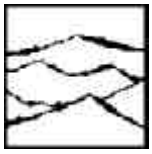


WAVECREST Corporation

DTS-2079
DTS-2077
DTS-2075

GPIB Programming Guide

207910-01 REV B

WAVECREST Corporation continually engages in research related to product improvement. New material, production methods, and design refinements are introduced into existing products without notice as a routine expression of that philosophy. For this reason, any current *WAVECREST* product may differ in some respect from its published description but will always equal or exceed the original design specifications unless otherwise stated.

WAVECREST Corporation
A Technologies Company
7626 Golden Triangle Drive
Eden Prairie, Minnesota 55344
(952) 831-0030
(800) 733-7128
www.wavecrest.com

U.S. Patent Nos. 4,908,784 and 6,185,509 and 6,194,925; other United States and foreign patents pending

GPIB INTERFACE GUIDE

Table of Contents

Section 5 System Commands

Section 6 Acquire Commands

Acquire Macros

Measure Commands (continued)

Figures and Tables

SECTION 1 - GPIB INTERFACE

1-1 INTRODUCTION TO REMOTE PROGRAMMING OF THE DTS

You can program the DTS to:

- Set up the DTS and start a measurement.
- Return the setup parameters and measurements to the GPIB controller.

Other tasks are accomplished by combining the basic functions.

It is assumed that you are familiar with the usage of the GPIB. If you are not, please consult your GPIB documentation. In particular, you should be familiar with the concepts of selecting an interface, device addressing, interface initialization as well as the command structure and format for programming an instrument over the GPIB.

1-2 DTS SYNTAX

The mnemonic representing the operation to be performed by the DTS is known as the "command header." There are different types of command headers that are discussed in more detail in the following paragraphs. Commands may be simple or compound. The simple command headers consist of a single mnemonic, while a compound command header contains two or more program mnemonics. The first mnemonic of a compound header selects a subsystem and the last mnemonic selects the desired function within the subsystem. Mnemonics, within a compound message, are separated by colons.

- To execute a simple command, the syntax is:
  <mnemonic><terminator>

  Example: `":RUN"`

- To execute a simple command with data:
  <mnemonic><separator><data><terminator>

  Example: `"*SAV 1"`

- To execute a single function in a subsystem (a compound command):
  <Subsystem>:<function><separator><data><terminator>

  Example: `"SYSTem:CHANnel 1"`

In addition to the simple and compound command headers, there are also common command headers to control generic functions in the DTS. An example of a common command function is "reset". The syntax for common command headers is:

***<command header><terminator>**

Example: `"*RST"`

Note that no space or other separator is allowed between the asterisk and the command header.

If a command header is immediately followed by a question mark, then the command is a query. After a query is received, the DTS responds by placing a response in the GPIB output queue. The response will stay in the queue until either the controller reads the response or another command is issued by the controller.

The program commands from the controller are case insensitive: either lower or uppercase letters may be used. The DTS will always respond using upper case. Either the long form (the complete spelling of a command) or the short form (abbreviated spelling) may be used.

The terminator for a message can be a NL (new line, ASCII 10) character, asserting the GPIB EOI (End-Or-Identify) signal or a combination of both. All three ways are equivalent.

It is possible to send multiple commands and queries to different subsystems at the same command by separating each command with a semicolon. Multiple commands may be any combination of compound and simple commands.

1-3 IEEE-488.2 BUS COMMANDS

IEEE-488.2 defines the action of the DTS for certain bus commands. A device clear (DCL) or selected device clear (SDR) command clears both the input and output buffers. The parser is reset, and any pending commands are cleared.

The group execute trigger (GET) command causes the same action as the RUN/GO command. The interface clear (IFC) command halts any bus activity. Control is returned to the system controller, and any command in progress is terminated.

1-4 IEEE-488.2 PROTOCOL

The IEEE-488.2 standard defines the overall scheme for communication with the DTS. Please consult the standard for further clarification of the protocol.

The communications subsystem of the DTS consists of an input buffer and an output buffer. The input buffer is a memory area where commands and queries from the controller are stored and processed. The input buffer holds 274 characters or bytes of data.

The output buffer is a memory area where data for the controller is stored until read. The output area is large enough to hold 510 characters or bytes of data. Larger blocks of data are handled by breaking the data into a series of blocks smaller than 510 bytes in size.

The DTS's command parser interprets commands from the controller, and determines what action to take in response.

After power up, or after receiving a device clear command, both the input and output buffers are cleared and the parser is reset. The controller and the DTS communicate by exchanging program and response messages. The controller should always terminate a program message before reading a response from the DTS.

If the controller sends a query message to the DTS, the next message from the controller should be a response message. The controller should read the entire response from the DTS before sending another query message.

Execution of commands by the DTS is in the order that the commands are received. This also includes reception of the group execute trigger (GET) bus command. The controller should not send a group execute trigger command in the middle of a program message.

It is possible to send multiple queries in a query message ("compound query") by use of semicolon message separators. The DTS responses to a multiple query will also be separated by semicolons.

1-4.1 PROTOCOL EXCEPTIONS

If the DTS is addressed to talk before the controller sent it a query, it will indicate a query error and not transmit any data bytes over the GPIB. If the DTS has no response because it was unable to execute the query because of an error, the DTS will not indicate a query error, and waits for the next message from the controller.

If a command error occurs, it is reported to the controller. An example of a command error would be a syntax error or an unrecognized command. A group execute trigger in the middle of a program message is also considered a command error.

If a parameter is out of range, or the current settings of the DTS do not allow execution of a requested command or query then an execution error is reported to the controller.

A device-specific error will be reported by the DTS if it is unable to execute a command for a strictly DTS dependent reason.

A query error will be reported if the proper protocol for a query is not followed. Query error include both "unterminated" and "interrupted" conditions.

If the controller attempts to read a response message before the program message has been terminated (an "unterminated" condition), the DTS reports a query error. The parser is reset, and any response is cleared from the output buffer, without being sent back to the controller.

If the controller fails to read the entire response message, and attempts to send another program message, the DTS responds with a query error. The unread portion of the response is discarded by the DTS. The program message from the controller is not affected, and will be processed normally by the DTS.

It is possible for the DTS to become deadlocked in a condition where both the input and output buffers are full. This can occur if the controller sends a very long program message which contains queries that generate a large number of data bytes in response. The DTS is unable to accept any more program message bytes under this condition, but the controller cannot read any of the response data bytes until the entire program message has been sent to the DTS. If this situation occurs, the DTS detects the condition, clears the output queue, and discards responses until it reaches the end of the program message. A query error bit is also set under this condition.

SECTION 2 – COMMON COMMANDS

2-1 SUMMARY OF DTS COMMANDS

In addition to the **Common** commands defined for all instruments by IEEE-488.2, the commands for the DTS are organized into 8 different subsystem sets. They are:

System - Controls some basic function of the DTS.

Acquire - Provides access to the parameters for acquiring and storing data.

Calibrate - Provides the selection of different calibrate functions and retrieves data generated by these functions.

Channel - Provides access to the parameters associated with the different channels.

Display - Provides access to the parameters for controlling how or what information will be written to screen.

Measure - Selects the measurements to be made.

Trigger - Controls the trigger modes and parameters for each trigger mode.

Hardcopy - Controls printing of instrument configurations and measurement statistics.

2-2 IEEE-488.1 BUS COMMANDS (HARDWARE)

The following commands are IEEE-488.1 bus commands (hardware line ATN true).

Device Clear - The device clear (DCL) command causes the DTS to perform a warm boot.

Group Execute - Will cause the same action as the trigger (GET), **RUN** and ***TRG** commands. The DTS will acquire data.

Clear Interface (IFC) - Halts all bus activity.

2-3 COMMON COMMANDS

The following are common commands defined by IEEE-488.2 and supported by the DTS.

***CLS**..... Clear Status.

***ESE** ..... Event Status Enable.

***ESE?** ..... Query.

***ESR?** ..... Event Status Register Query.

***IDN?** ..... Identification Query.

***OPC** ..... Operation Complete.

***OPC?**..... Query.

***RCL** ..... <0-10> .... Recall.

***RST**..... Reset. Resets the input and output buffers, resets the parser, and clears any pending commands.

***SAV** ..... <0-10> .... Save.

***SRE** ..... Service Request Enable.

***SRE?** ..... Query.

***STB?** ..... Status Byte Query.

***TRG** ..... Causes the DTS to initiate a measurement.

***TST?** ..... Test Instrument Query.

2-4 ROOT COMMANDS

:RUN Causes the DTS to initiate measurement. Does the same function as the *TRG.

:TER? The TER query will read the identified TRG Event Register. When the register is read, it is cleared. A one (1) informs the program that the trigger has occurred. Monitor this bit to know when a take sample (burst), pulse find, cable measure or an internal/external calibration is complete.

:LER? The LER query will read the Local Event Register. When the query is received and the register is read, it is cleared. A non-zero indicates that a reset is in progress.

:SDS? The SDS query reads the Special Device Register. When the query is received and the register is read, it is cleared. This register is used to indicate when some commands are complete when they don't set a TRG or MAV bit. Same as bit 3 of a serial poll.

The following is a listing of the commands, organized by the subsystems listed earlier in this document, used to support the DTS.

2-5 SYSTEM COMMANDS

:SYSTem:**ARM**ing</trigger source/trigger sequence/start arm/stop arm ..... Arming macro for speed.
/arm 1 ref/arm 2 ref/arm 1 slope/arm 2 slope/start count/stop count>

:SYSTem:**CHAN**nel<1|2|BOTH> ..... Set channel.
:SYSTem:**CHAN**nel**?** ..... Read channel.

:SYSTem:**DCCHAN**nel<1|2> ..... Select dc measurement channel.
:SYSTem:**DCCHAN**nel**?** ..... Read dc selected channel.

:SYSTem:**ELAP**sed/<OFF|ON|> ..... Select timed burst mode.
:SYSTem:**ELAP**sed**?** ..... Read current timed burst selection.

:SYSTem:**EVEN**t/<OFF|ON>/<1|2>/<RISe|FALl> ..... Select event mode.
:SYSTem:**EVEN**t**?** ..... Read current event selection

:SYSTem:**GAT**ing<ON|OFF> ..... Turn gating on or off.
:SYSTem:**GAT**ing**?** ..... Read gating selection.

:SYSTem:**GO** ..... Execute GO button.
:SYSTem:**NOGO** ..... Execute GO, but do not perform function.

:SYSTem:**HEAD**er<OFF|ON> ..... Select header type.
:SYSTem:**HEAD**er**?** ..... Read header type selected.

:SYSTem:**LONG**form<OFF|ON|0|1> ..... Select long or short form of headers.
:SYSTem:**LONG**form**?** ..... Read long or short-form selected.

:SYSTem:MACro</**f**unction/channel/trigger source/**t**rigger sequence ..... Macro used for speed.
/percent/start reference voltage/stop reference voltage>

:SYSTem:STAT/<AV><JI><MN><MX>..... Selects sets of statistics to be saved.
:SYSTem:STAT? ..... Reads number of sets acquired.

:SYSTem:STRObeARM<CH1|CH2|ARM1|ARM2|DC> ..... Select strobe arming input.
<POS|NEG|RISe|FALI>
:SYSTem:STRObeARM? ..... Read strobe arming input.

:SYSTem:STRObeCAL ..... Initiates a strobe calibration.

:SYSTem:STRObeCHANnel<1|2> ..... Select strobe input channel.
:SYSTem:STRObeCHANnel? ..... Read strobe channel.

:SYSTem:STRObeDELay<value> ..... Set strobe delay.
:SYSTem:STRObeDELay? ..... Read strobe delay.

:SYSTem:STRObeINC<value> ..... Set stepincrements|value.
:SYSTem:STRObeINC? ..... Read stepincrements|value.

:SYSTem:STRObeLEVel/<1|2>**/**<ARM1|ARM2|CH1|CH2>**/**max start delay ..... Set delays for strobe pulsefind.
/max end delay**/**max delta**/** min start delay**/**min end delay**/**min delta>

:SYSTem:STRObeSTARt<value> ..... Set start delay.
:SYSTem:STRObeSTARt? ..... Read start delay.

:SYSTem:STRObeSTOP<value> ..... Set stop delay.
:SYSTem:STRObeSTOP? ..... Read stop delay.

:SYSTem:STRObe#<value> ..... Set number of steps.
:SYSTem:STRObe#? ..... Read number of steps.

:SYSTem:TIMEout<value> ..... Timeout on pulse measurement.
10 seconds default. Integer seconds.
:SYSTem:TIMEout? ..... Read timeout value.

:SYSTem:WAVe<PEAK|FLAT|STRObe> ..... Set pulse find to locate peaks or flat spot (usually for a non sinewave).
:SYSTem:WAVe? ..... Read type of waveform search to be used by pulse find.

:SYSTem:WINDow</start value/stop value/<step increment|#of steps> ..... Set parameters.

2-6 ACQUIRE COMMANDS

:ACQuire:ALL<TT+|TT-|PW+|PW-|PERiod|TPD++|TPD- -|TPD+-|TPD- +|FREQ> ..... Select function and return all statistics.

:ACQuire:ANALysisFUNCtion</Func/Chan/LowStartCount/High StartCount ..... Select function/chan and
/StopCountIncrement Designator/Increment/DataDes> return function statistics

:ACQuire:ANALysisJITTer</Func/Chan/StartCount/LowStopCount ..... Select function/chan and
/HighStopCount/Increment/DataDes> return jitter statistics.

:ACQuire:ANALysisRANGe</Func/Chan/StartCount/LowStopCount ..... Select function/chan and
/HighStopCount/Increment/DataDes> return (Max-Min)/2.

:ACQuire:COMPlete? ..... Number of readings taken.

:ACQuire:COUNt<value> ..... Set Sample Size.
:ACQuire:COUNt? ..... Read Sample Size.

:ACQuire:DUTY ..... Returns duty cycle

:ACQuire**:FUNC**tion<TT+|TT-|PW+|PW-|PERiod|TPD++|TPD--|TPD+-|TPD-+|FREQ> ..... Select function.
:ACQuire**:FUNC**tion**?** ..... Read function selected.

:ACQuire**:LEV**el ..... Pulse Finder.

:ACQuire**:MEAS**ure ..... Takes a measurement and returns average & standard deviation.

:ACQuire**:RUN**<TT+|TT-|PW+|PW-|PERiod|TPD++|TPD- -|TPD+-|TPD- +|FREQ> ..... Select function and return average and jitter.

:ACQuire**:SETsCOUN**t<value> ..... Set sets size.
:ACQuire**:SETsCOUN**t**?** ..... Read sets size.

:ACQuire**:WIND**ow</start value/stop value/<step increment|#of points> ..... Set parameters and return voltage average.

2-7 CALIBRATE COMMANDS

:CALibrate**:DATA**<block> ..... Set external calibration values.
:CALibrate**:DATA?** ..... Read external calibration values.

:CALibrate**:EXT**ernal ..... Initiate external calibration.

:CALibrate**:INT**ernal ..... Initiate internal calibration.

:CALibrate**:REF?** ..... Return cal. reference voltage.

:CALibrate**:SIG**nal<“OFF, “8K,”1M”,”200M”> ..... Set internal calibration signal.
:CALibrate**:SIG**nal**?** ..... Read cal signal setting.

:CALibrate**:XINT**ernal<ASCII value> ..... Initiate extended internal calibration.

2-8 CHANNEL COMMANDS

:CHANnel<**STAR**t|**STOP**>**:EXT**ernalarm<ARM1|ARM2> ..... Select event external arm.
:CHANnel<**STAR**t|**STOP**>**:EXT**ernalarm**?** ..... Read selected external arm.

:CHANnel<**STAR**t|**STOP**>**:LEV**el<value> ..... Set event trip level.
:CHANnel<**STAR**t|**STOP**>**:LEV**el**?** ..... Read event level.

:CHANnel<**STAR**t|**STOP**>:<MIN|MAX>**?** ..... Read START or STOP min or max peaks.
:CHANnel<**STAR**t|**STOP**>**:COUN**t<1 to 131072> ..... Set event arm on Nth count.
:CHANnel<**STAR**t|**STOP**>**:COUN**t**?** ..... Read event arm on Nth count.

:CHANnel<**SWIT**ch**IDN?** ..... Returns Version of DSM-16
:CHANnel<**SWIT**ch<NN> ..... Select DSM-16 chan. and switch
:CHANnel<**SWIT**ch**?** ..... Returns selected chan. and switch
:CHANnel<**SWIT**ch<ON|OFF> ..... Enables/disables DSM-16 front panel switches.

2-9 DISPLAY COMMANDS

:DISPlay**:FILT**er<ON|OFF> ..... Select filtering on or off.
:DISPlay**:FILT**er**?** ..... Read selected filtering.

:DISPlay**:FILT**er<**MIN**imum|**MAX**imum><value> ..... Set filter value.
:DISPlay**:FILTer**<**MIN**imum|**MAX**imum>**?** ..... Read filter value.

:DISPlay**:LEV**el<value value> ..... Set percent.
:DISPlay**:LEV**el**?** ..... Read pulse percent.

:DISPlay**:LINE**<quoted string> ..... Display message on screen.

:DISPlay**:PAN**el<ON|OFF> ..... Turn front panel on or off.
:DISPlay**:PAN**el**?** ..... Read panel mode.

:DISPlay**:STAT**istics<ON|OFF> ..... Turn update statistics to display on/off.
:DISPlay**:STAT**istics**?** ..... Read statistics mode.

:DISPlay**:TEXT BLAN**k ..... Clear or restore display.

:DISPlay**:USER**<ON|OFF> ..... Selects user reference voltages for current function.
:DISPlay**:USER?** ..... Reads current user state.

2-10 MEASURE COMMANDS

:MEASure**:AVER**age**?** ..... Read average.

:MEASure**:DATA?** ..... Return measurement data.

:MEASure**:DATA4?** ..... Return measurement data as floating point.

:MEASure**:DATAT?** ..... Return timed burst elapsed times.

:MEASure**:DC**vlevel**?** ..... Read dc level.

:MEASure**:EVEN**t**?** ..... Returns event counter value (0-65536)

:MEASure**:JITT**er**?** ..... Read jitter.

:MEASure**:MIN?** ..... Read minimum measured value.

:MEASure**:MAX?** ..... Read maximum measured value.

:MEASure**:RANG**e**?** ..... Read range of measured values.

:MEASure**:SDEV**iation**?** ..... Read standard deviation.

:MEASure**:STAT4?** ..... Returns statistical data defined by :SYST:STAT for number of samples as float.

:MEASure**:STRO**be**VLEV**el**?** ..... Read strobed voltage level.

:MEASure**:VMAX**imum**?** ..... Read strobed maximum voltage.

:MEASure**:VMIN**imum**?** ..... Read strobed minimum voltage.

:MEASure**:VSDEV**iation**?** ..... Read voltage standard deviation.

:MEASure**:VDATA?** ..... Return strobed measured points.
:MEASure**:VDATA4?** ..... Return strobed measured as floating point.

:MEASure**:WIND**ow**?** ..... Return average strobed voltage.

2-11 TRIGGER COMMANDS

:TRIGger**:LEV**el<ARM1|<ARM2|GATe><value> ..... Set trigger level.
:TRIGger**:LEVel**<ARM1|<ARM2|GATe>**?** ..... Read trigger level setting.

:TRIGger**:<**MAX|MIN**>?**<ARM1|ARM2> ..... Read arming peaks.

:TRIGger**:SEQ**uence<STARt|STOP|STARtFIRST> ..... Selects trigger sequence.
:TRIGger**:SEQ**uence**?** ..... Reads trigger sequence.

:TRIGger**:SLOP**e<ARM1|<ARM2|GATe><POS|NEG|RISe|FALl|High|Low> ..... Set arming direction.
:TRIGger**:SLOP**e<ARM1|<ARM2|GATe>**?** ..... Read arming direction .

:TRIGger**:SOUR**ce<EXTernal|AUTomatic|MANual> ..... Selects trigger source.
:TRIGger**:SOUR**ce**?** ..... Read trigger source.

2-12 HARDCOPY COMMANDS

:HARDcopy**:HIST**ogram	Print burst histogram.
:HARDcopy**:PART**<alpha-numeric>	Enter part number.
:HARDcopy**:PART?**	Return part number.
:HARDcopy**:SER**ial<integer>	Enter serial number.
:HARDcopy**:SER**ial**?**	Return serial number.
:HARDcopy**:SER**ial**DECR**ement	Decrease serial # by one.
:HARDcopy**:SER**ial**INCR**ement	Increase serial # by one.
:HARDcopy**:SET**up	Print DTS set-up.
:HARDcopy**:STAT**istics	Print statistics w/abbrev. set-up information.

SECTION 3 - COMMON COMMANDS & STATUSING

3-1 DESCRIPTION OF THE COMMON COMMANDS & STATUS

IEEE-488.2 defines a set of common commands. These commands perform functions which are common to any type of instrument. They can therefore be implemented in a standard way across a wide variety of instrumentation. All the common commands of IEEE-488.2 begin with an asterisk. There is one key difference between the IEEE-488.2 common commands and the rest of the commands found in this instrument. The IEEE-488.2 common commands do not affect the parser's position within the command tree. Many of these commands are used for status.

Command	Command Name
***CLS**	Clear Status Command.
***ESE**	Event Status Enable Command.
***ESE?**	Event Status Enable Query.
***ESR?**	Event Status Register Query.
***IDN?**	Identification Query.
***OPC**	Operation Complete Command.
***OPC?**	Operation Complete Query.
***RCL** ..... <1-10>	Recall Command.
***RST**	Reset Command.
***SAV** ..... <1-10>	Save Command.
***SRE**	Service Request Enable Command.
***SRE?**	Service Request Enable Query.
***STB?**	Read Status Byte Query.
***TRG**	Trigger Command.
***TST?**	Test Instrument Query.

The bits in the status byte act as summary bits for the data structures residing behind them. In the case of queues, the summary bit is set if the queue is not empty. For registers, the summary bit is set if any enabled bit in the event register is set. The events are enabled via the correspond ing event enable register. Events captured by an event register remain set until the register is read or cleared. Registers are read with their associated commands. The "*CLS" command clears all event registers and all queues except the output queue. If "*CLS" is sent immediately following a <program message terminator>, the output queue will also be cleared.

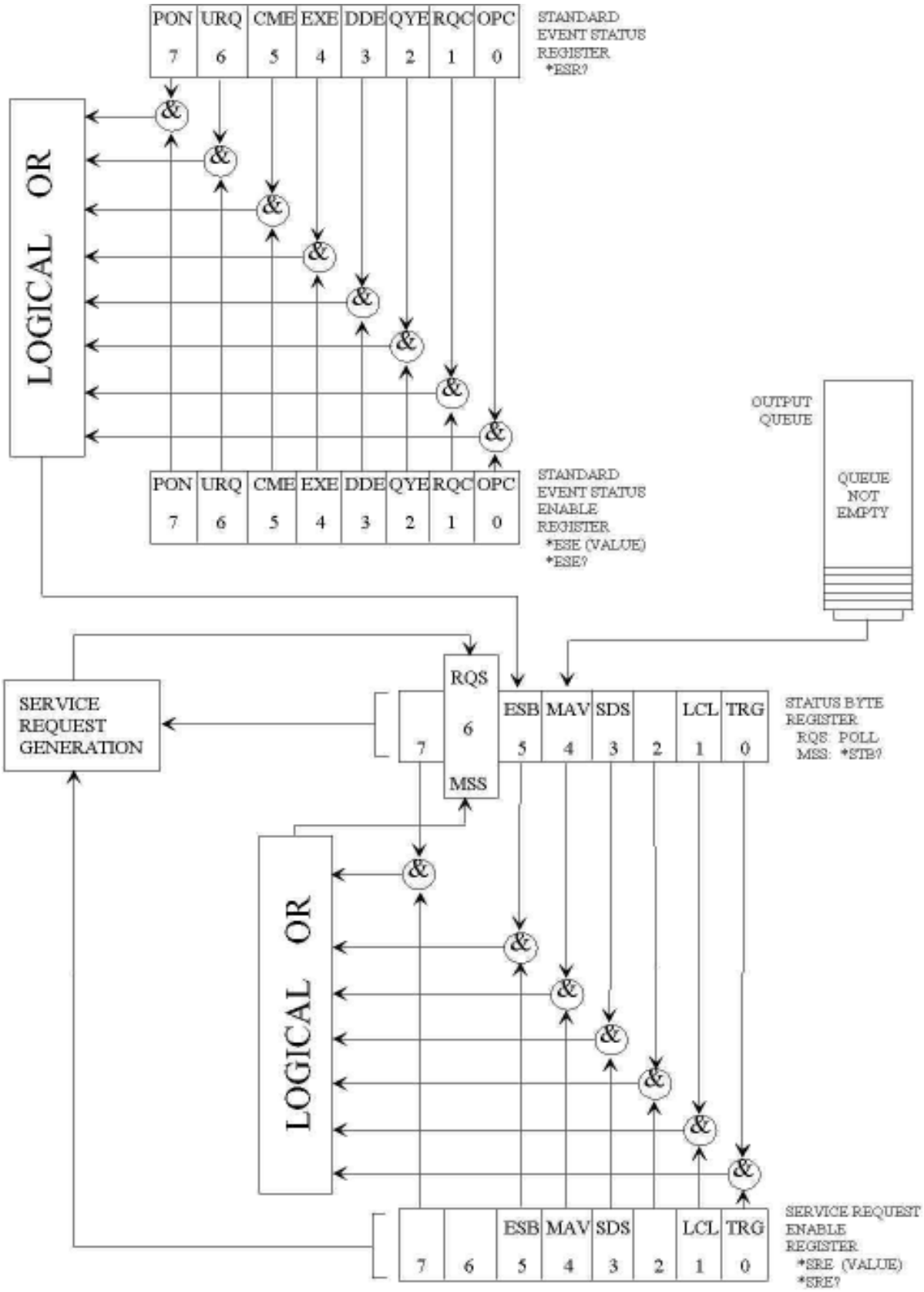


Figure 3-1 STATUS REPORTING

3-1.1 BIT DEFINITIONS

CME - Command error. Indicates whether the parser detected an error.

DDE - Device specific error. Indicates whether the device was unable to complete an operation for device dependent reasons.

ESB - Event status bit. Indicates if any of the conditions in the Standard Event Status Register are set and enabled.

EXE - Execution error. Indicates whether a parameter was out of range, or inconsistent with current settings.

LCL - Indicates whether a remote to local transition has occurred. Indicates when a Device Clear (DCL) is complete.

MAV - Message available. Indicates whether there is a response in the output queue.

MSS - Master summary status. Indicates whether the device has a reason for requesting service. This bit is returned for the *STB? query.

OPC - Operation complete. Indicates whether the device has completed all pending operations.

PON - Power on. Always 1 in the DTS.

QYE - Query error. Indicates whether the protocol for queries has been violated.

RQC - Request control. Indicates whether the device is requesting control. Asking for a simulated GO key to be executed.

RQS - Indicates if the device is requesting service. This bit is returned during a serial poll. RQS will be set to 0 after being read via a serial poll (MSS is not reset by *STB?).

SDS - Special device status.

TRG - Indicates whether a trigger has been received.

URQ - User request. Indicates whether a front panel key has been pressed.

3-1.2 KEY FEATURES

A few of the most important features of Status Reporting are shown below.

Operation Complete - The IEEE-488.2 structure provides one technique which can be used to find out if any operation is finished. The “OPC” command, when sent to the instrument after the operation of interest, will set the OPC bit in the Standard Event Status Register. If the OPC bit and the RQS bit have been enabled, a service request will be generated.

```
Send(0,5,”*SRE;*ESE1",11,EOI);      !enables an OPC service request.
Send(0,5,”*TRG;*OPC”,9,EOI);        !initiates data acquisition.
                                    !will generate a SRQ when the
                                    !acquisition is complete.
```

The Trigger Bit - The TRG bit indicates if the device has received a trigger. The TRG event register will stay set after receiving a trigger until it is cleared by reading it or using the *CLS command. If your application needs to detect multiple triggers, the TRG event register must be cleared after each one.

```
Send(0,5,"*SRE1",6,EOI);          !enables a trigger service request.
                                  !the next trigger will generate an SRQ.
Send(0,5,":TER?",5,EOI);          !queries the TRG event register, thus
Send(0,5,"*TRG",4,EOI);           !clearing it.
                                  !the next trigger can now generate an
Wait SRQ(0,result);               !SRQ.
```

Status Byte - If the device is requesting service (RQS set), and the controller serial polls the device, the RQS bit is cleared. The MSS bit (read with *STB?) will not be cleared by reading it. The status byte is not cleared when read, except for the RQS bit.

Serial Poll - The DTS supports the IEEE-488.1 serial poll feature. When a serial poll of the instrument is requested, the RQS bit is returned on bit 6 of the status byte.

Using Serial Poll - This example will show how to use the service request by conducting a serial poll of all instruments on the bus. In this example, assume that there are two instruments on the bus; a DTS at address 5 and a printer at address 1. These address assumptions are made throughout this manual, and it is also assumed that we are operating on GPIB controller board address 0.

The program command for serial poll using IEEE-488.2 in 'C' is ReadStatusByte (0,5,result);. The address 005 is the address of the DTS in this example. The command for checking the printer is ReadStatusByte (0,1,result); because the address of that instrument is 01 on bus address 0. This command reads the contents of the GPIB Status Register into the variable called result. At that time bit 6 of the variable result can be tested to see if it is set (bit 6=1).

The serial poll operation can be conducted in the following manner.

1. Enable interrupts on the bus. This allows the controller to "see" the SRQ line.
2. If the SRQ line is high (some instrument is requesting service)then check the instrument at address 1 to see if bit 6 of its status register is high.
3. Disable interrupts on the bus.
4. To check whether bit 6 of an instruments status register is high, use the following command line.

```
If (result & 0x40){
    then
    }
```

5. If bit 6 of the instrument at address 1 is not high, then check the instrument at address 5 to see if bit 6 of its status register is high.
6. As soon as the instrument with status bit 6 high is found, check the rest of the status bits to determine what is required.

The ReadStatusByte (0,5,result); command causes much more to happen on the bus than simply reading the register. This command clears the bus, automatically addresses the talker and listener, sends SPE (serial poll enable) and SPD (serial poll disable) bus commands, and reads the data. For more information about serial poll, refer to your controller manual, and programming language reference manuals.

After the serial poll is completed, the RQS bit in the DTS Status Byte Register will be reset if it was set. Once a bit in the Status Byte Register is set, it will remain set until the status is cleared with a *CLS command, or the instrument is reset.

Parallel Poll - The DTS does not support the parallel poll feature.

3-2 *CLS (Clear Status) command

The *CLS (clear status) common command clears the Event Status Register, the Status Byte Register, the trigger bit, the local bit and the error queue.

The Event Status Register is read by the *ESR? command. The Status Byte Register is read by the *STB? command or a serial poll.

Command syntax- *CLS

Example: `Send(0,5,"*CLS",4,EOI);`

Query Syntax- None

3-3 *ESE (Event Status Enable) command/query

The *ESE command sets the Standard Event Status Enable Register bits. The Standard Event Status Enable Register contains a mask value for the bits to be enabled in the Standard Event Status Register. A one (1) in the Standard Event Status Enable Register will enable the corresponding bit in the Standard Event Status Register, a zero will disable the bit. Refer to Table 3-1 for information about the Standard Event Status Enable Register bits, bit weights, and what each bit masks.

The *ESE query returns the current contents of the register.

Command Syntax- *ESE <mask>
<mask>::=0 to 255

Example: `Send(0,5,"*ESE 64",7,EOI);`

In this example, the *ESE 64 command will enable URQ, user request, bit 6 of the Standard Event Status Enable Register. Therefore, when a front-panel key is pressed, the event summary bit (ESB) in the Status Byte Register will be set also.

Event Status Enable Register
(High - Enables the ESR bit)

Bit	Weight	Enables
7	128	PON-Power On
6	64	URQ-User Request
5	32	CME-Command Error
4	16	EXE-Execution Error
3	8	DDE-Device Dependent Error
2	4	QYE-Query Error
1	2	RQC-Request Control
0	1	OPC-Operation Complete

Table 3-1 Standard Event Status Enable Register

Query Syntax- *ESE?

Returned Format: <mask><NL>
<mask>::=0 to 255

Example:
```
Send(0,5,"*ESE?",5,EOI);
Received(0,5,Event,1,EOI);
Printf("%d\n",Event);
```

3-4 *ESR? (Event Status Register) query

This ***ESR** query returns the contents of the Standard Event Status Register.

NOTE: Reading the register clears the Standard Event Status Register and the ESB bit in the STB register.

Query Syntax: *ESR?

Returned Format: <status><NL>
<status>::=0 to 255

Example:
```
Send(0,5,"*ESR?",5,EOI);
Receive(0,5,Event,1,EOI);
Printf("%d\n",Event);
```

With the example (*ESE=64), if a front-panel key has been pressed, the variable "event" will contain 64, the URQ (User Request bit).

Table 3-2 shows the Standard Event Status Register. The table shows each bit in the Standard Event Status Register, and the bit weight. When you read Standard Event Status Register, the value returned is the total bit weights of all bits that are high at the time you read the byte.

Event Status Register

Bit	Bit Weight	Bit Name	Condition
7	128	PON	0=not used-always zero
6	64	URQ	0=no front panel key has been pressed 1=front panel key has been pressed
5	32	CME	0=no command errors 1=a command error has been detected
4	16	EXE	0=no execution error 1=an execution error has been detected
3	8	DDE	0=no device dependent errors 1=a device dependent error has been detected
2	4	QYE	0-no query errors 1=a query error has been detected
1	2	RQC	0=request control
0	1	OPC	0=operation is not complete 1=operation is complete

0 = False = Low
1 = True = High

Table 3-2 Standard Event Status Register

3-5 *IDN? (Identification Number) query

The ***IDN?*** query allows the instrument to identify itself. It retuns the string: "WAVECREST, DTS-207(x), VERSION MAJOR, VERSION MINOR."

VERSION MAJOR = Major version of software release.
VERSION MINOR = Minor version of software release.

An *IDN? query must be the last query in a message. Any queries after the *IDN? in this program message will be ignored.

Query Syntax- *IDN?

Returned Format: `WAVECREST, DTS-207(x), NN.NN`

Example:
```
CHAR MESSAGE[50];
Send(0,5,"*IDN?",5,EOI);
Receive(0,5,MESSAGE,50,EOI);
Printf("%s\n",MESSAGE);
```

3-6 *OPC (Operation Complete) command/query

The ***OPC** (operation complete) command will cause the instrument to set the operation complete bit in the Standard Event Status Register when all pending device operations have finished. The *OPC? query places an ASCII "1" in the output queue when all pending device operations have finished.

Command Syntax- *OPC

Example:
```
Send(0,5,"*OPC",4,EOI);
```

Query Syntax- *OPC?

Example:
```
Send(0,5,"*OPC?",5,EOI);
Receive(0,5,data,1,EOI);
```

Returned format: "1"

3-7 *RCL (Recall) command

The *RCL command restores the state of the DTS from a specified set of saved setups. There can be ten (10) different setups (1 through 10).

Command Syntax- *RCL<specific setup>**#**

Example:
```
Send(0,5,"*RCL1",6.EOI);
```

Query Syntax- None

NOTE: See common command ***SAV** for information of specific information recalled/saved.

3-8 *RST (Reset) command

The *RST command place the instrument in a known state. The output buffer is cleared as well as the ESR and serial poll status registers. Use the interface clear (IFC) bus command to perform a hardware reset.

Command Syntax- *RST

Example:
```
int result;
Send(0,5,"*CLS",4,EOI);
Send(0,5,"*RST;*OPC",9,EOI);
result=0
while ((result&0X20 !=0){   /*wait for reset to finish*/
    ReadStatusByte(0,5,&result);
    }
/*reset complete*/
```

Query Syntax- None

3-9 *SAV (Save) command

The ***SAV** command stores the current settings of the DTS in non-volatile memory. This setup is saved and recalled by specifying a specific setup from 1 to 10. See the list below for the parameters saved. Notice that for each setting (1-10), each of the ten (10) functions has a number of settings saved.

Command Syntax- ***SAV**<specific setup>**#**

Example: `Send(0,5,"*SAV6",5,EOI);`

Query Syntax- None

During a SAVE or RECALL the following parameters are saved for later recall or recalled and used as DTS parameters:

Function Selection (defines edge direction)
- Channel selection (Ch1/Ch2/Both)
- Arming event arming sequence
- Start reference voltage
- Stop reference voltage
- External Arm reference voltage
- External Arm edge direction
- Pulse find levels (percentages)
- Start/Stop edge (rising or falling)
- Start/Stop arm on Nth count

Sample size
Sets size
Start/Stop VOH (max peak) voltage
Start/Stop VOL (min peak) voltage
Sample size
Sets size
Start/Stop VOH (max peak) voltage
Start/Stop VOL (min peak) voltage
Filter On/Off
Filter minimum
Filter maximum
Gating on/off
Start/Stop external arming inputs
Arming Source
Strobe start point
Strobe stop point
Strobe increment value
Strobe number of points
Strobe arming channel
Strobe input channel
Strobe delay
DC Channel

Notes: Cable measurement mode (on/off) not stored.
Parameters listed under Function Selection are saved for each function type.
The external calibration values are not saved on a SAVE.

3-10 *SRE (Service Request Enable) command/query

The ***SRE*** command sets the Service Request Enable Register bits. The Service Request Enable Register contains a mask value for the bits to be enabled in the Status Byte Register. A one in the Service Request Enable Register will enable the corresponding bit in the Status Byte Register, a zero will disable the bit. Refer to table 3-3 for the bits in the Service Requst Enable Register and what they mask.

The ***SRE*** query returns the current value.

Command Syntax- *SRE <mask>
<mask>::=0 to 255

Example: `Send(0,5,"*SRE16",7,EOI);`

NOTE: This example enables a service request to be generated when a message is available in the output queue. When a message is available, the MAV bit will be high.

Query Syntax- *SRE?

Returned Format: <mask><NL>
<mask>::=sum of all bits that are set - 0 through 255

Example:
```
Send(0,5,"*SRE?",5,EOI);
Receive(0,5,ENABLE,1,EOI);
Printf(%d\n",ENABLE);
```

Event Status Enable Register
(High - Enables the ESR bit)

Bit	Weight	Enables
7	128	not used
6	64	RQS-Request Service
5	32	ESR-Event Status Register
4	16	MAV-Message Available
3	8	SDS-Sub-Device Status
2	4	MSG-Message - Not Used
1	2	LCL-Local
0	1	TRG-Trigger

Table 3-3 Standard Event Status Enable Register

3-11 *STB? (Status Byte) query

Command Syntax- None

The ***STB** query returns the current value of the instrument's status byte. The **MSS** (Master Summary Status) bit and not **RQS** is reported on bit 6. The **MSS** indicates whether or not the device has at least one reason for requesting service. Refer to table 3-4 for the meaning of the bits in the status byte.

Note: To read the instrument's status byte with RQS reported on bit 6, use the GPIB Serial Poll.

Query Syntax- *STB?

Returned Format: <value><NL>
<value>::= 0 through 255

Example:
```
Send(0,5,"*STB?",5,EOI);
Receive(0,5,STATUS,1,EOI);
Printf("%d\n",STATUS);
```

Bit	Bit Weight	Bit Name	Condition
7	128	—-	0=not used
6	64	RQS/MSS	0=instrument has no reason for service 1=instrument is requesting service
5	32	ESR	0=no event status conditions have occurred 1=an enabled event status condition has occurred
4	16	MAV	0=no output messages are ready 1=an output message is ready
3	8	SDS	0=special device status
2	4	MSG Not Used	0=no message has been displayed 1=message has been displayed
1	2	LCL	0=a remote to local transition has not occurred 1=a remote to local transition has occurred
0	1	TRG	0=no trigger has occurred 1=a trigger has occurred

Table 3-4 Status Byte Register

3-12 *TRG (Trigger Event Register) command

The ***TRG** command initiates the DTS to take a measurement. This is the same effect as a Group Execute Trigger (GET) or sending the root command **RUN**. Use the root query, **:TER?**, to indicate when a measurement is complete.

Command Syntax- *TRG

Example:
```
int result, event_status;
Send (0,5,":TER?",5,EOI);   /*clears the TRG Event Register*/
result = 0
while((result & 0x01) !=0){
   ReadStatusByte(0,5,& result);
   }
Send(0,5,"*CLS",4,EOI);
Send(0,5,"*TRG",4,EOI);
while((result & 0x01) !=1){ /*wait for TRG bit of serial poll*/
   ReadStatusByte(0,5,& result);
   }
event_status = 0;
if((result & ESB) = = 1)    /*if ESB set*/
   {
   Send(0,5,"*ESR?",5,EOI);
   Receive(0,5,event_status,1,EOI);
   if((event_status & DDE) !=0)       /*if measurement bad*/
      Printf("failed measurement");
   }
```

Query Syntax- None

3-13 *TST? (Test Instrument) query

The ***TST?** query initiates a series of tests to be executed.

Command Syntax- None

Returned value: 0 = passed
-1 = failed

Query syntax- *TST?

Example:
```
Send(0,5,"*TST?",5,EOI);
Receive(0,5,status,1,EOI);
```

SECTION 4 - ROOT COMMANDS

4-1 DESCRIPTION OF THE ROOT COMMANDS

The **ROOT** commands are used to do a few basic instrument functions or read status.

Root commands: **:LER? :SDS?**
:RUN :TER?

4-2 LER?

The **LER?** query reads the Local Event Register. When the query is received and the register is read, it is also cleared. The status of the Local Event Register (0 or 1) is indicated by a serial poll status bit 1. When the LCL bit of a serial poll is a 1, the Device Clear (DCL) is complete. See the common command ***RST** for use with the *RST command.

Command syntax- None

Query syntax- :LER?

Example:
```
int result;
Send(0,5,":LER?",5,EOI);
ReadStatusByte(0,5,& result);
Printf("%d\n",result);
```

4-3 RUN

The **RUN** command initiates a measurement to be started in the DTS. Performs the same function as common command ***TRG**.

Command syntax- :RUN

Example:
```
Send(0,5,":RUN",4,EOI);
```

Query syntax- None

4-4 SDS?

The **SDS?** query reads the Special Device Status register. When the query is received the register value is returned and the register is cleared. The status of the Special Device Status register (0 or 1) is indicated by a serial poll or STB command on bit 3. This bit is used differently by specific instrument commands.

Recall storage ..... 1 = command complete
Display panel ON ..... 1 = command complete

Command syntax- None

Query syntax- :SDS?

Example:
```
int result
Send(0,5,":SDS?",5,EOI);
result = 1;
   while((result&0x08) !=0)  {
      ReadStatusByte(0,5,& result) ;
      }
```

```
Send(0,5,"*RCL5",5,EOI);
result = 0,
while((result&0x08 = =0)  {
   ReadStatusByte(0,5,& result);
   }
/*command complete*/
```

4-5 TER?

The **TER?** query allows the TRG Event Register to be read. When the TRG Event Register is read, it is cleared. A one (1) indicates a trigger has occurred. A zero (0) indicates a trigger has not occurred.

Command syntax- None

Query syntax- :TER?

Returned Format: Bit 1 of a serial poll will indicate the value of the TRG Event Register.

Example:
```
int result;
Send(0,5,":TER?",5,EOI);   /*clear TRG bit*/
while((result & 0x01) !=0){
   ReadStatusByte(0,5, & result);
   }
Send(0,5,"*TRG",4,EOI);
while((result & 0x01) !=1){
   ReadStatusByte(0,5, & result);
   }
/*command complete*/
```

Use the TER query to indicate when the following commands are complete:

Burst (*TRG)
Pulse Finder (:ACQ:LEV)
Internal Calibration
External Calibration
Strobe Calibration
Cable Measure

SECTION 5 - SYSTEM COMMANDS

5-1 DESCRIPTION OF SYSTEM COMMANDS

The **SYSTEM** commands control the way channels are selected, messages are formatted, front panel keys are simulated and how voltage measurement will be taken.

:SYSTem:<command syntax>

System commands:

ARMing	**LONG**form	**STRO**be**DEL**ay
CHANnel	**MAC**ro	**STRO**be**INCR**ement
DCCHANnel	**NOGO**	**STRO**be**LEV**el
ELAPsed	**STAT**istics	**STRO**be**STAR**t
EVENt	**STRO**be<points>	**STRO**be**STOP**
GATing	**STRO**be**ARM**	**TIM**eout
GO	**STRO**be**CAL**	**WAV**eform
HEADer	**STRO**be**CHAN**nel	**WIND**ow

5-2 ARMING

The **SYSTEM ARMING** command is a macro command to allow the sending of all commands related to arming the instrument in one command.

The parameters that can be sent are:

Trigger source ..... EXT/AUT/MAN
Arming enable sequence ..... STAR/STOP/STARTFIRST
Start arming input ARM1/ARM2
Stop arming input ARM1/ARM2
Arm1 input voltage reference ..... ±1.1
Arm2 input voltage reference ..... ±1.1
Arm1 input slope (edge) ..... POS/NEG/RIS/FAL
Arm2 input slope (edge) ..... POS/NEG/RIS/FAL
Start arm on count..... 1 to 131072
Stop arm on count..... 1 to 131072

The parameter's position is defined by a forward slash (/). If a parameter is not being set the forward slash must be used.

Command syntax- :**SYST**em:**ARM**ing/trigger source/enabling sequence/start arm /stop arm/arm1 ref/arm2 ref/arm1 slope /arm2 slope/start count/stop count

Example 1:
```
Send(0,5,":SYSTem:ARMing/EXT/STARTFIRST/ARM1
ARM2/+0.0001/+0.0001/POS/POS/2/256",69,EOI);
```

The following example only sets the arming reference voltage and slope.

Example 2:
```
Send(0,5,":SYSTem:ARMing/ / / / /+0/+0/NEG/NEG
    / /",34,EOI);
```

Query Syntax- None

5-3 CHANNEL

The **CHANNEL** command selects which channel is used for input from the front panel.

The **CHANNEL** query returns the presently selected channel.

Command syntax- :SYSTem:**CHAN**nel<1|2|BOTH>

Example: `Send(0,5,":SYSTem:CHANnel1",16,EOI);`

Query syntax- :SYSTem:**CHAN**nel**?**

Example: `Send(0,5,":SYSTem:CHANnel?",16,EOI);`
Response: `<"1"|"2"|"BOTH">`

5-4 DCCHANNEL

The **DCCHANNEL** command selects a DC measurement and the input channel that will be measured.

The **DCCHANNEL** query returns the channel presently selected.

Note: The selection of a DC measure excludes a STROBE measurement and the selection of a strobe arm excludes a DC measurement.

Command syntax- :SYSTem:**DCCHAN**nel<1|2>

Example: `Send(0,5,":SYSTem:DCCHANnel1",18,EOI);`

Query syntax- :SYSTem:**DCCHAN**nel**?**

Example: `Send(0,5,":SYSTem:DCCHANnel?",18,EOI);`
Response: `<"1"|"2">`

5-5 ELAPSED

The **ELAPSED** command enables the elapsed time counter to be initialized and it will be started when the proper edge gate is received on the ARM2 input.

Command syntax- :SYSTem:**ELAP**sed**/**<OFF|ON>

Example: `Send(0,5":SYSTem:ELAPsedON",16,EOI);`

Query syntax- :SYSTem:**ELAP**sed**?**

Example: `Send(0,5,":SYSTem:ELAPsed?",15,EOI);`
Response: `<"1"|"2">`

5-6 EVENT

The **EVENT** command enables the event counter to be initialized and counting will start on the next event. To stop or disable the counting, send the :**SYST:EVENOFF** command.

Command syntax- :SYSTem:**EVEN**t**/**<OFF|ON>/<1|2>/<RISe|FALl>/<Low|High>/volt. level

Example: `Send(0,5":SYSTem:EVENt/ON/1/RIS/LOW/1.1",14,EOI);`

Query systax- :SYSTem:**EVEN**t**?**

Example: `Send(0,5, ":SYSTem:EVENt?",13,EOI);`
Response: `<OFF|ON> <RIS|FAL> <L|H> <ASCII VOLTAGE>`
Example: `ON FAL L +0.50000`

5-7 GATING

The **GATING** command turns gating mode on or off. The selection of gating excludes the use of ARM2. When gating is selected the ARM2 edge and reference voltage is associated with gating. The **GATING** query returns the present setting of gating.

Command syntax- :SYSTem:GATing<ON|OFF>

Example: `Send(0,5,":SYSTem:GATingON",16,EOI);`

Query syntax- :SYSTem:GATing?

Example: `Send(0,5,":SYSTem:GATing?",15,EOI);`
Response: `<ON|OFF>`

5-8 GO

The **GO** command simulates a user pressing the front panel go button. This command would be used in conjunction with two (2) status bits of the Event Status Register (*ESR?). The host would look for the event status register bit 1, Request Control (asking for the GO key to be pressed). The host would then send the system go command and wait for the event status register bit 6, User Request to be set to a one (1) indicating the simulated pressing of the go key was completed.

Command syntax- :SYSTem:GO

Example:
```
char event_status;
Send(0,5,":CALibrate:EXTernal", 19,EOI);
event_status = 0;
while ((event_status&RQC==0)  {
   Send(0,5,"*ESR?",5,EOI);
   Receive(0,5, event_status,1,EOI);
   }
Send(0,5,":SYSTem:GO",10,EOI);
event_status = 0;
while ((event_status&URQ==0)  {
   Send(0,5,"*ESR?",5,EOI);
   Receive(0,5,event_status,1,EOI);
}
```

Query syntax- None

5-9 HEADER

The **HEADER** command allow the option of not having the header returned on a response from the DTS.

The **HEADER** query returns the type of header presently selected.

Command syntax- :SYSTem:HEADer<OFF|ON>

Example: `Send(0,5,":SYSTem:HEADerOFF",17,EOI);`

Query syntax- :SYSTem:HEADer?

Example: `Send(0,5,":SYSTem:HEADer?",15,EOI);`
Response: `<"0"|"1"> (OFF or ON)`

5-10 LONGFORM

The **LONGFORM** command selects whether a header is returned from the DTS is of a long form or short form. This command works with the **HEADER** command.

The **LONGFORM** query returns the presently selected long or short form.

Command syntax- :SYSTem:**LONG**form<OFF|ON>

Example: `Send(0,5,":SYSTem:LONGformOFF",19,EOI);`

Query syntax- :SYSTem:**LONG**form**?**

Example: `Send(0,5,":SYSTem:LONGform?",17,EOI);`

5-11 MACRO

The **SYSTEM MACRO** command can be used to send multiple commands for a few settings that usually change frequently.

The parameters that can be sent are:

Function ..... TPD++/TPD—/TPD+-/TPD-+/TT+/TT-/PW+/PW-/PER/FREQ
Channel 1/2/BOTH
Trigger Source ..... EXT/AUT/MAN
Arming Enable Sequence ..... STAR/STOP/STARTFIRST
Peak Percentage 50 50/80 20/20 80/90 10/10 90

Note: Any combination greater than zero (0) and less than 100 is valid over the GPIB interface

Start Input Voltage Reference ..... ±1.1
Stop Input Voltage Reference ..... ±1.1

Command syntax- :SYSTem:**MAC**ro/Function/Channel/Trigger source/Trigger sequence/percentage/start reference/stop reference

Example: `Send(0,5,":SYSTem:MACro/TT+/1/AUT/STOP/80 20/+0.003/-0.001",48,EOI);`

If a parameter is not used, that location can be left blank.

Example: `Send(0,5,":SYST:MAC/TPD++/BOTH/ / / / /",25,EOI);`

Query syntax- None

5-12 NOGO

The **NOGO** command simulates a user pressing the front panel go button, but does not perform the function. This command would be used in conjunction of two (2) status bits of the Event Status Register (*ESR?).

The host would look for the event status register bit 1, Request Control (asking for the GO key to be pressed). The host would then send the system nogo command and wait for the event status register bit 6, User Request to be set to a one (1) indicating the simulated pressing of the go key was completed.

Command syntax- :SYSTem:**NOGO**

Example:
```
char event_status;
Send(0,5,":CALibrate:EXTernal", 19,EOI);
event_status = 0;
while ((event_status&RQC==0)  {
   Send(0,5,"*ESR?",5,EOI);
   Receive(0,5, event_status,1,EOI);
   }
Send(0,5,":SYSTem:NOGO",12,EOI);
event_status = 0;
while ((event_status&URQ==0)  {
   Send(0,5,"*ESR?",5,EOI);
   Receive(0,5,event_status,1,EOI);
   }
```

Query syntax- None

5-13 STAT (Statistics)

The **STAT** command saves a selected group of statistics for each measurement: Average, Jitter, Minimum and Maximum. See the **:MEAS:STAT4** command for reading data.

The STAT query returns the selected group of statistics in 4-byte floating point form in the same order every time regardless of what order they were selected. The order is AV, JI, MN, MX.

Command syntax- :SYST:STAT/<ON|OFF>/<AV><JI><MN><MX>

Example: `Send(0,5,":SYST:STAT/ON/JIAVMXMN",22,EOI);`

Query syntax- :SYST:STAT?

Example: `Send(0,5,":SYST:STAT?",11,EOI);`
Response: <ON|OFF><AV><JI><MN><MX>

5-14 STROBE<POINTS>

The **STROBE**<POINTS> command sets the number of voltage measurement points. The first measurement will be at the start value.

The **STROBE**<POINTS> query will return the present window number of points value.

Command syntax- :**SYST**em:**STRO**be**#**<ASCII integer in picoseconds>

Example: `Send(0,5,":SYSTem:STRObe#20",17,EOI);`

Query syntax- :**SYST**em:**STRO**be**#?**

Example: `Send(0,5,":SYSTem:STRObe#?",16,EOI);`
Response: <ASCII integer>

5-15 STROBEARM

The **STROBEARM** command selects how a voltage measurement is taken. CH1, CH2, ARM1 or ARM2 selects the signal and edge controlling a strobed voltage measurement. The strobed point on a waveform can be controlled by moving the strobe signal, when not using the signal being stobed, or use the **STROBEDELAY** command.

If the strobe arm is not selected, then a DC measurement without strobing is defaulted. The **STROBEARM** query returns the strobe arm selected or DC if strobing is not selected.

Command syntax-
:SYSTem:STRObeARM<CH1|CH2|ARM1|ARM2|DC><POS|NEG|RISe|FALI>

Example: `Send(0,5,":SYSTem:STRObeARMCH1",17,EOI);`

Query syntax- :SYSTem:STRObeARM?

Example: `Send(0,5,":SYSTem:STRObeARM?",18,EOI);`
Response: <ASCII string>

5-16 STROBECAL

The **STRObeCAL** command initiates the DTS to calibrate the Strobing Digital Voltmeter (SDVM) delay settings (20 to 100,000 nanoseconds).

A complete calibration of the instrument should be completed before performing a strobe calibration due to the command's use of the precision measurement capabilities of the DTS.

1. External calibrate with DC.
2. Internal calibrate.
3. Strobe calibrate.

Command syntax- :SYSTem:STRObeCAL

Example: `Send(0,5,":SYSTem:STRObeCAL",17,EOI);`

Query syntax- None

5-17 STROBECHANNEL

The **STROBECHANNEL** command selects which input channel waveform will be strobed.

The **STROBECHANNEL** query returns the presently selected strobe channel.

Command syntax- :SYSTem:STRObeCHANnel<1|2>

Example: `Send(0,5,":SYSTem:STRObeCHANnel1",22,EOI);`

Query syntax- :SYSTem:STRObeCHANnel?

Example: `Send(0,5,":SYSTem:STRObeCHANnel?",22,EOI);`

5-18 STROBEDELAY

The **STROBEDELAY** command is used to allow strobed voltage measurements along the pulses of a selected channel. Strobing is armed from External Arm (ARM1 or ARM2).

With the same signal on a selected channel and on the selected arm channel, the strobed voltage value read will be 20ns from the beginning of the signal.

NOTE: To strobe at the beginning of a signal, delay the signal 20ns.

The **STROBEDELAY** query returns the present strobe delay setting.

The range of delay settings is from 20,000ps to 100,000,000ps.

Command syntax- :SYSTem:**STRO**be**DEL**ay<ASCII integer in picoseconds>

Example: `Send(0,5,":SYSTem:STRObeDELay20000",14,EOI);`

Query syntax- :SYSTem:**STRO**be**DEL**ay?

Example: `Send(0,5,":SYSTem:STRObeDELay?",13,EOI);`

5-19 STROBEINCREMENT

The **STROBEINCREMENT** command sets the increment between window strobe points. The increment is set in picoseconds.

The **STROBEINCREMENT** query returns the present window strobe delay increment.

NOTE: For any given delay, resolution at that delay is better than 0.2% of the delay.

Command syntax- :SYSTem:**STRO**be**INC**rement<ASCII integer in picoseconds>

Example: `Send(0,5,":SYSTemSTRObeINCrement10000",27,EOI);`

Query syntax- :SYSTem:**STRO**be**INC**rement?

Example: `Send(0,5,":SYSTem:STRObeINCrement?",24,EOI);`

5-20 STROBELEVEL

The **STROBELEVEL** command is a macro type command to set up the parameters for a strobe pulse find. The command provides for the selection of the channel, arming input, maximum peak window and minimum peak window. Each window is defined by a start delay, end delay and the delay increment between strobing points. The delays are in integer picoseconds. See Figure 5-1.

Command syntax- :SYSTem:**STRO**be**LEV**el/1|2/<ARM1|ARM2|CH1|CH2>/max start delay /max end delay/max step increment/min start delay/min end delay/min step increment

Example: 20megahertz signal (25 nanosecond positive and negative pulses)

```
Send(0,5,":SYSTem:WAVeSTRObe",18,EOI);
Send(0,5,":SYSTem:STRObeLEVel/1/ARM1/45000/45000
     /1000/25000/30000/1000",60,EOI);
Send(0,5,":ACQuire:LEVel",14,EOI);
```

Query syntax- :SYSTem:**STRO**be**LEV**el**?**

Example: `Send(0,5,":SYSTem:STRObeLEVel?",20,EOI);`

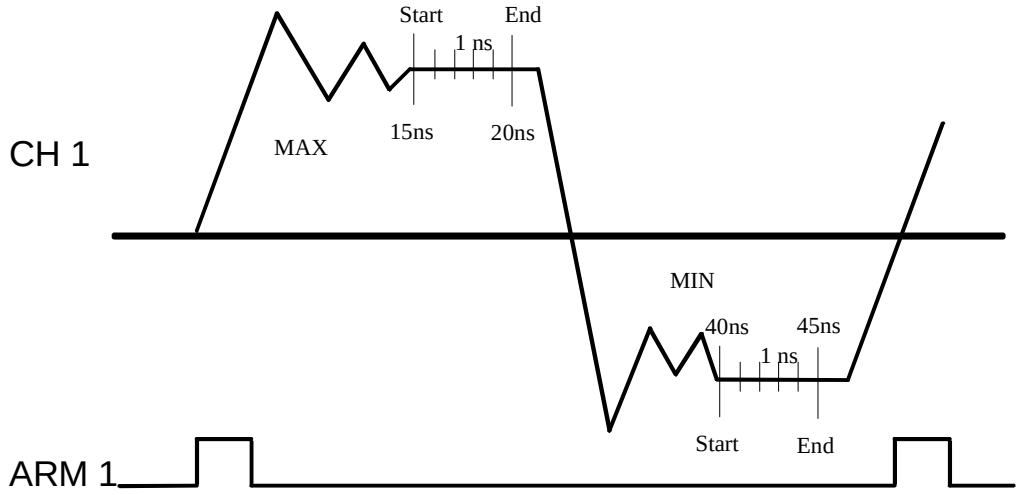


Figure 5.1 Strobelevel

5-21 STROBESTART

The **STROBESTART** command sets the start delay for the measure window command. The delay can be from 20,000ps to 100,000,000ps.

The **STROBESTART** query returns the present window start delay.

Command syntax- :SYSTem:**STRO**be**STAR**t**<**ASCII integer in picoseconds**>**

Example: `Send(0,5,"SYSTem:STRObeSTARt20000",25,EOI);`

Query syntax- :SYSTem:**STRO**be**STAR**t**?**

Example: `Send(0,5,":SYSTem:STRObeSTARt?",20,EOI) ;`

5-22 STROBESTOP

The **STROBESTOP** command sets the stop delay for the measure window command. The delay can be from 20,000ps to 100,000,000ps.

The **STROBESTOP** query returns the present window start delay.

Command syntax- :SYSTem:**STRO**be**STOP**<ASCII integer in picoseconds>

Example: `Send(0,5,":SYSTem:STRObeSTOP100000",24,EOI);`

Query syntax- :**SYST**em:**STRO**be**STOP?**

Example: `Send(0,5,":SYSTem:STRObeSTOP?,19,EOI);`

5-23 TIMEOUT

The **TIMEOUT** command sets the timeout value, in seconds, to wait before reporting "No Pulses Found", during a measurement.

The default, which is set on power up, is 10 seconds.

Command syntax- :SYSTem:**TIM**eout<value>

Example: `Send(0,5,":SYSTem:TIMeout15",17,EOI);`

Query syntax- :SYSTem:**TIM**eout**?**

Example: `Send(0,5,":SYSTem:TIMe?",13,EOI);`

5-24 WAVEFORM

The **WAVEFORM** command selects the mode of pulsefind. Use FLAT to locate the flatspot of a square wave and use PEAK to find the peaks of a sine waveform.

The **WAVEFORM** query returns the presently selected mode.

Command syntax- :SYSTem:**WAV**e<FLAT|PEAK|STRObe>

Example: `Send(0,5,":SYSTem:WAVePEAK",16,EOI);`

Query syntax- :SYSTem:**WAV**e?

Example: `Send(0,5,":SYSTem:WAVe?",13,EOI);`

NOTE: Use the :SYSTem:STRObeLEVel command to select the window for strobing.
Use the:ACQuire:LEVel?(pulsefind) to perform the pulsefind.

5-25 WINDOW

The **WINDOW** command is a macro command to allow the parameter setup for the measure window command. A window can be from 20ns to100µs given in picoseconds.

To describe a window three (3) parameters must be given, window start delay, window stop delay and either the measure point increment or the number of points to make a measurement.

To set the parameters and return an average voltage measurement of the window, see the acquire window command.

Command syntax- :SYSTem:**WIND**ow/start value/stop value/<step increment|# of points>

Example 1: `Send(0,5,":SYSTem:WINDow/20000/100000/10000",33,EOI);`
Example 2: `Send(0,5,":SYSTem:WINDow/20000/100000/#10",31,EOI);`

SECTION 6 - ACQUIRE COMMANDS

6-1 DESCRIPTION OF ACQUIRE COMMANDS

The **ACQUIRE** commands are used to set parameters used during a measure command.

:ACQuire:<command syntax>

Acquire commands: **ANAL**ysis
COMPlete
COUNt
DUTY
LEVel
SETsCOUNt

Acquire Macros: AC Measure - **ALL**
FUNCtion
MEASure
RUN
DC Measure - **WIIND**ow

6-2 ANALYSIS:FUNCTION|JITTER|RANGE

The **ANALYSISFUNCTION** command selects 1 of 10 functions and takes a measurement for the number of counts. The returned values are the mean of the measure, standard deviation, minimum and maximum in binary for each event where event is defined as a measurement. The returned values are in picoseconds except for frequency which returns the values in kilohertz.

Command syntax- **:ACQ**uire**:ANAL**ysis**FUNC**tion</FUNC/CHAN/LowStartCount /HighStartCount/StopCountDesignator/Increment/DataDes>

Example: `Send(0,5,":ACQuire:ANALysisFUNCtion/PW+/1/1/100/=/10/4",44,EOI);`
Example: `Send(0,5,":ACQuire:ANALysisFUNCtion/PER/2/1/100/+/10/4",44,EOI);`

If StopCount Designator = "1", Returns Arm start first,
= "+", Returns Start + 1
= "=", Returns Start Event

If DataDes = 2
Returns: Mean and standard deviation in binary
If DataDes = 4
Returns: The mean, standard deviation, minimum and maximum in binary.

Default: DataDes = 4

The **ANALYSISJITTER** command selects 1 of 10 functions and takes a measurement for the number of counts. The returned values are jitter, standard deviation, minimum and maximum in binary for each event where event is defined as a measurement. The returned value is in picoseconds, except for frequency which returns the values in kilohertz.

Command syntax- **ACQ**uire**:ANAL**ysis**JITT**er</FUNC/CHAN/StartCount/LowStopCount /HighStopCount/Increment/DataDes>

Example: `Send(0,5,":ACQuire:ANALysisJITTer/PW+/1/1/1/100/10/3",42,EOI);`
Example: `Send(0,5,":ACQuire:ANALysisJITTer/PER/2/1/2/100/10/3",42,EOI);`

If DataDes = 3
Returns: Jitter, i.e., standard deviation, min, max in binary.
If DataDes = 2
Returns: Jitter, i.e., standard deviation and mean.

Default: DataDes = 3

The **ANALYSISRANGE** command is similar to the **ANALYSISJITTER** command except the returned value is the range, (Max –Min)/2, with minimum and maximum in binary for each event where event is defined as a measurement.

Command syntax- :ACQuire**:ANAL**ysis**RANG**e</FUNC/CHAN/StartCount/LowStopCount /HighStopCount/Increment/DataDes>

Example: `Send(0,5,":ACQuire:ANALysisRANGe/PW+/1/1/1/100/10/3",41,EOI);`
Example: `Send(0,5,":ACQuire:ANALysisRANGe/PER/2/1/2/100/10/3",41,EOI);`

If DataDes = 3
Returns: Range, min, max in binary.
If DataDes = 2
Returns: Range, standard deviation and mean.

Default: DataDes = 3

6-3 COMPLETE

The **COMPLETE** command returns the number of measurements completed. The returned value will be an ASCII integer value.

Command syntax- :ACQuire**:COMP**lete**?**

```
Example: Send(0,5,":ACQuire:COMPlete?",18,EOI);
         Receive(0,5,data,1,EOI);
```
Response: <ASCII count>

6-4 COUNT

The **COUNT** command sets the number of measurements used to develop the statistics, average, minimum, maximum, range and standard deviation. The number of measurements can range from 1 to 1,000,000.

The **COUNT** query returns the present setting of the count value.

Command syntax- :ACQuire:**COUN**t<ASCII integer value>

Example: `Send(0,5,":ACQuire:COUNt200",17,EOI);`

Query syntax- :ACQuire**:COUN**t**?**

```
Example: Send(0,5":ACQuire:COUNt?",15,EOI);
         Receive(0,5,data,1,EOI);
```
Response: <ASCII integer>

6-5 DUTY

The **DUTY** command will calculate the duty cycle of the signal and return a three digit ASCII number. The percent will be of the positive pulse width in a format of xx.x%.

Command syntax- :ACQuire**:DUTY**

```
Example: Send(0,5,":ACQuire:DUTY",12,EOI);
Response: 49.8   (49.8%)
```

6-6 LEVEL

The **LEVEL** command causes the DTS to find the pulse levels on the start and/or stop channels depending on the channel selection. If the arming source selected is external, the levels of the arming channels are found as selected.

The levels are stored and can later be read by using the channel commands. The percent of the peak level found will be displayed and returned as the new start and stop references.

The levels found for each channel are the minimum and maximum peak and the selected percentage of these peaks.

Command syntax- **:ACQ**uire**:LEV**el

Example: `Send(0,5,":ACQuire:LEVel",14,EOI);`

Query syntax- None

6-7 SETSCOUNT

The **SETSCOUNT** command sets the count of a set of measurements which will create an average. This average is used with other set averages of sample size, to create the statistics available for return over the GPIB interface. The sets size value can range from 1 to 9999.

As an example, a sets size of a 100 and sample size of 1000 means that the statistics are of 1000 measurements of size 100.

The **SETSCOUNT** query returns the present setting of the sets size.

Command syntax- **:ACQ**uire**:SETsCOUN**t<ASCII integer value>

Example: `Send(0,5,":ACQuire:SETsCOUNt100",21,EOI);`

Query syntax- **:ACQ**uire**:SETsCOUN**t?

Example: `Send(0,5,":ACQuire:SETsCOUNt?",19,EOI);`
Response: <ASCII setscount>

ACQUIRE MACROS

6-8 ALL

The **ALL** command will select 1 of 10 functions, take a measurement and return the average, standard deviation, minimum and maximum. The function selected will force the following parameters to defaults:

Edges - Rising or falling

Channel - Single or both (if a single channel function, start or stop will be selected based on last single channel selected.

Arming - Auto-on-start, auto-on-stop, start first or stop first, based on the last arming sequence selected for that function.

Command syntax- **:ACQ**uire**:ALL**<TT+|TT-|PW+|PW-|PERiod|TPD++|TPD—|TPD+-|TPD-+|FREQ>

Example: `Send(0,5,":ACQuire:ALLTT+",15,EOI);`
`Receive(0,5,data,4,EOI);`

Query syntax- None

6-9 FUNCTION

The **FUNCTION** command will select 1 of 10 functions that will guide the DTS during time measurements. The function selected will force the follow parameters to defaults.

Edges to rising or falling

Channel to single or both (if a single channel function, start or stop will be selected based on last single channel selected.

Arming to auto-on-start, auto-on-stop, start first, or stop first based on the last arming sequence selected for that function.

The **FUNCTION** query will return the presently selected function.

Command syntax- **:ACQ**uire**:FUNC**tion<TT+|TT-|PW+|PW-|PERiod|TPD++|TPD—|TPD+-|TPD-+|FREQ>

Example: `Send(0,5,"ACQuire:FUNCtionTT+",19,EOI);`

Query syntax- **:ACQ**uire**:FUNC**tion**?**

Example: `Send(0,5,":ACQuire:FUNCtion?",18,EOI);`
Response: `<TT+|TT-|PW+|PW-|PER|TPD++|TPD—|TPD+-|TPD-+|FREQ>`

6-10 MEASURE

The **MEASURE** command will take a time measurement and return the average and standard deviation. The present function and reference voltages are used. This is a fast method of performing the acquire run command repetitively .

Command syntax- :**ACQ**uire:**MEAS**ure

Example: `Send(0,5,":ACQuire:MEASure",16,EOI);`
`Receive(0,5,data,2,EOI);`

Query syntax- None

6-11 RUN

The **RUN** command will select 1 of 10 functions, take a measurement and return the average and standard deviation. The function selected will force the following parameters to defaults:

Edges - Rising or falling

Channel - Single or both (if a single channel function, start or stop will be selected based on last single channel selected.

Arming - Auto-on-start, auto-on-stop, start first or stop first, based on the last arming sequence selected for that function.

The **RUN** command does not have a query.

Command syntax- **:ACQ**uire**:RUN**<TT+|TT-|PW+|PW-|PERiod|TPD++|TPD—|TPD+-|TPD-+|FREQ>

Example:
```
Send(0,5,":ACQuire:RUNTT+",15,EOI);
Receive(0,5,data,1,EOI);
```

Query syntax- None

6-12 WINDOW

The **WINDOW** command is a macro to set parameters and return the average (mean) voltage of the window. The window can be of a delay from 20,000ps to 100,000,000ps.

To describe a window, three (3) parameters can be given. If any parameter is omitted the foreward slash (/) must be placed in the command to indicate the proper spacing.

The three parameters are:

start delay value ..... 20,000ps to 100,000,000ps
stop delay value ..... 20,000ps to 100,000,000ps
increment between points ..... see system strobe increment command
or
number of measurement points..... see system strobe points command

Command syntax- **:ACQ**uire**:WIND**ow/start value/stop value/<step increment|#of points>

Example 1:
```
Send(0,5,":ACQuire:WINDow/25000/50000/1000",32,EOI);
Receive(0,5,voltage level,5,EOI);
```
Example 2:
```
Send(0,5,":ACQuire:WINDow/25000/50000/#100",32,EOI);
Receive(0,5,voltage level,5,EOI);
```

Query syntax- None

SECTION 7 - CALIBRATE COMMANDS

7-1 DESCRIPTION OF CALIBRATE COMMANDS

The **CALIBRATE** commands enables the host to perform an internal or external calibration and set or read the external calibration values.

:CALibrate:<command syntax>

Calibrate commands:

DATA	E**X**tended **INT**ernal
EXTernal	**REF?**
INTernal	**SIG**nal

NOTE: See the :SYSTem:STRObeCal command for performing a calibration of the Strobing Digital Voltmeter.

7-2 DATA

The **DATA** command can be used to enable the host to write the external calibration data values to the DTS.

When a DTS is calibrated externally, the values are stored in memory. These values are algebraically added to the measured value to determine the true measured value.

The user can apply these values in several ways:

1. Calibrate the DTS with equal length cables and then the DTS will return the actual measurements.
2. Enter the DTS values that represent the total calibration values (DTS values plus the user's fixture values). The DTS will then return the actual measurements.
3. Read into a host the external calibration values and then zero out the DTS values. The host can then compensate the returned measurement for both the DTS external calibration values plus the user's fixture delay values.

There are 16 external calibration values corresponding to the 10 functions in the 3 channel selections of BOTH/1/2.

SEQUENCE#	FUNCTION	CHANNEL
1	TPD++	BOTH
2	TPD—	BOTH
3	TPD+-	BOTH
4	TPD-+	BOTH
5	TT+	1
6	TT-	1
7	PW+	1
8	PW-	1
9	PER	1
10	FREQ	1
11	TT+	2
12	TT-	2
13	PW+	2
14	PW-	2
15	PER	2
16	FREQ	2

To store the skew value, a header plus 16 ASCII string values are sent. Each value must be separated by a space or +/- signs.

:CALibrate**:DATA**+50+60-93-80+45+20+65-32-48+43+50+101+93-87-80+70

The **DATA** query command is used to read the 16 values that are returned as an ASCII string of signed integer values in picoseconds.

Command syntax- **:CAL**ibrate**:DATA**<16 ASCII integers>

Example:
```
Send(0,5,":CALibrate:DATA+50+51+100+48+83+46+50+30+56
     +101+87+80-49+45+36+51",70,EOI);
```

To zero the cal data table, send the following:

```
Send(0,5,":CALibrate:DATA 0 0 0 0 0 0 0 0 0 0 0 0 0 0
     0  0",47,EOI);
```

Query syntax- **:CAL**ibrate**:DATA?**

Example:
```
char data [200];
Send(0,5,":CALibrate:DATA?",16,EOI);
Receive(0,5,data,200,EOI);
```

Response:
```
+50 +51 +100 +48 +83 +46 +50 +30 +56 + 101 +87 +80
−49 +45 +36 +51
```

7-3 EXTERNAL

The **EXTERNAL** command initiates the DTS to measure and build an external calibration table. This table can be external front end skew of the DTS only or the DTS and the user's fixture skew. See the calibrate data command for further detail on how the user can use this table.

Command syntax- **:CAL**ibrate**:EXT**ernal

Example:
```
Send(0,5,":CALibrate:EXTernal",19,EOI);
```

Query syntax- None

See Appendix A for a more complete example.

7-4 INTERNAL

The **INTERNAL** command initiates the DTS to build a calibration table for use during measurements.

Command syntax- **:CAL**ibrate**:INT**ernal

Example:
```
Send(0,5,":CALibrate:INTernal",19,EOI);
```

Query syntax- None

See Appendix A for a more complete example.

7-5 INTERNAL CALIBRATION - EXTENDED

The internal cal function will process 20,000,000 samples while taking 11 minutes to complete. With e**X**tended **INT**ernal calibration, the user can set a multiplier, 5.5 minutes, of the sample count from 1 to 25. Time for completion is extended by the same multiplier factor. A setting of 6 is recommended.

Extended Internal Calibration allows the user to reduce jitter due to the noise floor of the instrument through the use of longer internal calibration periods. The selected multiplier, from 1 to 25, extends the base calibration period of approximately 5.5 minutes by that factor. The table below shows typical results using the selected multipliers. Calibration times are approximated.

Command Syntax- :CALibrate:XINTernal<ASCII VALUE>

Example: `Send(0,5,":Calibrate:XINTernal6",21,EOI");`

Multiplier Factor	Cal Time	1-sigma jitter	0db Peak Noise Floor
1	5.5 min.	3.36	2.68ps
2	11 min.	2.56	1.46ps
6	33 min.	2.24	849fs
24	132 min.	2.19	827f

*** Embedded code version 1.98 or greater must be installed for extended internal calibration to work.

*** The extended internal calibration can only be initiated via GPIB commands with a user program or Virtual Instruments version 3.20 or greater. See GPIB manual for command.

Query Syntax- None

7-6 REF?

The **REF?** query will return the calibration reference voltages of the DTS' DC supplies. These specifications are referenced when calibration certification procedures are implemented.Refer to the User's Guide and Reference Manual for more information.

Command Syntax- None

Query syntax- :CALibrate**:REF?**

Example: `Send(0,5,":CALibrate:REF?",15,EOI);`

7-7 SIGNAL

The **SIGNAL** command will set the calibration signal to the specified parameter.

Command Syntax- :CALibrate**:SIG**nal<OFF|8K|1M|200M>

Example: `Send(0,5,":CALibrate:SIGnal1M",19,EOI);`

Query syntax- :CALibrate**:SIG**nal**?**

Example: `Send(0,5,":CALibrate:SIGnal?",18,EOI);`
Response: `<OFF|8K|1M|200>`

SECTION 8 - CHANNEL COMMANDS

8-1 DESCRIPTION OF CHANNEL COMMANDS

The **CHANNEL** commands write and read the channel start and stop reference voltages, the arm on *n*th counts and external arming selections..

:CHANnel<**STAR**t|**STOP**>:<command syntax>

Channel commands: **COUN**t **LEV**el **SWIT**ch
EXTernalarm **MIN**imum/**MAX**imum

8-2 COUNT

The **COUNT** command sets the arm-on-*n*th-event count for either the start or stop event. The range of the nth event is from 1 to 131072.

The **COUNT** query returns the count of either the start or stop event.

Command syntax- :CHANnel<**STAR**t|**STOP**>**:COUN**t<1|131072>

Example: `Send(0,5,":CHANnelSTARt:COUNt100",22,EOI);`

Query syntax- :CHANnel<**STAR**t|**STOP**>**:COUN**t**?**

Example: `Send(0,5,":CHANnelSTOP:COUNt?",19,EOI);`
Response: <ASCII count>

8-3 EXTERNALARM

The **EXTERNALARM** command selects which arming channel (ARM1 or ARM2) is associated with the start and stop events.

The **EXTERNALARM** query returns the arming selected for a specific (start/stop) event.

Command syntax- :CHANnel<**STAR**t|**STOP**>**:EXT**ernalarm<ARM1|ARM2>

Example: `Send(0,5,":CHANnelSTARt:EXTernalarmARM1",28,EOI);`

Query syntax- :CHANnel<**STAR**t|**STOP**>**:EXT**ernalarm**?**

Example: `Send(0,5,":CHANnelSTOP:EXTernalarm?",25,EOI);`
Response: `<ARM1|ARM2>`

8-4 LEVEL

The **LEVEL** command sets the start/stop reference levels. The range is ±1.1 volts in 150 microvolt resolution.

The **LEVEL** query returns the start/stop levels. The level returned is an ASCII value.

Command syntax- :CHANnel<**STAR**t|**STOP**>**:LEV**el<ASCII>

Example: `Send(0,5,":CHANnelSTARt:LEVel+1.33",23,EOI);`

Query syntax- :CHANnel**STAR**t**:LEV**el**?**

Example: `Send(0,5,":CHANnelSTARt:LEVel?",20,EOI);`
Response: <ASCII value>
Example: `+1.10000`

See Appendix C for more information regarding returned data formats.

8-5 MINIMUM/MAXIMUM

The **MINIMUM/MAXIMUM** query returns the minimum or maximum peak levels of the start or stop reference levels. The peak values measured when the last pulse find was initiated. This pulsefind could have been initiated from the front panel or with the :Acquire:Level command.

Command syntax: None

Query syntax- :CHANnel<**STAR**t|**STOP**>:<MIN|MAX>**?**

Example: `Send(0,5,":CHANnelSTARt:MIN?",18,EOI);`
Response: <ASCII MIN or MAX peak level>
Example: `-1.01890`

8-6 SWITCH ON/OFF

The **SWITCH ON/OFF** command enables or disables the switches on the front panel of the DSM-16.

Command syntax- :CHANnel**:SWIT**ch<ON|OFF>

Example: `Send(0,5,":CHANnel:SWITchON",17,EOI);`

Query syntax- None

8-7 SWITCH IDN

The **SWITCH IDN?** query returns the version number of the DSM–16. The returned value is an ASCII number representing the version major and minor (i.e. 1.1).

Command syntax- None

Query syntax- :CHANnel**:SWIT**ch**IDN?**

Example: `Send(0,5,":CHANnel:SWITchIDN?",19,EOI);`
Response: <ASCII number `1-16`>

8-8 CHANNEL:SWITCH number

The **SWITCH** number command identifies the DTS input channel to be selected (1 or 2). The DSM–16 is designed to be used as a 1 of 8 matrix to the DTS channel (1 of 8 to channel 1, and 1 of 8 to channel 2). The matrix inputs are assigned channel numbers 11-18 and 21-28.

NOTE: A small number of units are labeled 1 through 16.

Command syntax- :CHANnel**:SWIT**ch<11...18|21...28>

Example: `Send(0,5,":CHANnel:SWITch15",17,EOI);`

This will select the left bank of eight, fifth input from the left.

NOTE: The DSM-16 can be configured as a 1 of 15 matrix by connecting the eighth input from the left bank to the Channel 2 output.

Query syntax- :CHANnel**:SWIT**ch**?**

Example: `Send(0,5,":CHANnel:SWITch?",16,EOI);`

Response: <ASCII digits>

The returned format is ASCII digits representing both channel/switch numbers separated by a space (i.e. 15 21).

Example: `15 21`

SECTION 9 - DISPLAY COMMANDS

9-1 DESCRIPTION OF DISPLAY COMMANDS

The **DISPLAY** commands control the displaying of information on the front panel and if filtering is used in developing the statistics.

:DISPlay:<command syntax>

Display commands:

FILTer – ON/OFF	**PAN**el
MIN/MAX (Limits)	**STAT**istics
LEVel	**TEXT BLANK**
LINE	**USER**

9-2 FILTER (ON/OFF)

The **FILTER** (ON/OFF) command is used to select whether filtering will be used in calculating the statistics.

The **FILTER** query returns the current ON/OFF selection.

Command syntax- :DISPlay:**FILT**er<ON|OFF>

Example: `Send(0,5,":DISPlay:FILTerON",17,EOI);`

Query syntax- :DISPlay:**FILT**er**?**

Example: `Send(0,5,":DISPlay:FILTer?",16,EOI);`
Response: <ON|OFF>

9-3 FILTER (Limits)

The **FILTER** (limits) query commands return the ASCII floating point value of the presently set limits. Limits are ±2.4999999999 seconds. Maximum setting must be greater than Minimum setting.

A value of -999,999,999 is returned by the measure deviation query command if there were not any measurements within the limits.

Command syntax- :DISPlay:**FILT**er<**MIN**imum|**MAX**imum><signed ASCII floating point>

Example: `Send(0,5,":DISPlay:FILTerMINimum+0.000000000500",37,EOI);`

Query syntax- :DISPlay:**FILT**er<**MIN**imum|**MAX**imum>**?**

Example: `Send(0,5,":DISPlay:FILTerMINimum?",23,EOI);`
Response: <Signed ASCII floating point>
Example: `-5.000000e+001` (-0.5)

9-4 LEVEL

The **LEVEL** command sets the start and stop percentage level of peaks that the start and stop references will be set to. This percentage of peak will also be what the front panel start and stop references will be displaying.

The **LEVEL** query will return a start and stop percentage setting in two ASCII integers.

Command syntax- **:DISP**lay**:LEV**el<start value stop value>

Example: `Send(0,5,":DISPlay:LEVel20 80",19,EOI);`

Query syntax- **:DISP**lay**:LEV**el**?**

Example: `Send(0,5,":DISPlay:LEVel?",15,EOI);`
Response: <start value stop value>
Example: `20 80`

9-5 LINE

The **LINE** command displays a message on the front panel. The maximum length of the message is 40 characters.

Command syntax- **:DISP**lay**:LINE**<ASCII quoted string>

Example: `Send(0,5,":DISPlay:LINE"REMOTE OPERATION"",31,EOI);`

Query syntax- None

9-6 PANEL

The **PANEL** command selects if the front panel will accept input. If the DTS is being used in a secure environment then turn the front panel buttons and display off. Monitor the SDS bit after sending the **STATISTICS** command to determine when the DTS is no longer busy. Refer to section 4-3 for an explanation of the SDS bit.

The **PANEL** query returns the setting of the statistics updating.

Command syntax- **:DISP**lay**:PAN**el<ON|OFF>

Example: `Send(0,5,":DISPlay:PANelON",16,EOI);`

Query syntax- **:DISP**lay**:PAN**el**?**

Example: `Send(0,5,":DISPlay:PANel?",15,EOI);`
Response: `<ON|OFF>`

9-7 STATISTICS

The **STATISTICS** command selects if the statistics will be updated on the front panel after each measurement. If the DTS is being used only through the GPIB interface, time can be saved by not updating the front panel. Monitor the SDS bit after sending the **STATISTICS** command to determine when the DTS is no longer busy. Refer to section 4-3 for an explanation of the SDS bit.

The **STATISTICS** query command returns the setting of the update statistics.

Command syntax- **:DISP**lay**:STAT**istics<ON|OFF>

Example: `Send(0,5,":DISPlay:STATisticsON",21,EOI);`

Query syntax- **:DISP**lay**:STAT**istics**?**

Example: `Send(0,5,":DISPlay:STATistics?",20,EOI);`

Response: `<ON|OFF>`

9-8 TEXT BLANK

The **TEXT BLANK** command will clear the display.

Command syntax- :DISPlay**:TEXT BLANK**

Example: `Send(0,5,":DISPlay:TEXT BLANK",19,EOI);`

Query syntax- None

9-9 USER

The **USER** command selects the user set of references of the current function.

The DTS is capable of having the reference voltages set by two (2) methods.

1. Doing a pulse find and setting the references to a percentage of the peaks found.
2. Setting the start and stop voltage trip reference to a value.

When the user set the reference voltages directly, this is defined as a USER setting and is later selected by the display user command.

The **USER** query returns the setting of the user reference voltages.

Command syntax- :DISPlay**:USER**<ON|OFF>

Example: `Send(0,5,":DISPlay:USERON",15,EOI);`

Query syntax- :DISPlay**:USER?**

Example: `Send(0,5,":DISPlay:USER?",14,EOI);`
Response: `<ON|OFF>`

SECTION 10 - MEASURE COMMANDS

10-1 DESCRIPTION OF MEASURE COMMANDS

The **MEASURE** query returns the measurement statistics from the DTS to a host.

:MEASure**:**<command syntax>

Measure commands:

Time Measurement	**DC Measurement**
AVERage	**Single** -
DATA (Float or Double)	**DC**level
DATAT	**STRO**be**VLEV**el
EVENt	**Multiple** -
JITTer	**VDATA**
MAXimum	**VDATA4**
MINimum	**VMAX**imum
RANGe	**VMIN**imum
Standard **DEV**iation	**VSDEV**iation
STAT4	**WIND**ow

10-2 AVERAGE

The **AVERAGE** command returns the measured average of 1 to 1,000,000 measurements. The returned value is an ASCII floating point number.

Command syntax- None

Query syntax- **:MEAS**ure**:AVER**age**?**

Example: `Send(0,5,":MEASure:AVERage?",17,EOI);`
Response: <ASCII floating point>
Example: `-8.4566284e-011`

10-3 DATA/DATA4

The **DATA** query returns a selected number of measured values. These measured data values can be analyzed or used to provide a presentation. The measure data query supports two sizes of data types (See Appendix C) using IEEE standards for floating-point arithmetic (ANSI/IEEE Std. 754-1985):

The returned data stream is of the following format:

:MEASure**:DATAT#** xy...ddddddddd...
x = an ASCII digit representing the number of digits in y
y = a string of digits, of x length, which represents the number of bytes of information to be returned.
d=data

Command syntax- None

Query syntax- Float — **:MEAS**ure**:DATA4?**
:MEASure:Data#43200<200 bytes of data (50 measurements – 50x4)>

Double — **:MEAS**ure**:DATA?**
:MEASure:Data#3400<400 bytes of data (50 measurements – 50x8)>

Example:
```
char data[2048]
Send(0,5,:MEASure:DATA4?,15,EOI);
Receive(0,5,data,205,EOI);
```

Example:
```
char data[2048]
Send(0,5,:MEASure:DATA?,14,EOI);
Receive(0,5,data,405,EOI);
```

10-4 DATAT

The **DATAT** command returns the elapsed time measurements from a previous burst after the elapsed time counter has been turned on. With a sample size of 100 there will be 100 floating point time measurements returned.

The **DATAT** query returns a selected number of measured values. These measured data values can be analyzed or used to provide a presentation. See Appendix C for the returned data stream format types:

:MEASure**:DATAT#** xy...ddddddddd...
x = an ASCII digit representing the number of digits in y
y = a string of digits, of x length, which represents the number of bytes of information to be returned.
d=data

Query syntax: :MEASure**:DATAT#**3200<200 bytes of data (50 measurements - 50x4)>

Float — **:MEAS**ure**:DATAT?**

Example:
```
char data[2048]
Send(0,5,":MEASure:DATAT?",15,EOI);
Receive(0,5,data,205,EOI);
```

10-5 EVENT

The **EVENT** query command returns the current value (0 - 2,147,483,647) of the event counter after a **:SYST:EVEN** command has been performed. See Section 5-23 for more information on the **:SYST:EVEN** command.

Command syntax- None

Query syntax- :MEASure**:EVEN**t**?**

Example: `Send(0,5,":MEASure:EVENt?",15,EOI);`

10-6 JITTER

The **JITTER** query returns the standard deviation of the selected sample size.

Command syntax- None

Query syntax- :MEASure:JITTer**?**

Example: `Send(0,5,":MEASure:JITTer?",16,EOI);`
Response: <ASCII floating point>
Example: `+7.3441603e-012`

10-7 MAX

The **MAX** query command returns the maximum measured value of a set of measurements.

Command syntax- None

Query syntax- **:MEAS**ure**:MAX?**

Example: `Send(0,5,":MEASure:MAX?",13,EOI);`
Response: <ASCII floating point>
Example: `-6.5307617e-011`

10-8 MIN

The **MIN** query command returns the minimum measured value of a set of measurements.

Command syntax- None

Query syntax- **:MEAS**ure**:MIN?**

Example: `Send(0,5,":MEASure:MIN?",13,EOI);`
Response: <ASCII floating point>
Example: `-1.1169434e-010`

10-9 RANGE

The **RANGE** query command returns the plus or minus difference between the maximum and minimum values of a set of measurements.

Command syntax- None

Query syntax- **:MEAS**ure**:RANG**e**?**

Example: `Send(0,5,":MEASure:RANGe?",15,EOI);`
Response: <ASCII floating point>

10-10 SDEVIATION

The **SDEVIATION** query returns the standard deviation of the selected sample size.

Command syntax- None

Query syntax- **:MEAS**ure**:SDEV**iation**?**

Example: `Send(0,5,":MEASure:SDEViation?",20,EOI);`
Response: <ASCII floating point>
Example: `+7.3441603e-012`

10-11 STAT(istics)4

The **STAT4** query returns statistical data defined by :SYST:STAT for multiple SETS of measurements as float. The :SYST:STAT/ON command must be executed prior to using the **STAT4** command. Statistics are always returned in the order of AV, JI, MN and MX, depending on which ones are selected.

Command syntax- **:MEAS**ure**:STAT4?**

Example: `Send(0,5,":MEASure:STAT4?",15,EOI);`

Response: <4-Byte float (Intel)>
Example: <ON|OFF><AV><JI><MN><MX>

Query syntax- None

DC MEASUREMENT - Single

10-12 DCVLEVEL

The **DCVLEVEL** command returns the dc voltage measured on the selected input channel. The returned value is an ASCII string of five digits preceded by a (+) or (-) sign. The value is a signed integer with 100 microvolt resolution.

Command syntax- :MEASure**:DC**vlevel**?**

Example: `Send(0,5,":MEASure:DCvlevel?",18,EOI);`
Response: `-1.1444092e-004`

Query syntax- None

10-13 STROBEVLEVEL

The **STROBEVLEVEL** query returns the strobed dc voltage measured on the input channel selected. The strobing is provided through the arming channel. The strobing arm point can be controlled by the strobe delay or by external moving the arming signal.

The returned value is an ASCII string of five (5) digits preceded by a (+) or (-) sign. The value is a signed integer with 100 microvolt resolution.

To perform a strobed measurement, set up the following parameters:

STRObe **CHAN**nel

STRObe **ARM**ing

STRObe **DEL**ay

Command syntax- None

Query syntax- :MEASure**:STRO**be**VLEV**el**?**

Example:
```
Send(0,5,":SYSTem:STRObeCHANnel1",22,EOI);
Send(0,5,":SYSTem:STRObeARMARM1",21,EOI);
Send(0,5,":SYSTem:STRObeDELay25000",24,EOI);
Send(0,5,":MEASure:STRObeVLEVel?",21,EOI);
Receive(0,5,voltage level,5,EOI);
```

Response: <ASCII floating point>
Example: `-2.1731481e-003`

To perform multiple measurements that are averaged, see the **:MEAS**ure**:WIND**ow command.

DC MEASUREMENT - Multiple

10-14 VDATA

The **VDATA** query returns the voltage measurement points acquired in the previous measure window or acquire window command. The measured data values can be analyzed or used to provide a presentation.

Each voltage value is returned in 5 digits preceded by a (+) or (-) sign. The returned voltage is an ASCII integer string of 100 microvolt resolution.

Example: `+0.0001` (+100 uv) would be +1
`-1.0` (-1 v) would be -1000

The returned data stream is of the following format:

:MEASure**:VDATA**# xy...ddddddddd...
x = an ASCII digit representing the number of digits in y
y = an ASCII string of digits, of x length, which represents the number of bytes of information to be returned.

Example of returned data:

:MEASure**:VDATA**#210<60 bytes of data (10 measurements - 10x6)>

Command syntax- None

Query syntax- :MEASure:**VDATA?**

```
Example: char data [2048];
         Send(0,5,":MEASure:VDATA?",15,EOI);
         Receive(0,5,data,60,EOI);
```

10-15 VDATA4

The **VDATA4** query is the same as the VDATA command except that the data is returned as float for throughput. (See VDATA, Section 10-14.) See Appendix C for returned formats..

Command syntax- None

Query syntax- :MEASure**:VDATA4?**

```
Example: Send(0,5,":MEASure:VDATA4?",16,EOI);
```

10-16 VMAXIMUM

The **VMAXIMUM** query returns the maximum voltage value measured in the previous measure window or acquire window command.

Command syntax- None

Query syntax- :MEASure**:VMAX**imum**?**

```
Example: Send(0,5,":MEASure:VMAXimum?",18,EOI);
         Receive(0,5,voltage level,5, EOI);
```

Response: <Signed ASCII value>
Example: `-0.00758`

10-17 VMINIMUM

The **VMINIMUM** query returns the minimum voltage value measured in the previous measure window or acquire window command.

Command syntax- None

Query syntax- **:MEAS**ure**:VMIN**imum**?**

Example:
```
Send(0,5,":MEASure:VMINimum?",18,EOI);
Receive(0,5,voltage level,5,EOI);
```
Response: <Signed ASCII value>
Example: `-0.00821`

10-18 VSDEVIATION

The **VSDEVIATION** query returns the voltage standard deviation of the previous measure window or acquire window command. The returned value is a 6-digit ASCII string of a decimal number.

Command syntax- None

Query syntax- **:MEAS**ure**:VSDEV**iation**?**

Example:
```
char data [10];
Send(0,5,":MEASure:VSDEViation?",21,EOI);
Receive(0,5,data,7,EOI);
```
Response: <Signed ASCII value>

10-19 WINDOW

The **WINDOW** query instructs the instrument to take a series of strobed voltage measurements and then returns the average (mean) voltage. The following statistics are also available upon completion of the command.

VMAXIMUM..... Maximum voltage measured
VMINIMUM ..... Minimum voltage measured
VSDEVIATION ..... Standard deviation of voltages measured

The following parameters must be set up prior to sending a measure window query:

STRObe **CHAN**nel..... Select channel to be strobed
STRObe **ARM**ing channel ..... Select strobing (arming) input
STRObe **STAR**ting point delay ..... Set delay for the first strobed point
STRObe **STOP**ping point delay ..... Set delay for the last strobed point
STRObe **INCR**ement between points .... Set increment between strobed points. (the instrument will calculate the number of points between start and stop)
MEASure **WIN**dow**?** ..... Takes measurements and returns average

NOTE: Strobe increment defines a delay between each measurement. The instrument determines how many points to measure. An alternate method is to define the number of points between the first and last delayed points (**:SYST**em:**STRO**be**#**) and the instrument will determine the delay increment between measured points.1

```
Example: Send(0,5,":SYSTem:STRObeCHANnel1",22,EOI);
         Send(0,5,":SYSTem:STRObeARMARM1",21,EOI);
         Send(0,5,":SYSTem:STRObeSTArt25000",24,EOI);
         Send(0,5,":SYSTem:STRObeSTOP50000",23,EOI);
         Send(0,5,":SYSTem:STRObeINCRememt1000",27,EOI);
         Send(0,5,":MEASure:WINdow?",16,EOI);
         Receive(0,5,voltage level,6,EOI);
```

To measure a single strobe point, see the STROBEVLEVEL command.

To use a macro type of command to set up delays, take the measurements and return the voltage average, see the **:ACQ**uire**:WIN**dow command.

Command syntax- None

Query syntax- :MEASure**:WIND**ow**?**

```
Example: Send(0,5,":MEASure:WINDow?",16,EOI);
         Receive(0,5,data,5,EOI);
```

Response: <Signed ASCII value>

```
Example: -0.00758
```

SECTION 11 - TRIGGER COMMANDS

11-1 DESCRIPTION OF TRIGGER COMMANDS

The **TRIGGER** commands control the source and the level of the arming signal.

:TRIGger:<command syntax>

Trigger commands: **LEV**el
MINimum/**MAX**imum
SEQuence (start/stop/startfirst)
Arm **SLOP**e (Edge)
SOURce (external/manual/automatic)

11-2 LEVEL

The **LEVEL** command sets the trip level of the arming input (ARM1 or ARM2). The levels that can be selected are ±1.11 volts.

The **LEVEL** query returns the present trip setting of the specific arming input. The value is a 5-digit ASCII floating point number.

Command syntax- **:TRIG**ger:**LEV**el<ARM1|ARM2|GATe><ASCII value>

Example: `Send(0,5,":TRIGger:LEVelARM1 0.1",21,EOI);`

Query syntax- **:TRIG**ger:**LEV**el<ARM1|ARM2|GATe>?

Example: `Send(0,5,":TRIGger:LEVelARM2?",19,EOI);`
Response: `<ARM1|ARM2|GATE>`
Example: `+1.11000`

11-3 MINIMUM/MAXIMUM

The **MINIMUM/MAXIMUM** query returns the minimum or maximum peak levels of the ARM1 or ARM2 reference levels. The peak values were measured when the last pulse find was initiated. This pulse find could have been initiated from the front panel or with the acquire level command.

Command syntax- None

Query syntax- **:TRIG**ger:<MAXimum|MINimum><ARM1|ARM2>

Example: `Send(0,5,":TRIGger:MINimumARM1?",17,EOI);`
Response: <ASCII value>
Example: `+1.00123`

11-4 SEQUENCE

The **SEQUENCE** command selects the arming sequence between the START and STOP path. The three sequences are:

Arm on start
Arm on stop
Arm on startfirst

The **SEQUENCE** query returns the presently selected arming sequence.

Command syntax- **:TRIG**ger**:SEQ**uence<STARt|STOp|STARTFIRST>

Example: `Send(0,5,":TRIGger:SEQuenceSTARt",22,EOI);`

Query syntax- **:TRIG**ger**:SEQ**uence**?**

Example: `Send(0,5,":TRIGger:SEQuence?",18,EOI);`
Response: `<Start|Stop|Startfirst>`

11-5 SOURCE

The **SOURCE** command selects the arming signal that will initiate a measurement.

The **SOURCE** query returns the presently selected arming signal source.

The 3 source selections are:

<EXTernal|**MAN**ual|**AUT**omatic>

Command syntax- **:TRIG**ger**:SOUR**ce<EXTernal|MANual|AUTomatic>

Example: Send (0,5,":TRIGger:SOURceEXTernal",23,EOI);

Query syntax- **:TRIG**ger**:SOUR**ce**?**

Example: Send (0,5,":TRIGger:SOURce?",16,EOI);
Response: `<EXT|MAN|AUT>`

11-6 SLOPE

The **SLOPE** command sets the edge of a specific arming input (ARM1 or ARM2). This edge can be a positive going (rising) edge or a negative going (falling) edge. When ARM2 is used as a gate signal it can also be set to a high or low.

The **SLOPE** query returns the present setting of the specific external edge.

Command syntax- **:TRIG**ger**:SLOP**e<ARM1|ARM2|GATe><POS|NEG|RISe|FALl|High|Low>

Example: Send(0,5,":TRIGger:SLOPeARM1RIS",21,EOI);

Query syntax- **:TRIG**ger**:SLOP**e**ARM2?**

Example: Send(0,5,":TRIGger:SLOPeARM2?",19,EOI) ;
Response: `<ARM1|ARM2><NEG|POS>`

SECTION 12 - HARDCOPY COMMANDS

12-1 DESCRIPTION OF HARDCOPY COMMANDS

The **HARDCOPY** commands allow the printing of instrument configuration and statistics of the most recent measurement:

:HARDcopy:<command syntax>

Hardcopy commands:
HISTogram
PART
SERial
SERial **DECR**ement
SERial **INCR**ement
SETup
STATistics

12-2 HISTOGRAM

The **HISTOGRAM** command causes the instrument to print out the histogram of the previous measurement.

Command syntax- :HARDcopy:**HIST**ogram

Example: `Send(0,5, ":HARDcopy:HISTogram",19,EOI);`

Query syntax- None

12-3 PART

The **PART** number command allows an alpha-numeric string to be downloaded from a host to the instrument. This string will be printed at the top of form for all instrument printouts. The ASCII string can have a maximum of 10 digits.

Command syntax- :HARDcopy:**PART**<alpha-numeric string>

Example: `Send(0,5, "HARDcopy:PART745243:,20,EOI);`

Query syntax- :HARDcopy:**PART?**

Example: `Send(0,5, ":HARDcopy:PART?",15,EOI);`
Response: <ASCII string>
Example: `"DTS-2075#3120"`

12-4 SERIAL

The **SERIAL** command allows a numeric serial number to be downloaded from a host to the instrument. This number will be printed at the top of the form for all instrument printouts.

The number is a ASCII integer up to a maximum of 10 digits.

There are three forms of the **SERIAL** command:

:HARDcopy:**SER**ial<integer/**INCR**ement/**DECR**ement>

The **SERIAL** query returns a ASCII integer number.

Command syntax- :HARDcopy:**SER**ial<integer>

Example: `Send(0,5 ":HARDcopy:SERial1000",20,D\,EOI);`

Command syntax- :HARDcopy**:SER**ial**INCR**ement

Example: `Send(0,5 ":HARDcopy:SERialINCRement",25,EOI);`

Command syntax- :HARDcopy**:SER**ial**DECR**rement

Example: `Send(0,5 ":HARDcopy:SERialDECRement",25,EOI);`

Query syntax- :HARDcopy**:SER**ial**?**

Example: `Send(0,5, ":HARDcopy:SERial?",17,EOI);`
Response: <ASCII serial number>
Example: `32516`

12-5 SETUP

The **SETUP** command causes the instrument to print out the current selected configuration.

Command syntax- :HARDcopy**:SET**up

Example: `Send(0,5 ":HARDcopy:SETup".15,EOI);`

Query syntax- None

12-6 STATISTICS

The **STATISTICS** command causes the instrument to print out the measurement statistics of the previous measurement.

Command syntax- :HARDcopy**:STAT**istics

Example: `Send(0,5, ":HARDcopy:STATistics",20,EOI);`

Query syntax- None

APPENDIX A - INTERNAL, EXTERNAL & STROBE CALIBRATION

This appendix describes all the programming steps to perform an Internal, External and Strobe calibration from a host computer.

INTERNAL

The events required to perform an internal calibration:

Send the command to start internal calibration.
Respond with GO for message, “Press GO within 5s to continue”.
Wait for TRG bit of a serial poll to indicate completion.
Check DDE bit of the ESR register to determine if an error occurred.

The code to implement the above events follows:

```
int perform_int_calibration (void)
   {
   char event_status;
   char poll_status;
   Send(0,5,”*CLS”,4,EOI);
   Send(0,5,”:CAL:INT”,8,EOI);
   respond_to_go_request(1);   /*continue int cal*/
   poll_status = 0;
   while ((poll_status&TRG) ==0)   {
      ReadStatusByte(0,5,&poll_status);
      }
   Send(0,5,”*ESR?”,5,EOI);
   Receive(0,5,event_status,1,EOI);
   if((event_status& DDE)!=Ø)
      {
      printf(“InternalCalibrationFailed”);
      return(-1);
      }
   else
      {
      printf(“InternalCalibrationPassed”);
      return(0);
      }
   }
```

EXTERNAL

The events required to perform an external calibration are:

Send command to start external calibration.
Respond with GO for message, “Press GO within 5s to continue”.
Respond with NOGO for message, “Press GO within 5s to peform DC calibration phase”.
Respond with GO for message, “Connect Ch 1 to CAL 1”.
“Connect Ch 2 to CAL 2, Press GO”.
Respond with GO for message, “Cross Cable Connections At”
“CAL Signal Connectors, Press GO”.
Wait for TRG bit of a serial poll to indicate completion.
Check DDE bit of the ESR register to determine if an error occurred.

The code to implement the above events follows:

```
   int perform_ext_calibration(void)
      {
      char event_status;
      char poll_status;
      Send(0,5,”*CLS”,4,EOI);
      Send(0,5,”:CAL:EXT”,8,EOI);
      respond_to_go_request(1);  /*continue ext cal*/
      respond_to_go_request(0);  /*do not perform DC cal*/
      respond_to_go_request(1);  /*continue after con-
necting cables*/
      respond_to_go_request(1);  /*continue after crossing
cables*/
      poll_status = 0;
      while ((poll_status&TRG) ==0)  {
         ReadStatusByte(0,5,&poll_status);
         }
      Send(0,5,”*ESR?”,5,EOI);
      Receive(0,5,event_status,1,EOI);
      if((event_status& DDE)!=Ø))
         {
         printf(“External Calibration Failed”);
         return(-1);
         }
      else
      {
         printf(“External Calibration Passed”);
         return(0);
         }
      }
```

```
int respond_to_go_request (int go)
   {
   char poll_status;
   char event_status;
   event_status = 0;
   while((event_status&RQC) ==0)  {
      poll_status=0;
      while((poll_status&ESB) ==0)  {
         ReadStatusByte(0,5,&poll_status);
         }
      Send(0,5,"*ESR?",5,EOI);
      Receive(0,5,event_status,1,EOI);
      }
   if (go)
      Send(0,5,":SYST:GO",8,EOI);
   else
      Send(0,5,":SYST:NOGO"10,EOI);
   return(0);
   }
```

STROBE

The events required to perform a strobe calibration are:

Send the command to start strobe calibration.
Respond with GO for message, "Connect CAL1 to CH1 and CAL2 to ARM1".
Message displayed "StrobeCal in process – Please Wait".
Respond with GO for Message, "Move CAL2 from ARM1 to ARM2".
Respond with GO for Message, "Move CAL1 from CH1 to CH2".
Wait for TRG bit of a serial poll to indicate completion.
Check DDE bit of the ESR register to determine if an error occurred.

The code to implement the above events is:

```
int perform_strobe_calibration (void)
     {
     char event_status;
     char poll_status;
     Send(0,5,"*CLS",4,EOI);
     Send(0,5,":SYST:STROCAL",13,EOI);
     respond_to_go_request(1);   /*Connect cables*/
     respond_to_go_request(1);   /*Move cable to ARM2*/
     respond_to_go_request(1);   /*Move cable to CH2*/
     poll_status = 0;
     while ((poll_status&TRG) ==0)   {
        ReadStatusByte(0,5,&poll_status);
        }
```

```
   Send(0,5,"*ESR?",5,EOI);
   Receive(0,5,event_status,1,EOI);
   if((event_status& DDE)!=Ø)
      {
   printf("Strobe Cal Failed");
   return(-1);
   }
else
   {
   printf("Strobe Cal Passed");
   return(0);
   }
}
```

APPENDIX B - READING DATA

This appendix describes the programming steps to take a measurement and read back the values of the measurement. In this example a burst of 100 measurements is taken and the data read back in a 32-bit floating format.

```
void main (void)
    {
        int i;
        int no_of_bytes;
        char temp_string[2048];
        int c;
        int header;
        float this_reading[100];
        char *ptr;
        int result;
/*488 is initialized for controller and instrument*/
    no_of_bytes=100;
    Send(0,5,"*CLS",4,EOI);
    Send(0,5,":ACQ:COUN100",12,EOI)
    Send(0,5,"*TRG",4,EOI);
    result=0;
    while((result & 0x01) ==0)              /*Wait for TRG bit*/
        {
        ReadStatusByte(0,5,&result);
        }
    sprintf(temp_string,"%i",number_of_bytes*4);
    c=strlen(temp_string);
    header=c+2;             /*# of characters in header*/
    Send(0,5,":MEAS:DATA4?",12,EOI);
    result=0;
    while((result & 0x10) ==0)              /*Wait for MAV bit*/
        {
        ReadStatusByte(0,5,&result);
        }
    Receive(0,5,temp_string,(no_of_bytes*4)+header,EOI);      /*convert char string to
    floating point*/
    ptr=&temp_string [header];
    for(i=0; i<no_of bytes; i++)
        {
        this_reading [i] = *((float*)ptr);
        ptr = ptr+4;
        }
    }                                       /*end of main*/
```

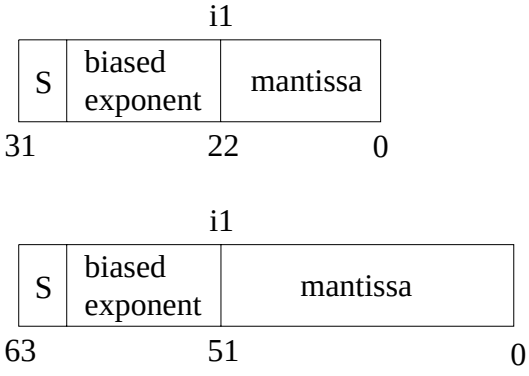
APPENDIX C - DATA TYPES

This appendix describes the data formats used for transferring data from the DTS unit over the GPIB for **:MEASure** commands.

The **:MEASure:DATA, :MEASure:DATA4 and :MEASure:DATAT** queries support two sizes of data types using IEEE standards for floating-point arithmetic (ANSI/IEEE Std. 754-1985):

Type	Size (bits)	Smallest Absolute value	Largest Absolute value	Number of Digit Accuracy
float	32	1.1×10^{-38}	3.4×10^{38} Scientific	6-digit precision
double	64	2.2×10^{-308}	1.7×10^{308} Scientific	15-digit precision

Data Representation:



s = Signbit (0 = positive, 1 = negative)
i = Position of implicit binary point (always 1)
1 = integer bit of mantissa
Exponent bias (normalized values)
float: 127(7FH)
double: 1023(3FFH)

WAVECREST Corporation

World Headquarters:
7626 Golden Triangle Drive
Eden Prairie, MN 55344
(952) 831-0030
FAX: (952) 831-4474
Toll Free: 1-800-733-7128
www.wavecrestcorp.com

West Coast Office:
1735 Technology Drive, Suite 400
San Jose, CA 95110
(408) 436-9000
FAX: (408) 436-9001
1-800-821-2272

Europe Office:
Lilenthalalle 25
D-80939 Munchen
011-49-89-32225330
FAX: 011-49-89-32225333