



VT6065

VXI SERIAL INTERFACE MODULE

USER'S MANUAL

**82-0033-000
Rev. April 7, 2003**

VXI Technology, Inc.

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



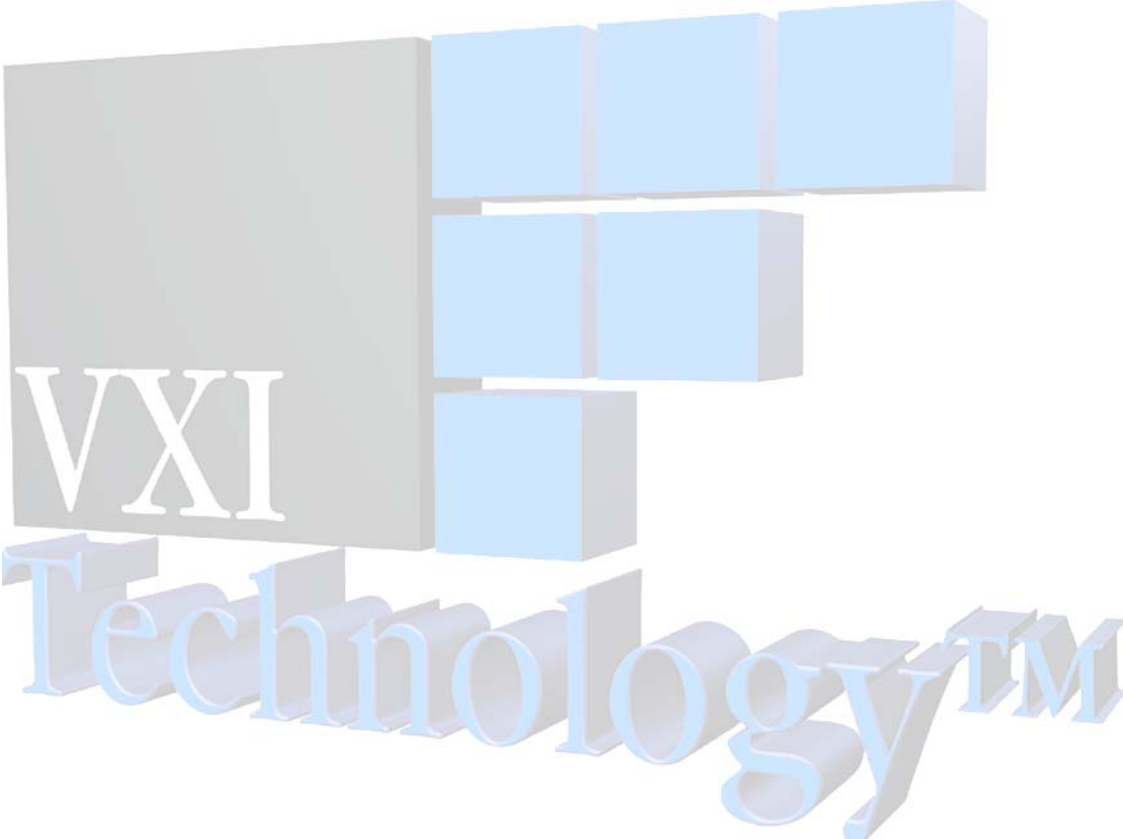


TABLE OF CONTENTS

INTRODUCTION	
Certification	7
Warranty	7
Limitation of Warranty	7
Restricted Rights Legend	7
DECLARATION OF CONFORMITY	8
GENERAL SAFETY INSTRUCTIONS	9
Terms and Symbols	9
Warnings	9
SUPPORT RESOURCES	11
SECTION 1	13
INTRODUCTION	13
Introduction	13
General Information	13
VT6065 Specifications	14
SECTION 2	15
PREPARATION FOR USE	15
Installation	15
Calculating System Power and Cooling Requirements	15
Setting the Chassis Backplane Jumpers	15
Setting the Logical Address	16
VXI Bus Interrupt Handler Setting	17
Mainframe Installation	17
Power-Up Self-Test Initialization	17
SECTION 3	19
PROGRAMMING	19
Introduction	19
Input Parameters	21
Block Mode	23
Termination Character And Termination Length	23
Connection Diagrams	25
SAMPLE PROGRAM	28
ERROR MESSAGES	34
SECTION 4	40
COMMAND DICTIONARY	40
Local Command Set	40
Alphabetical Command Listing	40
IEEE 488.2 COMMON COMMANDS	44
*CLS	44
*ESE	45
*ESR?	46
*IDN?	47
*OPC	48
*RCL	49
*RST	50
*SAV	51
*SRE	52
*STB?	53
*TRG	54
*TST?	55
*WAI	57

INSTRUMENT SPECIFIC SCPI COMMANDS	58
ABORt	58
FORMat	59
SERialCONTRol:CTS	61
SERial:CONTRol:DSR	62
SERial:CONTRol:DTR	63
SERial:CONTRol:RTS	65
SERial:BAUD	67
SERial:BITS	68
SERial:PACE?	69
SERial:PACE:THReshold:START	70
SERial:PACE:THReshold:STOP	71
SERial:PACE	72
SERial:PARity	73
SERial:SBITs	74
SERial:STANdard	75
SERial:TRANsmit:AUTO	76
SERial:TRANsmit:BAUD	77
SERial:TRANsmit:PACE	78
TERMinator:CHARacter	79
TERMinator:LENGth	80
TRACe:DATA	81
TRACe:DATA:LENGth?	82
TRACe:FREE?	83
TRACe:POINts	84
TRIGger:AUTO	85
TRIGger	86
TRIGger <channel>	87
TRIGger:SEQuence:SOURce	88
TRIGger:SEQuence:TIMer	89
REQUIRED SCPI COMMANDS	90
STATus:OPERation:CONDition?	91
STATus:OPERation:ENABle	92
STATus:OPERation?	93
STATus:PRESet	94
STATus:QUEStionable:CONDition?	95
STATus:QUEStionable:ENABle	96
STATus:QUEStionable?	97
SYSTem:ERRor?	98
SYSTem:VERSIon?	99
SECTION 5	100
THEORY OF OPERATION	100
Introduction	100
VXIbus MESSAGE-BASED INTERFACE	101
General	101
Microprocessor Clock	101
Microprocessor Memory	101
VXIbus Interface	101
Microprocessor Interrupts	102
Timer/Counter	102
VXIbus Interrupts	103
SYSFAIL and Access Indicators	103
Local Bus Interface	103
MAIN LOGIC BOARD	104
General	104
Microprocessor Bus	104

Serial UARTs.....	104
Line Drivers and Receivers.....	105
Self-Test Loop Back	106
Buffer RAMs	106
INDEX	108

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	VXI Serial Interface Module
MODEL NUMBER(S)	VT6065
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.

April 2003




 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 you suspect there is any damage to this product. ***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

This manual provides the necessary information to install the VXI Technology VT6065 Serial Interface Module in a VXIbus compatible chassis and to correctly operate the module.

GENERAL INFORMATION

The VT6065 Serial Interface module is a C-size VXIbus compatible card. The module is available with four or eight channels. Each channel can be configured as one of four standard interface buses.

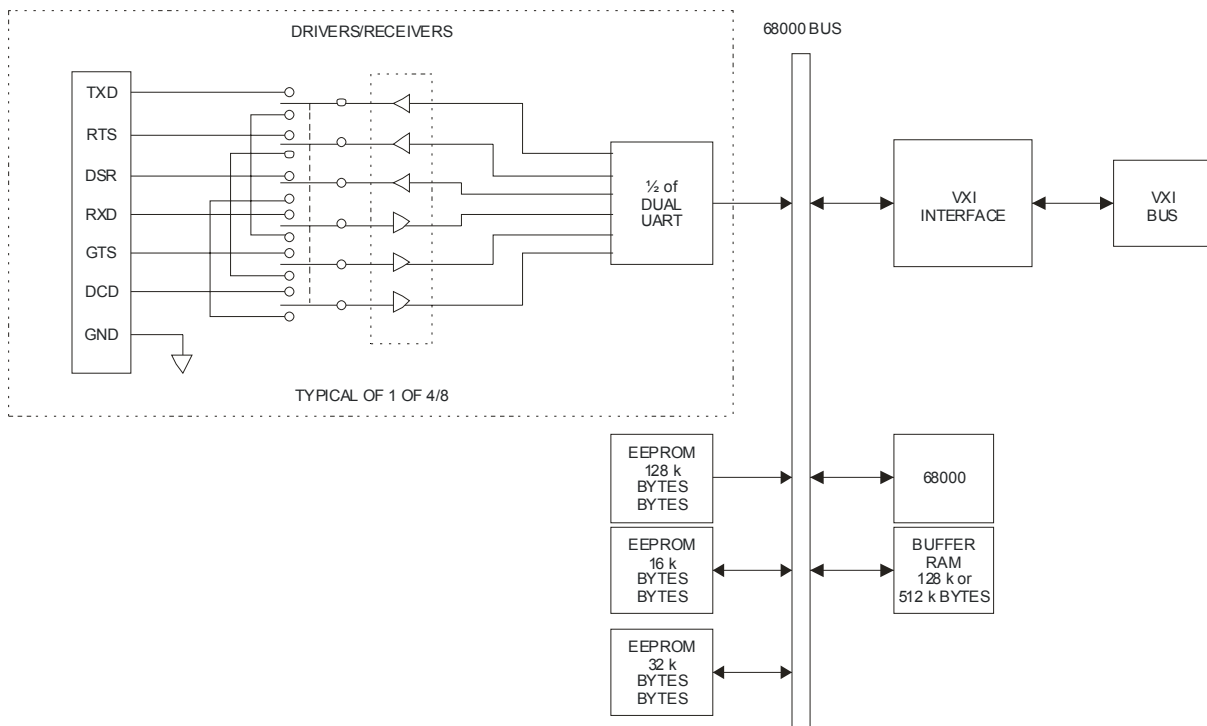


FIGURE 1-1 VT6065 BLOCK DIAGRAM

VT6065 SPECIFICATIONS

GENERAL SPECIFICATIONS	
CONFIGURATIONS	
VT6065-4	4-Channel C-Size Serial Interface Module
VT6065-4	8-Channel C-Size Serial Interface Module
BUFFER SIZE	
	128 k or 512 k bytes input and/or output
SERIAL PORT	
Baud Rate (Programmable)	300, 600, 1200, 2400, 4800, 9600, 19200, 38400
MINIMUM RECEIVE JITTER TOLERANCE	
Minimum Receive Jitter Tolerance	±4.5%
Data Length	5, 6, 7, 8
Parity	None, odd, even and mark or space
Stop Bits	1, 2
HANDSHAKING	
Software	XON/XOFF
Hardware	DSR and CTS for output pacing
POWER REQUIREMENTS (I_{PM})	
+5 V	2 A
±12 V	100 mA
±24 V	200 mA
WEIGHT	
	2.20 lb (1.0 kg)
DIMENSIONS	
	Single slot, C-size
VXIBUS	
	Message-Based Revision 1.4

SECTION 2

PREPARATION FOR USE

INSTALLATION

When the VT6065 is unpacked from its shipping carton, the contents should include the following items:

- (1) VT6065 VXIbus module.
- (1) VT6065 Serial Interface Module User's Manual (this manual).

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

Once the VT6065 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 VT6065. Once the chassis is found adequate, the VT6065's logical address and the backplane jumpers of the chassis should be configured before the VT6065'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's Manual for further details on setting the backplane jumpers.

SETTING THE LOGICAL ADDRESS

The logical address of the VT6065 is set by a single 8-position DIP switch located near the VMIP 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 signifies a logic 1; switches pushed away from the ON legend 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.

The VT6065 has an 8-position address switch used to set the module's logical address or set the module for dynamic configuration. The address switch is located on the "C" side of the module accessible through the sheet metal cover. Some sample examples are shown below:

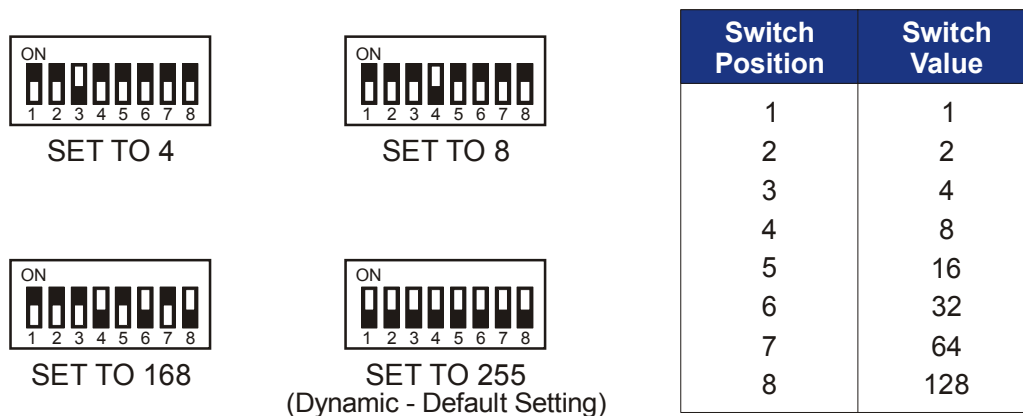


FIGURE 2-1 LOGICAL ADDRESS SWITCH SETTING EXAMPLES

An address of 255 (0xFF or all 1's) sets the module for dynamic configuration. In dynamic configuration, the Resource Manager sets the actual logical address of the module. Refer to your Resource Manager's manual for details about its addressing methods.

Any setting other than 255 indicates a static configuration. In a static configuration, the setting of the switches determines the address. Note that a logical address of 0 is not allowed.

A switch set to the ON position produces a 1 for that bit position. A switch set to OFF produces a 0 for that bit position. Note that the most significant bit of the address is the right hand switch labeled "8" and the least significant bit of the address is the left hand switch labeled "1".

VXIBUS INTERRUPT HANDLER SETTING

One programmable interrupt line is provided on the VT6065 module. This line is assigned by using the Assign Interrupter Line word serial protocol command (see p. 182 of the *VXibus System Specification Revision 1.4*). The Int_ID is set to 1. Normally the interrupt line is assigned by the Resource Manager during the power-up resource allocation.

The interrupt line is used for interrupts generated by changes in the status register. When a bit in the Service Request Enable register is a 1 and the corresponding bit in the status byte changes to a 1, the module generates a Request True event in the form of an interrupt. When all bits that cause Request True to be sent are removed, the module generates a Request False event in the form of an interrupt. It is recommended the user read Section 11 of *ANSI/IEEE-STD-488.2-1987* on device status reporting and Section E.4 of *VXibus System Specification Revision 1.4* on Protocol Events, as well as the section on interrupts in the user's manual for the Slot 0 being used.

MAINFRAME INSTALLATION

The VT6065 module is ready for operation when shipped. The VT6065 address switch is set to 255 at the factory to allow for dynamic configuration by the VXibus Slot 0 Resource Manager. See **LOGICAL ADDRESS SETTING** if static configuration (any switch setting other than 255) is desired.

To install the VT6065 in a C size VXibus chassis:

1. Ensure power on the VXI chassis is OFF.
2. Open the front cover of the chassis, if any.
3. Configure the interrupt daisy chain on the backplane so the installation slot will **not** have the slot bypassed. (Refer to the chassis operations manual for information on backplane configuration.)
4. Slide the VT6065 into the desired slot with the front panel nomenclature presented correctly.
5. Seat the module in the backplane.
6. Screw in the module's retaining screws.

POWER-UP SELF-TEST INITIALIZATION

At power-up, the VT6065 goes through a series of tests to assure proper operation and to establish the proper start-up state. Tests are performed on the module's ROM, RAM, non-volatile memory and the timer to make sure they are operating correctly.

Turn on the VXibus chassis. The FAIL LED should come on when power is applied. At the conclusion of the self-test (less than 4.9 seconds), the FAIL LED should be extinguished. An illuminated FAIL LED after 5 seconds is an indication of a module failure. Power should be turned off. The module should be removed from the VXibus chassis and examined for damage or improper installation.

SECTION 3

PROGRAMMING

INTRODUCTION

The VT6065 module is a VXIbus message-based device whose command set is compliant with the *Standard Command For Programmable Instruments* (SCPI) programming language. See the Sample Program in Appendix A for specific programming examples and command usage. Also refer to individual command descriptions.

All module commands are sent over the VXIbus backplane to the module using the VXI word serial **Byte Available** command. Each module command is terminated by setting the END bit in the **Byte Available** command with the last character of the command. All 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 utilizes 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. As an example, receive baud rate, receive parity, receive data bits, receive stop bits and receive pacing are all branches off the **SERial:RECEive** trunk. This makes it possible to combine several commands as follows:

```
SER2:REC:BAUD 1200;:PAR EVEN;:BITS 7;:SBIT 1
```

The above command is the same as the following:

```
SER2:REC:BAUD 1200
SER2:REC:PAR EVEN
SER2:REC:BITS 7
SER2:REC:SBIT 1
```

See the *Standard Command for Programmable Instruments (SCPI) Manual, Volume 1: Syntax & Style, Section 6*, for more information.

The SCPI commands are listed in upper and lower case. Character case is used to indicate different forms of the same command. Key words can have both a short form and a long form (some commands only have one form). The short form uses just the key word characters in uppercase. The long form uses the key word characters in uppercase plus the key word 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:

```
SER2:REC:BAUD 1200
SERIAL2:REC:BAUD 1200
SER2:RECEIVE:BAUD 1200
SERIAL2:RECEIVE:BAUD 1200
```

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

```
SERI2:REC:BAUD 1200
```

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 receive BAUD rate is on Channel 2, use the following command:

```
SER2:REC:BAUD ?
```

The response could be:

```
1200
```

This tells the user that the Channel 2 receive baud rate is set to 1200 baud.

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 2 format and receive baud rate could be set as follows:

```
FORM:DATA 2 INT;:SER2:REC:BAUD 1200
```

When combining commands, keep in mind the input buffer has a limit of 4095 characters. 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, place the ***rst** command in front of an initialization string as follows:

```
*RST;SER2:REC:BAUD 1200;:PAR EVEN;:BITS 7;:SBIT 1
```

Note that the **SER2:REC:BAUD 1200** command **did not** require a leading colon because there was no prior trunk of the SCPI tree.

INPUT PARAMETERS

Some commands have an input parameter of <boolean>. The actual parameter accepted can be on, off or a number. If the parameter is a number, it is evaluated as either zero or non-zero. Zero and off have the same effect. Non-zero and on have the same effect. When the query form of the command is used, the response is always numeric ASCII. The return value is either 0 or 1.

Some commands have optional parts and/or optional parameters. The parts between [and] are optional. If the optional part is a key word, the key word can be included or left out. If the optional part is a parameter, omitting the parameter will cause a default value to be used. The default value is 1 unless otherwise stated.

Some parameters are specified as <NRf>. <NRf> is a very general number format. It includes numbers we would call integers, numbers with decimal points, and numbers with exponential notation. Any of these forms may be used to specify the parameter. The query response of a <NRf> parameter depends on the actual command. See the example response for the specific query.

Spaces can be used freely among commands. The following commands are both correct and both have the same effect:

```
SER2 : REC : BAUD 1200
```

```
SER2:REC:BAUD1200
```

Some parameters are specified as <block>. <block> represents the general form of IEEE-STD-488.2 block. It specifies a group of data. Blocks can be used to pass any data including non-printable characters. The VT6065 utilizes two different types of blocks; definite length arbitrary blocks and indefinite length arbitrary blocks.

A definite length block specifies a group of data with a known length. A definite length block has the following general form:

```
#xyy..yzzzz...zzzz
```

The x is a single decimal digit from 1 to 9. It specifies the number of digits in the y's. The y's are a string of 1 to 9 decimal digits that represent the character's length of the data block. Note leading zeros are allowed in the y's. The z's represent the actual data.

Here are some examples of valid definite blocks:

#11A	A block with just the letter "A".
#12AB	A block with two letters "AB".
#19ABCDEFGHJI	A block with 9 letters "ABCDEFGHJI".
#210ABCDEFGHJIJ	A block with 10 letters "ABCDEFGHJIJ".
#9000000001A	A block with just the letter "A".

Here are some examples of **invalid** definite blocks:

#0	The first digit cannot be 0 (0 is for indefinite).
#11AB	The block says 1 character but there are two.
#AB	The character after the # must be 1 to 9.
#2AB	The y's must be decimal digits.

An indefinite length block specifies a group of data with an unknown length. An indefinite length block starts with #0 and ends with the newline (linefeed) character sent with the END indicator. If we represent data with z's and the newline with END as **NL**, an indefinite length block has the following general form:

#0zzzz...zzz**NL**

VXIbus message-based interfaces pass data with the **Byte Available** and **Byte Request** commands. Both commands pass 9 bits of information, 8 data bits and an END indication. When an indefinite length block is used to pass data, the **NL** is **not** considered part of the data.

Some examples may help to illustrate the indefinite length block. These examples are similar to the valid definite length blocks illustrated above:

#0 ANL	A block with just the letter "A".
#0 ABNL	A block with two letters "AB".
#0 ABCDEFGHJINL	A block with 9 letters "ABCDEFGHJI".
#0 ABCDEFGHJIJNL	A block with 10 letters "ABCDEFGHJIJ".

BLOCK MODE

Transmit channels can be in one of two major modes; character mode or block mode. In character mode, characters start transmitting as soon as they are placed in the transmit queue. Once a character is transmitted, it is no longer available. This mode of operation would be used when interacting with other instruments and where the data changes constantly. Block mode does not send the characters in the transmit queue until it is commanded to do so. The characters in the queue can be transmitted over and over on command. This might be used to stimulate a device over and over again. It might also be used to simulate a device that repeatedly sent the same data over and over again.

To place a channel in the block mode, use the **TRIG:AUTO** command to set auto mode **OFF** or **0**. Using this command clears the transmit queue. Characters can now be added to the queue using the **TRACe:DATA** command. Each new **TRACe:DATA** command adds characters to the end of the queue. When the queue contains the desired message, the transmission can be started.

Transmissions can be sent either as a single burst or at timed intervals. The **TRIG:SEQ:SOUR <channel> IMM** command sets the VT6065 for single burst. The **TRIG:SEQ:SOUR <channel> TIM** command says to use timed intervals. If timed intervals are used, the time value should be set with the **TRIG:SEQ:TIM <channel> <time_value>** command.

In either single burst or timed intervals, the transmission isn't started until the channel is triggered. If a channel is triggered before it has finished its previous transmission, an error is placed in the error queue and the transmission is stopped. There are three commands that can be used to trigger channels. The ***TRG** command and the **TRIG** command both act exactly the same. They start all block mode transmit channels that are not already running with timed intervals. This means it is possible to start many channels at nearly the same time. A single channel can be started with the **TRIG <channel>** command.

Once a timed interval block has been triggered, it automatically receives another trigger at the end of the timed interval. This continues until the **TRIG:SEQ:TIM <channel> <time_value>** command sets the time value to 0, or a ***RST** or **ABORT** command is received. Note that it will also stop if all the data hasn't been transmitted when it receives its next trigger. Setting the **<time_value>** to zero allows the current queue transmission to continue to the end of the queue. The ***RST** and **ABORT** commands will stop transmission as soon as possible.

TERMINATION CHARACTER AND TERMINATION LENGTH

When data is retrieved from the VT6065, it comes back in chunks or groups of data. The VT6065 needs to be told how to identify the end of those groups. The end of a group of data can be specified in three ways.

1. A particular character can be used to signal the end of the group. This can be used if every line sent to the VT6065 ended with a particular character, i.e., a serial interface where every line of received data always ended with a line-feed. A user could group the data into lines by specifying a line-feed character with the **TERM:CHAR 10** command (10 is the number for an ASCII line-feed character).

2. If the group of data always has the same number of characters in each group, the data group can be specified with a **TERM:LENG** command. This might be used if the received data had known fixed length character string, i.e., a peripheral that always sent 15 characters each time it reported. The user could tell the VT6065 to group data into individual reports by specifying a **TERM:LENG 15** command.
3. If the length of the data is not known and there is no single termination character that can be specified, the VT6065 can be told to group data into a "whatever is available" type group. This is done with a **TERM:LENG 0** command. Any data in the requested receive queue is returned when there is a request for data.

The end of data group indicator can also affect the format of retrieved data. If the termination length has been set to **0** (any characters available) and the format of the requested channel is set to **PACKed**, data is returned as an indefinite length block. If the data format is **PACKed** and the termination length is set to a positive number (a fixed record length), data is returned as a definite length block.

CONNECTION DIAGRAMS

Following are the pinouts for each of the available Serial Bus standards available on the VT6065.

RS-232 Configuration

First Channel as EIA/TIA-232-E Data Terminal Equipment (DTE)

<u>SCVT6065</u>		<u>DB25 Connector</u>	
GND	1	7	Signal Common
	2		N/C
TXDA	3	2	Transmitted Data
	4		N/C
RXDA	5	3	Received Data
RTSA	6	4	Request to Send
CTSA	7	5	Clear to Send
DTRA	8	20	Data Terminal Ready
DSRA	9	6	Data Set Ready

Second Channel as EIA/TIA-232-E Data Terminal Equipment (DTE)

<u>SCVT6065</u>		<u>DB25 Connector</u>	
GND	10	7	Signal Common
	11		N/C
TXDB	12	2	Transmitted Data
	13		N/C
RXDB	14	3	Received Data
RTSB	15	4	Request to Send
CTSB	16	5	Clear to Send
DTRB	17	20	Data Terminal Ready
DSRB	18	6	Data Set Ready

RS-422 Configuration

First Channel as EIA-422-A Data Terminal Equipment (DTE)

<u>SCVT6065</u>		<u>EIA-422-A</u>	
GND	1		Signal Common
TXDA+	2		TX+
TXDA-	3		TX-
RXDA+	4		RX+
RXDA-	5		RX-
RTSA	6		Request to Send (Out) (RS-423 levels)
CTSA	7		Clear to Send (In) (RS-423 levels)
DTRA	8		Data Terminal Ready (Out) (RS-423 levels)
DSRA	9		Data Set Ready (In) (RS-423 levels)

Second Channel as EIA-422-A Data Terminal Equipment (DTE)

<u>SCVT6065</u>		<u>EIA-422-A</u>	
GND	10	—————	Signal Common
TXDB+	11	—————	TX+
TXDB-	12	—————	TX-
RXDB+	13	—————	RX+
RXDB-	14	—————	RX-
RTSB	15	—————	Request to Send (Out) (RS-423 levels)
CTSB	16	—————	Clear to Send (In) (RS-423 levels)
DTRB	17	—————	Data Terminal Ready (Out) (RS-423 levels)
DSRB	18	—————	Data Set Ready (In) (RS-423 levels)

RS-423 Configuration

First Channel as RS-423-A Data Terminal Equipment (DTE)

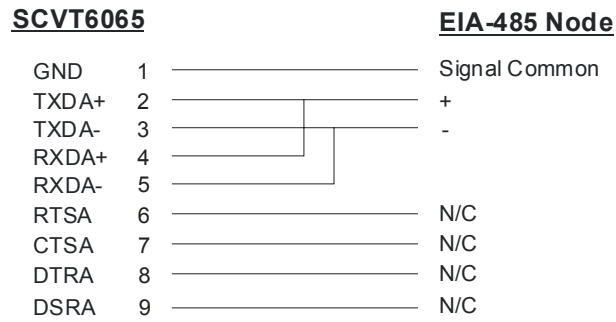
<u>SCVT6065</u>		<u>RS-423-A</u>	
GND	1	—————	Signal Common
	2	—————	N/C
TXDA	3	—————	Transmitted Data
	4	—————	N/C
RXDA	5	—————	Received Data
RTSA	6	—————	Request to Send
CTSA	7	—————	Clear to Send
DTRA	8	—————	Data Terminal Ready
DSRA	9	—————	Data Set Ready

Second Channel as RS-423-A Data Terminal Equipment (DTE)

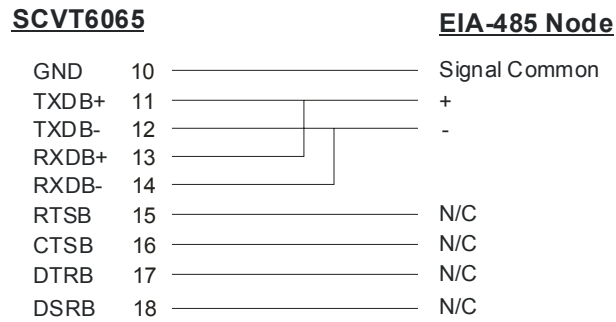
<u>SCVT6065</u>		<u>RS-423-A</u>	
GND	10	—————	Signal Common
	11	—————	N/C
TXDB	12	—————	Transmitted Data
	13	—————	N/C
RXDB	14	—————	Received Data
RTSB	15	—————	Request to Send
CTSB	16	—————	Clear to Send
DTRB	17	—————	Data Terminal Ready
DSRB	18	—————	Data Set Ready

RS-485 Configuration

First Channel as EIA-485 Node



Second Channel as EIA-485 Node



SAMPLE PROGRAM

Following is a single QUICK BASIC program showing programming examples of several of the functions of the VT6065.

```
' $INCLUDE: 'C:\GPIB-PC\QBDECL4.BAS'
REM *****/
REM NAME:      Model VT6065 SAMPLE PROGRAM           By: VXI Technology, Inc.  */
REM                                                    Date: 05-27-97          */
REM FUNCTION:   Familiarize User with Functions      */
REM                                                    */
REM            Demonstrate using SCPI command set to control */
REM                                                    */
REM            Model VT6065 VXIbus Serial Interface Module */
REM                                                    */
REM Programming */
REM Environment: PC with DOS and Quick Basic        */
REM                                                    */
REM            National VXI GPIB-PCIIA Handler      */
REM                                                    */
REM *****/

DIM resp AS STRING * 1000
DIM lin AS STRING * 100

REM First, establish your programming environment with necessary libraries

send1$ = "short_message"
send2$ = "#0ABCDEFGHIJKLMNPOQRSTUVWXYZ1234567890"
send3$ = "Hello, World"

REM Second, make contact with the controller

    CALL IBFIND("CTRL0", cr%)
    IF cr% < 0 THEN
        PRINT "Can't find the controller."
        STOP
    END IF

REM Next, Locate the VXI Module

    CALL IBFIND("VXITECH", bd%)
    IF bd% < 0 THEN
        CLS
        LOCATE 10, 10: PRINT "Unable to locate or communicate with the
VT-1."
        LOCATE 11, 10: PRINT "Check power and setup."
        STOP
    END IF
```

```
CALL IBWRT(bd%, "*RST")
```

```
REM Each channel has a transmit buffer TCHx and Receive Buffer RCHx
REM associated with the controlling hardware UART. You may alter the Baud,
REM parity, Number of bits (5,6,7,8), pacing method of flow control and
REM standard interface by sending the following module commands.
```

```
REM To select 4800 Baud for Channel 3
```

```
CALL IBWRT(bd%, "SYSTEM: COMMUNICATE:SERIAL 3 :RECEIVE:BAUD 4800")
```

```
REM OR , identically
```

```
CALL IBWRT(bd%, "ser3:baud 4800")
```

```
CALL IBWRT(bd%, "SYSTEM: COMMUNICATE:SERIAL 3 :TRANSMIT:BAUD 4800")
```

```
REM OR , identically
```

```
CALL IBWRT(bd%, "ser3:TRAN:baud 4800")
```

```
REM Commands are not case sensitive, and only the full command word or
REM it's specified abbreviation is accepted.
REM Command abbreviations are usually four characters long. See the manual
REM for each valid shortening.
REM Transmit autobaud is provided so the transmit baud follows the setting
REM of the receiver.
```

```
CALL IBWRT(bd%, "system:communicate:serial 3:transmit:auto ON")
```

```
REM OR
```

```
CALL IBWRT(bd%, "ser 3:tran:auto ON")
```

```
REM Here are the commands to set parity and word structure:
```

```
CALL IBWRT(bd%, "ser 3:rec:par EVEN")
```

```
REM AND
```

```
CALL IBWRT(bd%, "syst:comm:ser3:rec:bits 7")
```

```
REM OR
```

```
CALL IBWRT(bd%, "ser3:bits 7")
```

```
REM AND
```

```
CALL IBWRT(bd%, "syst:comm:ser:sbits 2")
```

```

REM          OR
          CALL IBWRT(bd%, "ser3:sbit 2")
REM Perhaps you want to select RS-422 on channel 4, then Type
          CALL IBWRT(bd%, "syst:comm:ser4:standard 422")
REM          OR
          CALL IBWRT(bd%, "SER4:STAN 422")
REM To select handshaking, you have two concerns, One set the desired
REM pacing mode, and two set valid start and stop thresholds.
          CALL IBWRT(bd%, "ser3:rec:pace XON")
REM          OR to set hardware handshaking select:
          CALL IBWRT(bd%, "ser3:control:RTS IBFull")
REM          AND
          CALL IBWRT(bd%, "ser3:REC:PACE:THReshold:STOP 3000")
REM          AND complementing
          CALL IBWRT(bd%, "ser3:REC:PACE:THR:START 500")

REM Examples for setting and reading transmit XON/XOFF or CTS handshaking on
REM channel 4 follow:
          CALL IBWRT(bd%, "ser4:cont:CTS ON")
REM          OR
          CALL IBWRT(bd%, "ser4:tran:paceXON")
REM          AND
          CALL IBWRT(bd%, "ser4:cont:CTS?")
          CALL ibrd(bd%, lin$)
REM          OR
          CALL IBWRT(bd%, "ser4:tran:pace?")
          CALL ibrd(bd%, lin$)
          CALL IBWRT(bd%, "")

```

REM The standard Model 6065 with 128K buffer memory has default buffer
REM allocations of 8K, which the user may reassign with the "trace:points"
REM **To reserve 4K for each transmit and receive on channel 2** Type

```
CALL IBWRT(bd%, "TRACe:POINtS RCH2,4096")  
CALL IBWRT(bd%, "TRACe:POINtS TCH2,4096")
```

REM To query the number of characters in these buffers Type

```
CALL IBWRT(bd%, "TRACe:DATA:LENGth? RCH1")  
CALL ibrd(bd%, lin$)  
CALL IBWRT(bd%, "TRACe:DATA:LENGth? TCH1")  
CALL ibrd(bd%, lin$)
```

REM At this time the response will be 0 .

REM **Command Verification**

REM Any of the commands can be verified by sending the system error request,
REM "SYSTEM:ERROR?", OR "SYS:ERR?" which will respond: "No error". The VT6065
REM has a two message error queue for any error output. If more than two
REM errors occur, the second is lost and replaced with "queue overflow".

REM Now do some house-keeping for future use.
REM The fixed length string allocates space for the driver's output to US.
REM Now that you have located the Model VT6065 and set the standard interface,
REM channel baud, and communications word structure, let's send a message
REM out serial channel 2 in RS-232 Standard Interface.
REM **Select data handling mode and variable length strings** for output as
REM follows:

```
CALL IBWRT(bd%, "FORMat:data 2 PACKed")  
CALL IBWRT(bd%, "TERM:LENGth 2 0")
```

REM To send a brief "Hello, World" message to channel 2 send the following.

```
CALL IBWRT(bd%, "trigger:AUTO 2 1")
```

REM trigger:auto sets the output channel 2 to character mode.
REM in this mode, characters are transmitted as available from the
REM IBWRT() routine.

```
CALL IBWRT(bd%, "trace:data tch2,#0 Hello, World")
```

REM The Hello message now appears out channel 2 with the configuration:
REM 4800 baud, 7 bits, EVEN Parity, 2 stop bits.

```

REM Block Mode
REM Another transmission mode is BLOCK, which provides for building a
REM message for one time or repetitive transmission on command trigger or
REM timed interval without further program intervention. To demonstrate,
REM channel 2 will continuously transmit "Start of test with any data here"
REM plus a data pattern of any length to fit in the TCHx buffer, perhaps:
REM "ABCDEFGHijklmnopqrstuvwxyz1234567890@"

```

```

    CALL IBWRT(bd%, "TRIGger:AUTO 2 0")
    CALL IBWRT(bd%, "TRIGger:SEQuence:SOURce 2 TIMer")
    CALL IBWRT(bd%, "TRIGger:SEQuence:TIMer 2 1")
    CALL IBWRT(bd%, "TRAC:DATA TCH2, #0Start a new test with any data
    here.")

```

```

    FOR time = 1 TO 10
        CALL IBWRT(bd%, "TRAC:DATA TCH2," + send2$)
    NEXT time

```

```

REM Now the transmit buffer TCH2 should have 36 + 10*36 = 396 characters.
REM The number of characters in any buffer may be read with a length query.

```

```

    CALL IBWRT(bd%, "TRACe:DATA:LENGth? TCH2")
    CALL Ibrd(bd%, resp$)
    RINT "number of characters in transmit 2 "; LEFT$(resp$, ibcnt%)

```

```

REM The command requesting the size is sent to the Model 6065 module. Then
REM the response must be read. One of the Basic subroutines supplies with
REM the National GPIB handler is IBRD called with the address of a fixed
REM length string as shown here. Notice above "DIM resp AS STRING * 4000".

```

```

REM Start continuous retransmission by sending a trigger. Stop the timed
REM retransmission with "trigger:tim 2: 0".

```

```

    CALL IBWRT(bd%, "TRIGger 2")
    CALL IBWRT(bd%, "trig:seq:tim 2 0")

```

```

REM This stops timed retransmission while maintaining the transmit buffer.

```

```

REM Receive data is constantly put into the channel receive buffer without
REM a command. The data must be removed from the buffer with the
REM "TRACE:DATA? RCHx" command as shown below. Once it is read, pointer are
REM updated and cannot be "re-read". Note resp$ is large here. Alternately,
REM a readback length may be set then short "lines" may be read.

```

```

    CALL ibwrt(bd%, "TRACe:data? RCH2")
    CALL IBRD(bd%, resp$)

```

```

REM To set a 36 char fixed length readback length use:

```

```

    CALL IBWRT(bd%, "term:leng 2 36")

```


REM To read the buffer back in small segments use:

```
CALL IBWRT(bd%, "term:leng 2 36")
CALL IBWRT(bd%, "TRAC:DATA:LENG? rch2")
CALL ibrd(bd%, lin$)
IF VAL(LEFT$(lin$, ibcnt%)) > 35 THEN
    CALL IBWRT(bd%, "TRACe:data? RCH2")
    CALL ibrd(bd%, lin$)
END IF

FOR lineno = 1 TO 10
    CALL IBWRT(bd%, "TRAC:data? RCH2")
    CALL ibrd(bd%, lin$)
    PRINT LEFT$(lin$, ibcnt%)
NEXT lineno
```

REM NOTICE the messages are preceded by #236?
REM The 2 is # of digits to follow for length.
REM The 36 is # of character of data to follow.

REM PRINT MID\$(lin\$,4,36) Would print only the data, without #236.

```
CALL IBWRT(bd%, "TRAC:DATA:LENG? rch2")
CALL ibrd(bd%, lin$)
WHILE VAL(LEFT$(lin$, ibcnt%)) > 35
    CALL IBWRT(bd%, "TRAC:data? RCH2")
    CALL ibrd(bd%, lin$)
    PRINT MID$(lin$, 5, 36)
    CALL IBWRT(bd%, "TRAC:DATA:LENG? rch2")
    CALL ibrd(bd%, lin$)
WEND
```

END

ERROR MESSAGES

When the module detects errors, an error message is placed in the error queue and one of the bits is set in the standard ESR. The error queue has a fixed length of two in the VT6065. If more than a maximum number of error messages are placed in the error queue without being read, the last error message is over written with the queue overflow error message. The SYST:ERR? query is used to read the error queue. If there are no errors in the queue, the **No error** message is returned. See *SCPI Volume 2: Command Reference* Section 19.7 for a detailed description of error processing.

Error messages in SCPI have a common format:

<error number>,"<description>[;<device dependent info>]"

The <error number> is negative for all errors and zero for the no error condition.

0, "No error"

The error queue is completely empty. Every error message has been read or the queue was purposely cleared by *CLS. The queue is empty at power-on.

Error numbers in the range of -100 to -199 indicate a syntax error.

-100, "Command error; Line too long, scan aborted"

The input buffer is big enough to hold 4095 characters. If more than 4095 characters are sent without a terminator, the input buffer overflows and this error is generated.

-101, "Invalid character; A comma was expected but not found"

Transmit characters may be entered as a list of numbers separated by commas. A list of transmit characters was partially recognized, but some problem was encountered. Typically, this results from a misplaced or missing comma in the list.

-101, "Invalid character; A number or block was expected but not found"

A command that required either a number (data) or a block didn't find what it was looking for. This may be due to a missing parameter or an invalid block.

-102, "Syntax error; Unknown command: ..."

The module didn't recognize the command. Up to 40 characters of the offending command are included in the error message (the characters are inserted in the message in lieu of the ellipses above).

-120, "Numeric data error; Buffers must have a size of at least 2"

All buffers have a minimum size of 2 characters. A command attempted to set a receive or transmit buffer to a size less than 2.

-120, "Numeric data error; Data values are 0 to 255"

Transmit characters may be entered as a list of numbers separated by commas. Since the largest number of data bits is 8, the largest data value allowed is 255. This error indicates that a command to load a transmit buffer attempted to enter a number outside of the 0 to 255 range.

-120, "Numeric data error; Invalid baud rate"

The baud rate of a channel can be programmed, but there are restrictions on what baud rates are allowed. An invalid baud rate was entered in a baud rate command. See the specific baud rate command for a list of valid baud rates.

-120, "Numeric data error; Invalid number of bits"

The number of data bits for a channel is programmable, but there are restrictions on the number of bits. An invalid number of data bits was entered in a data bits command. See the specific data bits command for a list of valid data bits.

-120, "Numeric data error; Invalid number of stop bits"

The number of stop bits for a channel is programmable, but there are restrictions on the number of bits. An invalid number of stop bits was specified in a stop bits command. See the specific stop bits command for list of valid stop bits.

-120, "Numeric data error; Threshold must be a positive number"

The start and stop thresholds for receive pacing for a channel are programmable, but the values must be positive and inside the buffer. A negative number was specified for a pacing threshold. See the specific threshold command for the valid range.

-120, "Numeric data error; Valid channel numbers are 1 to 4"

On a module with 4 channels, an attempt was made to specify a channel other than 1 to 4.

-120, "Numeric data error; Valid channel numbers are 1 to 8"

On a module with 8 channels, an attempt was made to specify a channel other than 1 to 8.

-120, "Numeric data error; Valid interfaces are 232, 422, 423 or 485"

An attempt was made to specify an interface standard other than those allowed.

-120, "Numeric data error; Valid receive trace names are RCH1 to RCH4"

On a module with 4 channels, an attempt was made to specify a receive trace buffer other than 1 to 4.

-120, "Numeric data error; Valid receive trace names are RCH1 to RCH8"

On a module with 8 channels, an attempt was made to specify a receive trace buffer other than 1 to 8.

-120, "Numeric data error; Valid SAV/RCL records are 1 to 16"

An attempt was made to access a non-volatile record other than 1 to 16.

-120, "Numeric data error; Valid termination lengths are 0 or larger"

A negative number was specified for the termination length of a receive buffer.

-120, "Numeric data error; Valid termination numbers are 0 to 255"

A termination character other than 0 to 255 was specified.

-120, "Numeric data error; Valid time values are 0 to 2147 seconds"

A number outside the range of 0 to 2147 was specified for a time value.

-120, "Numeric data error; Valid transmit trace names are TCH1 to TCH4"

On a module with 4 channels, an attempt was made to specify a transmit trace buffer other than 1 to 4.

- 120, "Numeric data error; Valid transmit trace names are TCH1 to TCH8"**
On a module with 8 channels, an attempt was made to specify a transmit trace buffer other than 1 to 8.
- 121, "Invalid character in number"**
A non-numeric character was encountered in what should have been a number. A missing semicolon can be one cause of this error message.
- 160, "Block data error; Block length was non-numeric"**
A definite length block was specified, but the length of the block was not a recognized number. Check the format of the definite length block.
- 160, "Block data error; Character after # wasn't a digit"**
A block was specified, but the character after the start of block character (#) wasn't a decimal digit. Check the format of the block.
- 161, "Invalid block data; Expected more data than what was supplied"**
When a definite length block is specified, there is a fixed length. A definite length block was specified, but the **END** indicator was recognized before all the data was found. Check the format of the definite length block.
- 200, "Execution error; Can't fill buffer while using it"**
This error occurs when a channel is used in block mode. When a block mode channel is transmitting, its buffer can't be filled. This error indicates there was an attempt to fill a transmit buffer while it was transmitting.
- 210, "Trigger error; A block was triggered before send was finished"**
This error occurs when a channel is used in block mode using timed transmit. The module attempts to send a channel buffer within the time period specified. If the time to start a new transmission occurs and the previous transmission hasn't finished, this error will occur. Some possible problems are that there is too much data to send in the specified time interval and/or handshaking is slowing down the transmission.
- 221, "Settings conflict; Not enough memory to allocate buffer"**
There was an attempt to allocate more memory to buffers than is actually available. Memory is allocated with each POINTs command. This error can occur when allocating non-symmetric buffers. Always allocate the smaller buffers first, then the larger buffers.
- 221, "Settings conflict; RTS mode can't be set in 485"**
In RS-485 mode, the module internally uses the RTS signal to control the enabling and disabling of the transmit drivers. The RTS signal can't be used in RS-485 mode.
- 221, "Settings conflict; Start threshold wasn't inside buffer"**
The start and stop thresholds for receive pacing for a channel are programmable, but the values must be positive and inside the buffer by a specific amount. Something was changed that made a start threshold larger than the allowed range. See the specific threshold command for the valid range.
- 221, "Settings conflict; Stop threshold wasn't inside buffer"**
The start and stop thresholds for receive pacing for a channel are programmable, but the values must be positive and inside the buffer by a specific amount. Something was changed that made a stop threshold larger than the allowed range. See the specific threshold command for the valid range.

-221, "Settings conflict; Termination length wasn't less than buffer size"

When using termination length to establish the end of a transmission, the length must be less than the size of the receive buffer for the specified channel. Something was changed that made the termination length larger than the allowed range.

-222, "Data out of range; Start threshold wasn't inside buffer"

The start and stop thresholds for receive pacing for a channel are programmable, but the values must be positive and inside the buffer by a specific amount. A command specified a start threshold larger than the allowed range. See the specific threshold command for the valid range.

-222, "Data out of range; Stop threshold wasn't inside buffer"

The start and stop thresholds for receive pacing for a channel are programmable, but the values must be positive and inside the buffer by a specific amount. A command specified a stop threshold larger than the allowed range. See the specific threshold command for the valid range.

-222, "Data out of range; Termination length wasn't less than buffer size"

When using termination length to establish the end of a transmission, the length must be less than the size of the receive buffer for the specified channel. A command specified a termination length larger than the allowed range.

-223, "Too much data; Transmit buffer full"

An attempt was made to put too much data into a transmit buffer. Either too much data is being put in the buffer or the data is not being sent fast enough and the buffer overflowed.

-231, "Data questionable; Framing error occurred on channel *x*"

The value *x* is a channel number from 1 to 4 on a 4-channel module and from 1 to 8 on an 8-channel module. A framing error occurred on the specified receive channel. Framing errors are usually an indication of different baud rates between the transmitting source and the receiver.

-231, "Data questionable; Overrun error occurred on channel *x*"

The value *x* is a channel number from 1 to 4 on a 4-channel module and from 1 to 8 on an 8-channel module. An overrun error occurred on the specified receive channel. Overrun errors are usually an indication of the module being too busy to service all need functions.

-231, "Data questionable; Parity error occurred on channel *x*"

The value *x* is a channel number from 1 to 4 on a 4-channel module and from 1 to 8 on an 8-channel module. A parity error occurred on the specified receive channel. Parity errors are usually an indication of different parity or data length between the transmitting source and the receiver.

-231, "Data questionable; Receive buffer overflow occurred on channel *x*"

The value *x* is a channel number from 1 to 4 on a 4-channel module and from 1 to 8 on an 8-channel module. An overflow error occurred on the specified receive channel. Overflow errors indicate the VT6065's master didn't read the receive data fast enough and an input buffer overflowed.

-350, "Queue overflow"

More errors occurred than the error queue could hold. The error queue is two messages deep. If a third error is detected, the second and third messages are thrown away and this message is entered in their place.

SECTION 4

COMMAND DICTIONARY

LOCAL COMMAND SET

This section contains a description of the individual commands recognized by the module. The commands are grouped into three categories.

The first category is IEEE-STD-488.2 Common Commands. These commands are included to comply with IEEE-STD-488.2 and SCPI. They deal with status reporting and the use of the status byte. They are also used for the control of interrupts and for device synchronization.

The second category is Device Dependent SCPI Commands. These are used to configure and operate the instrument.

The third category is Required SCPI Commands. They are included to comply with SCPI. These commands deal with SCPI specific status reporting including error reporting.

ALPHABETICAL COMMAND LISTING

The following tables provide an alphabetical listing of each command supported by the VM4018 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. Note that calibration values revert to the values stored in non-volatile memory upon reset. Using the CALibration:DEFault command will return calibration values back to known, factory preset values.

In order for CALibration commands/queries to be executed, calibration security must be turned off. If security is not turned off, a "-203, Command Protected" error will be returned. See the CALibration:SECure:CODE command for information on calibration security.

TABLE 4-1 IEEE 488.2 COMMON COMMANDS

Command	Description	*RST	*RST Value
*CLS	Clears the Status Register	X	
*ESE	Sets the Event Status Enable Register	X	
*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		
*RCL	Recalls a previously stored configuration		N/A
*RST	Resets the module to a known state		N/A
*SAV	Store the present configuration		N/A
*SRE	Set the service request enable register		
*STB?	Query the Status Byte Register		
*TRG	Causes a trigger event to occur		
*TST?	Starts and reports a self-test procedure		N/A
*WAI	Halts execution and queries	X	

TABLE 4-2 INSTRUMENT SPECIFIC SCPI COMMANDS

Command	Description	*RST	*RST Value
ABORt	Stops current block operations	X	N/A
FORMat	Set the data format for retrieving received characters	X	ASCIi
SERialCONTRol:CTS	Enable or disable Clear To Send (CTS) handshake mode		N/A
SERial:CONTRol:DSR	Enable or disable Data Set Ready (DSR) handshake mode	X	0
SERial:CONTRol:DTR	Set the DTR (Data Terminal Ready) handshake mode	X	OFF
SERial:CONTRol:RTS	Set the Request To Send (RTS) handshake mode	X	OFF
SERial:BAUD	Set the receive baud rate	X	9600
SERial:BITS	Set the number of data bits	X	8
SERial:PACE?	Query the receive software handshake mode		N/A
SERial:PACE:THReshold:STARt	Set the number of characters in the receive buffer to enable data handshaking	X	START threshold buffer size - 1024
SERial:PACE:THReshold:STOP	Set the number of characters in the receiver buffer to disable data handshaking	X	STOP threshold buffer size - 2048
SERial:PACE	Set the receive software handshake mode	X	NONE
SERial:PARity	Set the parity mode	X	NONE
SERial:SBITs	Set the number of stop bits	X	1
SERial:STANdard	Set the electrical interface standard	X	232
SERial:TRANsmit:AUTO	Couple or uncouple the transmit baud rate	X	1
SERial:TRANsmit:BAUD	Set the transmit baud rate on a specified channel	X	9600
SERial:TRANsmit:PACE	Set the transmit software handshake mode	X	NONE
TERMinator:CHARacter	Set the character for end of line	X	1
TERMinator:LENGth	Set the character count for end of line	X	1
TRACe:DATA	Load data into the specified transmit queue		N/A
TRACe:DATA:LENGth?	Query the number of characters in a channel queue		N/A
TRACe:FREE?	Query the amount of buffer memory unused in a queue		N/A
TRACe:POINts	Set the size of a transmit or receive queue	X	Equal allocation
TRIGger:AUTO	Set the mode of operation of a channel to either character or block mode	X	1
TRIGger	Trigger all channels for block transmission		N/A
TRIGger <channel>	Trigger one channel for block transmission		N/A
TRIGger:SEQUence:SOURce	Set the trigger source for a block mode transmit channel	X	IMM
TRIGger:SEQUence:TIMer	Set the time interval for re-sending data blocks	X	0 s

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 to which the module complies.		N/A

IEEE 488.2 COMMON COMMANDS

*CLS

Purpose	Clears the Status Register	
Type	IEEE 488.2 Common Command	
Command Syntax	*CLS	
Command Parameters	None	
*RST Value	*RST performs all the functions of *CLS	
Query Syntax	None – Command Only	
Query Parameters	N/A	
Query Response	N/A	
Description	This command clears all event registers, clears the request for OPC (Operation Complete) flag, and clears all queues (except for the output queue).	
Example	Command / Query	Response (<i>Description</i>)
	*CLS	
Related Commands	None	

***ESE**

Purpose	Sets the bits of the Event Status Enable Register	
Type	IEEE 488.2 Common Command	
Command Syntax	*ESE <mask>	
Command Parameters	<mask> = numeric ASCII value	
*RST Value	N/A	
Query Syntax	*ESE?	
Query Parameters	None	
Query Response	Numeric ASCII value	
Description	<p>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:</p> <p>Bit 0 - Operation Complete Bit 1 - Request Control (not used in VT6065) Bit 2 - Query Error Bit 3 - Device Dependent Error (not used in VT6065) Bit 4 - Execution Error Bit 5 - Command Error Bit 6 - User Request (not used in VT6065) Bit 7 - Power On</p> <p>The Event Status Enable query reports the current contents of the Event Status Enable Register.</p>	
Example	Command / Query	Response (Description)
	*ESE 36	<i>(This command enables the ESB in the status register to be set when there is a query error or a command error. The value 36 (00100100) comes from 4 (Query Error) + 32 (Command Error).)</i>
	*ESE?	<i>36 (Queries and reports that the Query Error (4) and the Command Error (32) events are enabled for generation of the ESB in the status register.)</i>
Related Commands	ESR?	

***ESR?**

Purpose	Queries and clears the Standard Event Status Register	
Type	IEEE 488.2 Common Command	
Command Syntax	None – Query Only	
Command Parameters	N/A	
*RST Value	N/A	
Query Syntax	*ESR?	
Query Parameters	None	
Query Response	Numeric ASCII value	
Description	<p>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:</p> <p>Bit 0 - Operation Complete Bit 1 - Request Control (not used in VT6065) Bit 2 - Query Error Bit 3 - Device Dependent Error (not used in VT6065) Bit 4 - Execution Error Bit 5 - Command Error Bit 6 - User Request (not used in VT6065) Bit 7 - Power On</p> <p>The Operation Complete bit is set when it receives an *OPC command.</p> <p>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.</p> <p>The Execution Error bit is set when an execution error is detected. Errors in the range of -200 to -299 are execution errors.</p> <p>The Command Error bit is set when a command error is detected. Errors in the range of -100 to -199 are execution errors.</p> <p>The Power-On bit is set when the module is powered-on, or it receives a reset via the VXIbus control register. Once this bit is cleared by the *ESR? command, it remains cleared.</p> <p>See <i>Section 3: ERROR MESSAGES</i> for a list of execution errors.</p>	
Example	Command / Query	Response (<i>Description</i>)
	*ESR?	4 (<i>The response means the Query Error event has occurred. A second query would produce a response of 0 because the query clears the register.</i>)
Related Commands	*ESE	

***IDN?**

Purpose	Queries the module for its identification string	
Type	IEEE 488.2 Common Command	
Command Syntax	None – Query Only	
Command Parameters	N/A	
*RST Value	N/A	
Query Syntax	*IDN?	
Query Parameters	None	
Query Response	ASCII character string	
Description	<p>The Identification Query returns the identification string of the VT6065 module. The response is character data. The response has four fields separated by commas. The first field is the name of the manufacturer. The second field is the model number. The third field is an optional serial number. If the serial number is not supplied, the third field contains a zero. The final field is the firmware revision number.</p>	
Example	Command / Query	Response (<i>Description</i>)
	* IDN?	VXI Technology Inc. VT6065,0,1.8 <i>(This tells the user that the manufacturer is VXI Technology Inc.. The model is VT6065. The serial number is not used, and therefore is zero. The firmware revision is 1.8. This is an example only – firmware revisions may change as features are enhanced).</i>
Related Commands	None	

***OPC**

Purpose	Sets the OPC bit in the Event Status Register	
Type	IEEE 488.2 Common Command	
Command Syntax	*OPC	
Command Parameters	None	
*RST Value	*RST removes any pending *OPC request	
Query Syntax	*OPC?	
Query Parameters	None	
Query Response	1	
Description	The Operation Complete command sets the OPC bit in the Event Status register when all pending operations have completed. Note that the VT6065 never has any pending operations so it sets the OPC bit when the command is received.	
Example	Command / Query	Response (Description)
	*OPC	<i>(Sets the OPC bit when all pending operations have finished.)</i>
	*OPC?	<i>1 (Indicates that all pending operations have completed.)</i>
Related Commands	*WAI	

***RCL**

Purpose	Recalls a previously stored configuration	
Type	IEEE 488.2 Common Command	
Command Syntax	*RCL <n>	
Command Parameters	<n> = 1 to 16	
*RST Value	N/A	
Query Syntax	None – Command Only	
Query Parameters	N/A	
Query Response	N/A	
Description	<p>The Recall Saved State recalls a previously stored VT6065 configuration. <n> (1 to 16) is the configuration number of the desired (previously stored) set-up. During power-up, location 1 is used to set the module configuration. The following information is saved in each configuration for all channels:</p> <p>FORMat:DATA SERial:CONTRol:CTS SERial:CONTRol:DSR SERial:CONTRol:DTR SERial:RECeive:BAUD SERial:RECeive:PARity SERial:RECeive:BITS SERial:RECeive:SBITs SERial:RECeive:PACe SERial:RECeive:PACe:THReshold:START SERial:RECeive:PACe:THReshold:STOP SERial:STANdard SERial:TRANsmit:AUTO SERial:TRANsmit:BAUD SERial:TRANsmit:PACe TERMinator:LENGth or TERMinator:CHARacter TRACe:POINts TRIGger:AUTO TRIGger:SEQuence:TIMer</p> <p>The *RST command does not affect this command.</p>	
Example	Command / Query	Response (Description)
	*RCL 4	
Related Commands	*SAV <n>	

***RST**

Purpose	Resets the module's hardware and software to a known state	
Type	IEEE 488.2 Common Command	
Command Syntax	*RST	
Command Parameters	None	
*RST Value	N/A	
Query Syntax	None – Command Only	
Query Parameters	N/A	
Query Response	N/A	
Description	<p>Resets the VT6065's hardware and firmware to a known state. The following items are set for each channel as shown:</p> <p> FORMat:DATA ASCii SERial:CONTRol:CTS 0 SERial:CONTRol:DSR 0 SERial:CONTRol:DTR OFF SERial:RECeive:BAUD 9600 SERial:RECeive:PARity NONE SERial:RECeive:BITS 8 SERial:RECeive:SBITS 1 SERial:RECeive:PACE NONE SERial:RECeive:PACE:THReshold:STARt buffer length – 2 k SERial:RECeive:PACE:THReshold:STOP buffer length – 1 k SERial:STANdard setting in RCL 1 SERial:TRANsmit:AUTO 0 SERial:TRANsmit:BAUD 9600 SERial:TRANsmit:PACE NONE TERMinator:LENGth 1 TRACe:POINts all to 1/n of total buffer size TRIGger:AUTO 1 TRIGger:SEQuence:TIMer 0.0 </p>	
Example	Command / Query	Response (Description)
	*RST	
Related Commands	None	

***SAV**

Purpose	Store the present configuration	
Type	IEEE 488.2 Common Command	
Command Syntax	*SAV <n>	
Command Parameters	<n> = Number used for storing the set-up (1 - 16)	
*RST Value	N/A	
Query Syntax	None – Command Only	
Query Parameters	N/A	
Query Response	N/A	
Description	The Save State command stores the present configuration for all channels in non-volatile memory. <n> (1 to 16) is the configuration number used for storing the set-up. See *RCL for the information stored. The *RST command does not affect this command. It uses the STANDARD value stored in configuration 1.	
Example	Command / Query	Response (Description)
	*SAV 4	
Related Commands	*RCL <n>	

***SRE**

Purpose	Sets the Service Request Enable Register	
Type	IEEE 488.2 Common Command	
Command Syntax	*SRE <mask>	
Command Parameters	<mask> = Numeric ASCII value	
*RST Value	N/A	
Query Syntax	*SRE?	
Query Parameters	None	
Query Response	Numeric ASCII value	
Description	<p>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.</p> <p>Note: Bit 6 is always internally cleared to zero as required by IEEE 488.2 section 11.3.2.3.</p> <p>The layout of the Service Request Enable Register is:</p> <ul style="list-style-type: none"> 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 	
Example	Command / Query	Response (Description)
	*SRE 4 *SRE?	(Enables interrupts when the Error Queue has data). 68 (Indicates interrupts are enabled for the Error Queue (value 4) and the Master Summary Status (value 64).)
Related Commands	None	

***STB?**

Purpose	Queries the Status Byte register	
Type	IEEE 488.2 Common Command	
Command Syntax	None – Query Only	
Command Parameters	N/A	
*RST Value	N/A	
Query Syntax	*STB?	
Query Parameters	None	
Query Response	Numeric ASCII value	
Description	<p>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:</p> <ul style="list-style-type: none"> 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 	
Example	Command / Query	Response (<i>Description</i>)
	*STB?	16 (<i>The response tells the user that there is a message available in the output queue.</i>)
Related Commands	None	

***TRG**

Purpose	Causes a trigger event to occur	
Type	IEEE 488.2 Common Command	
Command Syntax	*TRG	
Command Parameters	None	
*RST Value	N/A	
Query Syntax	None – Command Only	
Query Parameters	N/A	
Query Response	N/A	
Description	Trigger is used to generate a trigger event. This will trigger all transmit channels set to run in block mode that are not already running with a timed trigger. See BLOCK MODE for more details.	
Example	Command / Query	Response (Description)
	*TRG	<i>(If Channel 4 was set in block mode and ready to go, this command would start Channel 4 sending its transmit queue.)</i>
Related Commands	TRIGger:SEQuence:IMMediate	

***TST?**

Purpose	Causes a self-test procedure to occur and queries the results
Type	IEEE 488.2 Common Command
Command Syntax	None – Query Only
Command Parameters	N/A
*RST Value	N/A
Query Syntax	*TST? [<channel>]
Query Parameters	<channel> = Specify channel #
Query Response	Numeric ASCII value
Description	<p>The Self-Test Query performs a self-test and responds with the results. This command performs a series of tests on global resources and on individual channels. The module is isolated from outside connections by opening relays while the tests are performed. All data in the trace buffers, both transmit and receive, are lost. After the tests are completed, the module is placed in the same mode of operation as it was before the tests. The command without a channel performs the actual tests. The channel-less command must be run first to have good results. The command form with the channel reports further information about a particular channel, but does not perform any more tests.</p> <p>All tests are performed regardless of the results of previous tests. The tests are:</p> <ol style="list-style-type: none"> 1. Buffer RAM test 2. Determination of the number of channels installed 3. Channel tests <p>The channel tests are performed for each channel installed. The channels tests are:</p> <ol style="list-style-type: none"> 1. RTS-CTS wrap 2. DTR-DSR wrap 3. A non-interrupt test using RS-232 mode 4. An interrupt test using RS-232 mode 5. A non-interrupt test using RS-422 mode 6. An interrupt test using RS-422 mode 7. A non-interrupt test using RS-423 mode 8. An interrupt test using RS-423 mode 9. A non-interrupt test using RS-485 mode 10. An interrupt test using RS-485 mode 11. A non-interrupt test using the UART internal wrap mode 12. An interrupt test using the UART internal wrap mode <p>All channel tests except the internal UART wrap tests go through the driver/receiver chips used for that mode of operation. Relays are used to wrap the drivers back to receivers. After the tests are performed, the results are analyzed for failures. The information reported from the analysis is dependent on the form of the command used.</p>

Description (Cont.)	<p>The command form without a channel - Performs a self-test and returns a result that indicates which, if any, channels failed. A value of 0 indicates all tests passed for a particular channel. The format with no channel responds with a binary weighted decimal number. Each bit is a 1 for a channel failure and a 0 for a nonexistent or passed channel.</p>																										
	<table style="width: 100%; border: none;"> <tr> <td style="padding: 5px;">Bit 0 – Channel 1 Failed</td> <td style="padding: 5px;">Bit 5 – Channel 6 Failed</td> </tr> <tr> <td style="padding: 5px;">Bit 1 – Channel 2 Failed</td> <td style="padding: 5px;">Bit 6 – Channel 7 Failed</td> </tr> <tr> <td style="padding: 5px;">Bit 2 – Channel 3 Failed</td> <td style="padding: 5px;">Bit 7 – Channel 8 Failed</td> </tr> <tr> <td style="padding: 5px;">Bit 3 – Channel 4 Failed</td> <td style="padding: 5px;">Bit 8 – RAM Test Fail</td> </tr> <tr> <td style="padding: 5px;">Bit 4 – Channel 5 Failed</td> <td style="padding: 5px;">Bit 9 – Channel Count Failed</td> </tr> </table> <p>The RAM Test Failed bit says the buffer RAM (area used for the queues) failed to pass a pseudo random pattern test, or an all zeros test. The Channel Count Failed bit says the number of installed channels detected is different from the number of channels detected at power-up.</p> <p>The command form with a channel - Performs a self-test and returns a result which indicates which test failed on the selected channel. The format with a channel responds with a binary weighted decimal number. Each bit is a 1 for each test on that channel that failed.</p> <table style="width: 100%; border: none;"> <tr> <td style="padding: 5px;">Bit 0 – RAM Test</td> <td style="padding: 5px;">Bit 8 – RS-422 #2</td> </tr> <tr> <td style="padding: 5px;">Bit 1 – Channel Count</td> <td style="padding: 5px;">Bit 9 – RS-423 #1</td> </tr> <tr> <td style="padding: 5px;">Bit 2 – RTS-CTS</td> <td style="padding: 5px;">Bit 10 – RS-423 #2</td> </tr> <tr> <td style="padding: 5px;">Bit 3 – DTR-DSR</td> <td style="padding: 5px;">Bit 11 – RS-485 #1</td> </tr> <tr> <td style="padding: 5px;">Bit 4 – Timeouts</td> <td style="padding: 5px;">Bit 12 – RS-485 #2</td> </tr> <tr> <td style="padding: 5px;">Bit 5 – RS-232 #1</td> <td style="padding: 5px;">Bit 13 – Internal #1</td> </tr> <tr> <td style="padding: 5px;">Bit 6 – RS-232 #2</td> <td style="padding: 5px;">Bit 14 – Internal #2</td> </tr> <tr> <td style="padding: 5px;">Bit 7 – RS-422 #1</td> <td></td> </tr> </table> <p>The RAM Test and Channel Count bits are the same as in the channel-less form of the command. They are reported here again because they are an indication that a particular channel may not work properly.</p> <p>The Timeouts bit indicates the type of failure that occurred on the channel wrap tests. It is mainly for further diagnosis of a problem.</p> <p>Each electrical mode (RS-232, RS-422, RS-423, RS-485) is run with the appropriate drivers and receivers in two different tests. The first test sends and receives data on a character-by-character basis without relying on interrupts. The second test is similar, but uses interrupts for both receive and transmit.</p> <p>The internal tests are the same tests performed in the different electrical modes, but use the internal loopback mode of the UART, bypassing all drivers, receivers and relays.</p>		Bit 0 – Channel 1 Failed	Bit 5 – Channel 6 Failed	Bit 1 – Channel 2 Failed	Bit 6 – Channel 7 Failed	Bit 2 – Channel 3 Failed	Bit 7 – Channel 8 Failed	Bit 3 – Channel 4 Failed	Bit 8 – RAM Test Fail	Bit 4 – Channel 5 Failed	Bit 9 – Channel Count Failed	Bit 0 – RAM Test	Bit 8 – RS-422 #2	Bit 1 – Channel Count	Bit 9 – RS-423 #1	Bit 2 – RTS-CTS	Bit 10 – RS-423 #2	Bit 3 – DTR-DSR	Bit 11 – RS-485 #1	Bit 4 – Timeouts	Bit 12 – RS-485 #2	Bit 5 – RS-232 #1	Bit 13 – Internal #1	Bit 6 – RS-232 #2	Bit 14 – Internal #2	Bit 7 – RS-422 #1
Bit 0 – Channel 1 Failed	Bit 5 – Channel 6 Failed																										
Bit 1 – Channel 2 Failed	Bit 6 – Channel 7 Failed																										
Bit 2 – Channel 3 Failed	Bit 7 – Channel 8 Failed																										
Bit 3 – Channel 4 Failed	Bit 8 – RAM Test Fail																										
Bit 4 – Channel 5 Failed	Bit 9 – Channel Count Failed																										
Bit 0 – RAM Test	Bit 8 – RS-422 #2																										
Bit 1 – Channel Count	Bit 9 – RS-423 #1																										
Bit 2 – RTS-CTS	Bit 10 – RS-423 #2																										
Bit 3 – DTR-DSR	Bit 11 – RS-485 #1																										
Bit 4 – Timeouts	Bit 12 – RS-485 #2																										
Bit 5 – RS-232 #1	Bit 13 – Internal #1																										
Bit 6 – RS-232 #2	Bit 14 – Internal #2																										
Bit 7 – RS-422 #1																											
Example	Command / Query	Response (<i>Description</i>)																									
	*TST?	0																									
Related Commands	None																										

***WAI**

Purpose	Halts execution of commands and queries until the No Operation Pending message is true	
Type	IEEE 488.2 Common Command	
Command Syntax	*WAI	
Command Parameters	None	
*RST Value	N/A	
Query Syntax	None	
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 the previous commands have been executed before proceeding. It provides a way of synchronizing the module with its commander. Note that the VT6065 never has any pending operations so this command is effectively a “no operation” command.	
Example	Command / Query	Response (Description)
	*WAI	
Related Commands	*OPC	

INSTRUMENT SPECIFIC SCPI COMMANDS

ABORt

Purpose	Stops current block operations	
Type	Device dependent SCPI command	
Command Syntax	ABORt	
Command Parameters	None	
*RST Value	None	
Query Syntax	None – Command Only	
Query Parameters	N/A	
Query Response	N/A	
Description	This command stops the current block operations and all active timers; the buffers and settings are unchanged. This command is an event and has no associated, no query form and no *RST value.	
Example	Command / Query	Response (<i>Description</i>)
	ABOR	
Related Commands	None	

FORMat

Purpose	Set the data format for retrieving received characters
Type	Device dependent SCPI command
Command Syntax	FORMat[:DATA] [<channel>] <type>
Command Parameters	<channel> = If parameter is not supplied, defaults to Channel 1 <type> = ASCii, INTeger, HEXadecimal, OCTal, BINary, PACKed
*RST Value	ASCii
Query Syntax	FORMat?
Query Parameters	<channel> = If parameter is not supplied, defaults to Channel 1
Query Response	ASC INT HEX OCT BIN PACK
Description	<p>The command sets the data format for retrieving received characters. The actual output format is also dependent on the TERM commands which tells when a receive block should end. The TERM commands can cause blocks to be indefinite or definite. See the TERM commands (TERMINATION), in the previous section, for more details. This command does not allow the optional length parameter. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. Valid formats are:</p> <p>ASCii Data is transferred in NR1 format with 1, 2, or 3 significant digits. Multiple numbers are separated by commas. An example output for “ABC” is: 65,66,67</p> <p>INTeger Data is transferred as either a definite or an indefinite block. The length is fixed at 8 bits. An example of the indefinite block for “ABC” is: #0ABC</p> <p style="padding-left: 40px;">An example of a definite block for “ABC” is: #13ABC</p> <p>HEXadecimal Data is encoded as a non-decimal numeric, base 16, preceded by "#H" as specified in IEEE-STD-488.2. The length is fixed at 2 digits. An example output for “ABC” is: #H41,#H42,#H43</p> <p>OCTal Data is encoded as a non-decimal numeric, base 8, preceded by "#Q" as specified in IEEE-STD-488.2. The length is fixed at 3 digits. An example output for “ABC” is: #Q101,#Q102,#Q103</p> <p>BINary Data is encoded as a non-decimal numeric, base 2, preceded by "#B" as specified in IEEE-STD-488.2. An example output for “ABC” is: #B01000001,#B01000010,#B01000011</p> <p>PACKed PACKed is the same as INTeger above.</p> <p>At *RST, the format for all channels is set to ASCii.</p>

Description (Cont.)	The query reports the current data format for retrieving received characters. The response is character data that indicates one of the data formats. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The *RST command does not directly affect query commands.	
Example	Command / Query	Response (Description)
	FORM:ASC FORM 2 INT FORM? FORM? 2	<i>(Sets the format of Channel 1 to ASCII.)</i> <i>(Sets the format on Channel 2 to INTegeR.)</i> BIN <i>(Queries then reports the format setting for Channel 1.)</i> INT <i>(Queries then reports the format setting for Channel 2.)</i>
Related Commands	N/A	

SERialCONTRol:CTS

Purpose	Enable or disable Clear To Send (CTS) handshake mode	
Type	Device dependent SCPI command	
Command Syntax	[SYSTem:][COMMunicate:]SERial[<channel>]:CONTRol:CTS <boolean>	
Command Parameters	<channel> = If parameter is not supplied, defaults to Channel 1 <boolean> = This parameter takes on the values described earlier (typically 0 or 1) See INPUT PARAMETERS in the previous section.	
*RST Value	0	
Query Syntax	[SYSTem:][COMMunicate:]SERial[<channel>]:CONTRol:CTS?	
Query Parameters	<channel> = If parameter is not supplied, defaults to Channel 1	
Query Response	0 or 1	
Description	<p>The command will enable or disable Clear To Send (CTS) handshake mode for TRANSMISSION. The CTS signal is an input to the VT6065. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The parameter <boolean> can take on the values described earlier (typically 0 or 1). This hardware handshake scheme will stop transmission when the CTS signal is de-asserted. When CTS is removed mid-character, transmission will continue to complete the current character. The data line then goes to its idle (marking) state waiting for CTS to be re-asserted. This mode uses no CPU processing for handshaking. It is preferred to DSR on XON handshaking only for that reason.</p> <p>At *RST, this value is set to 0 - CTS handshake is disabled on all channels. While disabled, CTS is ignored and need not be connected.</p> <p>The query reports the CTS handshaking mode. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The response is a numeric value of either 0 or 1. A response value of 0 indicates the specified channel is not using CTS handshake. A return value of 1 indicates the specified channel is using CTS handshake. The *RST command does not directly affect query commands.</p>	
Example	Command / Query	Response (Description)
	SER2:CONT:CTS 1 SER:CONT:CTS?	(Enables CTS handshaking on Channel 2.) 1 (Queries then reports that CTS handshaking is enabled on Channel 2.)
Related Commands	N/A	

SERial:CONTRol:DSR

Purpose	Enable or disable Data Set Ready (DSR) handshake mode	
Type	Device dependent SCPI command	
Command Syntax	[SYSTem:][COMMunicate:]SERial[<channel>]:CONTRol:DSR <boolean>	
Command Parameters	<channel> = If parameter is not supplied, defaults to Channel 1 <boolean> = This parameter takes on the values described earlier (typically 0 or 1). See INPUT PARAMETERS in the previous section.	
*RST Value	0	
Query Syntax	[SYSTem:][COMMunicate:]SERial[<channel>]:CONTRol:DSR?	
Query Parameters	<channel> = If parameter is not supplied, defaults to Channel 1	
Query Response	0 or 1	
Description	<p>The command will enable or disable Data Set Ready (DSR) handshake mode for TRANSMISSION. The DSR signal is a polled input to the VT6065. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The parameter <boolean> can take on the values described earlier (typically 0 or 1). This handshake scheme will stop transmission when the DSR signal is de-asserted. When DSR is removed mid-character, transmission will continue to complete the current character, and may transmit one more character as well. The data line then goes to its idle (marking) state waiting for DSR to be re-asserted. This mode uses CPU processing for polling of the handshake line. After DSR is re-asserted, there is a time delay before transmission resumes. DSR and XON handshaking take CPU processing power to complete. Because of this, CTS is the preferred handshaking mode.</p> <p>At *RST, this value is set to 0 - DSR handshake is disabled on all channels. While disabled, DSR is ignored and need not be connected.</p> <p>The query reports the DSR handshaking mode. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The response is a numeric value of either 0 or 1. A return value of 0 indicates the specified channel is not using DSR handshake. A return value of 1 indicates the specified channel is using DSR handshake. The *RST command does not directly affect query commands.</p>	
Example	Command / Query	Response (Description)
	SER:CONT:DSR 1 SER:CONT:DSR?	(Enables DSR handshaking on Channel 3.) 1 (Queries then reports that DSR handshaking is enabled on Channel 3.)
Related Commands	N/A	

SERial:CONTRol:DTR

Purpose	Set the DTR (Data Terminal Ready) handshake mode
Type	Device dependent SCPI command
Command Syntax	[SYSTem:][COMMunicate:]SERial[<channel>]:CONTRol:DTR <ON OFF STANdard IBFull>
Command Parameters	<channel> = If parameter is not supplied, defaults to Channel 1 <ON OFF STANdard IBFull> = valid DTR modes
*RST Value	OFF
Query Syntax	[SYSTem:][COMMunicate:]SERial[<channel>]:CONTRol:DTR?
Query Parameters	<channel> = If parameter is not supplied, defaults to Channel 1
Query Response	ON OFF STANdard IBFull
Description	<p>The command sets the DTR (Data Terminal Ready) handshake mode. This mode is associated with the receive buffer for the channel. If the option <channel> parameter is not supplied, the channel defaults to Channel 1. Valid DTR modes:</p> <p>ON The ON parameter indicates the DTR output is asserted.</p> <p>OFF The OFF parameter indicates the DTR output is un-asserted.</p> <p>STANdard The STANdard mode indicates the DTR output is to follow the RS-232 standard. DTR is asserted when the receiver is ready to accept data. When the receiver is offline the DTR line will be de-asserted.</p> <p>IBFull This parameter indicates the DTR output is used to when the Input Buffer is Full. It is used to control the receive hardware handshake. DTR is asserted when the receive buffer is ready to receive. The number of characters in the buffer is compared with the STOP and START thresholds. These thresholds are set independently. As the number of characters in the receive buffer increases to the STOP threshold, DTR is de-asserted. DTR remains de-asserted until the number of characters in the buffer decrease to the START threshold. When the START threshold is reached DTR is re-asserted, and the receive buffer is again ready to receive. See the command SER:REC:PACE:THReshold(START and STOP) for details on the thresholds.</p> <p>The *RST command sets the DTR handshake mode to OFF for all channels.</p> <p>The query reports the DTR handshake mode. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The response is character data that indicates the current DTR handshake mode of the selected channel. Valid responses are: ON, OFF, STAN, and IBF. The *RST command does not directly affect query commands.</p>

Example	Command / Query	Response (<i>Description</i>)
	SER4:CONT:DTR IBF	<i>(Sets Channel 4 to use the IBFull mode of handshaking for the DTR line.)</i>
	SER1:CONT:DTR?	OFF (<i>Queries then reports that the current DTR mode of Channel 1 is OFF.</i>)
Related Commands	N/A	

SERial:CONTRol:RTS

Purpose	Set the Request To Send (RTS) handshake mode
Type	Device dependent SCPI command
Command Syntax	[SYStem:][COMMunicate:]SERial[<channel>]:CONTRol:RTS <ON OFF STANdard IBFull RFR>
Command Parameters	<channel> = If parameter is not supplied, defaults to Channel 1 <ON OFF STANdard IBFull RFR> = valid RTS modes
*RST Value	OFF
Query Syntax	[SYStem:][COMMunicate:]SERial[<channel>]:CONTRol:RTS?
Query Parameters	<channel> = If parameter is not supplied, defaults to Channel 1
Query Response	ON OFF STANdard IBFull RFR
Description	<p>The command sets the Request To Send (RTS) handshake mode. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. Valid RTS modes are:</p> <p>ON The ON parameter indicates that RTS output is asserted.</p> <p>OFF The OFF parameter indicates that RTS output is un-asserted.</p> <p>STANdard The STANdard mode indicates RTS is to follow the RS-232 standard. It is used to indicate transmit mode activity. RTS is asserted before any transmission. It continues to be asserted while the characters are sent. RTS is de-asserted after the last character is sent out of the transmitter. RTS could become de-asserted as soon as one bit time after the last stop bit.</p> <p>IBFull This parameter sets the RTS output for use in the receive hardware handshaking mode. RTS is asserted when the receive buffer is ready to receive. The number of characters in the buffer is compared with the STOP and START thresholds. These thresholds are set independently. As the number of characters in the receive buffer increases to the STOP threshold, RTS is de-asserted. RTS remains de-asserted until the number of characters in the buffer decrease to the START threshold. When the START threshold is reached RTS is re-asserted, and the receive buffer is again ready to receive. See the command ser:rec:pac:threshol (START and STOP) for details.</p> <p>RFR RFR stands for Ready For Receive. It is a synonym for IBFull.</p> <p>The *RST command sets the RTS handshake mode to OFF for all channels.</p> <p>The query reports the RTS handshake mode on a specified channel. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The response is character data that indicates the RTS handshake mode. The *RST command does not directly affect query commands.</p>

Example	Command / Query	Response (<i>Description</i>)
	SER:CONT:RTS IBF	<i>(Sets Channel 4 to use the IBFull mode of handshaking for the RTS line.)</i>
	SER:CONT:FTS?	OFF <i>(Queries then reports that RTS handshaking is not enabled on Channel 1.)</i>
Related Commands	N/A	

SERial:BAUD

Purpose	Set the receive baud rate	
Type	Device dependent SCPI command	
Command Syntax	[SYSTem:][COMMunicate:]SERial[<channel>][:RECEive]:BAUD <baud_rate>	
Command Parameters	<channel> = If parameter is not supplied, defaults to Channel 1 <baud_rate> = Valid baud rate	
*RST Value	9600	
Query Syntax	[SYSTem:][COMMunicate:]SERial[<channel>][:RECEive]:BAUD?	
Query Parameters	<channel> = If parameter is not supplied, defaults to Channel 1	
Query Response	Valid baud rate	
Description	<p>The command sets the receive baud rate on a specified channel. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. Valid values for <baud_rate> are 300, 600, 1200, 2400, 4800, 9600, 19200 and 38400.</p> <p>The *RST command causes the receive baud rate on all channels to be set to 9600 baud.</p> <p>The query reports the receive baud rate on a specified channel. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The response is a numeric value that is one of the valid baud rates. The *RST command does not directly affect query commands.</p>	
Example	Command / Query	Response (Description)
	SER4:BAUD 38400 SER3:BAUD?	(Sets the baud rate for Channel 4 to 38400.) 2400 (Queries then reports that the baud rate set on Channel 3 is 2400.)
Related Commands	N/A	

SERial:BITS

Purpose	Set the number of data bits	
Type	Device dependent SCPI command	
Command Syntax	[SYSTem:][COMMunicate:]SERial[<channel>][:RECeive]:BITS <bits>	
Command Parameters	<channel> = If parameter is not supplied, defaults to Channel 1 <bits> = 5, 6, 7 or 8	
*RST Value	8	
Query Syntax	[SYSTem:][COMMunicate:]SERial[<channel>][:RECeive]:BITS?	
Query Parameters	<channel> = If parameter is not supplied, defaults to Channel 1	
Query Response	5, 6, 7 or 8	
Description	<p>The command sets the number of data bits. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. Although this is under receive, it actually sets both transmit and receive number of bits. Valid values for <bits> are 5, 6, 7, and 8.</p> <p>The *RST command causes bits on all channels to be set to 8.</p> <p>The query reports the number of data bits. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The response is a numeric value that is one of the valid number of data bits. The *RST command does not directly affect query commands.</p>	
Example	Command / Query	Response (Description)
	SER1:BITS 8 SER1:BITS?	(Sets the data bits to 8.) 8 (Queries then reports that Channel 1 is 8 bits.)
Related Commands	N/A	

SERial:PACE?

Purpose	Query the receive software handshake mode	
Type	Device dependent SCPI command	
Command Syntax	None – Query Only	
Command Parameters	N/A	
*RST Value	N/A	
Query Syntax	[SYSTem:][COMMunicate:]SERial[<channel>][:RECEive]:PACE?	
Query Parameters	<channel> = If parameter is not supplied, defaults to Channel 1	
Query Response	XON, NONE	
Description	Queries the receive software handshake mode. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The response is character data that indicates the software handshake mode for the indicated channel. Valid responses are XON and NONE. The *RST command does not directly affect query commands.	
Example	Command / Query	Response (Description)
	SER3 : PACE?	NONE (<i>Queries then reports that there is no software handshaking being used on Channel 3.</i>)
Related Commands	N/A	

SERial:PACE:THReshold:STARt

Purpose	Set the number of characters in the receive buffer to enable data handshaking	
Type	Device dependent SCPI command	
Command Syntax	[SYSTem:][COMMunicate:]SERial[<channel>][:RECeive]:PACE:THReshold:STARt <NRf>	
Command Parameters	<channel> = If parameter is not supplied, defaults to Channel 1 <NRf> = 1 to (buffer size – 3)	
*RST Value	STARt threshold set to buffer size -1024	
Query Syntax	[SYSTem:][COMMunicate:]SERial[<channel>][:RECeive]:PACE:THReshold:STARt?	
Query Parameters	<channel> = If parameter is not supplied, defaults to Channel 1	
Query Response	Numeric ASCII value	
Description	<p>The command sets the number of characters in the receive buffer to enable data handshaking. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The valid range for <NRf> is 1 to (buffer size - 3). Buffer size is programmed using TRACe:POINts. This sets the point where handshaking will begin. Normally, the STARt threshold for a channel is set higher than the STOP threshold. If ser:pace:xon has been selected for this channel, it is the point where XOFF is sent. If ser:cont:dtr has been set to IBFull, this is the point where DTR is de-asserted. If ser:cont:rts has been set to IBFull, this is the point where RTS is de-asserted. If an attempt is made to set the start point outside the valid range, an error message is issued and the start point is set to 1/3 of the buffer size.</p> <p>At *RST, all channels have their STARt threshold set to buffer size -1024. Note that the buffer size is dependent on the number of channels and the size of buffer RAM. It can be set with the TRAC:POIN command. See the TRAC:POIN command for buffer size details.</p> <p>The query reports the pace handshaking threshold start point. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The response is a numeric ASCII value. The *RST command does not directly affect query commands.</p>	
Example	Command / Query	Response (Description)
	SER3 : PACE : THR : STAR 999	(Sets the start threshold on Channel 3 to 999. If the Channel 3 receiver ever reaches the point where 999 characters are in the queue, any requested actions (i.e. XOFF, DTR or RTS) will take place.)
	SER3 : PACE : THR : STAR?	999 (Queries then reports that the start threshold on Channel 3 is set to 999.)
Related Commands	N/A	

SERial:PACE:THReshold:STOP

Purpose	Set the number of characters in the receiver buffer to disable data handshaking	
Type	Device dependent SCPI command	
Command Syntax	[SYSTem:][COMMunicate:]SERial[<channel>][RECeive]:PACE:THReshold:STOP<NRf>	
Command Parameters	<channel> = If parameter is not supplied, defaults to Channel 1 <NRf> = 1 to (buffer size – 1)	
*RST Value	STOP threshold set to buffer size -2048	
Query Syntax	[SYSTem:][COMMunicate:]SERial[<channel>][RECeive]:PACE:THReshold:STOP?	
Query Parameters	<channel> = If parameter is not supplied, defaults to Channel 1	
Query Response	Numeric ASCII value	
Description	<p>The command sets the number of characters in the receive buffer to disable data handshaking. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The valid range for <NRf> is 1 to buffer size -1. This sets the point where handshaking will end. Normally, the STOP threshold is set lower than the START threshold for a channel. If ser:pacexon has been selected for this channel, it is the point where XON is sent. If ser:cont:dtr has been set to IBFull, this is the point where DTR is re-asserted. If ser:cont:rts has been set to IBFull, this is the point where RTS is re-asserted. If an attempt is made to set the start point outside the valid range, an error message is issued and the start point is set to 2/3 of the buffer size.</p> <p>At *RST, all channels have their STOP threshold set to buffer size -2048. Note that the buffer size is dependent on the number of channels and the size of buffer RAM. It can be set with the TRAC:POIN command. See the TRAC:POIN command for buffer size details.</p> <p>The query reports the pace handshaking threshold stop point. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The response is a numeric ASCII value. The *RST command does not directly affect query commands.</p>	
Example	Command / Query	Response (Description)
	SER:PACE:THR:STOP 500	(Sets the stop threshold on Channel 3 to 500. If the Channel 3 receive queue ever drops to the point where 500 characters are in the queue, any requested actions (i.e. XOFF, DTR or RTS) will take place.)
	SER3:PACE:THR:STOP?	(Queries then reports that the threshold on Channel 3 is set to 500.)
Related Commands	N/A	

SERial:PACE

Purpose	Set the receive software handshake mode	
Type	Device dependent SCPI command	
Command Syntax	[SYSTem:][COMMunicate:]SERial[<channel>][:RECeive]:PACE <XON NONE>	
Command Parameters	<channel> = If parameter is not supplied, defaults to Channel 1 <XON NONE> = Setting	
*RST Value	NONE	
Query Syntax	None – Command Only	
Query Parameters	N/A	
Query Response	N/A	
Description	<p>This command sets the receive software handshake mode. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. If the software handshake mode is set to NONE, no software handshaking is done. If the software handshake mode is set to XON, the stop and start thresholds are used to determine when to send the XON character (17 decimal) and when to send the XOFF character (19 decimal). As characters are received, the receive buffer fills. XOFF is sent when the receive buffer reaches the stop threshold. As characters are retrieved from the receive buffer, the number of characters in the receive buffer drop. When the number of characters drops below the start threshold, the XON character is sent.</p> <p>*RST, all channels have their software pace mode set to NONE. Also see the commands for setting the thresholds.</p>	
Example	Command / Query	Response (Description)
	SER3 :PACE XON	(Sets the pace mode to XON for Channel 3. XON and XOFF characters will be sent to regulate the quantity of characters in receive buffer 3 (RCH3).)
Related Commands	N/A	

SERial:PARity

Purpose	Set the parity mode	
Type	Device dependent SCPI command	
Command Syntax	[SYSTem:][COMMunicate:]SERial[<channel>][:RECeive]:PARity[:TYPE] <EVEN ODD NONE IGNore ZERO ONE>	
Command Parameters	<channel> = If parameter is not supplied, defaults to Channel 1 <EVEN ODDD NONE IGNore ZERO ONE> = Valid types	
*RST Value	NONE (All channels)	
Query Syntax	[SYSTem:][COMMunicate:]SERial[<channel>][:RECeive]:PARity[:TYPE]?	
Query Parameters	<channel> = If parameter is not supplied, defaults to Channel 1	
Query Response	EVEN ODDD NONE IGNore ZERO ONE	
Description	<p>The command sets the parity mode. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. Although this is under receive, it actually sets both transmit and receive parity. Valid types are:</p> <p>EVEN Even parity is sent with every character. Receive characters are checked for even parity.</p> <p>ODD Odd parity is sent with every character. Receive characters are checked for odd parity.</p> <p>NONE No parity is sent with the characters. No parity is checked on receive characters. Note that if the characters being received do have parity, it is probable this will show up as framing errors.</p> <p>IGNore This retains the previous parity mode for sending characters. Parity errors are ignored on receive (the user never sees the errors).</p> <p>ZERO A parity bit of 0 is sent with every character. Receive characters are checked for a 0 parity bit.</p> <p>ONE A parity bit of 1 is sent with every character. Receive characters are checked for a 1 parity bit.</p> <p>The query reports the parity mode. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The response is character data that indicates the current parity mode of the selected channel. Valid responses are: EVEN, ODD, NONE, IGN, ZERO, and ONE. The *RST command does not directly affect query commands.</p>	
Example	Command / Query	Response (Description)
	SER4:PAR ODD SER4:PAR?	<i>Sets the receive and transmit parity of Channel 4 to odd parity. ODD (Queries and reports the current parity mode of Channel 4 is odd.)</i>
Related Commands	N/A	

SERial:SBITs

Purpose	Set the number of stop bits	
Type	Device dependent SCPI command	
Command Syntax	[SYSTem:][COMMunicate:]SERial[<channel>][:RECeive]:SBITs <bits>	
Command Parameters	<channel> = If parameter is not supplied, defaults to Channel 1 <bits> = 1 or 2	
*RST Value	1	
Query Syntax	[SYSTem:][COMMunicate:]SERial[<channel>][:RECeive]:SBITs?	
Query Parameters	<channel> = If parameter is not supplied, defaults to Channel 1	
Query Response	1 or 2	
Description	<p>The command sets the number of stop bits. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. Although this is under receive, it actually sets both transmit and receive number of stop bits. Valid values for <bits> are 1 and 2.</p> <p>The *RST command causes the number of stop bits to be set to 1 on all channels.</p> <p>Query reports the number of stop bits. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The response is a numeric value that is one of the valid number of stop bits. The *RST command does not directly affect query commands.</p>	
Example	Command / Query	Response (Description)
	SER2:SBIT 2	<i>(Sets the number of stop bits on both transmit and receive.)</i>
	SER2:SBIT?	<i>(Queries and reports the number of stop bits on Channel 2 is 1 bit.)</i>
Related Commands	N/A	

SERial:STANdard

Purpose	Set the electrical interface standard	
Type	Device dependent SCPI command	
Command Syntax	[SYSTem:][COMMunicate:]SERial[<channel>]:STANdard <standard>	
Command Parameters	<channel> = If parameter is not supplied, defaults to Channel 1 <standard>= 232, 422, 423, 485	
*RST Value	232	
Query Syntax	[SYSTem:][COMMunicate:]SERial[<channel>]:STANdard?	
Query Parameters	<channel> = If parameter is not supplied, defaults to Channel 1	
Query Response	232, 422, 423, 485	
Description	<p>The command sets the electrical interface standard to use. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The valid values for <standard> are 232, 422, 423, 485. If <standard> is set to 232 this command configures the indicated channel for the RS-232 specification. If <standard> is set to 422, this command configures the indicated channel for the RS-422 specification. If <standard> is set to 423, this command configures the indicated channel for the RS-423 specification. If <standard> is set to 485, this command configures the indicated channel for the RS-485 specification.</p> <p>At *RST, all channels default to the standards saved in non-volatile location 1 (the factory set values in non-volatile location 1 are all RS-232).</p> <p>Query reports the current configuration standard for the indicated channel. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The response is a numeric ASCII value. Valid responses are 232, 422, 423, or 485. The *RST command does not directly affect query commands.</p>	
Example	Command / Query	Response (Description)
	SER4:STAN 422 SER4:STAN?	(Sets Channel 4 to the RS-422 standard interface.) 422 (Queries then reports Channel 4 is set for the RS-422 standard.)
Related Commands	N/A	

SERial:TRANsmit:AUTO

Purpose	Couple or uncouple the transmit baud rate	
Type	Device dependent SCPI command	
Command Syntax	[SYSTem:][COMMunicate:]SERial[<channel>]:TRANsmit:AUTO <boolean>	
Command Parameters	<channel>= If parameter is not supplied, defaults to Channel 1 <boolean>= 0 or 1	
*RST Value	1	
Query Syntax	[SYSTem:][COMMunicate:]SERial[<channel>]:TRANsmit:AUTO?	
Query Parameters	<channel> = If parameter is not supplied, defaults to Channel 1	
Query Response	0 or 1	
Description	<p>The command couples or uncouples the transmit baud rate to the receive baud rate. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. When <boolean> is ON or 1, the transmit baud rate is set to the same rate as the receive baud rate. If the receive rate is changed, so is the transmit rate. If <boolean> is OFF or 0, the transmit baud rate doesn't change when the receive rate is changed. <u>Note</u> that setting the transmit baud rate automatically turns coupling off for the specified channel. <u>Note</u> that the number of data bits and the channel parity are always the same for both transmit and receive, and are set with [RECeive] commands.</p> <p>At *RST, all channels default to coupled (auto 1).</p> <p>Query reports the coupling of the transmit and receive baud rates for the indicated channel. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The response is a numeric ASCII value. Valid responses are 0 and 1. The *RST command does not directly affect query commands.</p>	
Example	Command / Query	Response (Description)
	SER3:TRAN:AUTO 0	(Uncouples the Channel 2 transmit baud rate from the Channel 2 receive baud rate.)
	SER2:TRAN:AUTO?	0 (Indicates Channel 2 does not have receive and transmit baud rates coupled.)
Related Commands		

SERial:TRANsmit:BAUD

Purpose	Set the transmit baud rate on a specified channel	
Type	Device dependent SCPI command	
Command Syntax	[SYSTem:][COMMunicate:]SERial[<channel>]:TRANsmit:BAUD <baud_rate>	
Command Parameters	<channel> = If parameter is not supplied, defaults to Channel 1 <baud_rate> = 300, 600, 1200, 2400, 4800, 9600, 19200 or 38400	
*RST Value	9600	
Query Syntax	[SYSTem:][COMMunicate:]SERial[<channel>]:TRANsmit:BAUD?	
Query Parameters	<channel> = If parameter is not supplied, defaults to Channel 1	
Query Response	300, 600, 1200, 2400, 4800, 9600, 19200 or 38400	
Description	<p>The command sets the transmit baud rate on a specified channel. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. Valid values for <baud_rate> are 300, 600, 1200, 2400, 4800, 9600, 19200 and 38400. <u>Note</u> that setting a transmit baud rate automatically turns off the auto coupling between transmit and receive baud rates.</p> <p>At *RST, all channels default to 9600 with auto coupling to the receive baud rates.</p> <p>Query reports the transmit baud rate on the indicated channel. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The response is a numeric ASCII value. Valid responses are 300, 600, 1200, 2400, 4800, 9600, 19200, 38400. The *RST command does not directly affect query commands.</p>	
Example	Command / Query	Response (Description)
	SER3:TRAN:BAUD 19200 SER:TRAN:BAUD?	(Sets the transmit baud rate on Channel 3 to 19200 baud.) 3 (Queries and reports Channel 3 transmit baud rate is set to 19200 baud.)
Related Commands	N/A	

SERial:TRANsmit:PACE

Purpose	Set the transmit software handshake mode	
Type	Device dependent SCPI command	
Command Syntax	[SYSTem:][COMMunicate:]SERial[<channel>]:TRANsmit:PACE <XON NONE>	
Command Parameters	<channel> = If parameter is not supplied, defaults to Channel 1 <XON NONE> = Transmit software handshake mode	
*RST Value	NONE	
Query Syntax	[SYSTem:][COMMunicate:]SERial[<channel>]:TRANsmit:PACE?	
Query Parameters	<channel> = If parameter is not supplied, defaults to Channel 1	
Query Response	XON or NONE	
Description	<p>The command sets the transmit software handshake mode. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. Transmit pacing uses XON and XOFF characters received on the receive side of a channel to pace data being transmitted on the transmit side. When pace is set to XON, the receive side monitors the received characters looking for XON and XOFF. When an XOFF is received, the transmitter stops transmitting. After the transmitter has been stopped by XOFF, receiving an XON character will start the transmitter transmitting again.</p> <p>At *RST, all channels default to NONE.</p> <p>Query reports the transmit software handshaking mode. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The responses are XON and NONE. The *RST command does not directly affect query commands.</p>	
Example	Command / Query	Response (Description)
	SER4:TRAN:PACE XON	(Sets the transmit software handshake mode to use XON/XOFF on the receive channel to pace the transmitter.)
	SER4:TRAN:PACE?	(Queries then reports Channel 4 transmit software handshaking is active.)
Related Commands		

TERMinator:CHARacter

Purpose	Set the character for end of line	
Type	Device dependent SCPI command	
Command Syntax	TERMinator:CHARacter [<channel>] <character_number>	
Command Parameters	<channel> = If parameter is not supplied, defaults to Channel 1 <character_number> = Numeric ASCII value from 0 to 255	
*RST Value	1	
Query Syntax	TERMinator:CHARacter [<channel>]?	
Query Parameters	<channel> = If parameter is not supplied, defaults to Channel 1	
Query Response	Numeric ASCII value from 0 to 255	
Description	<p>The command sets the character to use for end of line recognition. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The valid range of <character_number> is 0 to 255. This tells the VT6065 what character to use to recognize the end of a record. Note that setting a termination character automatically sets the termination length to 0.</p> <p>At *RST, all channels default to a termination length of 1 which turns termination character off.</p> <p>Query reports the character to use for end of line recognition. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The responses are: OFF or an ASCII numeric value in the range of 0 to 255. The *RST command does not directly affect query commands.</p>	
Example	Command / Query	Response (Description)
	TERM:CHAR 2 10	(Sets the termination character on Channel 2 to 10, an ASCII line-feed character.)
	TERM:CHAR? 2	OFF (Queries then reports Channel 2 is not using any termination character at this time.)
	TERM:CHAR? 2	10 (This example shows Channel 2 using ASCII line-feed character (10) as the termination character)
Related Commands	N/A	

TERMinator:LENGth

Purpose	Set the character count for end of line	
Type	Device dependent SCPI command	
Command Syntax	TERMinator:LENGth [<channel>] <length>	
Command Parameters	<channel> = If parameter is not supplied, defaults to Channel 1 <length> = 0 to buffer length -3	
*RST Value	1	
Query Syntax	TERMinator:LENGth [<channel>]?	
Query Parameters	<channel> = If parameter is not supplied, defaults to Channel 1	
Query Response	0 to buffer length -3	
Description	<p>The command sets the character count to use for end of line recognition. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The valid range of <length> is 0 to length of buffer -3. The length tells the VT6065 how many characters to accumulate in a receive queue before returning any data. Note that setting a termination length of 0 allows all characters in the receive queue to be returned. Note that setting a termination length automatically turns off the termination character.</p> <p>At *RST, all channels default to a termination length of 1 which turns termination character off.</p> <p>Query reports the character count to use for end of line recognition. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The response is a numeric ASCII value. Valid values are 0 to buffer length -3. The *RST command does not directly affect query commands.</p>	
Example	Command / Query	Response (Description)
	TERM:LENG 3 15	(Sets the termination length on Channel 3 to 15 characters.)
	TERM:LENG? 3	0 (Queries then reports Channel 3 will return all characters in the receive queue.)
	TERM:LENG? 3	15 (This example reports Channel 3 will return characters in groups of 15 characters.)
Related Commands	N/A	

TRACe:DATA

Purpose	Load data into the specified transmit queue	
Type	Device dependent SCPI command	
Command Syntax	TRACe:DATA <trace_name>,(<block> <NRf>{,<NRf>})	
Command Parameters	<trace_name> = RCH#	
*RST Value	N/A	
Query Syntax	TRACe:DATA <trace_name>?	
Query Parameters	<trace_name> = RCH#	
Query Response	Setting dependent	
Description	<p>The command loads data into the specified transmit queue. <trace_name> selects which channel's transmit queue gets loaded. Valid values for <trace_name> on a 4-channel card are TCH1, TCH2, TCH3 and TCH4. Valid values for <trace_name> on an 8-channel card are the same as a 4-channel card plus TCH5, TCH6, TCH7 and TCH8. Data can be loaded either as a block or as a series of numbers. See BLOCK MODE for more details on blocks. If data is loaded as a series of numbers, the numbers are sequentially placed in the queue.</p> <p>The *RST command does not affect the data in the queues.</p> <p>Query retrieves the characters in the specified input queue. <trace_name> selects which channel's receive queue the characters are retrieved from. Valid values for <trace_name> on a 4-channel card are RCH1, RCH2, RCH3 and RCH4. Valid values for <trace_name> on an 8-channel card are the same as a 4-channel card plus RCH5, RCH6, RCH7 and RCH8. The format of the data returned can vary. See the FORM:DATA command for more details. The amount of data returned depends on the term:char and TERM:LENG commands. See TERMINATION CHARACTERS and TERMINATION LENGTH for more details on how the end of data is determined. The *RST command does not directly affect query commands.</p>	
Example	Command / Query	Response (Description)
	TRAC:DATA TCHL1,#13ABC	(Loads the characters ABC into the transmit queue of Channel 1. It uses a block to load the characters.)
	TRAC:DATA? RCHL1	65,66,67 (Queries and reports 3 characters were returned, ABC. The data format was ASC.)
	TRAC:DATA? RCHL1	#13ABC (Queries and reports 3 characters were returned and the data format was INT with a fixed data length of 3.)
Related Commands	N/A	

TRACe:DATA:LENGth?

Purpose	Query the number of characters in a channel queue	
Type	Device dependent SCPI command	
Command Syntax	None – Query Only	
Command Parameters	N/A	
*RST Value	N/A	
Query Syntax	TRACe:DATA:LENGth? <trace_name>	
Query Parameters	<trace_name> = RCH#	
Query Response	Numeric ASCII value	
Description	<p>Query reports the number of characters in one of the channel queues. <trace_name> selects which channel's queue gets checked for length. Valid values for <trace_name> on a 4-channel card are RCH1, RCH2, RCH3, RCH4, TCH1, TCH2, TCH3 and TCH4. Valid values for <trace_name> on an 8-channel card are the same as a 4-channel card plus RCH5, RCH6, RCH7, RCH8, TCH5, TCH6, TCH7 and TCH8. The response is a numeric ASCII value. Valid values are 0 to buffer length.</p> <p>The *RST command does not directly affect query commands.</p>	
Example	Command / Query	Response (<i>Description</i>)
	TRAC:DATA:LENG? RCH2	0 (<i>Queries then reports there are no characters in the Channel 2 receive buffer.</i>)
	TRAC:DATA:LENG? RCH2	(<i>Queries then reports there are 123 characters in the Channel 2 receive buffer.</i>)
Related Commands	N/A	

TRACe:FREE?

Purpose	Query the amount of buffer memory unused in a queue	
Type	Device dependent SCPI command	
Command Syntax	None – Query Only	
Command Parameters	N/A	
*RST Value	N/A	
Query Syntax	TRACe:FREE? <trace_name>	
Query Parameters	<trace_name> = RCH#	
Query Response	Numeric ASCII value	
Description	<p>Queries the amount of buffer memory which is unused in a queue. <trace_name> selects which channel's queue gets checked. Valid values for <trace_name> on a 4-channel card are RCH1, RCH2, RCH3, RCH4, TCH1, TCH2, TCH3 and TCH4. Valid values for <trace_name> on an 8-channel card are the same as a 4-channel card plus RCH5, RCH6, RCH7, RCH8, TCH5, TCH6, TCH7 and TCH8. The response is a numeric ASCII value. Valid values are 0 to the size of the queue. The returned value is the number of bytes not filled. Note that each character takes up two bytes in a queue. The *RST command does not directly affect query commands.</p>	
Example	Command / Query	Response (Description)
	TRAC:FREE? TCH1	0 (Queries then reports the transmit queue for Channel 1 is full; no more characters can be added to the queue.)
	TRAC:FREE? TCH1	8192 (This example indicates there is enough buffer memory available to add 4096 (8192/2=4096) more characters to the transmit queue for Channel 1.)
Related Commands	N/A	

TRACe:POINts

Purpose	Set the size of a transmit or receive queue																					
Type	Device dependent SCPI command																					
Command Syntax	TRACe:POINts <trace_name>,<points>																					
Command Parameters	<trace_name>= RCH#, TCH# <points> = 2 to the size of buffer space																					
*RST Value	Equal allocation																					
Query Syntax	TRACe:POINts? <trace_name>																					
Query Parameters	<trace_name> = RCH#																					
Query Response	Numeric ASCII value																					
Description	<p>The command sets the size of a transmit or receive queue. Valid values for <trace_name> on a 4-channel card are RCH1, RCH2, RCH3, RCH4, TCH1, TCH2, TCH3 and TCH4. Valid values for <trace_name> on an 8-channel card are the same as a 4-channel card plus RCH5, RCH6, RCH7, RCH8, TCH5, TCH6, TCH7 and TCH8. The valid range of <points> is 2 to the size of buffer space. <u>Note</u> that all buffers are re-allocated when a TRAC:POIN command is received. This should not be used when the module is actively transmitting or receiving. <u>Note</u> that the <points> is specified in bytes and each character in a queue requires two bytes. If more memory is requested than is available, an error is reported and latter buffers are shorted. It is recommended that small buffers be allocated first. This will avoid unnecessary error messages by freeing up space before allocating larger buffers.</p> <p>At *RST, all buffer memory is allocated equally to all queues. The actual size is dependent on the number of channels on the board and the amount of buffer space available:</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Models</th> <th>Total Buffer Space</th> <th>Bytes</th> <th>Characters</th> </tr> </thead> <tbody> <tr> <td>VT6065-4</td> <td>128 k</td> <td>16 k</td> <td>8 k</td> </tr> <tr> <td>VT6065-8</td> <td>128 k</td> <td>8 k</td> <td>4 k</td> </tr> <tr> <td>VT6065-4</td> <td>512 k</td> <td>64 k</td> <td>32 k</td> </tr> <tr> <td>VT6065-8</td> <td>512 k</td> <td>32 k</td> <td>16 k</td> </tr> </tbody> </table> <p>Query reports the size of a transmit or receive queue. The response is a numeric ASCII value that indicates the number of bytes allocated to a receive or transmit queue. Note that two bytes are required for every character in a queue. The *RST command does not directly affect query commands.</p>		Models	Total Buffer Space	Bytes	Characters	VT6065-4	128 k	16 k	8 k	VT6065-8	128 k	8 k	4 k	VT6065-4	512 k	64 k	32 k	VT6065-8	512 k	32 k	16 k
Models	Total Buffer Space	Bytes	Characters																			
VT6065-4	128 k	16 k	8 k																			
VT6065-8	128 k	8 k	4 k																			
VT6065-4	512 k	64 k	32 k																			
VT6065-8	512 k	32 k	16 k																			
Example	<table border="1" style="width: 100%;"> <thead> <tr> <th style="text-align: left;">Command / Query</th> <th style="text-align: left;">Response (Description)</th> </tr> </thead> <tbody> <tr> <td>TRAC:POIN RCH1 , 500</td> <td>(Sets the size of the receive queue for Channel 1 to 500 bytes, which is room for 250 characters.)</td> </tr> <tr> <td>TRAC:POIN? RCH1</td> <td>500 (Queries then reports Channel 1 receive queue has 500 bytes allocated to it.)</td> </tr> </tbody> </table>	Command / Query	Response (Description)	TRAC:POIN RCH1 , 500	(Sets the size of the receive queue for Channel 1 to 500 bytes, which is room for 250 characters.)	TRAC:POIN? RCH1	500 (Queries then reports Channel 1 receive queue has 500 bytes allocated to it.)															
Command / Query	Response (Description)																					
TRAC:POIN RCH1 , 500	(Sets the size of the receive queue for Channel 1 to 500 bytes, which is room for 250 characters.)																					
TRAC:POIN? RCH1	500 (Queries then reports Channel 1 receive queue has 500 bytes allocated to it.)																					
Related Commands	N/A																					

TRIGger:AUTO

Purpose	Set the mode of operation of a channel to either character or block mode	
Type	Device dependent SCPI command	
Command Syntax	TRIGger:AUTO [<channel>] [<boolean>]	
Command Parameters	<channel> = If parameter is not supplied, defaults to Channel 1 <boolean> = 0 or 1	
*RST Value	1	
Query Syntax	TRIGger:AUTO? [<channel>]	
Query Parameters	<channel> = If parameter is not supplied, defaults to Channel 1	
Query Response	Numeric ASCII value	
Description	<p>The command sets the mode of operation of a channel to either character mode or block mode. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The valid range for <channel> is 1 to 4 on a 4-channel board and 1 to 8 on an 8-channel board. When set to 1 or ON, the channel operates in the character mode. Each character is sent as soon as possible after being placed in the transmit queue, i.e. it is automatically sent. When set to 0 or OFF, the channel operates in the block mode. Characters are not sent but accumulate in the transmit queue. When the channel is triggered, the characters in the transmit queue are sent. The characters are still retained in the transmit queue and can be sent over and over. The source of the trigger can be either the timer or from a command. Either the *trg or the trig command can trigger the transmission. See the section on BLOCK MODE for more details.</p> <p>The *RST command sets all channels to 1 - character mode.</p> <p>Query reports the character/block mode of operation of a channel. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The valid range for <channel> is 1 to 4 on a 4-channel board and 1 to 8 on an 8-channel board. The response is a numeric ASCII value. Valid responses are 0 and 1. A response of 1 indicates the channel is in character mode. A response of 0 indicates the channel is in block mode. The *RST command does not directly affect query commands.</p>	
Example	Command / Query	Response (Description)
	TRIG:AUTO 2	(Sets Channel 2 to block mode.)
	TRIG:AUTO? 2	0 (Queries and reports Channel 2 is operating in block mode.)
Related Commands	N/A	

TRIGger

Purpose	Trigger all channels for block transmission	
Type	Device dependent SCPI command	
Command Syntax	TRIGger[:IMMediate]	
Command Parameters	None	
*RST Value	N/A	
Query Syntax	None – Command Only	
Query Parameters	N/A	
Query Response	N/A	
Description	<p>Trigger all channels for block transmission. This command triggers all transmit channels set to run in block mode that are not already running with a timed trigger. This is exactly the same as the *TRG command. See the TRIG:AUTO command and the section on BLOCK MODE for more details.</p> <p>The *RST command stops all block mode transmissions but does not directly affect this command.</p>	
Example	Command / Query	Response (Description)
	TRIG	
Related Commands	TRIGger[:IMMediate] <channel>	

TRIGger <channel>

Purpose	Trigger one channel for block transmission	
Type	Device dependent SCPI command	
Command Syntax	TRIGger[:IMMEDIATE] <channel>	
Command Parameters	<channel> = Specify channel #	
*RST Value	N/A	
Query Syntax	None – Command Only	
Query Parameters	N/A	
Query Response	N/A	
Description	<p>Trigger one channel for block transmission. The valid range for <channel> is 1 to 4 on a 4-channel board and 1 to 8 on an 8-channel board. This command triggers one transmit channel set to run in block mode. If the channel is already running an error is generated. See the TRIG:AUTO command and the section on BLOCK MODE for more details.</p> <p>The *RST command stops all block mode transmissions but does not directly affect this command.</p>	
Example	Command / Query	Response (Description)
	TRIG 2	(Starts a Channel 2 block mode transmission.)
Related Commands	TRIGger[:IMMEDIATE]	

TRIGger:SEQuence:SOURce

Purpose	Set the trigger source for a block mode transmit channel	
Type	Device dependent SCPI command	
Command Syntax	TRIGger:SEQuence:SOURce [<channel>] IMMEDIATE TIMER	
Command Parameters	<channel> = If parameter is not supplied, defaults to Channel 1	
*RST Value	IMM	
Query Syntax	TRIGger:SEQuence:SOURce? [<channel>]	
Query Parameters	<channel> = If parameter is not supplied, defaults to Channel 1	
Query Response	IMM or TIM	
Description	<p>The command sets the trigger source for a block mode transmit channel. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The valid range for <channel> is 1 to 4 on a 4-channel board and 1 to 8 on an 8-channel board. This only has an effect on block mode channels. Note that block mode is set by the TRIG:AUTO <channel> 0 command. When IMM is set, the transmit operation is started by a TRIG or *TRG command. When TIM is set, the timer value controls the trigger. See the BLOCK MODE section for more details.</p> <p>The *RST command sets the source for all triggers to immediate.</p> <p>Query reports the trigger source for a block mode channel. If the optional <channel> parameter is not supplied the channel defaults to Channel 1. The valid range for <channel> is 1 to 4 on a 4-channel board and 1 to 8 on an 8-channel board. The returned value is either IMM or TIM. The return value of IMM means the specified channel will be triggered by commands and not by the timer. The return value of TIM means the timer will trigger the specified channel. The *RST command does not directly affect query commands.</p>	
Example	Command / Query	Response (Description)
	TRIG:SEQ:SOUR 2 IMM	(Sets Channel 2 so transmission will be started by a TRIG 2 command.)
	TRIG:SEQ:SOUR 3 TIM	(Sets Channel 3 so transmission will be started by the timer.)
	TRIG:SEQ:SOUR? 2	IMM (Queries then reports Channel 2 will trigger by commands and not by the timer.)
Related Commands	N/A	

TRIGger:SEQuence:TIMer

Purpose	Set the time interval for re-sending data blocks	
Type	Device dependent SCPI command	
Command Syntax	TRIGger:SEQuence:TIMer [<channel>] <time_value>	
Command Parameters	<channel> = If parameter is not supplied, defaults to Channel 1 <time_value> = 0.001 to 2147.483 seconds	
*RST Value	0	
Query Syntax	TRIGger:SEQuence:TIMer? [<channel>]	
Query Parameters	<channel> = If parameter is not supplied, defaults to Channel 1	
Query Response	Numeric ASCII value	
Description	<p>The command sets the time interval to be used for re-sending blocks of data. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The valid range for <channel> is 1 to 4 on a 4-channel board and 1 to 8 on an 8-channel board. The <time_value> parameter sets the time interval between the starts of block transmissions. The valid range of <time_value> is 0.001 to 2147.483 seconds. The value of 0 is also allowed. Setting the value to zero turns the timer off. Turning the timer off is a way of gracefully stopping transmissions because the current transmission will continue but no new transmission will be started. This only has an effect on block mode channels. Note that block mode is set by the TRIG:AUTO <channel> 0 command. When IMM is set, the transmit operation is started by a TRIG or *TRG command. When TIME is set, the timer value controls the trigger. See BLOCK MODE for more details.</p> <p>The *RST command sets all timer values to zero.</p> <p>Query reports the time interval to be used for re-sending blocks of data. If the optional <channel> parameter is not supplied, the channel defaults to Channel 1. The valid range for <channel> is 1 to 4 on a 4-channel board and 1 to 8 on an 8-channel board. The response is a numeric ASCII value (the number of seconds between the start of block transmissions). The *RST command does not directly affect query commands.</p>	
Example	Command / Query	Response (Description)
	TRIG:SEQ:TIM 3 0.1	(Sets the Channel 3 timer so transmission will be repeated every 100 ms.)
	TRIG:SEQ:TIM 3 0	(This example could be used to stop Channel 3 transmissions after the current transmission is finished.)
	TRIG:SEQ:TIM? 3	0.100000 (Queries then reports Channel 3 will re-send blocks every 100 ms.)
Related Commands	N/A	

REQUIRED SCPI COMMANDS

Required SCPI commands are available in addition to the commands specifically for the VT6065. Refer to the SCPI standard *Volume 1: Syntax & Style* for more information.

These commands deal with two 16-bit registers. They are the operation status register and the questionable status register. The VT6065 doesn't modify any questionable status register bits. It is included for SCPI compatibility. The VT6065 doesn't change any of the lower 8 bits of the operation status register. It is also included for SCPI compatibility. The layout of the lower 8 bits of the operation status register is:

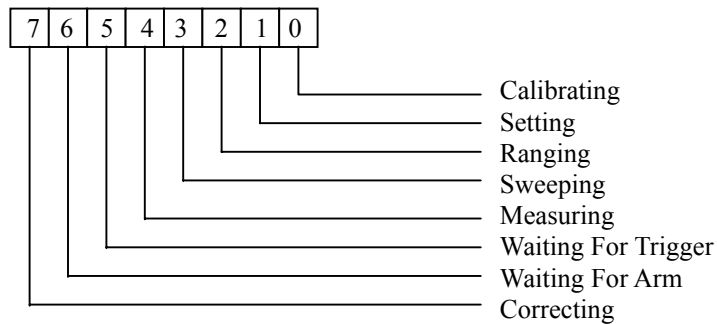


FIGURE 4-7 LOWER 8 BITS, OPERATION STATUS REGISTER

For more information on status data structures, the user is referred to *ANSI/IEEE-STD- 488.2-1987*, Section 11, Device Status Reporting.

STATus:OPERation:CONDition?

Purpose	Queries the Operation Status Condition Register	
Type	Required SCPI command	
Command Syntax	None – Query Only	
Command Parameters	N/A	
*RST Value	N/A	
Query Syntax	STATus:OPERation:CONDition?	
Query Parameters	None	
Query Response	Numeric ASCII value	
Description	Queries the Operation Status Condition Register.	
Example	Command / Query	Response (<i>Description</i>)
	STAT : OPER : COND?	64 (<i>This means the module is currently waiting for an arm to occur.</i>)
Related Commands	None	

STATus:OPERation:ENABLE

Purpose	Sets the Operation Status Enable Register	
Type	Required SCPI command	
Command Syntax	STATus:OPERation:ENABLE <NRf>	
Command Parameters	<NRf> = Numeric ASCII value from 0 to 32767	
*RST Value	0	
Query Syntax	STATus:OPERation:ENABLE?	
Query Parameters	None	
Query Response	Numeric ASCII value from 0 to 32767	
Description	<p>The command sets the Operation Status Enable register. The limits are 0 - 32767. Note that the VT6065 doesn't change any bits. This command is included for SCPI compatibility.</p> <p>The query reports the status of the Operation Status register. The response is a numeric ASCII value.</p>	
Example	Command / Query	Response (Description)
	STAT:OPER:ENAB 0 STAT:OPER:ENAB?	<i>(This disables all operation status register events.)</i> 0 <i>(This confirms that all operation status events are disabled.)</i>
Related Commands	None	

STATus:OPERation?

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	Numeric ASCII value	
Description	Queries the Operation Status Event register.	
Example	Command / Query	Response (<i>Description</i>)
	STAT:OPER?	0 (<i>This means no operation events have occurred.</i>)
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.	
Example	Command / Query	Response (<i>Description</i>)
	STAT:PRES	
Related Commands	None	

STATus:QUEStionable:CONDition?

Purpose	Queries the Questionable Status Condition Register	
Type	Required SCPI command	
Command Syntax	None – Query Only	
Command Parameters	N/A	
*RST Value	N/A	
Query Syntax	STATus:QUEStionable:CONDition?	
Query Parameters	None	
Query Response	Numeric ASCII value	
Description	Queries the Questionable Status Condition Register.	
Example	Command / Query	Response (<i>Description</i>)
	STAT:QUES:COND?	0
Related Commands	None	

STATus:QUEStionable:ENABle

Purpose	Sets the Questionable Status Enable Register	
Type	Required SCPI command	
Command Syntax	STATus:QUEStionable:ENABle <NRf>	
Command Parameters	<NRf> = Numeric ASCII value from 0 to 32767	
*RST Value	NRf must be supplied	
Query Syntax	STATus:QUEStionable:ENABle?	
Query Parameters	None	
Query Response	Numeric ASCII value	
Description	<p>The Status Questionable Enable command sets the bits in the Questionable Status Enable Register. The limits are 0 - 32767. Note that the VT6065 does not modify any bits in the questionable status register. This command is for SCPI compatibility.</p> <p>The Status Questionable Enable query reports the contents of the Questionable Status Enable Register. The VT6065 does not alter the bit settings of this register and will report the last programmed value.</p>	
Example	Command / Query	Response (<i>Description</i>)
	STAT:QUES:ENAB 64 STAT:QUES:ENAB?	64
Related Commands	None	

STATus:QUEStionable?

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	ASCII numeric value	
Description	Queries the Questionable Status Event Register.	
Example	Command / Query	Response (<i>Description</i>)
	STAT:QUES?	0
Related Commands	None	

SYSTem:ERRor?

Purpose	Queries the Error Queue	
Type	Required SCPI command	
Command Syntax	None – Query Only	
Command Parameters	N/A	
*RST Value	N/A	
Query Syntax	SYSTem:ERRor?	
Query Parameters	None	
Query Response	ASCII string	
Description	<p>The System Error query is used to retrieve error messages from the error queue. The error queue will maintain up to 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 <i>Standard Commands for Programmable Instruments (SCPI) Standard Volume 2: Command Reference</i> for details on errors and reporting them. Refer to the ERROR MESSAGES section of this manual for specific details regarding the reported errors.</p>	
Example	Command / Query	Response (Description)
	SYST:ERR?	0 (Indicates no errors have occurred since the last time the error queue was cleared.)
	SYST:ERR?	-350, "Queue overflow" (Indicates more errors have occurred than the error queue could hold.)
Related Commands	None	

SYSTem:VERSion?

Purpose	Queries the SCPI version number	
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 the version of the SCPI standard with which the VT6065 complies.	
Example	Command / Query	Response (<i>Description</i>)
	SYST:VERS?	1992.0
Related Commands	None	

SECTION 5

THEORY OF OPERATION

INTRODUCTION

The VT6065 VXIbus Serial Interface Module is available as a 4-Channel or 8-Channel unit depending upon whether the second four channels are installed on the main logic board of the module. The Serial Interface Module consists of a main logic board and a VXIbus Message-Based Interface daughter card. The main logic board contains:

- Serial UARTs (Universal Asynchronous Receiver/Transmitter)
- RS-232, RS-422, RS-423 and RS-485 drivers and receivers
- Self-test loop back relays and buffer RAMs

The VXIbus Message-Based Interface daughter card contains:

- VXIbus Registers
- 68000 CPU
- EPROM containing the module's program
- EEPROM for storing configuration data
- System RAM
- System timer

VXIBUS MESSAGE-BASED INTERFACE

GENERAL

The VXIbus Message-Based Interface implements a complete interface per the VXIbus Specification Revisions 1.3 and 1.4. The daughter card contains all the VXIbus registers implemented in dual-ported memory, one 68000 CPU running at 8 MHz for the 4-Channel Module or at 12.5 MHz for the 8-Channel Module. The interface daughter card also includes 128 k bytes of program memory, 16 k bytes of EEPROM, 64 k bytes of system RAM, one MC68230 Timer/Counter and a Local Bus Interface (which is not used in the VT6065).

MICROPROCESSOR CLOCK

The microprocessor clock is derived from two possible sources depending upon whether the interface is running at 8 MHz or at 12.5 MHz. If the interface is running at 8 MHz, the VXIbus system clock is buffered by U33 and routed to the CPU via jumper JP5. If the interface is running at 12.5 MHz, a 25 MHz TTL output oscillator is installed at U13, and no jumper is installed at JP5. Both clock sources are divided by two by U7 providing a clean 50% duty cycle clock for the CPU.

MICROPROCESSOR MEMORY

The Message-Based Interface uses a single 64 k by 16 EPROM (U1) for its program memory, two 8 k by 8 EEPROMs (U5 and U6) for non-volatile storage of configuration data, and two 32 k by 8 static RAMs for system use (U2 and U3). Decoding for the memories, along with the other peripherals on the daughter card, is implemented by PAL (programmable array logic) U12. U12 also handles the logic required for the DTACK and VPA functions. U4 provides the necessary logic to control the write functions for the RAMs and EEPROMs allowing byte write operations to occur.

VXIBUS INTERFACE

The VXIbus is connected to the interface daughter card via P101 and P102. The data bus is buffered by U37 and U38 which are octal bi-directional buffer/latches configured to buffer the data coming into the card and latch the data going out, depending upon the type of bus cycle (read or write). The address modifier lines are decoded by U34 and U35, and provide an enable to U26 when an A16 access is occurring on the VXIbus in the upper quadrant (#HC000 through #HFFFF). The specific board address is determined by the setting of SW1, and is compared by U26 to produce the signal VREGENA! indicating this specific interface is being accessed.

Dynamic Configuration of the interface's address is provided by U41 and U31. The configured offset address is written into U41 and, if it is output enabled, it will over-drive the SW1 setting. Dynamic Configuration is enabled by setting SW1 to all ones (#HFF). This is seen by U42, and the FF! signal enables the offset to be written into U41 via U39 and U40.

In order for the CPU to have knowledge of its address, U27 provides a method of reading the address setting.

The principal state machine, which controls VXIbus transfers, is contained in PAL U36. It provides timing for the data buffers U37 and U38 as well as write and enable signals to the dual ported RAMs, U28 and U29, which implement the VXI registers.

When a VXIbus register is written to, U36 provides the enables (REGCE0! and REGCE1!) and the write strobe (VWRITE!) to U28 and U29. U37 and U38 are also enabled to pass the data through to the dual-port RAMs (signal IEVDB!). When a VXIbus register is read, U36 enables U28 and U29, and latches their output into U37 and U38 with the DCLK signal. U37 and U38 are output enabled to the VXIbus by the OEVDB0 and OEVDB1 signals which are controlled by the VDS0! and VDS1! signals respectively.

When the dual-port RAMs are accessed, the address of the register (which was accessed) is latched by U30 and U31. U11 prevents further accesses to the interface until the CPU can read the register which has changed. This process produces an interrupt on IRQ6! which causes the event to be serviced and the interface to be re-enabled.

MICROPROCESSOR INTERRUPTS

The Message-Based Interface MC68000 is configured for auto-vectored interrupts. This allows for seven unique interrupt levels. The interrupts are encoded by U14. The highest level interrupt is NMI! and is not used. IRQ6 is used by the VXIbus interface. IRQ5! is used by the VXIbus trigger inputs. IRQ4! is available to the user circuitry and is brought to P100. IRQ3! is available to the local bus. IRQ2! is also available to the user circuitry, and is also brought to P100 and the lowest priority interrupt. IRQ1! is used by the MC68230 timer/counter.

TIMER/COUNTER

The Message-Based Interface has an MC68230 Timer/Counter U19 which provides a timing function and some parallel I/O. It is used to provide the real time clock function of the interface and control of the VXIbus TTL trigger lines. The clock source for the MC68230 is the same as the microprocessor clock divided by 2 by U7.

The parallel I/O provides the source for the Dynamic Configuration enable signal DCENA!, the trigger source from the timer function, and the TTL trigger bus.

Signals from port PA0 through PA7 set the trigger output line, the trigger input line and the trigger input and output enables. These control lines are decoded by U16 for the output function and U15 for the input function. U21 buffers the trigger outputs and provides the required open collector drive for these lines. The selected trigger input may cause an interrupt to the CPU as well as being made available to the user connector P100.

Signal TROUT! enables the TTL trigger output driver; SYSFAIL causes the VSYFAIL! to go true; FAILED causes the SYSFAIL LED to light.

The test point EI provides the CPU with the running speed. If a jumper is installed between pins 3 and 4, the interface is expected to run at 8 MHz. If it is left out, it is expected to run at 12.5 MHz. This affects the timer function. In addition, EI pins, 1 and 2, are used to clear out the non-volatile memory. If pins 1 and 2 are shorted at power-up, the default values are written into the EEPROMs, and provide a method to restore the interface to the original configuration.

VXIBUS INTERRUPTS

Signals from the U19 ports PB0, PB1 and PB2 set the VXIbus interrupt level, and PB4 reads to see if an interrupt is pending. The setting of IRQSEL0 through 3 determines which interrupt level to drive, and which acknowledge response. U17 decodes the three lines into the seven IRQ lines on the back plane which are buffered by the open collector driver U22. The IRQSEL0 through 3 also sets the compare value for U18, U8 and U24 to respond to during the interrupt acknowledge cycle. This cycle drives the VIACKOUT! line when a VIACKIN! signal and matching address are found. U11 and U20 provide synchronization of VXIbus interrupts so none are lost due to the asynchronous nature of an interrupt event.

The VXIbus interrupt is started by clocking the SETVXIRQ! signal and may be cleared, if necessary, by clocking the CLRVXIRQ! signal.

SYSFAIL AND ACCESS INDICATORS

The Message-Based Interface provides an access LED to indicate the interface is being communicated with. When U36 runs through an access cycle, it sends a signal ALDTACK! to U25 which stretches the length of the pulse to about 100 ms in length. U25 then drives Q1 which in turn lights the LED.

The SYSFAIL LED is driven by Q2 which is driven by U19 under program control. It indicates the interface has not passed its self-test successfully.

LOCAL BUS INTERFACE

The VT6065 does not use the local bus interface feature of the Message-Based Interface.

MAIN LOGIC BOARD

GENERAL

The serial interface is implemented using dual UART. Each dual UART implements two channels of serial interface. The 4-channel module has two of these integrated circuits while the 8-channel module has four. To describe the operation of the VT6065 Serial Interface Module, refer to Channels 1 and 2 only. Note that this information applies equally to the other channels.

MICROPROCESSOR BUS

The microprocessor bus is brought to the main logic board via P7. U50, U53 and U56 provide address bus buffering on the main logic board to reduce loading on the 68000 CPU. U59 and U62 provide data bus buffering also to reduce CPU loading. U65 is a PAL (programmable array logic) device which implements the necessary control logic for the main logic board to transfer data to and from the VXIbus Message-Based Interface daughter card. It provides control signals to the data buffers, enabling them and setting their data direction. It provides a decoded enable signal for the devices on the main logic board (signal named PORTENA* (Note that any signal name which ends in "*" is a low true signal). It provides enables for the buffer RAMs (named RAM0ENA* and RAM1ENA*) and a DTACK signal to the 68000 CPU (named XDTACK*).

The PORTENA* signal is further decoded by U68 and U69 which provide the ultimate decoded enables for the eight output ports (signals named PORT0* through PORT7*), four dual UARTs (signals named DUART0* through DUART3*), an unused read back port (named READPORT*), and an unused enable named FPGACE*.

The output ports are implemented using 74HCT273 octal latches with resets. The latches are cleared during power-up reset (by the signal named RESET*), and can be written to thereafter by the CPU. Six of the output ports (U51, U54, U57, U60, U63, U66) are used to set the self-test configuration relays. They drive the ULN-2803A relay driver ICs (U52, U55, U58, U61, U64 and U67 respectively). These relay drivers in turn control the relays (K1 through K48) on the main logic board. Note that the four channel VT6065 only requires K1 through K24 and their respective drivers and latches.

SERIAL UARTS

The dual UARTs (U2, U3, U4 and U5) are enabled with the DUART0* through DUART3* signals, and use the buffered address and data bus to communicate with the 68000 CPU. The dual UART requires separate read and write control signals (named READ* and WRITE*) which are generated by PAL U78. The timing for these signals is derived from the R/W*, AS* and PORTENA* signals. R26 and C65 provide termination for the WRITE* signal, while R27 and C66 provide termination for the READ* signal.

The dual UARTs require an external source to set their baud rate. U1 provides a 3.6864 MHz clock (named BAUDCLK) to the UARTs. C1 and R4 provide a termination for this clock signal to prevent excessive ringing.

Each dual UART implements all the necessary logic to provide two bi-directional serial communications channels. They provide two transmit outputs, two receive inputs, two RTS (Request To Send) outputs, two CTS (Clear To Send) inputs, five general purpose inputs and six general purpose outputs.

Two of the general purpose inputs (IP2 and IP3) are used to implement the DSR (Data Set Ready) signals, and two of the general purpose outputs are used to implement the DTR (Data Terminal Ready) signals. The remaining general purpose inputs and outputs are wired to pads for ease of customization. The unused inputs are pulled up to a high logic level to insure consistent data read back.

The serial interface signals are routed through P3 to allow the signals to be intercepted for customized serial interface hardware (such as electrically isolated interfaces or fiber optic interfaces). +5 and ± 12 volts are also provided on P3 to support such interfaces.

In the standard VT6065 Serial Interface Module, the signals are jumpered across P3 (pin 1 to pin 2, pin 3 to pin 4, etc.), and are then routed to the on-board drivers and receivers.

LINE DRIVERS AND RECEIVERS

The VT6065 Serial Interface Module is capable of transmitting and receiving in either RS-232, RS422, RS-423 or RS-485 compatible modes. The configuration is selectable through software control, and does not require the module be opened.

The VT6065 uses a UA9636A for its transmit and handshake buffers in the RS-232 and RS-423 modes. It takes a TTL input signal and produces an output which swings from +6 volts to -6 volts, which is consistent with the requirements of these two standards. When RS-422 or RS-485 is required, the VT6065 uses a 26LS31C line driver for its transmit buffers. Note that in the RS-422 mode, the handshake lines are RS-423 compatible and in the RS-485 mode, handshake lines are not used.

For its receivers, the VT6065 uses the SN75189 to support the RS-232 mode, and uses the 26LS32A for the RS-422, RS-423 and RS-485 modes. The 26LS32A input is wired differentially for the RS-422 and RS-485 modes. It is wired for single-ended input in the RS-423 mode by connecting the positive input to ground at the front panel connector and driving the negative input.

On Channel 1, the UART's transmit output drives both U6 and U8. The RTS and DSR signals are also routed to U13. If RS-232 or RS-423 is selected, K1 is disabled routing U6's transmit output to the front panel connector J100. K5 and K6 are then enabled to disconnect the output of two U8 from J100. Additionally, K3 and K10 are disabled connecting U13's RTS and DSR outputs to J100. If RS-422 or RS-485 is selected, K1 is enabled, and K5 and K6 are disabled routing the output of U8 to J100.

The primary electrical difference between RS-422 and RS-485 is that the output drive should only be enabled when it is commanded to transmit data. This is achieved by using the UARTs' RTS output to control the transmit driver's enable in the RS-485 mode. This is handled by the PAL U16. When the RS-485 mode is selected, RTS enables U8's output. If RS-422 is selected, U8 is always enabled.

The Serial Interface Module's inputs (receive, CTS and DCD) are routed to either the SN75189 line receivers (U7 and U11) for RS-232 mode), or to the 26LS32A line receivers (U9 and U14) for the RS-422, RS-423 or RS-485 modes. In the RS-232 mode, K1, K3 and K9 are disabled, and K5, K6 and K9 are enabled routing the input signals to U7 and U1. In the other modes, K1, K3 and K9 are enabled while K5, K6 and K9 are disabled.

The outputs of the line receivers are selected depending on the mode selected by PAL U12. If RS-232 is selected, the outputs of U7 and U11 are selected. While in the other modes, the outputs of U9 and U14 are selected. The signal REC422A0 controls the input source and is controlled by U72.

SELF-TEST LOOP BACK

The VT6065 provides for driver level loop back self-test. This ensures the drivers and receivers are fully functional.

When K1 is energized, it routes the RS-232 compatible output of U6 to the input of U7 instead of to J100. During self-test, data is transmitted in this mode and received by the same UART. The data is compared for integrity. Similarly, energizing K3 wraps RTS back to U7, K10 wraps DSR back to U14. In the RS-422 mode, K5 and K6 wrap transmit back to U9. The RS-423 and RS-485 modes are confirmed by checking both the RS-232 and RS-422 modes as the required drivers and receivers have been checked during those tests.

The self-test loop back procedure is run in both the RS-232 and RS-422 modes for each channel to confirm the integrity of the Serial Interface Module.

BUFFER RAMS

The main logic board also contains the buffer RAMs which hold the transmit and received data patterns. These RAMs (U73, U74, U75 and U76) may be either 32 k by 8 or 128 k by 8 static RAMs providing a total of either 128 k bytes or 512 k bytes of total buffer memory.

U78 provides the control signals to output enable the RAMs (signals named LRAMOE* and URAMOE*), and provide the write strobes for the RAMs (signals named LRAMWRITE* and URAMWRITE*). These signals are derived from signals AS*, R/W*, LDS* and UDS.

INDEX

*		IEEE-STD-488.2 Common Commands	21, 39
*CLS	43	IGNore	72
*ESE	44	INTeger	58
*ESR?	45		
*IDN?	46	L	
*OPC	47	line-feed character	23
*RCL	48	logical address	15, 16
*RST	42, 49		
*SAV	50	M	
*SRE	51	message-based	103
*STB?	52	message-based interface	101, 102
*TRG	53		
*TST?	54	N	
*WAI	56	NONE	72
		NRf parameter	21
A		O	
ABORt	57	OCTal	58
ASCii	58	ODD	72
		OFF	62, 64
B		ON	62, 64
backplane	16	ONE	72
backplane jumpers	15	OPC	40
BINary	58		
block mode	23	P	
block parameter	21	PACKed	24, 58
boolean parameter	21	power	15
byte	19, 22	Q	
		Query	20
C		R	
channel parameter	23	receivers	104
character mode	23	request false	17
CLS	40	request true	17
cooling	15	required SCPI commands	39
CTS (clear to send)	104	resource manager	16, 17
		RFR	64
D		RS-232 configuration	25
Device Dependent SCPI Commands	39	RS-422 configuration	25
DSR (data set ready)	104	RS-423 configuration	26
dynamic configuration	16, 101	RS-485 configuration	27
		RST	40, 41
E		S	
ESE	40	SCPI Commands	19
ESR?	40	SCPI Tree	19, 21
EVEN	72	SERial:BAUD	66
		SERial:BITS	67
F		SERial:CONTRol:CTS	60
FAIL LED	17	SERial:CONTRol:DSR	61
FORMat	58	SERial:CONTRol:DTR	62
		SERial:CONTRol:RTS	64
H		SERial:PACE	71
HEXadecimal	58	SERial:PACE:THReshold:STARt	69
		SERial:PACE:THReshold:STOP	70
I			
IBFull	62, 64		
IDN?	40		

SERial:PACE?	68
SERial:PARity	72
SERial:SBITs	73
SERial:STANdard	74
SERial:TRANsmit:AUTO	75
SERial:TRANsmit:BAUD	76
SERial:TRANsmit:PACE	77
Service Request Enable	17
Slot 0	17
SRE	40
STANdard	62, 64
static configuration	16, 17
STATus:OPERation:CONDition?	42, 90
STATus:OPERation:ENABle	42, 91
STATus:OPERation?	92
STATus:OPERation[:EVENT]?	42
STATus:PRESet	42, 93
STATus:QUEStionable:	42
STATus:QUEStionable:CONDition?	94
STATus:QUEStionable:ENABle	42, 95
STATus:QUEStionable?	96
STATus:QUEStionable[:EVENT]?	42
STB?	40
SYSFAIL LED	102
SYSTem:ERRor?	42, 97
SYSTem:VERsion?	42
SYSTem:VERsion?	98

T

TERM:LENG command	24
termination character	24
termination length	24
TERMinator:CHARacter	78
TERMinator:LENGth	79
time_value parameter	23
TRACe:DATA	80
TRACe:DATA:LENGth?	81
TRACe:FREE?	82
TRACe:POINts	83
Transmit and Handshake Buffers	104
TRG	40
TRIGger	85, 86
TRIGger:AUTO	84
TRIGger:SEQuence:SOURce	87
TRIGger:SEQuence:TIMer	88
TST	40

U

UART	103, 104
------	----------

V

VXI chassis	17
VXIbus	15, 19, 22, 99, 100, 101
VXIbus chassis	17

W

WAI	40
-----	----

Z

ZERO	72
------	----