Digital Storage Oscilloscope

IDS-700 & IDS-800 Series

Programming Manual

Due to continuous improvements in the IDS-700 & 800 series Digital Storage Oscilloscopes, information contained in this manual is subject to change without notice. Contact RS Components for revisions and corrections.

This document supports firmware version v1.07 and previous version

RS Components Ltd.

PO Box 99, Corby,

Northants., NN17 9RS

United Kingdom.

Tel: +44 (0) 1536 201234

Fax: +44 (0) 1536 405678

Internet: rswww.com

1. Introduction

The IDS-820/IDS-840 Digital Storage Oscilloscope can be driven using the GPIB (General Purpose Interface Bus) or the RS-232 serial connection system with a computer. Commands sent over either interface can read or set any IDS-820/840's instruction. This chapter explains how to carry out the following tasks.

Note: Not all interface options are available on all models – refer to individual product specifications for applicable options.

Notes for GPIB installation

If you are setting up the IDS-820/IDS-840 with a GPIB system, please comply with the following requirements:

- ❖ Only a maximum of 15 devices can be connected to a single GPIB bus.
- ❖ Do not use more than 20 m of cable to connect devices to a bus.
- ❖ Connect one device for every 2 m of cable used.
- ❖ Each device on the bus needs a unique device address. No two devices can share the same device address.
    - ❖ Turn on at least two-thirds of the devices on the GPIB system while you use the system.
- ❖ Do not use loop or parallel structure for the topology of the GPIB system.

Notes for RS-232 Configuration

This oscilloscope contains a DB 9-pin, male RS-232 connector for serial communication with a computer or terminal. The RS-232 interface of this oscilloscope is configured as an RS-232 "Data Terminal Equipment", so that data is sent from pin 3 and received on pin 2. For remote controls, the RS-232 interface has to be connected with a computer or terminal.

Pin Assignments

The pin assignments for RS-232 interface of the IDS-820/IDS-840 series are listed below.

1   No connection
2   Receive Data(RxD) (input)
3   Transmit Data(TxD) (output)
4   No connection
5   Signal Ground (GND)
6   No connection
7   No connection
8   No connection
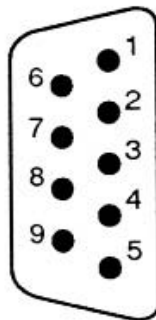9   No connection

Figure 1-1: Pin assignments for the IDS-820's RS232 connector

DB9 to DB9 Wiring

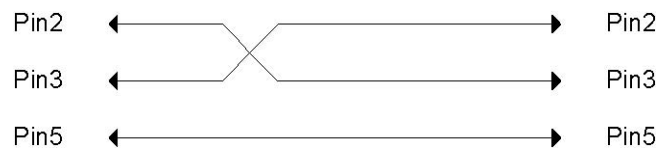The wiring configuration is used for computer with DB9 connectors that configured as Data Terminal Equipment.

Figure 1-2: DB9 to DB9 wiring

When the oscilloscope is set up with an RS-232 interface, please check the following points:

◆ Do not connect the output line of one DTE device to the output line of the other.

◆ Many devices require a constant high signal on one or more input pins.

◆ Ensure that the signal ground of the equipment is connected to the signal ground of the external device.

◆ Ensure that the chassis ground of the equipment is connected to the chassis ground of the external device.

◆ Do not use more than 15m of cable to connect devices to a PC.

◆ Ensure the same configurations are used on the device as the one used on PC terminal.

◆ Ensure the connector for the both side of cable and the internal connected line are met the demand of the instrument.

Computer's Connection for RS-232

A personal computer with a COM port is the essential facilities in order to operate the digitizing oscilloscope via RS-232 interface.

The connections between oscilloscope and computer are as follows:

I. Connect one end of a RS-232 cable to the computer.

II. Connect the other end of the cable to the RS-232 port on the oscilloscope.

III. Turn on the oscilloscope.

IV. Turn on the computer.

Figure 1-3, shows the GPIB port and RS-232 port's locations on the rear panel of the IDS-820/IDS-840 Digital storage Oscilloscope.
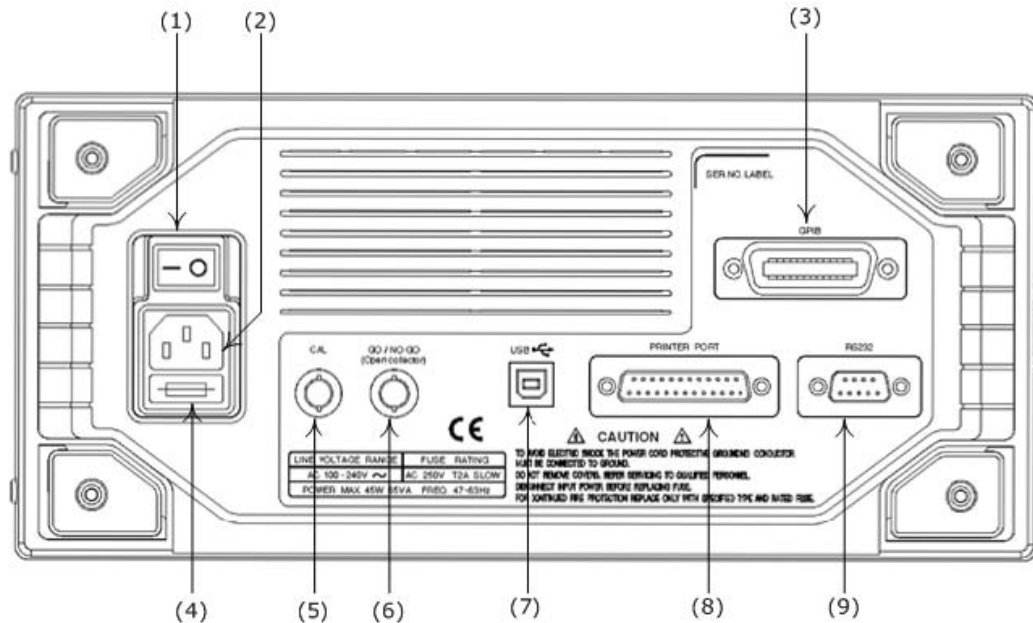
Figure 1-3. Rear Panel of IDS-820/840

(1): Main power switch   (6): "GO/NO GO" BNC output

(2): AC power socket   (7): USB connector

(3): GPIB port (option)   (8): Printer port

(4): Fuse drawer   (9): RS-232 port

(5): "SELF CAL" BNC output

2. Computer Connection

A personal computer with a GPIB card is the essential stuff in order to operate the IDS-820/IDS-840 via GPIB interface.

Connect between IDS-820/IDS-840 Digital storage Oscilloscope and computer as follows:

I. Connect one end of a GPIB cable to the computer.

II. Connect the other end of the GPIB cable to the GPIB port on the IDS-820/IDS-840 Digital storage Oscilloscope.

III. Turn on the IDS-820/IDS-840 Digital storage Oscilloscope.

IV. Turn on the computer.

The GPIB interface capabilities:

The GPIB interface of the IDS-820/IDS-840 corresponds to the standard of IEEE488.1-1987, IEEE488.2-1992 and SCPI-1994. The GPIB interface functions are listed as follows:

SH1(Source Handshake):	The IDS-820/IDS-840 can transmit multilane messages across the GPIB.
AH1(Acceptor Handshake):	The IDS-820/IDS-840 can receive multilane messages across the GPIB.
T6(Talker):	Talker interface function includes basic talker, serial poll, and unaddress if MLA capabilities, without talk only mode function.
L4 (Listener):	The IDS-820/IDS-840 becomes a listener when the controller sends its listen address with the ATN (attention) line asserted. The IDS-820/IDS-840 does not have listen only capability.

SR1 (Service Request): The IDS-820/IDS-840 asserts the SRQ (Service request) line to notify the controller when it requires service.

RL1 (Remote/Local): The IDS-820/IDS-840 responds to both the GTL (Go to Local) and LLO (Local Lock Out) interface messages.

PP0 (Parallel Poll): The IDS-820/IDS-840 has no Parallel Poll interface function.

DC1 (Device Clear): The IDS-820/IDS-840 has Device clear capability to return the device to power on status.

DT0 (Device Trigger): The IDS-820/IDS-840 has no Device Trigger interface function.

C0 (Controller) : The IDS-820/IDS-840 can not control other devices.

GPIB address setting

To change the GPIB address, use the following steps:

� Press the UTILITY button on the front panel. The utility menu provides

Interface Menu sub-menu by pressing the F2 soft-key in which the GPIB sub menu

is included. Press the F1 soft-key to select the GPIB setting menu.

<u>For GPIB sub menu</u>

◆ Type GPIB: Select GPIB port.

◆ Addr 1~30: select the appropriate address for GPIB.

◆ Previous Menu: back to previous menu.

GPIB connection testing

If you want to test the whether the GPIB connection is working or not, use the National Instrument's "Interactive Control utility" for instance, you communicate with the GPIB devices through calls you interactively type in at the keyboard.

The Interactive Control can help you to learn about the instrument and to troubleshoot problems by displaying the following information on your screen after you enter a command:

◆ Results of the status word (ibsta) in hexadecimal notation

◆ Mnemonic constant of each bit set in ibsta

◆ Mnemonic value of the error variable (iberr) if an error exists (the ERR bit is set in ibsta)

◆ Count value for each read, write, or command function

◆ Data received from your instrument

You can access online help in Interactive Control by entering help at the prompt, or you can get help on a specific function by entering help <function> at the prompt, where <function> is the name of the function for which you want help.

To start Interactive Control within National Instrument's "Measurement & Automation Explorer", complete the following steps:

1 Select Tools→I-488.2 Utilities→Interactive Control.
2 Open either a board handle or device handle to use for further NI-488.2 calls. Use ibdev to open a device handle, ibfind to open a board handle, or the set 488.2 command to switch to a 488.2 prompt.

The following example uses ibdev to open a device, assigns it to access board gpib0, chooses a primary address of 7 with no secondary address, sets a timeout of 10 seconds, enables the END message, and disables the EOS mode.

:ibdev

    enter board index: 0

    enter primary address: 7

    enter secondary address: 0

    enter timeout: 13

    enter 'EOI on last byte' flag: 1

    enter end-of-string mode/byte: 0

ud0:

Note: If you type a command and no parameters, Interactive Control prompts you for the necessary arguments. If you already know the required arguments, you can type them at the command prompt, as follows:
:ibdev 0 7 0 13 1 0 ud0:

Note: If you do not know the primary and secondary address of your GPIB instrument, right-click on your GPIB interface in Measurement & Automation Explorer and select Scan for Instruments. After Explorer scans your interface, it displays your instrument address in the right window panel.

3. After you successfully complete ibdev, you have a ud prompt. The new prompt, ud0, represents a device-level handle that you can use for further NI-488.2 calls. To clear the device, use ibclr, as follows:

ud0: ibclr [0100] (cmpl)
       To write data to the device, use ibwrt.ud0: ibwrt "*IDN?"[0100] (cmpl)count: 5
       To read data from your device, use ibrd. The data that is read from
theinstrument is displayed. For example, to read 28 bytes, enter the following:

ud0: ibrd 28

[0100] (cmpl) count: 28

47 57 2C 20 47 44 53 2D	, IDS-
38 32 30 2C 20 50 39 32	820, P92
30 31 33 30 2C 20 56 2E	0130, V.
31 2E 30 39	1.09

6. When you finish communicating with the device, make
sure you put it offline using the ibonl command, as follows:

ud0: ibonl 0 [0100] (cmpl)

: The ibonl command properly closes the device handle and the ud0 prompt is no
longer available.

7. To exit Interactive control, type q.

For the details, please refer to National Instrument's manual.

If you do not receive a proper response from the IDS-820/IDS-840, please check the
power is on, the GPIB address is correct, and all cable connections are active,

RS232 connection testing

If you want to test whether the RS-232 connection is working or not, you can send a
command from computer. For instance, using a terminal program send the query
command

*idn?

should return the Manufacturer, model number, serial number and firmware version in
the following format:

RS ,IDS-820,P920130,V.1.09

If you do not receive a proper response from the oscilloscope, please check if the
power is on, the RS-232 configurations are the same on both sides, and all cable
connections are active.

3. Remote Control's Commands

The IDS-820/IDS-840 can be operated from computer via the GPIB port or RS-232 port. The remote commands of IDS-820/IDS-840 are compatible with IEEE-488.2 and SCPI standards partially.

SCPI

SCPI (Standard Commands for Programmable Instruments) is a standard created by an international consortium of the major test and measurement equipment manufacturers. The IEEE-488.2 syntax has been adopted by SCPI to furnish common commands for the identical functions of different programmable instruments.

SCPI

Common Command & Queries

Syntax & Status Data Structure

Interface Function

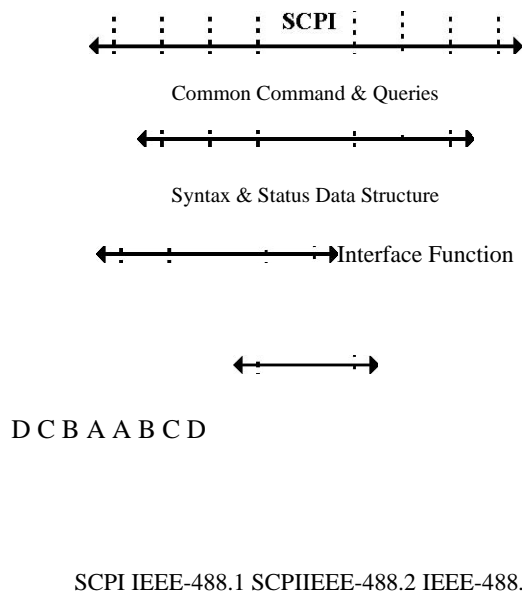D C B A A B C D

SCPI IEEE-488.1 SCPIIEEE-488.2 IEEE-488.2

Figure 3-1: the relationship between IEEE-488.1, IEEE-488.2, and SCPI.

As shown in the figure 3-1, the IEEE-488.1 standard is located at layer A and layer A belongs to the protocol of interface function on the GPIB bus. The source handshake (SH), acceptor handshake (AH) and talker are included in this layer (10 interface functions in total). At layer B, the syntax and data structure are the essence of the entire IEEE 488.2 standard. The syntax defines the function of the message communication, which contain the <PROGRAM MESSAGE> (or simply "commands") and <RESPONSE MESSAGE>. The two kinds of messages represent the syntax format of a device command and return value. The data structure is the foundation of

the status reporting, which IEEE-488.2 standard have been defined. The common commands and queries are included in layer C. Commands and queries can be divided into two parts: mandatory and optional. Commands modify control settings or tell the instrument to perform a specific action. Queries cause the instrument to send data or status information back to the computer. A question mark at the end of a command identifies it as a query. Layer D is interrelated with device information. Different devices have different functions. SCPI command sets belong to this layer.

Command Syntax

If you want to transfer any of the instructions to an instrument and comply with SCPI, there are three basic elements which must be included as follows.

  ◆ Command header
  ◆ Parameter (if required)
  ◆ Message terminator or separator

Command Header

The command header has a hierarchical structure that can be represented by a command tree (Figure 3-2).
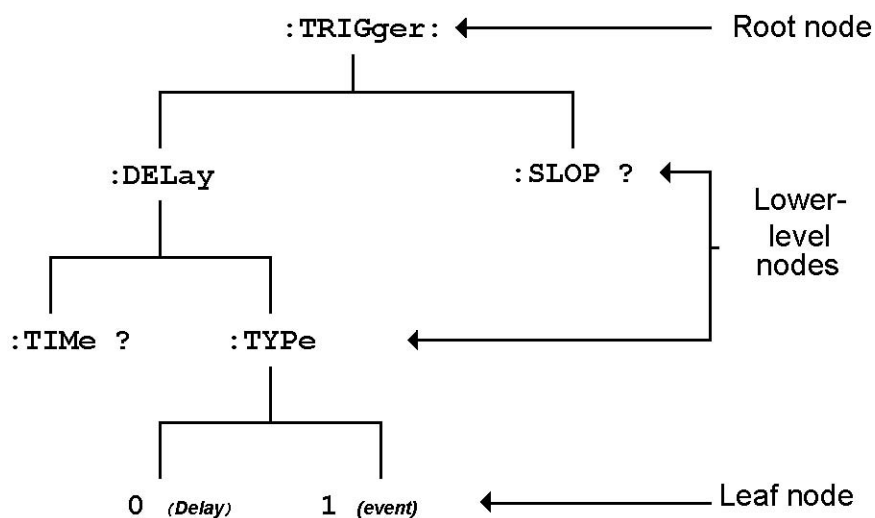

Figure 3-2: Tree hierarchy

The top level of the tree is the root level. A root node is located at the root level. A root node and one or more lower-level nodes form a header path to the last node called the leaf node.

The command header is configured by header path and leaf node. Figure 3-3 shows the command header for the leaf node.

:TRIGger:DELay:EVENt ?

Root Node

Header Path   Leaf Node
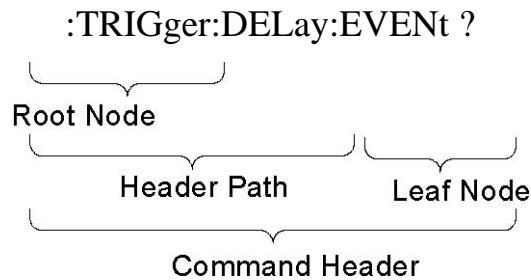
Command Header

Figure 3-3: Command header

Parameter

If the command has parameters, the values have to be included. In this manual, when we expressed the syntax of the command, the < > symbols are used for enclosing the parameter type. For instance, the syntax of the command in Figure 8-5 includes the Boolean parameter type

NOTE: Do not include the <, >, or | symbols when entering the actual value for a parameter.

:TRIGger:DELay:MODe <Boolean>

Parameter
Type

Space

Figure 3-4: Command Header with Parameter

Table 3-1 defines the Boolean and other parameter types for the IDS-820/IDS-840.

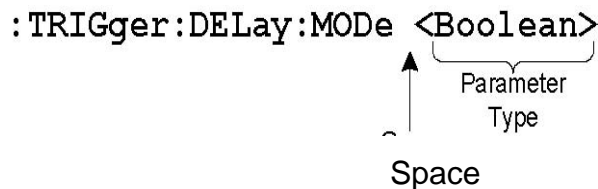| Parameter Type	Description	Example
Boolean	Boolean numbers or values	1 0
NR1	Integers	0, 1, 18
NR2	Decimal numbers	1.5, 3.141, 8.4
NR3	Floating point numbers	4.5E-1, 8.25E+1
NRf	NR1, NR2, or NR3	1, 1.5, 4.5E-1

Table 3-1: Parameter Types for Syntax Descriptions

For the actual value of the parameter type <Boolean>, you have to enter 0 instead of "OFF" or enter 1 instead of "ON".

The following example includes both the header and a value for the parameter type:

:TRIGger:NREJ 0

Parameter values that appear in this manual are usually separated by a vertical line. This vertical line has the same meaning as the word "or," For example, values for the parameter <Boolean> are

0|1

This is the same as saying "0 (off) or 1 (on)" Any single value is a valid parameter.

Message Terminator and Message Separator

In accordance with the IEEE 488.2 standard, the IDS-820/IDS-840 accepts any of the following message terminators:

- � LF$^\wedge$END Line feed code (hexadecimal 0A) with END message
- � LF Line feed code
- � <dab>$^\wedge$END Last data byte with END message

These terminators are compatible with most application programs. A semicolon separates one command from another when the commands appear on the same line.

Entering Commands

The standards that govern the command set for the IDS-820/IDS-840 allow for a certain amount of flexibility when you enter commands. For instance, you can abbreviate many commands or combine commands into one message that you send to the IDS-820/IDS-840. This flexibility, called friendly listening, saves programming time and makes the command set easier to remember and use.

Command Characters

The IDS-820/IDS-840 is not sensitive to the case of command characters. You can enter commands in either uppercase or lowercase.

You can precede any command with white space characters. You must, however, use at least one space between the parameter and the command header.

Abbreviating Commands

Most commands have a long form and a short form. The listing for each command in this section shows the abbreviations in upper case. For instance, you can enter the query

:TIMebase:SCALe ?

simply as

:TIM:SCAL ?

Combining Commands

You can use a semicolon (;) to combine commands and queries. The IDS-820/IDS-840 executes coherent commands in the order it receives them. When you coherent queries, the IDS-820/IDS-840 will combine the responses into a single response message. For example, if the frequency and amplitude of the signal are equal to 100kHz and 1V, the command

:MEASure:FREQuency?;:MEASure:VAMPlitude?

returns the message

100kHz 1v

4. Details of Command Reference

Each command in this chapter will be given a brief description. The examples of each command will be provided and what query form might return.

*CLS (no query form)

Clears all event status data register. This includes the Output Queue, Operation Event Status Register, Questionable Event Status Register, and Standard Event Status Register.

Syntax

*CLS

Examples

*CLS clears all event registers.

*ESE

Sets or returns the bits in the Event Status Enable Register (ESER). The ESER enables the Standard Event Status Register (SESR) to be summarized on bit 5 (ESB) of the Status Byte Register (SBR).

Syntax

*ESE<NR1> *ESE?

Arguments

<NR1> is a number from 0 to 255. The binary bits of the ESER are set according to this value.

Returns

<NR1> is a number from 0 to 255 that indicates the decimal value of the binary bits of the ESER.

Examples

*ESE 65 sets the ESER to binary 0100 0001.If the ESER contains the binary value 1000 0010, the *ESE? will return thevalue of 130.

*ESR? (query only)

Returns and clears the contents of the Standard Event Status Register (SESR).

Syntax

∗ESR?

Returns

<NR1> is a number from 0 to 255 that indicates the decimal value of the binary bits of the ESER.

Examples

If the ESER contains the binary value 1100 0110, the ∗ESR? will return the value of 198.

*IDN? (query only)

Returns the unique identification code of the IDS-820/IDS-840.

Syntax

∗IDN?

Examples

∗IDN?

Returns ,IDS-820/IDS-840,0,<Firmware version>

*LRN? (query only)

Returns the string that the IDS-820/IDS-840 settings will be listed.

Syntax

∗LRN?

Returns

```
:ACQuire:ACCumulate 0;AVERage 2;LENGth 500;MODe 1;POINt;:CHANnel1:BWLimit 0;COUP
ling 1;DISPlay 1;INVert 0;MATH 0;OFFSet 1.128e+00;PROBe 0;SCALe 2.000e-01;:CHANn
el2:BWLimit 0;COUPling 1;DISPlay 1;INVert 0;MATH 0;OFFSet 1.214e+01;PROBe 0;SCAL
e 1.000e+00;:TIMebase:DELay 2.540e-03;SCALe 1.000e-03;SWEep 0;WINDow:DELay 2.540
e-03;SCALe 5.000e-07;:TRIGger:COUPle 1;LEVel 1.636e+00;MODe 2;NREJ 0;REJect 0;SL
OP 0;SOURce 0;ADVance:DELay 1.000e-07;EVENt 2;LEVel 0.000e+00;MODe 0;TYPe 0;TV:F
IELd 0;LINe 0;POLarity 0;TYPe 1;:DISPlay:WAVeform 0;BRIGhtness:DATa 40;GRATicule
   40;:MEASure:SOURce 1;FALL 0.000e+00;FREQuency 0.000e+00;NWIDth 0.000e+00;PDUTy
0.00%;PERiod 0.000e+00;PWIDth 0.000e+00;RISe 0.000e+00;VAMPlitude 0.000e+00;VAVe
rage 0.000e+00;VHI 0.000e+00;VLO 0.000e+00;VMAX 0.000e+00;VMIN 0.000e+00;VPP 0.0
00e+00;VRMS 0.000e+00;:MEASure:SOURce 2;FALL 0.000e+00;FREQuency 0.000e+00;NWIDt
h 0.000e+00;PDUTy 0.00%;PERiod 0.000e+00;PWIDth 0.000e+00;RISe 0.000e+00;VAMPlit
ude 0.000e+00;VAVerage 0.000e+00;VHI 0.000e+00;VLO 0.000e+00;VMAX 0.000e+00;VMIN
   0.000e+00;VPP 0.000e+00;VRMS 0.000e+00;:AUToset;:PRINt;:REFResh;:RUN;:STOP
```

The command form (∗OPC) sets the operation complete bit (bit 0) in the

Standard Event Status Register (SESR) when all pending operations finish. The query

form (∗OPC?) tells the oscilloscope to place an ASCII 1 in the Output Queue when the

oscilloscope completes all pending operations.

Syntax

∗OPC ∗OPC?

Returns ∗RCL

Recall the setting data from memory which were previously saved. The settings of the

RS-232 (or GPIB) can be stored in memories M1 to M15. However, if the user recalls a

stored memory in which the settings of the RS-232 or GPIB are different to the present

settings, the RS-232 (or GPIB) settings will remain unchanged. The current RS-232

(or GPIB) settings will not to be affected by the recalled setting of RS-232 (or GPIB) for

the moment.

Syntax

∗RCL <NR1>

Arguments: 1~15

Examples

∗RCL 1 recalls the setting data located at the- first position of memory address.

∗RST (no query form)

Sets all control settings of the oscilloscope to their default values, but does not purge

stored setting.

Syntax

∗RST

*SAV

Saves the setting data to memory.

Syntax

∗SAV <NR1>

Arguments

1~15
Examples

∗SAV 2 saves the setting data to the second position of memory queue.

*SRE

Setup the contents of the Service Request Enable Register (SRER). The query form returns the contents of the SRER. Bit 6 of the SRER is always zero. The bits on the SRER correspond to the bits on the SBR.

Syntax

∗SRE <NR1> ∗SRE?

Arguments

<NR1> is an integer from 0 to 255.

Returns

<NR1>

Examples

∗SRE 7sets bits of the SRER to 0000 0111.If the ∗SRE? returns 0000 0011, the setting

of ∗SRE is 3.

*STB? (query only)

Query of the Status Byte register (SBR) with ∗STB? will return a decimal number representing the bits that are set (true) in the status register.

Syntax

∗STB?

Returns

<NR1>

Examples

∗STB? returns 81 if SBR contains the binary value 0101 0001.

*WAI (no query form)

WAI prevents the programming instrument from executing further commands or queries until all pending operations finish.

Syntax

∗WAI

:ACQuire:AVERage

Select the average number of waveform acquisition. The range for averaging is from 2 to 256 in powers of 2.

Note: Before using this instrument, please apply ":ACQuire:MODe 2" in advance!

Syntax

:ACQuire:AVERage {1|2|3|4|5|6|7|8}

:ACQuire:AVERage?

Arguments

1→Average number is 2   2→Average number is 4

3→Average number is 8   4→Average number is 16

5→Average number is 32   6→Average number is 64

7→Average number is 128   8→Average number is 256

Returns

<NR1>

:ACQuire:LENGth

Select the record length. The IDS-820/IDS-840 provides record
lengths of 500, 1250, 2500, 5000, 12500, 25000, 50000, and 125000.

Syntax

:ACQuire:LENGth {0|1|2|3|4|5|6|7}

:ACQuire:LENGth?

Arguments

$0 \to$ Record length is 500 $1 \to$ Record length is 1250 $2 \to$ Record length is 2500 $3 \to$ Record length is 5000 $4 \to$ Record length is 12500 $5 \to$ Record length is 25000 $6 \to$ Record length is 50000 $7 \to$ Record length is 125000

Returns

<NR1>

:ACQuire:MODe

Select the waveform acquisition mode. There are four different acquisition

modes: sample, peak detection, average and accumulate.

Syntax

:ACQuire:MODe {0|1|2}

:ACQuire:MODe?

Arguments

$0 \to$ Select the sample mode $1 \to$ Select the peak detection mode

$2 \to$ Select the average mode

Returns

<NR1>

Note: Please select the specific acquire mode before implementing any acquisition.
:ACQuire<X>:POINt

Transfer waveform data (500 data-points in total) from the IDS-820/IDS
840. Each point is composed of two bytes (the integer value of 16 bits). The high byte
(MSD) is transferred first.

Syntax

:ACQuire<X>:POINt

Arguments

<X>→Specify the channel number (1|2)

Returns

The string of data is following.

#	Data size digit	Data size	Sample rate	Channel indicator	Waveform data size	Waveform data

#: Begin a transmission of data string.

Data size digit: Indicate the digits of following data string amount (1 digit).

Data size: the amount of current data string (4 digits).

Sample rate: The corresponding sample rate of received waveform data (4

bytes). The sample rate is indicated by a floating point format compatible with IEEE

754 standards.

Channel indicator: Shows the channel which sent the waveform data.

Waveform data size: The indicator of total amount of waveform data (4 bytes).
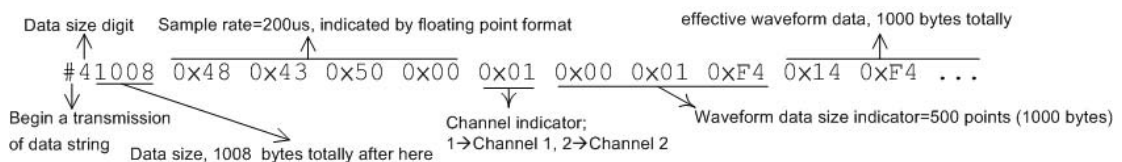
Waveform data: The effective waveform data which covers 500 points (1000bytes) in

total.

Example

Transfer the waveform data of channel 1 at 200μs per division:

:ACQuire1:POINt

The IDS-820/IDS-840 should return the following messages:

:AUToset

Perform an automatic setup in order to optimize the acquisition parameters.

Syntax

:AUToset

:CHANnel<X>:BWLimit

Enable or disable the bandwidth limit function.

Syntax

:CHANnel<X>:BWLimit

{0|1} :CHANnel<X>:BWLimit?

Arguments

<X>→Specify the channel number (1|2)0→Disable bandwidth limit 1→Enable bandwidth

limitReturns <NR1>

:CHANnel<X>:COUPling

Select the different coupling states for IDS-820/IDS-840.

Syntax

:CHANnel<X>:COUPling

{0|1|2} :CHANnel<X>:COUPling?

Arguments

<X>→Specify the channel number (1|2) 0→Place scope in AC coupling state 1→Place

scope in DC coupling state 2→Place scope in grounding state

Returns

<NR1>
:CHANnel<X>:DISPlay

Enable or disable the channel's display.

Syntax

:CHANnel<X>:DISPlay

{0|1} :CHANnel<X>:DISPlay?

Arguments

<X>→Specify the channel number (1|2)0→Disable channel <X> display 1→Enable

channel <X> display

Returns

<NR1>

:CHANnel<X>:INVert

Enable or disable the waveform invert function.

Syntax

:CHANnel<X>:INVert

{0|1} :CHANnel<X>:INVert?

Arguments

<X>→Specify the channel number (1|2)0→Disable invert function 1→Enable invert

function

Returns

<NR1>

:CHANnel<X>:MATH

Set the math expression.

Syntax

:CHANnel<X>:MATH {0|1|2|3}

Arguments

<X>→Specify the channel number (1|2)

0→Turn off math funciton   1→Select the add operator

2→Select the subtract operator   3→Select the FFT operation

Returns

<NR1>

:CHANnel<X>:OFFSet

Sets or query the offset voltage.

Syntax

:CHANnel<X>:OFFSet <NR3> :CHANnel<X>:OFFSet?

Arguments

<X>→Specify the channel number (1|2)<NR3> is the desired offset value in volts. The

range is dependent on the scale and the probe attenuation factor. The offset ranges

are as follows:

Offset Range:

2mV/div ~ 50mV/div	±0.5V
100mV/div ~ 500mV/div	±5V
1V/div ~ 5V/div	±50V

The next table shows the relationship between the <NR3> value and matching offset voltage.

0.002�2mV	0.01�10mV	0.1�100mV	1�1V
0.005�2mV	0.02�20mV	0.2�200mV	2�2V
	0.05�50mV	0.5�500mV	5�5V

Returns

<NR3>

:CHANnel<X>:PROBe

Select the different probe attenuation factor.

Syntax

:CHANnel$<$X$>$:PROBe {0|1|2} :CHANnel$<$X$>$:PROBe?

Arguments

$<$X$>\rightarrow$Specify the channel number (1|2) $0\rightarrow$1X $1\rightarrow$10X $2\rightarrow$100X

Returns

<NR1>
:CHANnel<X>:SCALe

Sets or query the vertical scale of the specified channel.

Syntax

:CHANnel$<$X$>$:SCALe

$<$NR3$>$:CHANnel$<$X$>$:SCALe?

Arguments

$<$X$>\rightarrow$Specify the channel number (1|2)$<$NR3$>$ is the desired gain value in volts per

division. The range is 2mV/div to5V/div (with 1X probe).

Next table shows the relationship between the <NR3> value and matching scale.

0.002�2mV	0.01�10mV	0.1�100mV	1�1V
0.005�2mV	0.02�20mV	0.2�200mV	2�2V
	0.05�50mV	0.5�500mV	5�5V

Returns

<NR3>

Examples

:CHANnel1:SCALe 1, setup the channel one at 1V per division.
:CURSor:X<X>Position

Select the cursors position of X axis.

Syntax

:CURSor:X<X>Position

<NR1> :CURSor:X<X>Position?

Arguments

<X>→Specify the cursor (1|2)<NR1> is the desired position. For x-axis operation, the range is 0 to 250; for y-axis operation, the range is 0 to 200.

Returns

<NR1>

CURSor:Y<X>Position

Select the cursors position of Y axis.

Syntax

:CURSor:Y<X>Position

<NR1> :CURSor:Y<X>Position?

Arguments

⟨X⟩→Specify the cursor (1|2)⟨NR1⟩ is the desired position. For x-axis operation, the range is 0 to 250; for y-axis operation, the range is 0 to 200.

Returns

<NR1>

:CURSor:<X>DELta? (query only)

Return the time or voltage difference between the two vertical or horizontal cursors.

Syntax

:CURSor:XDELta? :CURSor:

YDELta?

Arguments

⟨X⟩→Specify the time or voltage difference (X|Y)

Returns

<NR3>

:CURSor:XDISplay

Enable or disable the cursor display for X axis.

Syntax

:CURSor:XDISplay {0|1}

Arguments

0→Disable cursors display 1→Enable cursors display

Returns

<NR1>

:CURSor:YDISplay

Enable or disable the cursors display for Y axis.

Syntax

:CURSor:YDISplay {0|1}

Arguments

0→Disable cursors display 1→Enable cursors display

Returns

<NR1>
:CURSor:SOURce

Select which channels' cursor is active for front panel control.

Syntax

:CURSor:SOURce

{1|2|3} :CURSor:SOURce?

Arguments

1→Select channel 1 for cursor measurement 2→Select

channel 2 for cursors measurement 3→Select math

function for cursors measurement

Returns

<NR1>

:DISPlay:ACCumulate

Select the accumulate display mode.

Syntax

:DISPlay:ACCumulate

{0|1} :DISPlay:ACCumulate?

Arguments

0→Disable accumulate display mode 1→Enable accumulate display mode

Returns

<NR1>
:DISPlay:CONTrast:DATa

Select contrast level of LCD screen for data readout and waveform displays.

Syntax

:DISPlay:CONTrast:DATa

<NR1> :DISPlay:CONTrast:DATa?

Arguments

<NR1> is the desired brightness level. The range is from 0~20 (0% to 100%).

Returns

<NR1>

:DISPlay:GRATicule

Select graticule display type for LCD screen.

Syntax

:DISPlay:GRATicule

{0|1|2} :DISPlay:GRATicule?

Arguments

0→Select full grids 1→Select cross type 2→Only the frame is displayed

Returns

<NR1>
:DISPlay:WAVeform

Select the dots (or vectors) display for data-points.

Syntax

:DISPlay:WAVeform

<0|1> :DISPlay:WAVeform?

Arguments

0→Enable vectors display 1→Enable dots display

Returns

<NR1>

:MEASure:FALL? (query only)

Return the value of the timing measurement taken from the falling edge of the first pulse in the waveform.

Syntax

:MEASure:FALL?

Returns

<NR3>.

Note: Please select the specific channel before implementing any measurement. For explanation see ":MEASure:SOURce"

:MEASure:FREQuency? (query only)

Return the value of Frequency measurement.

Syntax

:MEASure:FREQuency?

Returns

<NR3>.

Note: Please select the specific channel before implementing any measurement. For explanation see ":MEASure:SOURce"

:MEASure:NWIDth? (query only)

Return the value of timing measurement from the first negative pulse in the waveform.

Syntax

:MEASure:NWIDth?

Returns

<NR3>.

Note: Please select the specific channel before implementing any measurement. For explanation see ":MEASure:SOURce"

:MEASure:PDUTy? (query only)

Return the ratio of the positive pulse width to the signal period.

Syntax

:MEASure:PDUTy?

Returns

<NR2>. is the percentage of ratio. The range is from 1 to 99.

Note: Please select the specific channel before implementing any measurement. For explanation see

":MEASure:SOURce"

:MEASure:PERiod? (query only)

Return the timing value of period measurement.

Syntax

:MEASure:PERiod?

Returns

<NR3>.

Note: Please select the specific channel before implementing any measurement. For explanation see ":MEASure:SOURce"

:MEASure:PWIDth? (query only)

Return the value of timing measurement of the first positive pulse in the waveform.

Syntax

:MEASure:PWIDth?

Returns

<NR3>.

Note: Please select the specific channel before implementing any measurement. For explanation see ":MEASure:SOURce"

:MEASure:RISe? (query only)

Return the value of timing measurement taken for the rising edge of the first pulse in the waveform.

Syntax

:MEASure:RISe?

Returns

<NR3>.

Note: Please select the specific channel before implementing any measurement. For explanation see ":MEASure:SOURce"

:MEASure:SOURce

Select the measured channel (channel 1 or 2). The default setting of measured

channel is channel one.

Note: Please select the specific channel before implementing any measurement.

Syntax

:MEASure:SOURce {1|2}

Arguments

1→Enable the measurement functions for channel 1

2→Enable the measurement functions for channel 2

Returns

<NR1>.

:MEASure:VAMPlitude? (query only)

Return the voltages of high value minus the low value.

Syntax

:MEASure:VAMPlitude?

Returns

<NR3>.

Note: Please select the specific channel before implementing any measurement. For explanation see
":MEASure:SOURce"
:MEASure:VAVerage? (query only)

Return the average voltages.

Syntax

:MEASure:VAVerage?

Returns

<NR3>.

Note: Please select the specific channel before implementing any measurement. For explanation see
":MEASure:SOURce"

:MEASure:VHI? (query only)

Return the value of the global high voltage.

Syntax

:MEASure:VHI?

Returns

<NR3>.

Note: Please select the specific channel before implementing any measurement. For explanation see ":MEASure:SOURce"

:MEASure:VLO? (query only)

Return the value of global low voltage.

Syntax

:MEASure:VLO?

Returns

<NR3>.

Note: Please select the specific channel before implementing any measurement. For explanation see ":MEASure:SOURce"

:MEASure:VMAX? (query only)

Return the value of maximum amplitude.

Syntax

:MEASure:VMAX?

Returns

<NR3>.

Note: Please select the specific channel before implementing any measurement. For explanation see ":MEASure:SOURce"

:MEASure:VMIN? (query only)

Return the value of minimum amplitude.

Syntax

:MEASure:VMIN?

Returns

<NR3>.

Note: Please select the specific channel before implementing any measurement. For explanation see

":MEASure:SOURce"

:MEASure:VPP? (query only)

Return the value of V_{max} minus V_{min}.

Syntax

:MEASure:VPP?

Returns

<NR3>.

Note: Please select the specific channel before implementing any measurement. For explanation see ":MEASure:SOURce"

:MEASure:VRMS? (query only)

Return the value of true Root Mean Square voltage.

Syntax

:MEASure:VRMS?

Returns

<NR3>.

Note: Please select the specific channel before implementing any measurement. For explanation see ":MEASure:SOURce"

:PRINt

Begin a hardcopy to a specified printer.

Syntax

:PRINt

:REFResh

Refresh the waveform data of the LCD screen and re-display the waveform data.

Syntax

:REFResh

:RUN

Controls the RUN state of the trigger system. The acquisition cycle will follow each qualified trigger in the RUN state.

Syntax

:RUN

:STOP

Controls the STOP state of the trigger system. The acquisition cycle only triggered when the :RUN command is received.

Syntax

:STOP

:SYSTem:UNLock

The front panel keyboard and knobs of IDS-820/IDS-840 will be disabled after any one of the remote control command are received. Use this command in order to re-activate the front panel keyboard and knobs.

Syntax

:SYSTem:UNLock

TIMebase:DELay

Sets the horizontal position (delay timebase) parameter.

Syntax

:TIMebase:DELay

<NR3> :TIMebase:DELay?

Arguments

<NR3> is the desired delay time.

Returns

<NR3>
:TIMebase:SCALe

Sets the horizontal timebase scale per division (SEC/DIV).

Syntax

:TIMebase:SCALe <NR3> :TIMebase:SCALe?

Arguments

Sec/div	NR3	Sec/div	NR3	Sec/div	NR3
1ns	1e-9	5µs	5e-6	25ms	25e-3

2.5ns	2.5e-9	10μs	10e-6	50ms	50e-3
5ns	5e-9	25μs	25e-6	100ms	100e-3
10ns	10e-9	50μs	50e-6	250ms	250e-3
25ns	25e-9	100μs	100e-6	500ms	500e-3
50ns	50e-9	250μs	250e-6	1s	1
100ns	100e-9	500μs	500e-6	2.5s	2.5
250ns	250e-9	1ms	1e-3	5s	5
500ns	500e-9	2.5ms	2.5e-3	10s	10
1μs	1e-6	5ms	5e-3		
2.5μs	2.5e-6	10ms	10e-3		

<NR3> is the desired timebase scale per division.

Returns

<NR3>

Examples

:TIMebase:SCALe 5e-3 sets timebase scale to 5ms per division.

:TIMebase:SWEep

Selects the horizontal timebase sweep mode. This command is equivalent to setting the horizontal menu.

Syntax

:TIMebase:SWEep

<0|1|2|3|4> :TIMebase:SWEep?

Arguments

0→Main timebase

1→Window 2→Window

Zoom 3→Roll mode 4→XY

mode

Returns

<NR1>

:TIMebase:WINDow:DELay

Set and query the zoomed area (the grey coloured area) for zoomed window display.

Syntax

:TIMebase:WINDow:DELay

<NR3> :TIMebase:WINDow:DELay?

Arguments

⟨NR3⟩ is the desired position (delay time).

Returns

<NR3>
:TIMebase:WINDow:SCALe

Set and query the scale (length) of the zoomed window timebase.

Syntax

:TIMebase:WINDow:SCALe

⟨NR3⟩ :TIMebase:WINDow:SCALe?

Arguments

⟨NR3⟩ is the desired scale (length) of the zoomed window timebase.

Returns

<NR3>

:TRIGger:COUPle

Select and query the type of trigger coupling.

Syntax

:TRIGger:COUPle <0|1> :TRIGger:COUPle?

Arguments

0→AC 1→DC

Returns

<NR1>
:TRIGger:DELay:TIMe

Set and query the user-defined delay trigger time.

Syntax

:TRIGger:DELay:TIMe

⟨NR3⟩ :TRIGger:DELay:TIMe?

Arguments

⟨NR3⟩ is the desired user-defined delay time. The range is from 100ns~1.3ms.

Returns

<NR3>

Note: Please select the specific delay type before implementing any measurement. For explanation see ":TRIGger:DELay:TYPe"

:TRIGger:DELay:EVENt

Sets and query the user-defined delay trigger events.

Syntax

:TRIGger:DELay:EVENt

⟨NR1⟩ :TRIGger:DELay:EVENt?

Arguments

⟨NR1⟩ is the desired user-defined delay trigger events. The range is from 2~65000.

Returns

<NR1>

Note: Please select the specific delay type before implementing any measurement. For explanation see ":TRIGger:DELay:TYPe"

:TRIGger:DELay:LEVel

Set and query the user-defined start trigger signal level.

Syntax

:TRIGger:DELay:LEVel

⟨NR3⟩ :TRIGger:DELay:LEVel?

Arguments

⟨NR3⟩ is the desired user-defined start trigger signal level. The range is ±12.

Returns

<NR3>

:TRIGger:DELay:MODe

Select and query the start trigger (i.e. external trigger) signal level.

Syntax

:TRIGger:DELay:MODe

<0|1|2> :TRIGger:DELay:MODe?

Arguments

0→TTL   1→ECL

2→USR

Returns

<NR1>
:TRIGger:DELay:TYPe

Select and query the different delay trigger settings.

Syntax

:TRIGger:ADVance:TYPe <0|1> :TRIGger:ADVance:TYPe?

Arguments

0→Time setting 1→Event setting

Returns

<NR1>

:TRIGger:FREQuency? (query only)

Return the readout value of the trigger frequency counter.

Syntax

:TRIGger:FREQuency?

Returns

<NR3>.

:TRIGger:LEVel

Select and query the trigger level.

Syntax

:TRIGger:LEVel <NR3> :TRIGger:LEVel?

Arguments

<NR3> is the desired trigger level voltage.

Returns

<NR3>
:TRIGger:MODe

Select and query the trigger mode.

Syntax

:TRIGger:MODe <0|1|2|3> :TRIGger:MODe?

Arguments

0→Auto Level 1→Auto 2→Normal 3→Single

Returns

<NR1>

:TRIGger:NREJ

Switch and query the noise rejection mode.

Syntax

:TRIGger:NREJ <0|1> :TRIGger:NREJ?

Arguments

0→OFF 1→ON

Returns

<NR1>
:TRIGger:PULSe:MODe

Switch and query different pulse trigger types.

Syntax

:TRIGger:PULSe:MODe

<0|1|2|3> :TRIGger:PULSe:MODe?

Arguments

0→< 1→> 2→= 3→≠

Returns

<NR1>

:TRIGger:PULSe:TIMe

Select the time value for pulse width.

Syntax

:TRIGger:PULSe:TIMe

<NR3> :TRIGger:PULSe:TIMe?

Arguments

<NR3> is the desired time value of the pulse width with the units is in seconds.

Returns

<NR3>
:TRIGger:REJect

Select and query the frequency rejection mode.

Syntax

:TRIGger:REJect <0|1|2> :TRIGger:REJect?

Arguments

0→OFF 1→Low frequency reject mode 2→High frequency reject mode

Returns

<NR1>

:TRIGger:SLOP

Switch and query the rising or falling trigger slope.

Syntax

:TRIGger:SLOP <0|1> :TRIGger:SLOP?

Arguments

0→Rising slope 1→Falling slope

Returns

<NR1>
:TRIGger:SOURce

Select and query the trigger source.

Syntax

:TRIGger:SOURce <0|1|2|3> :TRIGger:SOURce?

Arguments

0→Channel 1 1→Channel 2 2→External trigger 3→AC line voltage

Returns

<NR1>

:TRIGger:TYPe

Select and query the trigger type.

Syntax

:TRIGger:TYPe <0|1|2|3> :TRIGger:TYPe?

Arguments

0→Edge 1→Video 2→Pulse 3→Delay

Returns

<NR1>
:TRIGger:VIDeo:FIELd

Select and query the field on which the video trigger mode will be triggered.

Syntax

:TRIGger:VIDeo:FIELd

<0|1|2> :TRIGger:VIDeo:FIELd?

Arguments

0→Line 1→Odd frame (Field 1) 2→Even frame (Field 2)

Returns

<NR1>

:TRIGger:VIDeo:LINe

Select and query the specified line for a video signal.

Syntax

:TRIGger:VIDeo:LINe

<NR1> :TRIGger:VIDeo:LINe?

Arguments

<NR1> is the desired line.

For the NTSC system, the line range is from 1~263 for Odd frames, 1-262 for even

frames. For the PAL system, the line range is from 1~313 for Odd frames, 1-312 for

even frames.

Returns

<NR1>
:TRIGger:VIDeo:POLarity

Select and query the input video polarity.

Syntax

:TRIGger:VIDeo:POLarity <0|1> :TRIGger:VIDeo:POLarity?

Arguments

0→Positive-going sync pulses 1→Negative-going sync pulses

Returns

<NR1>

:TRIGger:VIDeo:TYPe

Select and query the TV broadcast system.

Syntax

:TRIGger:VIDeo:TYPe <0|1|2> :TRIGger:VIDeo:TYPe?

Arguments

0→PAL 1→NTSC 2→SECAM

Returns

<NR1>
:WMEMory<X>:DISPlay

Select whether the stored waveform will be displayed after being saved.

Syntax

:WMEMory<X>:DISPlay

<NR1> :WMEMory<X>:DISPlay?

Arguments

<X>→Specify the location of RefA or RefB memory (1|2) 0→OFF 1→ON

Returns

<NR1>

:WMEMory<X>:ERASe

Select whether the stored waveform will be erased after being saved.

Syntax

:WMEMory<X>:ERASe

Arguments

<X>→Specify the location of RefA or RefB memory (1|2)

:WMEMory<X>:LOCate

Set the position of the stored waveform.

Syntax

:WMEMory<X>:LOCate <NR1>

Arguments

<X>→Specify the location of RefA or RefB memory (1|2) <NR1> is the

desired position. The rage is from −200 to +200.
:WMEMory<X>:OFFSet

After the ":WMEMory<X>:LOCate" command is specified, you can adjust the position up or down by this command.

Syntax

:WMEMory⟨X⟩:OFFSet ⟨NR1⟩

Arguments

⟨X⟩→Specify the location of RefA or RefB memory (1|2)⟨NR1⟩ is the desired offset

position. The range is from −100 to +100.

:WMEMory⟨X⟩:SAVe

Select whether the memory set will be saved.

Syntax

:WMEMory⟨X⟩:SAVe ⟨1|2|3⟩

Arguments

⟨X⟩→Specify the location of RefA or RefB memory (1|2) 1→Channel 1 2→Channel 2

3→Math function

5. Status Reports

A set of status registers allows the user to quickly determine the Digital Storage
Oscilloscopes' internal processing status. The status register, as well as the status
and event reporting system, adhere to SCPI recommendations.

Structure of System

The sketch of the status and event reporting system is shown in figure 5-1. Each
component of the sketch represents a set of registers and queues which can read,
reported on, or can enable the occurrence of certain events within the system. If a
specific event in the Digital Storage Oscilloscope sets a bit in a status register, reading
the status registers can tell you what types of events have occurred. Each bit in the
status register corresponds to a bit in an enable register; the enable bit must be high
for the event to be reported to the Status Byte Register. A Service Request (SRQ) is
the last event to occur. The SRQ requests an interrupt on the GPIB to report events to
the system controller.

Status Registers

Two kinds of status registers are included within the IDS-820/IDS-840 Digital Storage Oscilloscope.

◆ OPERation Status Registers (CONDition, EVENt, and ENABle)

◆ QUEStionable Status Registers (CONDition, EVENt, and ENABle) The STATus subsystem is the hierarchical set of commands (Figure 5-2) that read the SCPI defined status registers. The lower level nodes: QUEStionable and OPERation each have three 16 bits registers: CONDition, EVENt, and ENABle. Figure 5-3 shows the sequential relationship between these three types of registers and the commands which relate to each register.
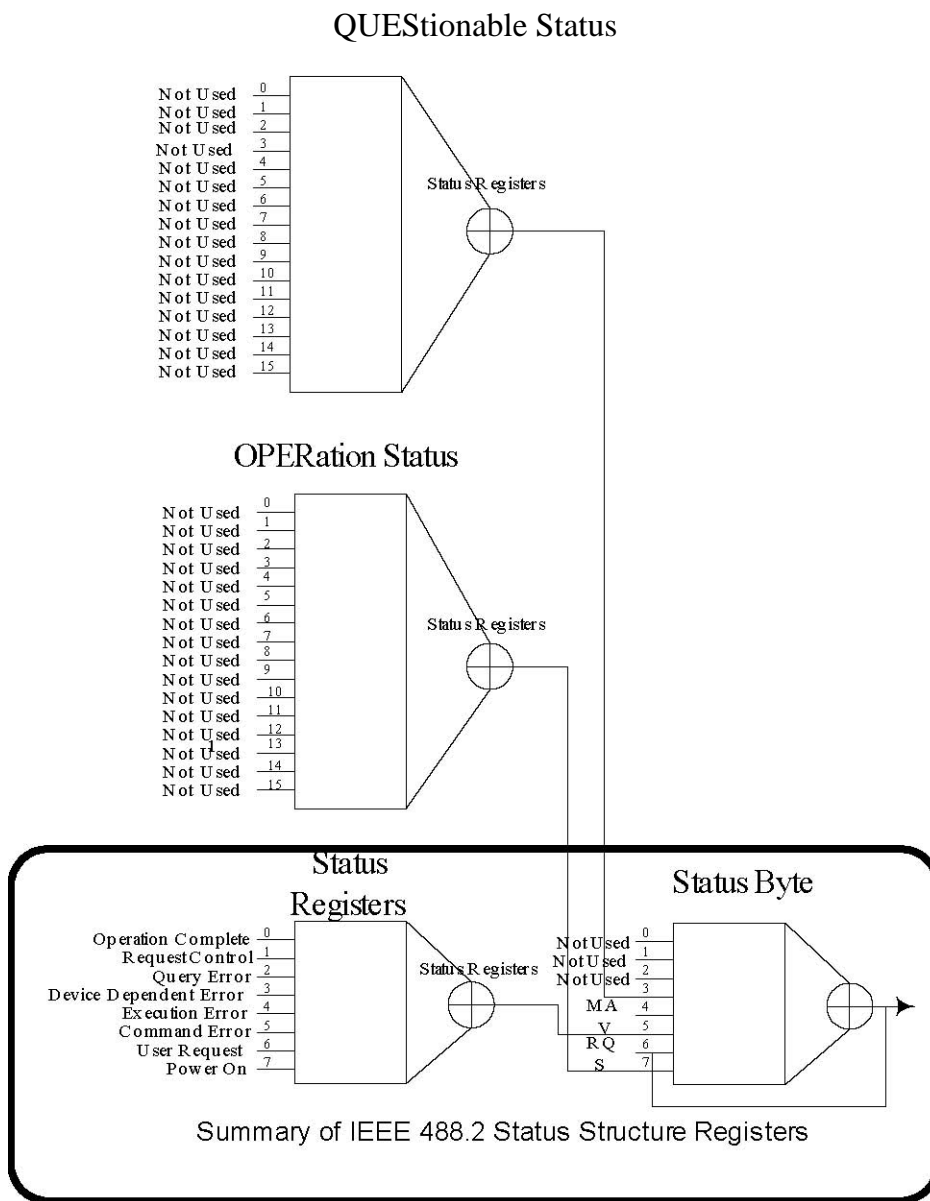

Figure 5-1. A graphic represents the status registers and their connections.

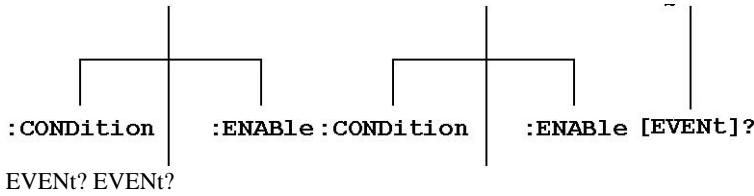:CONDition   :ENABle :CONDition   :ENABle [EVENt]?

EVENt? EVENt?

Figure 5-2: STATus hierarchy of SCPI defined register
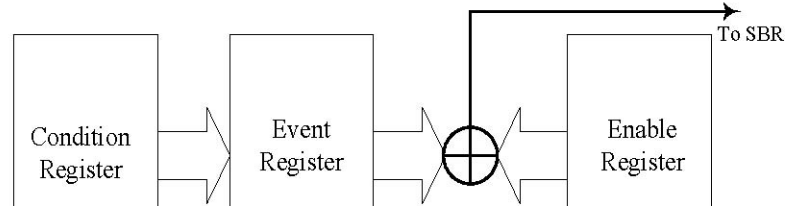
Figure 5-3: Status registers and related commands.

The CONDition register is a read-only register which monitors the present state of the instrument. The CONDition register updates in real time and the inputs are not latched or buffered. When a condition monitored by the CONDition register becomes true, the bit for that condition also becomes true (1). When the condition is false, the bit is 0. The read-only EVENt register latches any false-to-true change in condition. Once the bit in the EVENt register is set, it is no longer affected by changes in the corresponding bit of the CONDition register. The bit remains set until the controller reads it. The command *CLS (Clear Status) clears the EVENt registers.

QUEStionable Status Registers.

Table 5-1 shows the bit designations of the 16 bit QUEStionable Status Register.

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
NU	◆NU	NU	NU	NU	NU	NU	NU
32768	16384	8192	4096	2048	1024	512	256
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
NU	NU	NU	NU	NU	NU	NU	NU
128	64	32	16	8	4	2	1

Table 5-1: QUEStionable Status Register

OPERation Status Registers.

Table 5-2 shows the bit designations of the 16 bit OPERation Status Register.

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
NU	NU	NU	NU	NU	NU	NU	NU
32768	16384	8192	4096	2048	1024	512	256
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
NU	NU	NU	NU	NU	NU	NU	NU
128	64	32	16	8	4	2	1

Table 5-2: OPERation Status Register (NU: not used)

Status Registers.

There are two status registers in the IDS-820/IDS-840 digital oscilloscope which are defined by IEEE-488.1 and IEEE-488.2 standards.

- ◆ Status Byte Register (SBR)
- ◆ Standard Event Status Register (SESR)

Status Byte Register (SBR): The SBR (Table 5-3) summaries the status of all other registers and queue.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OPER	RQS	ESB	MAV	QUES	NU	NU	NU
128	64	32	16	8	4	2	1

Table 5-3: Status Byte Register (SBR)

Bits 0 to bit 2 are not used; these bits are always zero. Bit 3 (QUES, QUEStionable) is the summary bit for the QESR (QUEStionable Event Status Register). When this bit is high it indicates that status is enabled and present in the QESR. Bit 4 (MAV, Message Available) indicates that output is available in the output queue. Bit 5 (ESB, Event Status Bit) is the summary bit for the Standard Event Status Register (SESR). When this bit is high it indicates that status is enabled and present in the SESR. Bit 6 (RQS, Request Service) is obtained from a serial poll. This bit shows that the IDS-820/IDS-840 Digital storage oscilloscope requests service from the GPIB controller. Bit 7 (OPER, OPERation) is the summary bit for the OESR (OPERation EVENt STATus Register).

Use serial poll or the $*STB?$ Query to read the contents of the SBR. The bits in the SBR are set and cleared depending on the contents of the Standard Event Status Register (SESR), the Standard Event Status Register (SESR), and the Output Queue.

Standard Event Status Register (SESR): Table 5-4 shows the SESR

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PON	USR	CME	EXE	DDE	QYE	NU	OPC
128	64	32	16	8	4	2	1

Table 5-4: Standard Event Status Register (SESR)

Bit 0 (OPC, Operation Complete) shows that the operation is complete. This bit is active when all pending operations are complete following an $*OPC$ command. Bit 1 is always zero. Bit 2 (QYE, Query Error) indicates a command or query protocol error.The bit 3 (DDE, Device Error) shows that a device error occurred. Bit 4 (EXE, Execution Error) shows that an error occurred while the IDS-820/IDS-840 Digital storage oscilloscope was execution a command or query. Bit 5 (CME, Command Error) shows that an error occurred while theIDS-820/IDS-840 Digital storage oscilloscope was paring a command or query. Bit 6 (USR, User Request) is ignored here for the IDS-820/IDS-840. Bit 7 (PON, Power On) shows that the IDS-820/IDS-840 Digital Storage Oscilloscope was powered on.

Use the $*ESR?$ Query to read the SESR. Reading the SESR clears the bits of the registers so that the register can accumulate information about new events.

Enable Registers.

The enable registers determine whether certain events are reported to the Status Byte Register and SRQ. The IDS-820/IDS-840 Digital Storage Oscilloscope has the following enable registers.

- ❖ Event Status Enable Register (ESER)
- ❖ OPERation Enable Register
- ❖ QUEStionable Enable Register
- ❖ Service Request Enable Register (SRER)

When one of the bits of the enable registers is high and the corresponding bit in the status register is high, the enable registers will perform a logical OR function, the output that controls the set bit of the Status Byte Register is high. Various commands set the bits in the enable registers. The following sections describe the enable registers and the commands that set them. Event Status Enable Register (ESER): The ESER controls which types of events are summarized by the Event Status Bit (ESB) in the SBR. The bits of the ESER correspond to the bits of the SESR. Use the *ESE command to set the bits in ESER. Use the *ESE? command to read it. OPERation Enable Register: Even though the OPERation Enable Register is present in this digital storage oscilloscope, the OPERation registers do not report any conditions. QUEStionable Enable Register: The QUEStionable Enable Register controls which types of events are summarized by the QUES status bit in the SBR. Service Request Enable Register (SRER): The SRER controls which bits in the SBR generate a service request. Use the *SRE command to set the SRER. Use the *SRE? command to read it.

Queues

The output queue is included with this Digital Storage Oscilloscope. Output Queue: This Digital Storage Oscilloscope stores query responses in the output queue by succeeding the IEEE 488.2 protocol. If the Digital Storage Oscilloscope receives a lot of un-read query data simultaneously, the output buffer of the Digital Storage Oscilloscope will be repeatedly addressed and the output buffer will probably generate errors. The computer must read a query response before it sends the next command (or query) or it loses response to earlier queries. When an error or event occurs, the output queue stores the message. The output queue stores and reports the messages on a FIFO (first in first out) state.

6. Error Messages

Table 6-1 lists the SCPI error messages for this digital storage oscilloscope.

Error Code	SCPI Error Code / Explanation
-100	Command error
-102	Syntax error
/*Execution Error*/	
-220	Parameter error
-221	Settings conflict
-222	Data out of range
-223	Too much data
-224	Illegal parameter value
-232	Invalid format

7. Program Template for GPIB

```
/* Filename - IDS820ex1.c *
        * This is an example program written in C. We use a NI's GPIB interface
        * card and one X86 PC to control IDS-820/IDS-840. This program could
        * get the waveform data from IDS-820/IDS-840, and save them to a file.
        * You can use Microsoft Visual C++ or Borland C++ Builder to compile this
        * file. And you must link this file with an object file
        * (BORLANDC_GPIB-32.OBJ) for Borland C++ or GPIB-32.OBJ for Visual C++ )
        * that provided by National Instruments Corporation. DECL-32.H is a
        * Win32 C/C++ include file, that contains NI-488.2 function prototypes
        * and various pre-defined constants. It's also provided by NI.
 *

        * Copyright GOOD WILL INSTRUMENT
        * All Rights Reserved.
 */


#include <stdio.h> #include

<stdlib.h> #include <string.h>


/*
        * Include the WINDOWS.H and DECL-32.H files. The standard Windows
        * header file, WINDOWS.H, contains definitions used by DECL-32.H and
        * DECL-32.H contains prototypes for the GPIB routines and constants.

  */ #include <windows.h> #include "decl-32.h"
```

```c
#define ARRAYSIZE 1100 /* Size of read buffer */ #define BDINDEX 0 /* Board Index
*/
#define PRIMARY_ADDR_OF_DMM     7                    /* Default primary address of
                                                        IDS820/840*/
#define NO_SECONDARY_ADDR       0                    /* Secondary address of device
*/
#define TIMEOUT                 T3s                  /* Timeout value = 5 seconds
*/

#define EOTMODE                 1     /* Enable the END message
*/
#define EOSMODE                 0     /* Disable the EOS mode
*/
void Acquire(void);
void Delay(int);
void ShowIbsta(char);

int             Dev;                  /* Device handle
*/
unsigned char ReadBuffer[201];                 /* Read data buffer
*/

char ErrorMnemonic[21][5] = {"EDVR",
                                "ECIC",
                                "ENOL",
                                "EADR",
                                "EARG",
                                "ESAC",
                                "EABO",
                                "ENEB",
                                "EDMA",
                                "",
                                "EOIP",
                                "ECAP",
                                "EFSO",
```

```
                                                  "",
                                                  "EB
                                                  US",
                          "ESTB", "ESRQ", "", "", "", "ETAB"}; unsigned
char WaveBuf[ARRAYSIZE];

int _cdecl main(void)
{
int i, count=0;
unsigned char ch;

        Dev = ibdev (BDINDEX, PRIMARY_ADDR_OF_DMM,
                     NO_SECONDARY_ADDR, TIMEOUT,
                     EOTMODE, EOSMODE); if(ibsta & ERR)
    printf("\n\rUnable to open device"); ibclr (Dev); if(ibsta & ERR)
        printf("\n\rUnable to clear device(ibsta= %x), (iberr=%x)",ibsta,iberr);

      ibwrt (Dev, "*CLS\n", 5L); /*Clear ths
   status registers and*/ printf("\n\r*CLS\n\r");
   /*Output buffer of IDS-820/IDS
840. */
if((ibsta&ERR)||(ibsta&TIMO)){ ShowIbsta(1); return 0;
}
    Delay(1000);
     ibwrt (Dev, "*IDN?\n", 6L);              /*Get the unique identification */
     printf("*IDN?\n\r");                     /*code of the IDS-820/IDS-840.
*/

 if((ibsta&ERR)||(ibsta&TIMO)){ ShowIbsta(1); return 0;
     } while(1){ ibrd (Dev, ReadBuffer, 100);
     /*Read datas from input bufer. */
 if((ibsta&ERR)||(ibsta&TIMO)){ ShowIbsta(0); return 0;
}
for(i=0;i<ibcntl;i++){ ch=ReadBuffer[i]; WaveBuf[count++]=ch;
}
if(ch=='\n'){ WaveBuf[count]=0x00; printf("%s\n\r",WaveBuf); break;
     } }

 Delay(1000);
 ibwrt (Dev, ":RUN\n", 5L); /*Let the IDS-820/IDS-840 run.

*/ printf(":RUN\n\r"); if((ibsta&ERR)||(ibsta&TIMO)){
   ShowIbsta(1); return 0; }
    Delay(1000);
     ibwrt (Dev, ":CHANnel1:DISPlay 1\n", 20L);/*Set channel 1 display on
```

```
*/ printf(":CHANnel1:DISPlay 1\n\r"); if((ibsta&ERR)||(ibsta&TIMO)){
        ShowIbsta(1);
        return 0;
        }


    Delay(1000);
     ibwrt (Dev, ":TIMebase:SCALe 2.0e-4\n", 23L);/*Set timebase: 200us/div

*/ printf(":TIMebase:SCALe 1.0e-4\n\r"); if((ibsta&ERR)||(ibsta&TIMO)){
        ShowIbsta(1);
        return 0;
        }


    Delay(1000);
     ibwrt (Dev, ":CHANnel1:OFFSet 0\n", 19L); /*Set offset voltage: 0V

*/ printf(":CHANnel1:OFFSet 0\n\r"); if((ibsta&ERR)||(ibsta&TIMO)){
        ShowIbsta(1);
        return 0;
        }


    Delay(1000);
     ibwrt (Dev, ":CHANnel1:SCALe 0.5\n", 20L); /*Set vertical scale:

                                            500mV/div */
     printf(":CHANnel1:SCALe 0.5\n\r");
     if((ibsta&ERR)||(ibsta&TIMO)){

        ShowIbsta(1);
        return 0;
        }

    Delay(1000);
     ibwrt (Dev, ":ACQuire:MODe 0\n", 16L); /*Set acquire mode:

                                            sample mode */
     printf(":ACQuire:MODe 0\n\r");
     if((ibsta&ERR)||(ibsta&TIMO)){


        ShowIbsta(1
```

```
  );
    return 0;
    }


Delay(1000);
  ibwrt (Dev, ":TRIGger:LEVel 0.3\n", 19L); /*Set trigger level: 0.3V*/
  printf(":TRIGger:LEVel 0.3\n\r");
  if((ibsta&ERR)||(ibsta&TIMO)){


    ShowIbsta(1
    );
     return 0;
     }


Delay(1000);
  ibwrt (Dev, ":TRIGger:MODe 1\n", 16L); /*Set trigger mode: AUTO */
  printf(":TRIGger:MODe 1\n\r");
  if((ibsta&ERR)||(ibsta&TIMO)){


    ShowIbsta(1
    );
     return 0;
     }


        Delay(10000); /*Waiting
    for acquisition process and
    IDS-820/IDS840 internal*/
    /*process. The delay time
    depend on the timebase that*/
    /*you selected. Low speed
    division require much more */
    /*time for acquisition. It's
    recommended a minimum */
    /*200ms delay time after the
    last command, before */
    /*getting the waveform data
    from IDS-820/IDS-840. */
    Acquire(); /*Get waveform data.
    */
ibwrt (Dev, ":SYSTEM:UNLOCK\n",15L);              /*Unlock IDS-820/IDS-840
```

from

remote control.*/

```
printf(":SYSTEM:UNLOCK\n\r");
if((ibsta&ERR)||(ibsta&TIMO)){
ShowIbsta(1);
return 0;
}
ibonl(Dev,0);                                      /*Take the device
offline. */
return 1;
}


void Acquire(void)
{
short i, j;
FILE *writeP;
char writeFilename[15] = "wavedata.txt";
short wave;
int tmp, count=0;


ibwrt (Dev, ":ACQUIRE1:POINT\n", 16L); printf(":ACQUIRE1:POINT\n\r");


Delay(1000);
if((ibsta&ERR)||(ibsta&TIMO)){
ibclr (Dev);
ibwrt (Dev, "*CLS\n", 5L);
ShowIbsta(1);
Delay(100000);
ibwrt (Dev, ":ACQUIRE1:POINT\n", 16L); /*Try again! */
printf(":ACQUIRE1:POINT\n\r");
     if((ibsta&ERR)||(ibsta&TIMO)){
          ShowIbsta(1);
         Delay(100000);
          exit (1);

    } } while(1){
    ibrd(Dev, ReadBuffer, 100);
    if((ibsta&ERR)||(ibsta&TIMO)){
         ShowIbsta(0);
         Delay(100000);
         exit (1);

    }
    for(i=0;i<ibcntl;i++)
```

```
        WaveBuf[count++]=ReadBuffer[i];
    printf("\rReceived: %7d ", count);
    if(count>=1014){

    printf("\rReceived: %7d bytes.\n\r", count);
    break; } }
    /*Open file "wavedata.txt" and write waveform datas to it.*/
     writeP=fopen(writeFilename, "w");
    if(writeP == NULL){
    printf("error: cannot write '%s'\n", writeFilename);
    exit(1);
     }
     j=14;
     for(i=0;i<500;i++){

        wave=WaveBuf[j++]<<8;
        wave+=WaveBuf[j++];
        fprintf(writeP, "%d\n", wave);

    }
    tmp = fclose(writeP);
    if(tmp == EOF){

    printf("error: cannot write '%s'\n\r",
    writeFilename); exit(1); } }

void Delay(int i)
{
int j,k;


        for(j=0;j<i;j++){ for(k=
        0;k<30000;k++) ; } }
    void ShowIbsta(char c) { if(ibsta &
    ERR){ if(c) printf("\n\rUnable write to
    device(ibsta= %x),
    (iberr=%x)\n\r",ibsta,iberr); else
    printf("\n\rUnable read from
    device(ibsta= %x),
(iberr=%x)\n\r",ibsta,iberr); } else{
        if(c) printf("\n\rWrite, ibsta= %x",ibsta);
            else printf("\n\rRead, ibsta=
            %x",ibsta); } }
```