Agilent 86130A BitAlyzer® Error Performance Analyzer Programmer's Guide

Agilent Technologies

Notices

Manual Part Number

86130-90051

Edition

Third edition, June 2002

Printed in USA

Agilent Technologies, Inc.
Lightwave Division
3910 Brickway Boulevard
Santa Rosa, CA 95403, USA

Warranty

Technology Licenses

Restricted Rights Legend

Safety Notices

CAUTION

Caution denotes a hazard. It calls attention to a procedure which, if not correctly performed or adhered to, could result in damage to or destruction of the product. Do not proceed beyond a caution sign until the indicated conditions are fully understood and met.

WARNING

Warning denotes a hazard. It calls attention to a procedure which, if not correctly performed or adhered to, could result in injury or loss of life. Do not proceed beyond a warning sign until the indicated conditions are fully understood and met.

Contents

1

Remote Operation

Introduction

This chapter contains the information required to operate the instrument remotely using a suitable controller. The aspects of remote operation covered are as follows:

- What is the General Purpose Interface Bus (GPIB)
- Using the GPIB

What is the General Purpose Interface Bus (GPIB)?

The General Purpose Interface Bus (GPIB) is Agilent Technologies' implementation of IEEE standard 488.2-1992.

The GPIB Interface is easy to use. It allows flexibility in both communicating and controlling data between a controller and the 86130A. It is also one of the easiest methods of constructing automatic test systems.

Devices on the bus fall into one of two categories, controller or non-controller. For example, the simplest system (two non-controllers) where one instrument is configured to send data continuously - known as TALKING, and the other instrument (such as a printer) is configured to receive data continuously - known as LISTENING. Most devices can perform both roles, TALK or LISTEN, but not simultaneously. Usually a controller controls which instrument TALKS and which instrument LISTENS. The 86130A can TALK and LISTEN when instructed to do so by a suitable controller. In addition it can operate without a controller when logging results or sending screen dumps to an external printer.

The controller may also manage other instruments connected in the same bus configuration, addressing only one instrument, to carry out the data transfer or TALK function.

Further information on GPIB standards and concepts is available in the following publications:

- IEEE Interface Standard 488.2-1992

GPIB to 86130A Connection

System Configuration

The 86130A error performance analyzer is factory preset to GPIB address 17.

Cabling Arrangements

Connect a GPIB cable from the Controller to the 86130A error performance analyzer.

Use of GPIB

You should consider the following when connecting the instrument for operation over the GPIB.

- Operating Distances
- Instrument Mode at Power On
- Address Configuration
- Local and Remote Modes
- Using Local and Remote Commands
- GPIB Required Commands
- Sending Commands Over the GPIB
- Using Non-HP Controllers
- Invalid Commands
- Reading Data
- Message Format

Operating Distances

Up to 30 instruments can be connected on a local bus system, but it is important to ensure that the maximum GPIB cable length between instruments is less than 2 meters. In addition the total cabling should not exceed 20 meters.

Some useful cable part numbers are listed in Table 1-1.

Table 1-1. Part Numbers of GPIB Cables

Description	HP/Agilent Part Number
1m	10833A
2m	10833B
4m	10833C
0.5m	10833D

For distances up to 1250m, a suitable bus extender such as a 37204A can be used. Two bus extenders are required, one at the local bus and one at the remote bus. For distances beyond 1250m, two 37204A bus extenders with suitable modems must be employed.

Instrument Mode at Power On

At power on, the 86130A will return to the same mode as it was powered down. Normally, at power on, the 86130A is ready for either front panel operation or remote operation.

CAUTION No GPIB activity should take place within 3 minutes of system power up, as this will effect the system power up routine and may result in system hang up.

Address Configuration

When configuring a GPIB-based system, it is essential that each device on the GPIB has a unique address. The device address can be in the range of 1 to 30. For a controller to communicate with a device over the GPIB, it must send the commands to the appropriate GPIB device address.

Local and Remote Modes

The 86130A can be operated in one of two modes: local or remote.

In local operation, all the front panel controls are responsive and control the instrument.

In remote operation, the front panel controls are inoperative and the instrument is controlled by the GPIB controller. The front panel display reflects the remote programming commands received.

Local and Remote Commands

At power on the instrument is in local mode and is sent into remote mode by sending any command string to the instrument. The instrument will recognize the command string, set itself to the remote mode and then act on that command.

There are two ways to return the instrument back to local mode. The first method is to press the front panel `Local` key. The second method is to cycle power to the instrument.

GPIB Required Commands

The required commands perform the most basic remote functions over GPIB and are common to all GPIB controllable instruments. The commands are as follows:

- DEVICE CLEAR
- SERIAL POLL

Device Clear (CLEAR)

This command initializes the instrument GPIB hardware.

The command format using HP 200/300 Series Basic is:

Example CLEAR 717

Serial Poll (SPOLL)
A serial poll will retrieve the value of the primary status byte. This byte contains useful information about the current state of the instrument.

Example SPOLL(717)

Commands Sent Over GPIB

To send commands over the GPIB involves sending the command string via the interface select code to the device address. HP computers use the Basic instruction OUTPUT to send command strings. The structure of a command line is as follows:

OUTPUT interface select code + device address; “command string”

Note

The semi-colon symbol is the command separator and must be included.
The command string must be enclosed in inverted commas.

Using a HP 300 Series Controller with its GPIB interface set at select code 7 and a device at address 17, a typical command line to reset the instrument would appear as follows:

OUTPUT 717;”*RST”

Non-HP Controllers

With non-HP controllers, it may be necessary to send a suitable command terminator after the data message, the terminator can be:

- ASCII newline (identical to the linefeed character, LF)
- ASCII carriage return + 1 linefeed (CR/LF)
  In most HP controllers the CR/LF is sent automatically when HP Basic OUTPUT statements are used.

Invalid Commands

A command will be rejected if:

- It contains a syntax error
- It cannot be identified
- It has too few or too many parameters
- A parameter is out of range
- It is out of context

Query Responses

It is possible to interrogate the individual settings and status of a device using query commands. Retrieving data is a two stage operation.

The query command is sent from the controller using the OUTPUT statement and the data is read from the device using the ENTER statement. A typical example, using the SCPI IEEE 488.2 Common Command *IDN? querying the identity of a device, is given as follows:

```
OUTPUT 717;"*IDN?"
ENTER 717;Identity$
PRINT Identity$
```

Typically, this would display the identity string
“AGILENT TECHNOLOGIES,86130A,3331U00101,A.01.01”.

Note

When sending strings to the instrument, either the double quote (“) or the single quote may be used (‘), the former being more suited to PASCAL programs, which make use of a single quote; the latter being more suited to use in BASIC programs, which uses a double quote as a delimiter. In this manual, the double quote has been used throughout.

Message Format

The 86130A has the capability of returning data in the following formats:

- STRING
- NUMERIC
- BOOLEAN
- BLOCK DATA

String
The following example returns an ASCII string representing the instrument serial number, enclosed in quotes. This should be entered into a string variable.

Example

```
10 OUTPUT 717;"*IDN?"
20 ENTER 717;Serial$
30 PRINT Serial$
40 END
```

Possible Result = "AGILENT TECHNOLOGIES,86130A,3331U00101,A.01.01"

Numeric
Returns one of three numeric formats and can be entered into a string or numeric variable.

The three formats are:

- An integer
- A number with embedded decimal point.
- A number with embedded decimal point and exponent.

Integer

Example

```
10 OUTPUT 717;"*STB?"
20 ENTER 717;Status_byte$
30 PRINT Status_byte$
40 END
```

Requests the contents of the status byte. Possible Result = +64

A Number with Embedded Decimal Point.

Example

```
10 OUTPUT 717;":SENSe1:VOLTAGE:ZOTHRESHOLD?"
20 ENTER 717;Level$
30 PRINT Level$
40 END
```

Requests the current voltage threshold at which the system is operating.

A Number with Embedded Decimal Point and Exponent.

Example

```
10 OUTPUT 717;"FETCH:ECOUNT?"
20 ENTER 717;Error_count
30 PRINT Error_count
40 END
```

Requests the number of errors counted thus far.

Possible Result = +9.91000000E+012

Boolean

Boolean parameters are used to indicate whether a condition is true or false. A numeric value is returned where 1= true and 0= false.

Block Data

Block data is used when large quantities of related data is being returned. A definite length block is suitable for sending blocks of 8-bit binary information when the length is known beforehand. An indefinite length block is suitable for sending blocks of 8-bit binary information when the length is not known beforehand or when computing the length beforehand is undesirable.

2

Programming the 86130A

Introduction

This section gives information on how to begin programming the 86130A.

The section covers the following topics:

- The 86130A Command Language
- Important Points about SCPI
- SCPI Command Structure
- Behavior at Power On
- Recommended Programming Techniques
- Order-Sensitive Commands

The 86130A Command Language

The 86130A is compatible with the standard language for remote control of instruments. Standard Commands for Programmable Instruments (SCPI) is the universal programming language for instrument control.

SCPI can be subdivided into two distinct command sets.

- Common Commands
- Instrument Control Commands

SCPI Common Commands

This is a common command set. It is compatible with IEEE 488.2 and contains general housekeeping commands.

The common commands are always headed by an asterisk. A typical example is the reset command

```
OUTPUT717;"*RST"
```

The IEEE 488.2 command set also contains query commands. Query commands always end with a question mark. A typical example is the command querying the identity of a device at address 717.

```
OUTPUT717;"*IDN?"
ENTER 717;Identity$
```

To see a full list of commands, refer to "System Command Reference Section" on page 4-1.

SCPI Instrument Control Commands

The programming commands are compatible with the Standard Commands for Programmable Instruments (SCPI) standard. For more detailed information regarding the GPIB, the IEEE 488.2 standard, or the SCPI standard, refer to the following books:

SCPI Consortium. SCPI–*Standard Commands for Programmable Instruments*, 1997. (http://www.scpiconsortium.org)

International Institute of Electrical and Electronics Engineers. *IEEE Standard 488.1-1987, IEEE Standard Digital Interface for Programmable Instrumentation*. New York, NY, 1987.

International Institute of Electrical and Electronics Engineers. *IEEE Standard 488.2-1987, IEEE Standard Codes, Formats, Protocols and Common commands For Use with ANSI/IEEE Std 488.1-1987*. New York, NY, 1987.

Note

The response of the instrument to the *RST or *RCL commands may be up to 3 seconds. Any GPIB program using these commands should have a time out of greater than 3 seconds.

IEEE 488.2 Mandatory and Optional Commands

In order to comply with the SCPI model as described in IEEE 488.2, the instrument implements certain mandatory commands. Other commands are implemented optionally. For more detail on the IEEE 488.2 mandatory and optional commands, refer to “System Command Reference Section” on page 4-1.

Important Points about SCPI

There are a number of key areas to consider when using SCPI for the first time.

These are as follows:

- Instrument Model
- Command Syntax
- Optional Parts of Commands
- Sending Commands
- Command Separators

Instrument Model

SCPI guidelines require that the 86130A is compatible with an instrument model. This ensures that when using SCPI, functional compatibility is achieved between instruments which perform the same tasks. For example, if two different instruments have a programmable clock frequency setting, then both instruments would use the same SCPI commands to set their frequency. The instrument model is made up of a number of subsystems.

The sub-system defines a group of functions within a module and has a unique identifier under SCPI which is called the Root Keyword.

For more detail on the instrument model, refer to “Interrogating the Instrument Status” on page 3-1.

Command Syntax

Commands may be up to twelve characters long. A short-form version is also available which has a preferred length of four characters or less. In this document the long-form and short-form versions are shown as a single word with the short-form being shown in upper-case letters. For example, the long-form node command

SOURce has the short-form SOUR. Using the short form saves time when entering a program, however using the long form makes a program more descriptive and easy to understand.

In Chapter 4, “System Command Reference Section”, any command used to set the value of any configurable parameter also has a query form. Where a command ending in a question mark does appear, it is a query only command.

Optional Command Keywords

Some layers in the SCPI command structure are optional. These optional keywords are indicated by square brackets ([]). A typical use for these types of keywords is with a command is unique to one module. In this case, the top layer (Root Keyword) of the command structure may be omitted.

For example, the following two segments of command code are functionally identical:

```
[SOURce[1]:]PATTern:MDENsity[:DENSity] <numeric value>
PATTern:MDENsity <numeric value>
```

Notice that it is not necessary to include the syntax inside the square brackets ([]).

Sending Commands

Commands are sent over the GPIB in the same way that GPIB and IEEE 488.2 common commands are sent. The difference is that the SCPI command is “nested” into the programming language of choice. The programming language of choice may be a language such as HP BASIC, C++, or SICL.

Command Separators

The SCPI command structure is hierarchical and is governed by a number of symbols. For example, a change in the command hierarchy is indicated by a colon, similar level commands are separated by a semi-colon and parameters are separated by a comma. The following command illustrates this:

```
SENSe[1]:PATTern:UPATtern<n>:IDATa [A|B,] <start_bit>, <length_in_bits>,
<block_data>
```

Notice that the command hierarchy is indicated by colons and that the parameters (beginning with [A|B,]), are separated by commas.

For more information, refer to Chapter 2, “SCPI Command Structure” in the following section.

SCPI Command Structure

As previously stated, the SCPI command set has a hierarchical layered structure.

The structure is as follows:

Root Keyword + Command Keyword + Parameter(s)

Root Keyword

The Root Keyword is the top layer in the command structure. It identifies a subsystem within a module, which is contained in the modular measurement system.

Refer to Table 4-1 on page 4-2 in Chapter 4, "System Command Reference Section". Each of the pattern generator and error detector port names identifies a SCPI subsystem in the 86130A, where the port name is used as the Root Keyword for all commands effecting that subsystem.

There are 5 SCPI subsystems not associated with input/output ports:

FETCh/PFETch	Measurement Subsystem
SYSTem	System Subsystem
STATus	Status Subsystem
MMEMory	Mmemory Subsystem
TEST	Test Subsystem
PLUGin	Plugin Subsystem

Some root keywords may be optional if the destination of the command is implicit in the Command Keyword.

NOTE

86130A does not include the DISPlay Subsystem, so the following commands are not supported:
DISPLay:WINDow
DISPLay:WINDow[:RESults] <paramter>
DISPLay:WINDow:CONFig <parameter>

DISPLay:REPort PREVious|CURRent
DISPLay:UPAGE[:DEFine] <parameter>
DISPLay:UPAGE:CLEar3

Command Keyword The layer below the Root Keyword is the Command Keyword. It describes the feature on the system which is to be changed. It will always be present in any command string and may have additional associated commands.

Parameter The command parameters are the lowest layer in the SCPI command structure. They may be required by the Command Keyword and are numeric, string, boolean or block data.

Command Structure Example

Taking the pattern generator pattern selection command as an example, the command structure can be examined.

The pattern command can be illustrated as follows:

Root Keyword	Command Keyword	Sub Command	Parameter(s)
[SOURce[1]:]	PATTern		
		[:SELect]	PRBS<n> PRBS<n> | PRBN<n> | ZSUBstitut<n> | MDENsity<n> | UPATtern<n> | FILENAME, <string>
		[:SELect]?	PRBS<n> | PRBN<n> | ZSUB<n> | MDEN<n> | UPAT

[SOURce [1]:]	This is the top layer of the command structure and identifies the pattern generator source sub-system.
PATTern	This is the next layer and is the equivalent of setting the front panel pattern selection field.
PRBS(n)	This is the parameter required by the PATTern command keyword.

Note

Any optional commands are enclosed in square brackets [ ] and any optional characters are shown in lower case.
A colon indicates a change of level in the command hierarchy. Commands at the same level in the hierarchy may be included in the same command line, if separated by a semi colon.
The bar symbol (|) indicates mutually exclusive commands.

To translate this syntax into a command line, follow the convention that is described above. Remember, however, that the command line can be created in several different ways. It can be created with or without optional keywords, and in a long or short form. The following example gives three possible forms of the command line; all are acceptable.

In long form:

```
OUTPUT 717;”SOURce1:PATTern:SELect PRBS7”
```

In shortform:

```
OUTPUT 717;”SOUR1:PATT:SEL PRBS7”
```

With optional commands removed:

```
OUTPUT 717;”PATTern PRBS7”
```

The long form is the most descriptive form of programming commands in SCPI. It is used for the examples in this manual.

Behavior at Power On

At power-on, the state of the registers and filters will be:

Normal operation,	The initial state of the registers and transition filters will be saved in the event of a power failure.
Initial power-on,	All registers and filters are disabled except the PON, CME and EXE bits of the Standard Event Status Register and its summary bit in the Status Byte. The transition filters will be set to allow all conditions and events to pass.

The event registers and the error queue are cleared at each and every power-up.

Recommended Programming Techniques

Accurate bit error ratio measurements are dependant on good programming techniques. These techniques include such things as making allowances for variations of command behavior and executing instrument setups in recommended order.

Overlapped and Sequential Commands

IEEE 488.2 defines the distinction between overlapped and sequential commands. A sequential command is one which finishes executing before the next command starts executing. An overlapped command is one which does not finish executing before the next command starts executing. The 86130A has the following overlapped commands:

```
SENSe[1]:GATE[:STATe] ON | 1 (when GATE:MODE SINGle)
SENSe[1]:EYE:TCENter|:TCENtre ONCE | ON | 1
SENSe[1]:EYE:ACENter|:ACENtre ONCE | ON | 1
SENSe[1]:EYE:ALIGn:AUTO ONCE | ON | 1
SENSe[1]:EYE:QUICk:TCENter ONCE | ON | 1
SENSe[1]:EYE:QUICk:ACENter ONCE | ON | 1
SENSe[1]:EYE:QUICk:ALIGN:AUTO ONCE | ON | 1
```

NOTE It is not be reliable to use wait statements in the GPIB control program to facilitate the use of overlapped commands.

Because these commands may allow the execution of more than one command at a time, special programming techniques must be used to ensure valid results. The common commands *OPC, *WAI, and *OPC? can be used for this purpose. They help synchronize a device controller with the execution of overlapped commands. The behaviors of these commands, in brief, are as follows:

***OPC**

The *OPC command sets the Operation Complete (OPC) bit of the Standard Event Status Register (SESR) when the No Operation Pending flag is TRUE (No Operation Pending flag is attached to each overlapped command). Until that time, the controller may continue to parse and execute previous commands. It is good technique, then, to periodically poll the OPC bit to determine if the overlapped command has completed.

*WAI

The *WAI commands allows no further execution of commands or queries until the No Operation Pending flag is true, or receipt of a Device Clear (dcas) message, or a power on.

*OPC?

The *OPC? query returns the ASCII character “1” in the Output Queue when the No Operation Pending flag is TRUE. A the same time, it also sets the Message Available (MAV) bit in the Status Byte Register. The *OPC? will not allow further execution of commands or queries until the No Operation Pending flag is true, or receipt of a Device Clear (dcas) message, or a power on.

NOTE The command behaviors described above are for overlapped commands. When the same commands are used with sequential commands, the behaviors may be different.

Operation Pending Events

For 86130A, six conditions can change an operation pending flag. Notice that the first four correspond to the four overlapped commands.:

- A *single, timed* accumulation period has expired.
- The automatic eye-time-centering operation has expired.
- The automatic eye-amplitude-centering operation has expired.
- An automatic alignment has occurred.
- The requested operation failed.
- The operation was aborted by the user.

Programming Example

The following BASIC language example shows how a *WAI command can be used to synchronize the controller and device. In this example, the device is configured to have a single accumulation period of 10 seconds, followed by a query of the bit count. Because “`SENSe1:GATE:STATe ON`” is an overlapped command, the bit count could be queried immediately after the start of accumulation. In this example, however, the query is held off until the accumulation period has elapsed.

```
INTEGER GatePeriod
REAL BitCount
ASSIGN @Bert TO 717
GatePeriod=10 ! seconds
OUTPUT @Bert;"SENSE1:GATE:MODE SING"
OUTPUT @Bert;"SENSE1:GATE:MANNER TIME"
OUTPUT @Bert;"SENSE1:GATE:PER:TIME " <escVAL$(GatePeriod)
OUTPUT @Bert;"SENSE1:GATE:STATE ON" ! run gating
OUTPUT @Bert;"*WAI"
OUTPUT @Bert;"FETCH:SENSE2:BCOUNT?"
ENTER @Bert;BitCount
```

Pattern Changes and Settling Time

Whenever the pattern is changed, there is a settling time for the hardware and its controlling firmware that can last seconds, or even minutes in some cases. This is particularly true for text-based (ASCII) patterns. In these instances, it is good programming technique to hold off any important bit error ratio measurements until conditions have settled.

There are two methods for determining if conditions have settled.

- Read the SYNC LOSS bit (bit 10) using the following command segment:

```
OUTPUT 717;”STATus:QUEStionable:CONDition?’
ENTER 717;Question_con_reg
PRINT Question_con_reg
```

If the variable returns a value that includes bit 10 (“1024”), this is an indication that the new pattern has not yet synchronized. In this case, the settling is not yet complete.

- Query the bit error ratio using a command segment, such as the following:

```
OUTPUT 717;"SENSe1:GATE:STATe ON”
OUTPUT 717;”*OPC?”
OUTPUT 717;"FETCh:SENSe1:ERATio?”
ENTER 717;Settling_BER
PRINT Settling_BER
```

The above programming segment may be repeatedly run until the returned value of Settling_BER is lower than a desired bit error ratio.

Note

Accessing large patterns can take several minutes. Control programs must be prepared for I/O time outs of this order.

Order-Sensitive Commands

Because some instrument-setup commands are actually subsets of other instrument-setup commands, it is good technique to program them in the recommended order. This ensures that the instrument is set up efficiently and error-free.

User Pattern Edits

One way to quickly edit a user pattern is to import a block of data. This can be done by first identifying the user pattern type with the following command:

[SOURce[1]:]PATTern:UPATtern<n>:USE STRaight | APATtern

When the parameter STRaight is selected, the whole of the pattern is repeatedly sent out. When the parameter APATtern is selected, the pattern is composed of two halves.

For user patterns in the STRaight mode, it is recommended that the following four commands be executed *in order*:

1 SOURce1:PATTern:UPATtern<n>:USE STRaight

1 SOURce1:PATTern:UPATtern<n>[:LENGth] <numeric value>:

1 SOURce1:PATTern:UPATtern<n>:DATA <block data>

1 SOURce3:TRIGger:UPATtern<n> <numeric value>

For user-patterns in the APATtern mode, it is recommended that the following five commands be executed *in order*:

1 SOURce1:PATTern:UPATtern<n>:USE APATtern

2 SOURce1:PATTern:UPATtern<n> [:LENGth]<numeric value>

3 SOURce1:PATTern:UPATtern<n> DATA A,<block data>

4 SOURce1:PATTern:UPATtern<n> DATA B,<block data>

5 SOURce3:TRIGger:APATtern<n> ABCHange | SOPattern

The sequences above should also be used when programming with the UFILe: commands. These commands include:

```
[SOURce[1]:]PATTern:UFILe:USE <filename>, STRaight | APATtern
[SOURce[1]:]PATTern:UFILe:[:LENGth] <filename>, <numeric value>
[SOURce[1]:]PATTern:UFILe:DATA [A | B,]<filename>, <block data>
```

Alternate Pattern Setup

An alternate pattern is a user pattern that is composed of two halves. You can control the output of these two halves internally or externally. Internal control indicates that control of the alternate pattern output is from the instrument front panel or from remote programming commands. External control indicates that the alternate pattern output is dependant on the voltage level at **Aux In**. This is a fundamental distinction because alternate pattern output behavior is dependent on the parameter in the APCHange:SOURce command and its interplay with other commands that follow.

It is recommended that the following four commands be executed *in order*:

1. SOURce[1]:PATTern:UPATern<n>:USE APATtern
2. [SOURce[1]:]PATTern:APCHange:SOURce EXTernal | INTernal
3. [SOURce[1]:]PATTern:APCHange:MODE ALTernate | ONEShot | LLEVel | REDGe
4. [SOURce[1]:]PATTern:APCHange:SELect AHALf | BHALf | ABHAlf

The combination of the settings for MODE and SOURce dictate alternate pattern output behavior.

The following table gives an indication of the differences.

Table 2-1.

	SOURce INTernal	SOURce EXTernal	Output Behavior
MODE ALTernate		X	The signal at **Aux In** controls which half of the pattern is output. If **Aux In**=logic high, pattern B is sent. If **Aux In**=logic low, pattern A is sent.
	X		The :APCHange:SELect command controls which half of the pattern is output. The selections are pattern A only (AHALf) pattern B only (BHALf) alternating A B (ABHAlf)
MODE ONEShot		X	When a rising edge occurs at the **Aux In**, a single occurrence of pattern B is inserted into a continuous pattern A output.
	X		The :APCHange:IBHalf command is used to insert one instance of pattern B into the output.
MODE LLEVel		X	The signal at **Aux In** controls which half of the pattern is output. If **Aux In**=logic high, pattern B is sent. If **Aux In**=logic low, pattern A is sent.
	X		The :APCHange:SELect command controls which half of the pattern is output. The selections are pattern A only (AHALf) pattern B only (BHALf) alternating A B (ABHAlf)
MODE REDGe		X	When a rising edge occurs at the **Aux In**, a single occurrence of pattern B is inserted into a continuous pattern A output.
	X		This combination is not supported.

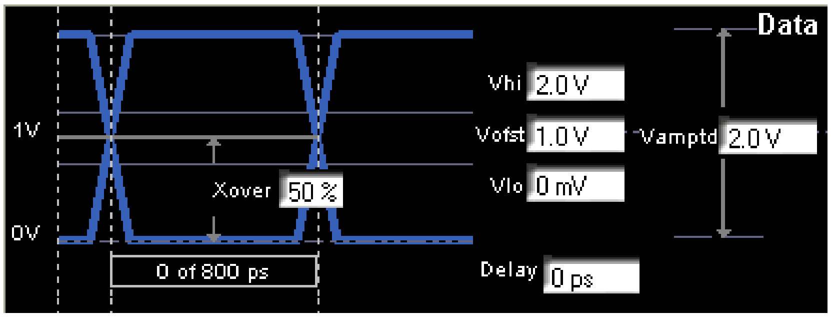
Automatic Level Controls

The data and clock outputs are coupled together in the following ways:

- Vhi, Vlo, Vofst are coupled together at all times. For example, if Vhi is incremented by 10mV, the values of Vlo and Vofst are incremented by 10mV as well. When changing either Vhi, Vlo or Vofst, the value of Vamptd remains fixed. Resolution of Vhi, Vlo and Vofst is 10mV.
- Changing Vamptd also changes the values of Vhi and Vlo. The value of Vofst remains fixed. The resolution of Vamptd is 20mv.

The instrument supports only those parameters which set the output levels within supportable range. If a level is set outside its limit, the instrument behavior is the following:

- If Vhi, Vlo and Vofst are set outside their limit, then they are set to their maximum available level. Vamptd is maintained constant.
- If Vamptd is set outside its limit, then it is set to the maximum available level without taking Vhi or Vlo outside their ranges.


Data

Vhi 2.0 V

Vofst 1.0 V

Vamptd 2.0 V

Vlo 0 mV

Xover 50 %

0 of 800 ps

Delay 0 ps

Output Level Adjustments

The values of Vhi are limited when they are below -1V. This is because voltage characteristics such as amplitude and offset are coupled together; refer to "Automatic Level Controls" on page 2-19. In such cases, it is important to use recommended programming techniques when adjusting output levels. This ensures an error-free adjustment.

The recommended programming technique is described as follows:

- If the desired amplitude is higher, send the Vhi value first.
- If the desired amplitude is lower, send the Vamptd value first.

or

- If Vhi is below -1V, program Vhi to 0 to -1V first, followed by the desired values of Vamptd and Vhi.

For your convenience, the following table summarizes the voltage levels that are supported.

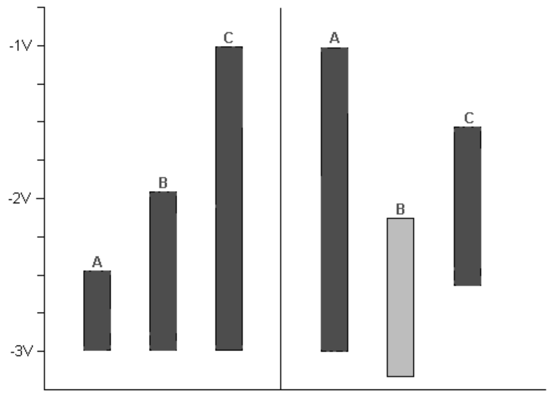
Termination	**Min Vamptd**	**Max Vamptd**	**Max Vhi**	**Min Vlo**
0 or +1.3V	0.5V	2.0V	2.5V	-3V
-2V	0.5V	2.0V	0.0V	-3V

The diagram below illustrates why the recommended programming technique is important. The left side of the diagram illustrates a group of valid Vamptd and Vhi. The right side of the diagram illustrates the same group after voltage level adjustments.

In the case of the “A” level, the amplitude is increased from 0.5V to 2.0V. Here, the programmer has sent the command for the new value of Vhi, followed by the desired value for Vamptd. The result is a successful adjustment.

In the case of the “B” level, the value of Vhi is changed without first programming Vhi to the range of 0 to -1V. With the resulting adjustment, the amplitude crosses the lower limit of -3V, creating an error.

In the case of the “C” level, the amplitude is decreased from 2.0V to 1.0V. Here, the programmer has sent the command for the desired Vamptd first. The result is a successful adjustment.



Interrogating the Instrument Status

Introduction

This section explains how to use the status reporting features which are contained in the 86130A.

It explains the structure of the internal registers with examples on how to interrogate them. It also explains the concept of interrupt programming using the Service Request.

The section covers the following topics:

- 86130A Status Reporting
- Status Register Group Model
- 86130A Register Model
- Description of the Status Registers
- Interrupt Programming and using the Service Request

86130A Status Reporting

The 86130A has status reporting features which give important information about events and conditions within the instrument. For example, setting a flag may indicate the end of a measurement or perhaps a command error. To access this information requires interrogating a set of registers using Standard Commands for Programmable Instruments (SCPI).

Internal Registers

The registers contained in the 86130A are as follows:

Internal Registers
Status Byte
Standard Event Status
Questionable Data Status
Operation Status
Clock Loss

The internal registers are read using a combination of SCPI common commands and SCPI status commands. For more information about reading the instrument's registers, refer to "Status Byte Register Group" on page 3-8.

Status Register Group Model

SCPI guidelines specify a register group model which is the building block of the SCPI status reporting system. The SCPI generalized status register group model is shown in Figure 3-1.

.

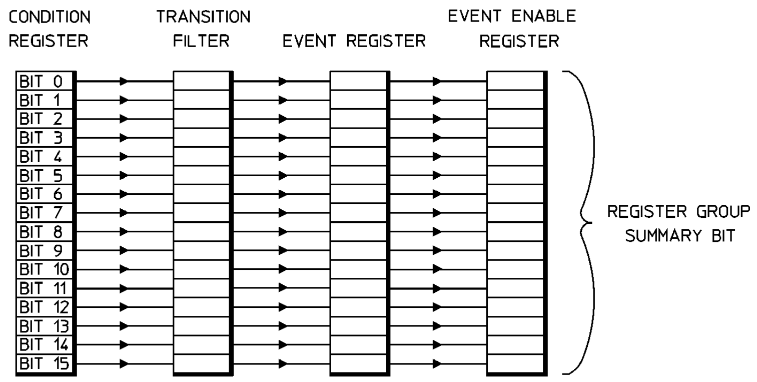


Figure 3-1. Generalized Status Register Group

Condition Register	The condition register monitors the hardware and firmware status of the instrument. There is no latching of conditions in this register; it is updated in real time. This register is *not* in the Standard Event Status or Status Byte Register groups.
Transition Filter	The transition filter determines whether positive or negative transitions (0 to 1 or 1 to 0) in the condition register sets the event register. This register is *not* in the Standard Event Status or Status Byte Register groups.
Event Register	The event register latches a wide variety instrument events, only if masked by an enable register. For the Questionable Status and Operation Status Registers, the event register is preceded by a condition register and transition filter.
Enable Register	The enable register acts as a mask on the event register. It determines which bits in the event register set the summary bit in the Status Byte.

This reporting structure is the basis of generating interrupts or service requests. For more information, refer to "Service Request Example" on page 3-26

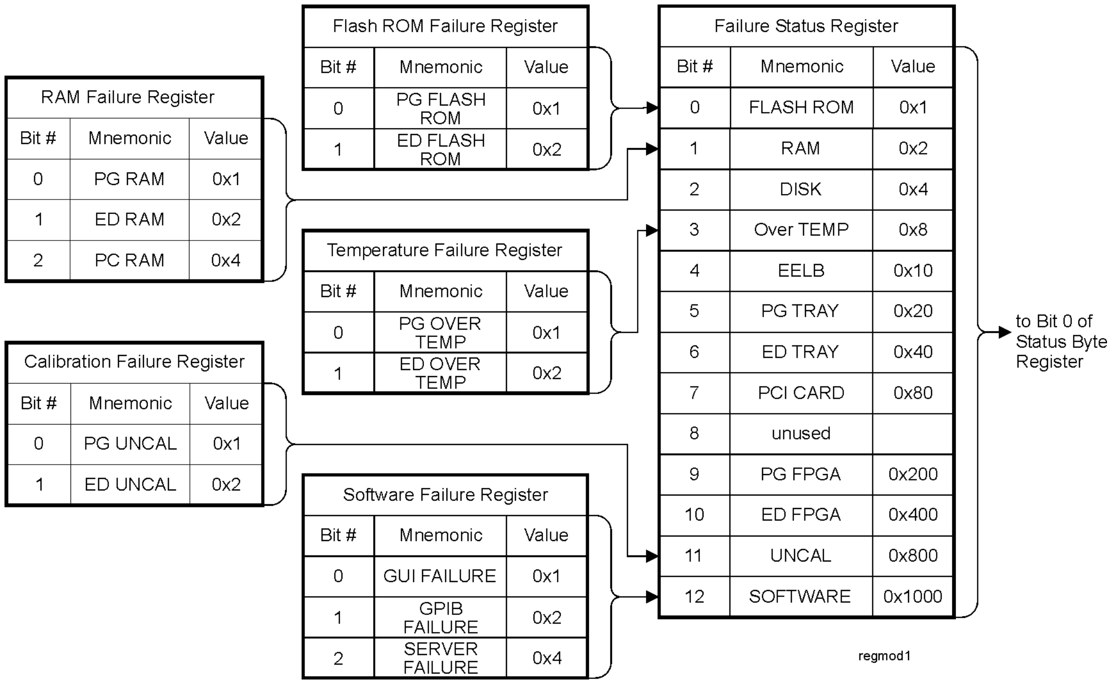
Interrogating Register Groups

The status register groups are interrogated using the STATus commands. The format consists of a command identifier, register group identifier, and register title. The following example queries the event register in the Operation Status Register group:

```
STATus:OPERation[:EVENt]?
```

86130A Register Model

The 86130A register model is shown below and on the following page. Full descriptions of each register follow, beginning with the "Status Byte Register Group" on page 3-8.



RAM Failure Register

Bit #	Mnemonic	Value
0	PG RAM	0x1
1	ED RAM	0x2
2	PC RAM	0x4

Calibration Failure Register

Bit #	Mnemonic	Value
0	PG UNCAL	0x1
1	ED UNCAL	0x2

Flash ROM Failure Register

Bit #	Mnemonic	Value
0	PG FLASH ROM	0x1
1	ED FLASH ROM	0x2

Temperature Failure Register

Bit #	Mnemonic	Value
0	PG OVER TEMP	0x1
1	ED OVER TEMP	0x2

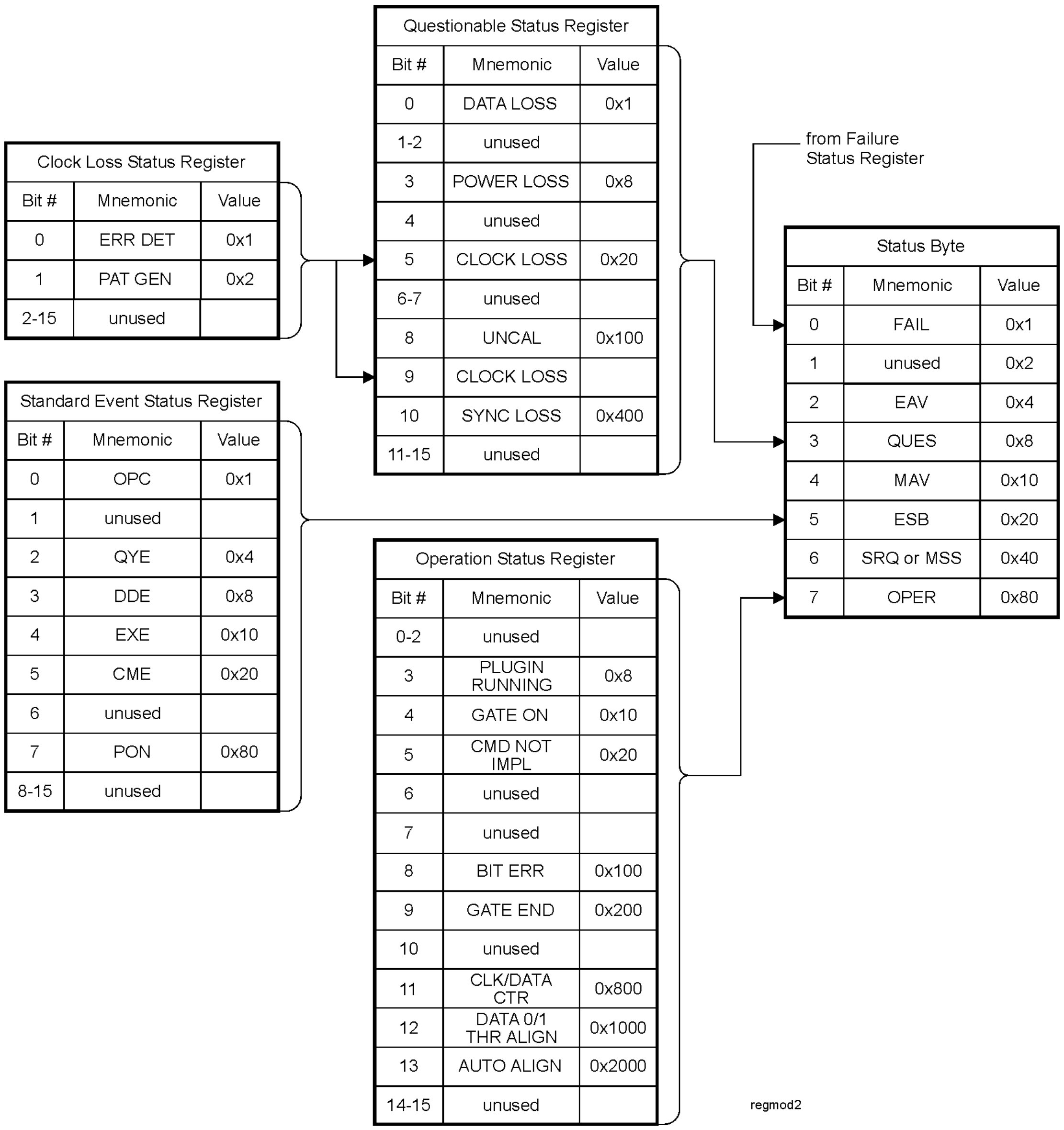
Software Failure Register

Bit #	Mnemonic	Value
0	GUI FAILURE	0x1
1	GPIB FAILURE	0x2
2	SERVER FAILURE	0x4

Failure Status Register

Bit #	Mnemonic	Value
0	FLASH ROM	0x1
1	RAM	0x2
2	DISK	0x4
3	Over TEMP	0x8
4	EELB	0x10
5	PG TRAY	0x20
6	ED TRAY	0x40
7	PCI CARD	0x80
8	unused	
9	PG FPGA	0x200
10	ED FPGA	0x400
11	UNCAL	0x800
12	SOFTWARE	0x1000

Figure 3-2. Error Performance Analyzer Register Model



Questionable Status Register		
Bit #	Mnemonic	Value
0	DATA LOSS	0x1
1-2	unused	
3	POWER LOSS	0x8
4	unused	
5	CLOCK LOSS	0x20
6-7	unused	
8	UNCAL	0x100
9	CLOCK LOSS	
10	SYNC LOSS	0x400
11-15	unused	

Clock Loss Status Register		
Bit #	Mnemonic	Value
0	ERR DET	0x1
1	PAT GEN	0x2
2-15	unused	

Standard Event Status Register		
Bit #	Mnemonic	Value
0	OPC	0x1
1	unused	
2	QYE	0x4
3	DDE	0x8
4	EXE	0x10
5	CME	0x20
6	unused	
7	PON	0x80
8-15	unused	

Status Byte		
Bit #	Mnemonic	Value
0	FAIL	0x1
1	unused	0x2
2	EAV	0x4
3	QUES	0x8
4	MAV	0x10
5	ESB	0x20
6	SRQ or MSS	0x40
7	OPER	0x80

Operation Status Register		
Bit #	Mnemonic	Value
0-2	unused	
3	PLUGIN RUNNING	0x8
4	GATE ON	0x10
5	CMD NOT IMPL	0x20
6	unused	
7	unused	
8	BIT ERR	0x100
9	GATE END	0x200
10	unused	
11	CLK/DATA CTR	0x800
12	DATA 0/1 THR ALIGN	0x1000
13	AUTO ALIGN	0x2000
14-15	unused	

Error Performance Analyzer Register Model (continued)

Status Byte Register Group

The Status Byte is the summary register to which the other registers report. Each reporting register is assigned a bit in the Status Byte Register. When a bit is selected by a mask, only the corresponding reporting register is permitted to pass its contents in a service request. When interrogated, the Status Byte Register returns a weighted-value number which represents the instrument status in summarized form.

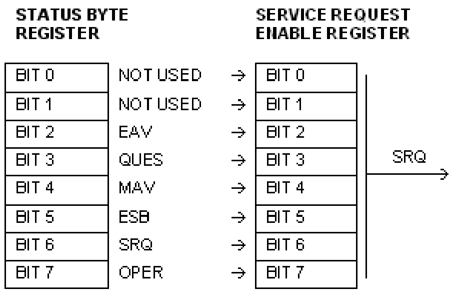


Figure 3-3. Status Byte Register

Example

Note

For a more detailed description on service request programming, refer to "Service Request Example" on page 3-26.

The bits in the Status Byte Register are defined as follows:

Table 3-1. Status Byte Register

Bit #	Mnemonic	Description	Bit Value
0	FAIL	Not Used	1
1	-	Not used.	
2	EAV	Error is available.	4
3	QUES	Questionable Data Status register summary bit.	8
4	MAV	Output queue summary bit.	16
5	ESB	Standard Event register summary bit.	32
6	SRQ or MSS	SRQ or master status summary bit.	64
7	OPER	Operation Status register summary bit.	128

EAV Summary Bit Bit 2 indicates that the error queue contains one or more messages.

QUES Summary Bit Bit 3 indicates that a bit has been set in the Questionable Data Status register. The bits in the Questionable Data Status register indicate when a signal is of questionable quality.

MAV Summary Bit Bit 4 is the message available summary bit. This bit remains set until all the output messages are read from the instrument. The instrument stores its messages in an output queue. These messages are read by addressing the instrument to talk and reading the data. The availability of this data is summarized by the MAV bit.

ESB Summary Bit	Bit 5 indicates that a bit in the Standard Event register has been set.
SRQ or MSS Summary Bit	Bit 6 of the Status Register has two definitions, depending on the method used to access the register. If the value of the register is read using the serial poll (SPOLL), bit 6 is referred to as the Service Request (SRQ) Bit. It is used to interrupt and inform the active controller that the instrument has set the service request control line, SRQ If the register is read using the common command *STB? , then bit 6 is referred to as the master summary bit or MSS bit. This bit indicates that the instrument has requested service. The MSS bit is not cleared when the register is read using the *STB? command. It always reflects the current status of all the instrument's status registers.
OPER Summary Bit	Bit 7 is the Operation Status register summary bit.

Reading the Status Byte Register
The Status Byte Register may be read in two methods; serial polling and common command. The following two programming segments read the Status Byte and place the contents in the variable Status_Value.

```
SPOLL          10 Status_value= SPOLL(717)
               20 PRINT Status_value

*STB?          10 OUTPUT 717;"*STB?"
               20 ENTER 717;Status_value
```

Status Byte Service Request Enable Register

The Service Request Enable Register is an 8-bit register which acts as a mask on the Status Byte. The Service Request Enable register is programmed using the SCPI common command *SRE. This enable register is programmed with a value that corresponds to the desired reporting register.

For example, to have the instrument issue a service request when bits are set in the Questionable Status Register and Operation Status Register, bits 3 and 7 must be set in the Service Request Enable register.

The following code segment sets bit 3 and 7 in the Service Request Enable register. Notice that the value passed is the weighted sum of the two bits.

```
OUTPUT 717;"*SRE 136"
```

Standard Event Status Register Group

The Standard Event Status register group is a 16 bit register group which gives general-purpose information about the instrument. It is programmed using SCPI common commands.

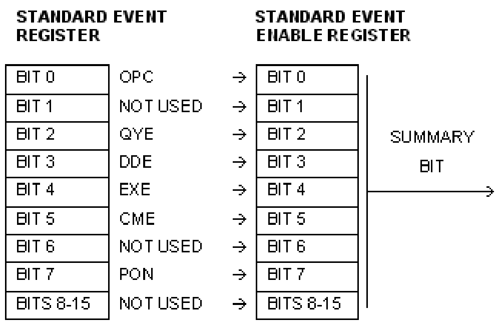


Figure 3-4. Standard Event Status Register

Note

This register is compatible with the generalized status register model. It is comprised of an event and enable register, but no condition register or transition filter. All positive transitions in this register are latched.

The bits in the Standard Event Status register group are defined as follows:

Table 3-2. Standard Event Status Register

Bit #	Mnemonic	Description	Bit Value
0	OPC	This bit is the operation complete bit.	1
1	-	Not used.	
2	QYE	This bit is the query error bit.	4
3	DDE	This bit is the device dependent Error bit.	8

Table 3-2. Standard Event Status Register

Bit #	Mnemonic	Description	Bit Value
4	EXE	This bit is the execution error bit.	16
5	CME	This bit is the command error bit.	32
6	-	Not used.	64
7	PON	This bit is the power on bit.	128
8-15	-	These bits are not used.	

OPC Summary Bit	Bit 0 is the operation complete bit. It is set in response to the *OPC command, only if the instrument has completed all its pending operations. *OPC is an overlapped command. For more information, refer to “Overlapped and Sequential Commands” on page 2-13
QYE Summary Error Bit	Bit 2 is the query error bit. It indicates that there is a problem with the output data queue. There has been an attempt to read the queue when it is empty, the output data has been lost, or the query command has been interrupted.
DDE Summary Bit	Bit 3 is the device-dependent-error bit. It is set when an instrument-specific error has occurred.
EXE Summary Bit	Bit 4 is the execution error bit. It is set when a command (GPIB instrument specific) cannot be executed due to an out of range parameter or some instrument condition existing that prevents the execution.
CME Summary Bit	Bit 5 is the command error bit. It is set whenever the instrument detects an error in the format or content of the program message (usually a bad header, missing argument, or wrong data type etc.).
PON Summary Bit	Bit 7 is the power-on bit. It is set each time the instrument is powered from off to on.

Reading the Standard Event Status Register

The Standard Event register can be interrogated using the *ESR? common command. The register is cleared after it is read.

The following code segment interrogates the Status Event Register, places the contents in a variable, and prints the variable:

```
OUTPUT 717;"*ESR?"
ENTER 717;Event_reg$
PRINT Event_reg$
```

The Standard Event Register may also be cleared without interrogating it. This is done by using the "*CLS" command.

Standard Event Enable Register

The Standard Event Enable register is a 16 bit register which acts as a mask on the Standard Event Status register. It allows one or more event bits in the Standard Event register to set the ESB summary bit (bit 5) in the Status Byte.

For example, if bit 0 is set in the Standard Event Enable register, then when the OPC bit in the Standard Event register sets, the ESB summary bit is set in the Status Byte.

The Standard Event Enable register is set using the "*ESE" command.

The following code segment sets bit 0 and 3 in the Standard Event Enable register: Notice that the value 9 is the weighted sum of bits 0 and 3.

```
OUTPUT 717;"*ESE 9"
```

Clock Loss Register

The Clock Loss Register group indicates whether the pattern generator or error detector has experienced a clock signal loss. The output of this register sets bit 5 and 9 (Clock Loss) in the Questionable Status Register.

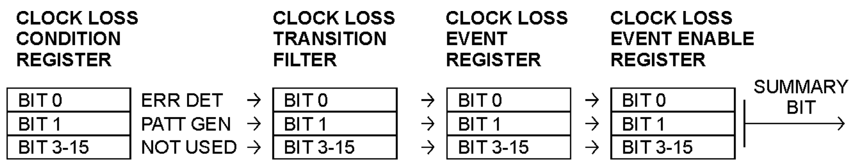

CLOCK LOSS CONDITION REGISTER
CLOCK LOSS TRANSITION FILTER
CLOCK LOSS EVENT REGISTER
CLOCK LOSS EVENT ENABLE REGISTER
BIT 0
BIT 1
BIT 3-15
ERR DET
PATT GEN
NOT USED
SUMMARY BIT

Table 3-3. Clock Loss Register

Bit #	Description	Bit Value
0	ERR DET Clock Loss	1
1	PAT GEN Clock Loss	2

ERR DET	Bit 0 sets when the error detector detects a clock loss condition.
PAT GEN	Bit 1 sets when the pattern generator detects a clock loss condition.

Clock Loss Condition Register

The condition register latches when the pattern generator or error detector experiences a clock loss. The register can be interrogated in the following code segment example:

```
OUTPUT 717;"STATus:CLOSs:CONDition?
ENTER 717;Closs_con_reg
PRINT Closs_con_reg
```

Clock Loss Event Register

The event register latches the bit(s) that have been set in the condition register and passed along by the enable register and transition filter. The event register can be interrogated in the following code segment example:

```
OUTPUT 717;"STATus:CLOSs:EVENt?"
ENTER 717;Closs_evt_reg
PRINT Closs_evt_reg
```

Clock Loss Transition Filter

This transition filter is set in the same way as the Questionable Data Transition Filter. For more information, refer to "Questionable Data Transition Filter" on page 3-20

The following code segment sets the Clock Loss Transition Filter to pass positive transitions from bit 1 (Pat Gen):

```
OUTPUT 717;"STATus:CLOSs:PTRAnsition 2".
```

Clock Loss Event Enable Register

This enable register acts as a mask on the Clock Loss Event register.

The following code segment sets bits 0 and 1 (Err Det and Pat Gen) in the Clock Loss Event Enable register. Notice that the number passed is the weighted sum of the two values:

```
OUTPUT 717;"STATus:CLOSs:ENABle 3"
```

In the above code segment, if either the Err Det or Pat Gen bit is set in the Clock Loss Event register, the appropriate contents are passed to bits 5 and 9 in the Questionable Data Register.

Questionable Status Register Group

The Questionable Status Register group is comprised of 16 bit registers. Its output indicates that the measurement is of questionable quality. The registers of the group are programmed using the SCPI Status command set.

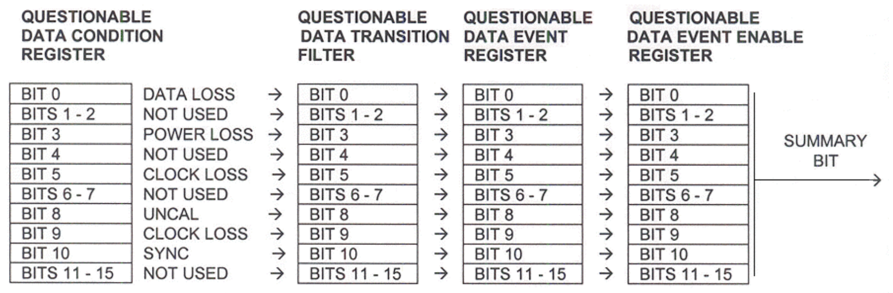


Figure 3-5. Questionable Data Status Register Group

The Questionable Data Status register group is compatible with the SCPI Status Register model and is defined as follows:

Table 3-4. Questionable Data Condition Register

Bit #	Mnemonic	Description	Bit Value
0	DATA LOSS	This bit indicates data loss at the error detector.	1
1-2	not used.		
3	POWER LOSS	This bit indicates power loss at error detector or pattern generator.	8
4	not used		
5	CLOCK LOSS	This bit indicates a loss of clock signal at the error detector or pattern generator	32
6-7	not used		

Table 3-4. Questionable Data Condition Register

Bit #	Mnemonic	Description	Bit Value
8	UNCAL	This bit indicates a mismatch between installed hardware and calibration tables.	256
9	CLOCK LOSS	This bit indicates a loss of clock signal at the error detector or pattern generator	512
10	SYNC LOSS	This bit indicates that the incoming pattern does not match reference pattern.	1024
11-15	not used.		

DATA LOSS	This bit is set when the data source is turned off, not connected, or your cables or device is faulty. This bit can also set when the 0/1 threshold is not in the eye limits of the incoming data signal. In this last case, use Auto Align or select Avg 0/1 Threshold.
POWER LOSS	This bit is set when the instrument experiences a power loss.
CLOCK LOSS	This bit is set when the pattern generator receives no external clock signal or the error detector receives no clock input signal. To find out which of the 2 events is causing this bit to set, you must poll the Clock Loss Status Register; refer to “Clock Loss Register” on page 3-15
UNCAL	This bit is set when the serial number of the installed pattern generator or error detector tray does not match the calibration file in the instrument.
SYNC LOSS	This bit is set when the error detector pattern does not match the incoming data pattern or the BER of your device is higher than the sync threshold.

Questionable Data Condition Register

The condition register indicates what problem source is making the measurement quality questionable. The register can be interrogated in the following code segment example:

```
OUTPUT 717;”STATus:QUEStionable:CONDition?’
ENTER 717;Question_con_reg
PRINT Question_con_reg
```

Questionable Data Event Register

The event register latches which bit has been set in the condition register. The event register can be interrogated in the following code segment example:

```
OUTPUT 717;”STATus:QUEStionable:EVENt?”
ENTER 717;Question_evt_reg
PRINT Question_evt_reg
```

Questionable Data Transition Filter

This transition filter is set using the ":PTRAnsition" and ":NTRAnsition" commands. The transition filter can be set to pass positive transitions, negative transitions, *or both*. The default setting of the transition filter is to pass positive transitions only.

The easiest way to understand transition filters is by example. The following code segment is a typical manipulation of the filters:

```
10 OUPTPUT 717;"*RST"
20 OUPTPUT 717;"STATus:QUEStionable:PTRAnsition?
30 ENTER 717;pos_trans_total
40 PRINT pos_trans_total
50 OUPTPUT 717;"STATus:QUEStionable:NTRAnsition?
60 ENTER 717;neg_trans_total
70 PRINT neg_trans_total
80 OUPTPUT 717;"STATus:QUEStionable:NTRAnsition 1024
90 OUPTPUT 717;""STATus:QUEStionable:PTRAnsition 31744
```

Line 10	This line resets the instrument. All transition filters are set to pass only positive transitions.
Line 20	This line queries the number of bits in the transition filter that are set to pass positive transitions.
Line 30	This line places the returned number into the variable "pos_trans_total."
Line 40	This line prints "32768." All filter transitions are set positive only by default.
Line 50	This line queries the number of bits in the transition filter that are set to pass negative transitions.
Line 60	This line places the returned number into the variable "neg_trans_total."
Line 70	This line prints "0." No filter transitions are set negative at reset.
Line 80	This line sets only bit 10 (Sync Loss) to a negative filter transition.

Line 90 This line turns on all positive filter transitions, *except* at bit 10 (Sync Loss). Bit 10 is now set to pass only negative transitions.

NOTE If you wish to reset all bits in a transition filter to negative or positive, pass the number 0. For example, the following programming segment disables the positive filter transitions for the entire Questionable Data Transition Filter Register:
OUTPUT 717;”STATus:QUEStionable: PTRAnsition 0”

Questionable Data Event Enable Register

This enable register acts as a mask on the Questionable Data Event register.

The following code segment sets bits 3 and 5 (Power Loss and Clock Loss) in the Questionable Data Event Enable register. Notice that the number passed is the weighted sum of the two values assigned to those locations:

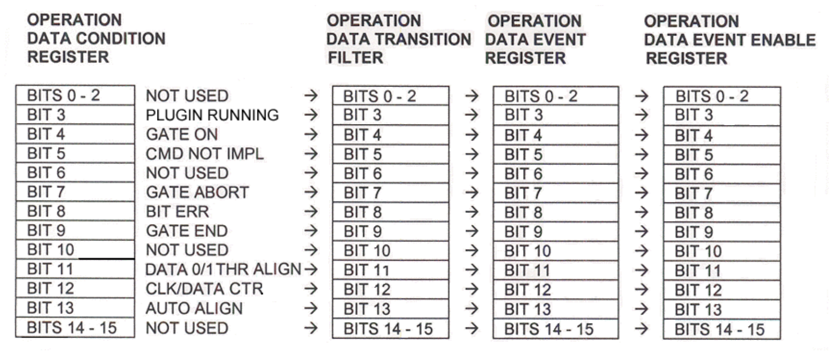
```
OUTPUT 717;”STATus:QUEStionable:ENABle 40”
```

In the above code segment, if either the Power Loss or Clock Loss bit is set in the Questionable Data Event register, the appropriate contents are passed to the QUES summary bit (bit 3).

Operation Status Register Group

The Operation Status Register group is comprised of 16 bit registers, of which only 7 bits are used. Its output gives information about the current operation the instrument is performing. The registers of the group are programmed using the SCPI Status command set.

The Operation Status register group is defined as follows:



OPERATION DATA CONDITION REGISTER			OPERATION DATA TRANSITION FILTER		OPERATION DATA EVENT REGISTER		OPERATION DATA EVENT ENABLE REGISTER
BITS 0 - 2	NOT USED	→	BITS 0 - 2	→	BITS 0 - 2	→	BITS 0 - 2
BIT 3	PLUGIN RUNNING	→	BIT 3	→	BIT 3	→	BIT 3
BIT 4	GATE ON	→	BIT 4	→	BIT 4	→	BIT 4
BIT 5	CMD NOT IMPL	→	BIT 5	→	BIT 5	→	BIT 5
BIT 6	NOT USED	→	BIT 6	→	BIT 6	→	BIT 6
BIT 7	GATE ABORT	→	BIT 7	→	BIT 7	→	BIT 7
BIT 8	BIT ERR	→	BIT 8	→	BIT 8	→	BIT 8
BIT 9	GATE END	→	BIT 9	→	BIT 9	→	BIT 9
BIT 10	NOT USED	→	BIT 10	→	BIT 10	→	BIT 10
BIT 11	DATA 0/1 THR ALIGN	→	BIT 11	→	BIT 11	→	BIT 11
BIT 12	CLK/DATA CTR	→	BIT 12	→	BIT 12	→	BIT 12
BIT 13	AUTO ALIGN	→	BIT 13	→	BIT 13	→	BIT 13
BITS 14 - 15	NOT USED	→	BITS 14 - 15	→	BITS 14 - 15	→	BITS 14 - 15

Figure 3-6. Operation Status Register Group

The bit in the Operation Status register group are defined as follows:

Table 3-5. Operation Status register

Bit #	Mnemonic	Description	Bit Value
0-2	These bits are not used.		
3	PLUGIN RUNNING	This bit indicates that a plugin is processing an overlapped command.	8
4	GATE ON	This bit indicates an accumulated measurement is in progress.	16
5	CMD NOT IMPL	Command is not implemented	32
6	not used		
7	GATE ABORT	This bit indicates accumulation has been aborted by the user.	128
8	BIT ERR	This bit indicates that a bit error has occurred.	256
9	GATE END	This bit indicates that a repetitive measurement period has ended.	512
10	not used		
11	CLK/DATA CTR	This bit indicates that the clock/data alignment is in progress.	2048
12	DATA 0/1 THR ALIGN	This bit indicates that the 0/1 threshold center align is in progress.	4096
13	AUTO ALIGN	This bit indicates that auto alignment is in progress.	8192
14-15	These bits are not used.		

PLUGIN RUNNING	This bit indicates that a plugin is processing an overlapped command.
GATE ON	This bit indicates that an accumulation period is in progress.
CMD NOT IMPL	This bit indicates that the executed SCPI command is not implemented. Event register only for this bit.
GATE ABORT	This bit indicates that the current accumulation period has been aborted.
BIT ERR	This bit indicates that the instrument has detected a bit error.
GATE END	This bit indicates that the current repetitive accumulation period has ended.
CLK DATA CTR	This bit indicates that the clock/data alignment is in progress.
DATA 0/1 THR ALIGN	This bit indicates that the 0/1 threshold alignment is in progress.
AUTO ALIGN	This bit indicates that the auto align is in progress.

Operation Data Condition Register

The condition register indicates the current operation that the instrument is performing. The register can be interrogated using the following code segment example:

```
OUTPUT 717;”STATus:OPERation:CONDition?”
ENTER 717;Operation_con_reg
PRINT Operation_con_reg
```

Operation Data Event Register

The event register latches which bit has been set in the condition register. The event register can be interrogated in the following code segment example:

```
OUTPUT 717;”STATus:OPERation:EVENt?”
ENTER 717;Operation_con_reg
PRINT Operation_con_reg
```

Operation Data Transition Filter

This transition filter is set in the same way as the Questionable Data Transition Filter. For more information, refer to "Questionable Data Transition Filter" on page 3-20

The following code segment sets the Operation Data Transition Filter to pass positive transitions from bit 8 (Bit Err):

```
OUTPUT 717;"STATus:OPERation:PTRAnsition 256"
```

Operation Data Event Enable Register

This enable register acts as a mask on the Operation Data Event Register.

The following code segment sets bits 7 and 9 (Gate Abort and Gate End) in the Operation Data Event Enable register. Notice that the number passed is the weighted sum of the two values assigned to those locations:

```
OUTPUT 717;"STATus:OPERation:ENABle 640"
```

In the above code segment, if either the Gate Abort or Gate End bit is set in the Operation Data Event Register, the appropriate contents are passed to the OPER summary bit (bit 7) in the Status Byte register.

Use of Interrupts

You may want to know when a particular event occurs, without having to continually poll the reporting register. The best way to do this is with the use of interrupts.

Service Request Example

Interrupts or Service Requests (SRQ) allow the instrument to pause the controller when the contents of a particular register change. The controller can then suspend its present task, service the instrument, and then return to its initial task.

The basic steps involved in generating a service request (SRQ) are as follows:

- Decide which particular event should trigger a service request.
- Locate the corresponding status register.
- Set the transition filter to pass the chosen transition of that event.
- Set the enable register from that register group to pass that event to set the summary bit in the Status Byte Register.
- Set the Status Byte Enable Register to generate an SRQ on the chosen summary bit being set.

The process is best explained by looking at an actual example. The following example generates an SRQ from an event in the Operating Status group.

The following example causes the error detector to generate a service request at the end of a measurement period using bit 9 of the Operation Status. See Figure 3-7.

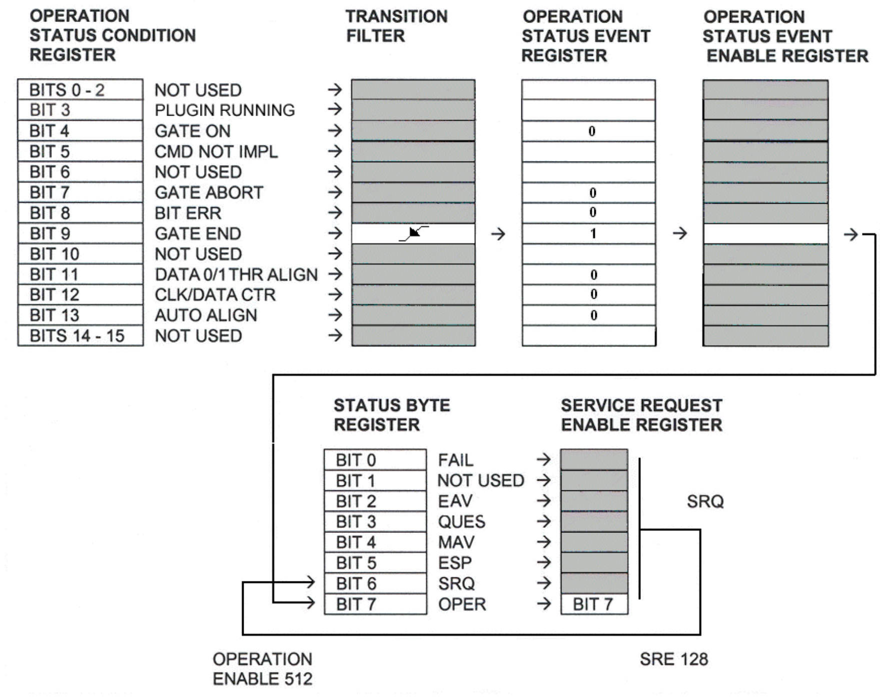


Figure 3-7. Service Request Illustration

Note

The SRQ enable bit, bit 6, of the Status Byte is the master status summary bit and will automatically be set on the occurrence of a service request.

The basic steps involved in setting the instrument to generate this service request are as follows

1 Set the transition filter to pass the desired transition.

Note

The default condition of the transition filters is to pass *only* positive transitions.

2 Program the Operation Status Event Enable Register to allow bit 9 (GATE END) in

the event register to set the summary bit in the Status Byte Register.

3 Program the Service Request Enable Register to generate a service request when the Operation Status summary bit (OPER) is set in the Status Byte register.

Translating these three steps into to SCPI command lines it appears as follows:

```
10 Error_det=717
15 OUTPUT Error_det;"STATus:OPERation:PTRansition 512"
20 OUTPUT Error_det;"STATus:OPERation:NTRansition 0"
30 OUTPUT Error_det;"STATus:OPERation:ENABle 512"
40 OUTPUT Error_det:"*SRE 128"
50 END
```

Line 10	This line assigns the error detector address to the variable Error_det.
Line 15	This line enables the passing of a positive transition through the transition filter at bit 9 (GATE END). All other bits have their position transitions disabled.
Line 20	This line disables all negative transitions through the transition filter.
Line 30	This line sets bit 9 (GATE END) in the Operation Status Enable Register.
Line 40	This line sets the Service Request Enable Register to produce a service request if bit 7 (OPER) is set in the Status Byte Register.
Line 50	This line terminates the program.

4

System Command Reference Section

Command Definition Quick Reference

Table 4-1. Definition of Input/Output Ports

Port	**Pattern Generator**		**Error Detector**	
1	Data output port:	SOURce1	Data input port:	SENSe1
		OUTPut1		INPut1
2	Clock output port:	SOURce2	Clock input port:	SENSe2
		SOURce9		INPut2
		OUTPut2		
3	Trigger output port:	SOURce3		
6	Clock input port:	SENSe6		
7			Trigger output port:	SOURce7
8			Error output port: (commands not implemented)	OUTPut8
9				
10	$\overline{\text{Data}}$ out port:	SOURce10		
		OUTPut10		
11	$\overline{\text{Clock}}$ out port:	SOURce11		
		OUTPut11		

Definition of Store Numbers

Store numbers specify how numerical data is returned by the instrument.

NRf	This number only interprets 0 (OFF) and 1 (ON)
NR1	This number is positive or negative. N = 0 through 9.
NR2	This number is positive or negative and adds a decimal. N = 0 through 9.
NR3	This number is positive or negative, may add a decimal, and adds the capability of an exponent. N = 0 through 9.

IEEE 488.2 Common Commands and Queries

IEEE 488.2 defines commands that begin with *. Some of these are mandatory in all instruments and others are optional.

Mandatory Common Commands and Queries

The 86130A supports the following IEEE 488.2 mandatory commands. Mandatory commands give all SCPI-compliant instruments a common "look and feel".

`*CLS`	Clear Status Command
`*ESE`	Standard Event Status Enable Command
`*ESE?`	Standard Event Status Enable Query
`*ESR?`	Standard Event Status Register Query
`*IDN?`	Identification Query
`*OPC`	Operation Complete Command
`*OPC?`	Operation Complete Query
`*RST`	Reset Command
`*SRE`	Service Request Enable Command
`*SRE?`	Service Request Enable Query
`*STB?`	Read Status Byte Query
`*TST?`	Self-Test Query
`*WAI`	Wait-to-Continue Command

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
`*CLS`	`---`	`---`	`event; no query`
`*ESE`	`<numerical value>`		
`*ESE?`		`<NR1>`	
`*ESR?`		`<NR1>`	`query only`
`*IDN?`		`<character data>`	`query only`
`*OPC`	`---`		
`*OPC?`		`<NR1>`	
`*RST`			`event; no query`
`*SRE`	`<numerical value>`		
`*SRE?`		`<NR1>`	
`*STB?`		`<NR1>`	`query only`
`*TST?`		`<NR1>`	`query only`
`*WAI`			`command only`

*CLS

This commands clears all status data structures in a device. For the 86130A, these registers include:

SESR	IEEE 488.2
OPERation Status Register	SCPI
QUEStionable Status Register	SCPI

Execution of *CLS also clears any additional status data structures implemented in the device. The corresponding enable registers are unaffected.

*ESE <numerical value>

The Standard Event Status Enable Command (*ESE) programs the Standard Event Enable Register. This register acts like a mask, so that the next time a selected bit goes high, the ESB bit in the status byte is set. The programming of the Standard Event Enable Register allows one or more event bits in the Standard Event Enable Register to set the ESB summary bit 5 in the status byte.

For example, if bit 0 is set in the Standard Event Enable Register, then when the OPC bit in the Standard Event register goes true, the ESB summary bit is set in the Status Byte.

*ESE?

The Standard Event Status Enable Query (*ESE?) returns the contents of the Standard Event Enable Register.

*ESR?

This query form interrogates the Standard Event Status Register. The register is cleared after it is read.

*IDN?

For 86130A, the Identification Query (*IDN?) response semantics are organized into four fields separated by commas. The field definitions are as follows:

Field 1 Manufacture	Agilent Technologies
Field 2 Model	86130A
Field 3 Serial Number	USxxxxxxxx
Field 4 Firmware Level	A.x.x.xxx

*OPC

A device is in the Operation Complete Command Active State (OCAS) after it has executed *OPC. The device returns to the Operation Complete Command Idle State (OCIS) whenever the No Operation Pending flag is TRUE, at the same time setting the OPC bit of the ESR TRUE. The following events force the device into OCIS without setting the No Operation Pending flag TRUE and without setting the OPC bit of the ESR:

- power on
- receipt of a dcas message (device clear)
- execution of *CLS
- execution of *RST

Implementation of the *OPC command is straightforward in devices which implement only sequential commands. When executing *OPC, the device simply sets the OPC bit of ESR.

In devices which implement overlapped commands, the implementation of *OPC is a more complicated. After executing *OPC, the device must not set the OPC bit of ESR until the device returns to OCIS, even though it continues to parse and execute commands.

NOTE For 86130A, *OPC can be used with overlapped commands. For more information, refer to "Overlapped and Sequential Commands" on page 2-13

*OPC?

A device is in the Operation Complete Query Active State (OQAS) after it has executed *OPC?. The device returns to the Operation Complete Query Idle State (OQIS) whenever the No Operation Pending flag is TRUE, at the same time placing a "1" in the Output Queue. The following events force the device into OQIS without setting the No Operation Pending flag TRUE and without placing a "1" in the Output Queue:

- power on
- receipt of the dcas message (device clear)

Implementation of the *OPC? query is straightforward in devices which implement only sequential commands. When executing *OPC? the device simply places a "1" in the Output Queue.

The implementation of overlapped commands in a device complicates the implementation of *OPC? and places some restrictions on the implementation of the Message Exchange Protocol (MEP). IEEE 488.2 dictates that devices shall send query responses in the order that they receive the corresponding queries. Although IEEE 488.2 recommends that *OPC? be the last query in a program message, there is nothing to prevent a controller program from ignoring this suggestion. This is why *OPC? must be sequential.

NOTE

For 86130A, *OPC(?) can be used with overlapped commands. For more information, refer to “Overlapped and Sequential Commands” on page 2-13

*RST

The Reset Command (*RST) sets the device-specific functions to a known state that is independent of the past-use history of the device. The command has the same effect as the front-panel PRESET key.

In addition, receipt of *RST by the error detector will cause all past results to be reset to zero.

*SRE <numerical value>

The Service Request Enable Command (*SRE) programs the Service Request Enable Register. This acts as a mask on the Status Byte, defining when the instrument may issue a service request. For a service request to be issued the summary bit in the Status Byte must match the bit in the Service Request Enable Register. More than one bit may be set by the *SRE command.

*SRE?

The query returns the current contents of the Service Request Enable Register.

*STB?

The Read Status Byte Query (*STB?) allows the programmer to read the status byte and Master Summary Status bit. When the register is accessed using the *STB command, bit 6 of the Status Byte is referred to as the Master Summary (MSS) bit. With this query, the register is not cleared when the value is read. It always reflects the current status of all the instrument’s status registers.

*TST?

The self-test query causes all internal self-tests and places a response into the output queue indicating whether or not the device completed the self-tests without any detected errors. It returns a “0” for success; a “1” if a failure was detected.

Upon successful completion of *TST?, the device settings are restored to their values prior to the *TST?

NOTE For more control over self tests, refer to “The TEST Subsystem” on page 4-108.

*WAI

For sequential commands, the Wait-to-Continue Command (*WAI) is a no-operation.
For overlapped commands, *WAI allows no further execution of commands or queries until the No Operation Pending flag is TRUE, or receipt of a dcas message, or a power on.

NOTE For 86130A, *WAI can be used with overlapped commands. For more information, refer to “Overlapped and Sequential Commands” on page 2-13

Optional Common Commands and Queries

The 86130A supports the following IEEE 488.2 optional commands. The instrument behavior is compatible with the remote behavior as described in the SCPI instrument model.

`*RCL`	Recall Device Setup
`*SAV`	Save Device Setup

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
`*RCL`	`<numeric value | string>`		`event; no query`
`*SAV`	`<numeric value | string>`		`event; no query`

*RCL <numeric value | string>

This command recalls the setup from a numbered store or from a full path filename. The range of store numbers is 0 through 9.

In addition, receipt of *RCL by the error detector will cause all past results to be reset to zero.

Note

The GPIB bus will be held off for approximately 8 seconds following receipt of this command to allow the instrument to settle fully.

*SAV <numeric value | string>

This command saves the setup into a numbered store or into a full path filename. The range of store numbers is 0 through 9.

The following IEEE 488.2 commands are not supported:

*OPT?
*PSC
*PSC?

FETCh Measurement Subsystem

The FETCh commands query the current value. All GPIB commands in the following section are query only.

The FETCh Measurement Subsystem

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
FETCh			
[:SENSe[1]]			
:BURSt			
:BCOunt?			query only
:DCYCle?			query only
:SRATio?			query only
:TCOunt?			query only
:ECOunt			
[:ALL]			
[:FULL]			
[:TOTal]?		<NR3>	query only
:DELTa?		<NR3>	query only
:OASZero			
[:TOTal]?		<NR3>	query only
:ZASone			

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
[:TOTal]?		<NR3>	query only
:EFINterval			
:DSEConds?		<NR3>	query only
:SEConds?		<NR3>	query only
:EINTerval			
:DSEConds?		<NR3>	query only
:SEConds?		<NR3>	query only
:ERATio			
[:ALL]			
[:FULL]			
[:TOTal]?		<NR3>	query only
:DELTa?		<NR3>	query only
:OASZero			
[:TOTal]?		<NR3>	query only
:ZASone			
[:TOTal]?		<NR3>	query only
:GATE			
:ELAPsed?		<NR3>	query only
:LOSS			
:POWer?		<NR3>	query only
:SYNChronizat?		<NR3>	query only
:SENSe2			
:BCOunt?		<NR3>	query only
:FREQuency			
[:CW|:FIXed]?		<NR3>	query only

FETCh[:SENSe[1]]:BURSt:BCOunt?

This query returns the Bad Burst Count since the start of the accumulation period. If Burst mode is OFF, it will return 9.91E+37 (Not-A-Number, NAN).

FETCh[:SENSe[1]]:BURSt:DCYCle?

This query returns the Burst Duty Cycle since the start of the accumulation period. If Burst mode is OFF, it will return 9.91E+37 (Not-A-Number, NAN).

FETCh[:SENSe[1]]:BURSt:SRATio?

This query returns the Burst Synchronization Ratio since the start of the accumulation period. If Burst mode is OFF, it will return 9.91E+37 (Not-A-Number, NAN).

FETCh[:SENSe[1]]:BURSt:TCOunt?

This query returns the Total Burst Count since the start of the accumulation period. If Burst mode is OFF, it will return 9.91E+37 (Not-A-Number, NAN).

FETCh[:SENSe[1]]:ECOunt[:ALL][:FULL][:TOTal]?

The total number of errors accumulated since the start of the accumulation period.

FETCh[:SENSe[1]]:ECOunt[:ALL][:FULL]:DELTa?

The number of errors in the last decisecond. This is intended to give a result that corresponds to the "instantaneous" error count. This value is available even when accumulation is turned off.

FETCh[:SENSe[1]]:ECOunt:OASZero[:TOTal]?

This is a contraction of the phrase `One received AS Zero`. The query returns the number of erred ones (a true data one received a data zero).

FETCh[:SENSe[1]]:ECOunt:ZASone[:TOTal]?

This is a contraction of the phrase `Zero received AS One`. The query returns the number of erred zeros (a true data zero received a data one.)

FETCh[:SENSe[1]]:EFINterval:DSEConds?
FETCh[:SENSe[1]]:EFINterval:SEConds?

This query is a contraction of `Error Free INterval` and returns a count of the number of time intervals during which no error was detected. The time intervals are are deciseconds (:DSEConds?) and seconds (:SEConds?).

FETCh[:SENSe[1]]:EINTerval:DSEConds?
FETCh[:SENSe[1]]:EINTerval:SEConds?

This query is a contraction of the phrase `Errored INTerval` and returns a count of the number of time intervals during which one or more errors were detected. The time intervals are deciseconds (:DSEConds?) and seconds (:SEConds?).

FETCh[:SENSe[1]]:ERATio[:ALL][:FULL][:TOTal]?

This query is a contraction of the phrase `Error RATio`. It is the ratio of the number of errors to the number of bits received in a time interval, specified by the next level in the command.

FETCh[:SENSe[1]]:ERATio[:ALL][:FULL]:DELTa?

The "instantaneous" error ratio calculated from the counts obtained in the last decisecond. This value is available even when accumulation is turned off.

FETCh[:SENSe[1]]:ERATio:OASZero[:TOTal]?

This is a contraction of the phrase `One received AS Zero`. The command returns the erred ones ratio (each true data one is received as a data zero).

FETCh[:SENSe[1]]:ERATio:ZASone[:TOTal]?

This is a contraction of the phrase `Zero received AS one`. The command returns the erred zero ratio (each true data zero is received as a data one).

FETCh[:SENSe[1]]:GATE:ELAPsed?

This query returns information about the degree to which the accumulation period has progressed. If SENSe[1]:GATE:MANNer TIME is selected, then this command returns the elapsed time into the accumulation period in units of seconds. If SENSe[1]:GATE:MANNer ERRors is selected, then this command returns the elapsed errors into the accumulation period. IF SENSe[1]:GATE:MANNer BITS is selected, then this command returns the elapsed clock bits into the accumulation period.

FETCh[:SENSe[1]]:LOSS:POWer?

This query command returns the total number of seconds that power was lost since the start of the accumulation period.

FETCh[:SENSe[1]]:LOSS:SYNChronizat?

This query command returns of the total number of seconds for which the incoming pattern was not synchronized to the reference pattern during the accumulation period.

The sub-command SYNChronizat may also be spelled SYNChronisat

FETCh:SENSe2:BCOunt?

This query command returns the accumulated bit count since the start of accumulation.

FETCh:SENSe2:FREQuency[:CW|:FIXed]?

This command returns the current frequency of the signal on the clock input. This measurement is independent of the accumulation period.

This command is superseded by SENSe2:FREQuency[:CW|:FIXed]?. It is also retained for backwards compatibility with 71600B systems.

The following FETCh command is not supported:

FETCh[:SENSe[1]]:LTEXt?

INPut Commands

The INPut commands control the error detector's Data and Clock inputs.

INPut1: The Data Input

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
INPut[1]			
:DELay	<numeric value>		
:DELay?		<NR3>	
:POLarity	NORMal|INVerted		
:POLarity?		NORM|INV	
:TERMination	0 | -2 | 1.3		changed from 716xxB
:TERMination?		<NR2>	

INPut[1]:DELay <numeric value> INPut[1]:DELay?

This command sets the Data input delay in seconds. The value is rounded to the nearest one picosecond.

Data input delay, one of two components to sampling point, controls the point in time at which the data signal is measured. Specifically, it is the delay in time from the active clock edge to the time at which the data is actually sampled. This delay can be set as high as 1 bit period or 10 nS, whichever is less.

The response form returns current data delay in seconds.

INPut[1]:POLarity NORMal | INVerted
INPut[1]:POLarity?

This command sets the polarity of the error detector reference pattern. This function is useful if your device inverts data.

The response form returns the current polarity of the Data input.

INPut[1]:TERMination 0 | –2 | 1.3
INPut[1]:TERMination?

This command sets the Data input termination level to –2 volts (ECL), 0 volts (SCFL), or 1.3 V (LVPECL). Agilent 86130A has added 1.3 V termination selection.

If input termination and 0/1 threshold level are to be set up, then the input termination should be set up first.

The Data port is connected to a 50 ohm load impedance (or termination) within the error detector. Data termination refers to the voltage level at the end of this load. The logic output from a device requires receiving equipment, including the error detector, to have specific termination voltage.

The response form returns the current Data input termination at the error detector.

NOTE This command is *not* precisely the same as 716xxB command. The 86130A adds the +1.3V termination selection.

WARNING Selecting the wrong terminations can damage your device.

INPut2: The Clock Input

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
INPut2			
:TERMination	0 | - 2 | 1.3		changed from 716xxB
:TERMination?		<NR2>	

INPut2:TERMination 0 | –2 | 1.3
INPut2:TERMination?

This command sets the Clock input termination level to –2 volts (ECL), 0 volts (SCFL), or 1.3 V (LVPECL). Agilent 86130A has added 1.3 V termination selection.

If input termination and 0/1 threshold level are to be set up, then the input termination should be set up first.

The response form returns the current Clock input termination of the error detector.

NOTE

This command is *not* precisely the same as 716xxB command. The 86130A adds the +1.3V termination selection.

WARNING

Selecting the wrong terminations can damage your device.

MMEMory Subsystem

The MMEMory commands control system memory and file operation functions.

The MMEMory Subsystem

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
MMEMory			
:CATalog?	[directory_name]	<NR3>,<NR3> {,<file entry>}	changed from 716xxB
:COPY	<file_source>, <file_destination> [,0 | 1 | OFF | ON]		command only
:DATA	<file_name>, <block_data>		
:DATA?	<file_name>	<block_data>	

MMEMory:CATalog? [directory_name]

The CATalog? command is query-only and returns information on the current contents and state of the specified directory. The information returned is composed of two numeric parameters followed by as many strings as there are files in the directory list. The first parameter indicates the total amount of storage currently used in bytes. The second parameter indicates the total amount of storage available, also in bytes. Each <file entry> is a string and indicates the name, type and size of one file or directory in the directory list:

<file name>,<file type>,<file size>

The <file size> is returned in bytes.

The default path is the drive of the UserPatternPath.

Hidden and System files will not be returned.

NOTE This command is *not* precisely the same as 716xxB command. 86130A adds an optional parameter to specify a directory name.

MMEMory:COPY <file_source>, <file_destination> [, 0 | 1 | OFF | ON]

This command copies the file from file_source to file_destination. The file specifications can be full path that include drive letters or network paths. The optional parameter is to specify whether to overwrite the destination file or not.

0 | OFF	don’t overwrite	default value
1 | ON	overwrite	

MMEMory:DATA <file_name>, <block_data> MMEMory:DATA? <file_name>

This command loads block_data into the file file_name. The file specifications can be full path that include drive letters or network paths. Maximum length of block_data allowed is 16 Mb.

The query returns block_data from the file file_name. Maximum length of block_data returned is 16 Mb.

The following MMEMory commands are not supported:

MMEMory:CPDisk <store number>
MMEMory:DELete <file name>
MMEMory:ICPDisk <dest. store number>, AHALf|BHALf, <start_bit>, <end_bit>
MMEMory:INITialize
MMEMory:MPResent?

OUTPut Commands

The OUTPut commands control the pattern generator’s Data, $\overline{\text{Data}}$, Clock, and $\overline{\text{Clock}}$ outputs.

OUTPut1: The Data Output

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
`OUTPut[1]`			
`:COUPling`	`AC|DC`		changed from 716xx
`:COUPling?`		`AC|DC`	
`:DATA`			
`:XOVer`	`<numeric value>`		
`:XOVer?`	`[MINimum|MAXimum]`	`<NR3>`	
`:DELay`	`<numeric value>`		
`:DELay?`		`<NR3>`	
`:POLarity`	`NORMal|INVerted`		
`:POLarity?`		`NORM|INV`	
`[:STATe]`	`0 | 1 | OFF | ON`		
`[:STATe]?`		`0 | 1`	
`:TERMination`	`0 | - 2 | 1.3`		changed from 716xx
`:TERMination?`		`<NR2>`	

OUTPut[1]:COUPling AC | DC
OUTPut[1]:COUPling?

This command sets the Data output coupling to AC or DC.

The response form returns the current Data output coupling.

NOTE This command is *not* precisely the same as 716xxB command. If coupling is AC, the DC termination levels default to 0 on bootup.

OUTPut[1]:DATA:XOVer <numeric value>
OUTPut[1]:DATA:XOVer? [MINimum | MAXimum]

This commands sets the Data output eye crossover. The percentage range is 25% to 75%.

The response form returns the current crossover percentage.

OUTPut[1]:DELay <numeric value>
OUTPut[1]:DELay?

This command sets the delay of the active edge of the clock output relative to the Data output. The units are seconds. The value is rounded to the nearest one picosecond.

The response form returns the current data to clock delay value.

OUTPut[1]:POLarity NORMal | INVerted
OUTPut[1]:POLarity?

This command sets the polarity of the Data output in the pattern generator.

The response form returns the current polarity of the Data output.

OUTPut[1][:STATe] 0 | 1 | OFF | ON
OUTPut[1][:STATe]?

This command controls the Data output. When OFF, the output is set to 0V.

The response form returns the current state of the Data output.

OUTPut[1]:TERMination 0 | –2 | 1.3
OUTPut[1]:TERMination?

This command sets the Data termination level to –2 volts (ECL), 0 volts (SCFL/AC), and +1.3 volts (LVPECL).

The response form returns the DC value of the data output termination. For more information on how this relates to AC coupling, refer to the 86130A online Help topic, “AC Coupling and Bias Tees”.

NOTE

This command is *not* precisely the same as 716xxB command. The 86130A adds the +1.3V termination selection.

OUTPut2: The Clock Output

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
OUTPut2			
:COUPling	AC | DC		external coupling/ changed from 716xxB
:COUPling?		AC | DC	external coupling
[:STATe]	0 | 1 | OFF | ON		new for 86130A
[:STATe]?		0 | 1	
:TERMination	0 | -2 | 1.3		changed from 716xxB
:TERMination?		<NR2>	

OUTPut2:COUPling AC | DC
OUTPut2:COUPling?

This command sets the Clock output coupling to AC or DC.

The query form returns the current coupling at the Clock output.

NOTE This command is *not* precisely the same as 716xxB command. If coupling is AC, the DC level defaults to 0 on bootup.

OUTPut2[:STATe] 0 | 1 | OFF | ON
OUTPut2[:STATe]?

This command controls the Clock output. When OFF, the output is set to 0V.

The query form returns the current state of the Clock output.

NOTE This command is new for 86130A.

OUTPut2:TERMination 0 | –2 | 1.3
OUTPut2:TERMination?

This command sets the Clock termination level to –2 volts (ECL), 0 volts (SCFL/AC), and +1.3 volts (LVPECL).

The response form returns the DC value of the Clock output termination. For more information on how this relates to AC coupling, refer to the 86130A online Help topic, “AC Coupling and Bias Tees”.

NOTE This command is *not* precisely the same as 716xxB command. The 86130A adds the +1.3V termination selection.

OUTPut10: The $\overline{\text{Data}}$ Output

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
OUTPut10			
:COUPling	AC | DC		ext coupling/ new for 86130A
:COUPling?		AC | DC	ext coupling
:DATA			
:XOVer	<numeric value>		
:XOVer?	[MINimum|MAXimum]	<NR3>	
[:STATe]	0 | 1 | OFF | ON		
[:STATe]?		0 | 1	
:TERMination	0 | -2 | 1.3		new for 86130A
:TERMination?		<NR2>	

OUTPut10:COUPling AC | DC
OUTPut10:COUPling?

This command sets the $\overline{\text{Data}}$ coupling to AC or DC.

The query form returns the current $\overline{\text{Data}}$ output coupling.

NOTE

This command is new for 86130A.

OUTPut10:DATA:XOVer <numeric value> OUTPut10:DATA:XOVer? [MINimum|MAXimum]

This command sets the $\overline{\text{Data}}$ output eye crossover voltage. The percentage range is 25% to 75%.

The query form returns the current crossover value in percentage.

OUTPut10[:STATe] 0 | 1 | OFF | ON OUTPut10[:STATe]?

This command controls the $\overline{\text{Data}}$. When OFF, the output is set to 0V.

The query form returns the current state of the $\overline{\text{Data}}$ output.

OUTPut10:TERMination 0 | –2 | 1.3 OUTPut10:TERMination?

This command sets the $\overline{\text{Data}}$ termination level to –2 volts (ECL), 0 volts (SCFL/AC), and +1.3 volts (LVPECL).

The query form returns the current termination at the $\overline{\text{Data}}$ output.

NOTE This command is new for 86130A.

OUTPut11: The $\overline{\text{Clock}}$ Output

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
OUTPut11			
:COUPling	AC|DC		external coupling/ new for 86130A
:COUPling?		AC | DC	external coupling
[:STATe]	0 | 1 | OFF | ON		new for 86130A
[:STATe]?		0 | 1	
:TERMination	0 | -2 | 1.3		new for 86130A
:TERMination?		<NR2>	

OUTPut11:COUPling AC | DC
OUTPut11:COUPling?

This command sets the $\overline{\text{Clock}}$ output coupling to AC or DC.
The query form returns the current coupling at the $\overline{\text{Clock}}$ output.

NOTE

This command is new for 86130A.

OUTPut11[:STATe] 0 | 1 | OFF | ON
OUTPut11[:STATe]?

This command controls the $\overline{\text{Clock}}$ output. When OFF, the output is set to 0V. The query form returns the current state of the $\overline{\text{Clock}}$ output.

NOTE This command is new for 86130A.

OUTPut11:TERMination 0 | –2 | 1.3
OUTPut11:TERMination?

This command sets the $\overline{\text{Clock}}$ termination level to –2 volts (ECL), 0 volts (SCFL/AC), and +1.3 volts (LVPECL).

The query form returns the current termination of the $\overline{\text{Clock}}$ output.

NOTE This command is new for 86130A.

The following OUTPut commands are not supported:

OUTPut4:COUPling AC|DC
OUTPut4:COUPling?
OUTPut4:TERMination -2|0
OUTPut4:TERMination?
OUTPut5:TERMination -2|0
OUTPut5:TERMination?
OUTPut5:COUPling AC|DC
OUTPut5:COUPling?
OUTPUT8:PLENgth RZ|STRetched
OUTPUT8:PLENgth?

PFETch Measurement Subsystem

The PFETch commands query the value that is immediately previous to the current value. All GPIB commands in the following section are query only.

The PFETch Measurement Subsystem

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
PFETch			
[:SENSe[1]]			
:BURSt			
:BCOunt?			query only
:DCYCle?			query only
:SRATio?			query only
:TCOunt?			query only
:ECOunt			
[:ALL]			
[:FULL]			
[:TOTal]?		<NR3>	query only
:DELTa?		<NR3>	query only
:OASZero			
[:TOTal]?		<NR3>	query only
:ZASone			

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
[:TOTal]?		<NR3>	query only
:EFINterval			
:DSEConds?		<NR3>	query only
:SEConds?		<NR3>	query only
:EINTerval			
:DSEConds?		<NR3>	query only
:SEConds?		<NR3>	query only
:ERATio			
[:ALL]			
[:FULL]			
[:TOTal]?		<NR3>	query only
:DELTa?		<NR3>	query only
:OASZero			
[:TOTal]?		<NR3>	query only
:ZASone			
[:TOTal]?		<NR3>	query only
:GATE			
:ELAPsed?		<NR3>	query only
:LOSS			
:POWer?		<NR3>	query only
:SYNChronizat?		<NR3>	query only
:SENSe2			
:BCOunt?		<NR3>	query only
:FREQuency			
[:CW|:FIXed]?		<NR3>	query only

PFETch[:SENSe[1]]:BURSt:BCOunt?

This query returns the Bad Burst Count since the start of the accumulation period. If Burst mode is OFF, it will return 9.91E+37 (Not-A-Number, NAN).

PFETch[:SENSe[1]]:BURSt:DCYCle?

This query returns the Burst Duty Cycle since the start of the accumulation period. If Burst mode is OFF, it will return 9.91E+37 (Not-A-Number, NAN).

PFETch[:SENSe[1]]:BURSt:SRATio?

This query returns the Burst Synchronization Ratio since the start of the accumulation period. If Burst mode is OFF, it will return 9.91E+37 (Not-A-Number, NAN).

PFETch[:SENSe[1]]:BURSt:TCOunt?

This query returns the Total Burst Count since the start of the accumulation period. If Burst mode is OFF, it will return 9.91E+37 (Not-A-Number, NAN).

PFETch[:SENSe[1]]:ECOunt[:ALL][:FULL][:TOTal]?

The total number of errors accumulated since the start of the accumulation period.

PFETch[:SENSe[1]]:ECOunt[:ALL][:FULL]:DELTa?

The number of errors in the last decisecond. This is intended to give a result that corresponds to the "instantaneous" error count. This value is available even when accumulation is turned off.

PFETch[:SENSe[1]]:ECOunt:OASZero[:TOTal]?

This is a contraction of the phrase `One received AS Zero`. The query returns the number of erred ones (a true data one received a data zero).

PFETch[:SENSe[1]]:ECOunt:ZASone[:TOTal]?

This is a contraction of the phrase `Zero received AS One`. The query returns the number of erred zeros (a true data zero received a data one.)

PFETch[:SENSe[1]]:EFINterval:DSEConds?
PFETch[:SENSe[1]]:EFINterval:SEConds?

This query is a contraction of `Error Free INterval` and returns a count of the number of time intervals during which no error was detected. The time intervals are are deciseconds (:DSEConds?) and seconds (:SEConds?).

PFETch[:SENSe[1]]:EINTerval:DSEConds?
PFETch[:SENSe[1]]:EINTerval:SEConds?

This query is a contraction of the phrase `Errored INTerval` and returns a count of the number of time intervals during which one or more errors were detected. The time intervals are deciseconds (:DSEConds?) and seconds (:SEConds?).

PFETch[:SENSe[1]]:ERATio[:ALL][:FULL][:TOTal]?

This query is a contraction of the phrase `Error RATio`. It is the ratio of the number of errors to the number of bits received in a time interval, specified by the next level in the command.

PFETch[:SENSe[1]]:ERATio[:ALL][:FULL]:DELTa?

The "instantaneous" error ratio calculated from the counts obtained in the last decisecond. This value is available even when accumulation is turned off.

PFETch[:SENSe[1]]:ERATio:OASZero[:TOTal]?

This is a contraction of the phrase `One received AS Zero`. The command returns the erred ones ratio (each true data one is received as a data zero).

PFETch[:SENSe[1]]:ERATio:ZASone[:TOTal]?

This is a contraction of the phrase `Zero received AS one`. The command returns the erred zero ratio (each true data zero is received as a data one).

PFETch[:SENSe[1]]:GATE:ELAPsed?

This query returns information about the degree to which the accumulation period has progressed. If SENSe[1]:GATE:MANNer TIME is selected, then this command returns the elapsed time into the accumulation period in units of seconds. If SENSe[1]:GATE:MANNer ERRors is selected, then this command returns the elapsed errors into the accumulation period. IF SENSe[1]:GATE:MANNer BITS is selected, then this command returns the elapsed clock bits into the accumulation period.

PFETch[:SENSe[1]]:LOSS:POWer?

This query command returns the total number of seconds that power was lost since the start of the accumulation period.

PFETch[:SENSe[1]]:LOSS:SYNChronizat?

This query command returns of the total number of seconds for which the incoming pattern was not synchronized to the reference pattern during the accumulation period.

The sub-command SYNChronizat may also be spelled SYNChronisat

PFETch:SENSe2:BCOunt?

This query command returns the accumulated bit count since the start of accumulation.

PFETch:SENSe2:FREQuency[:CW|:FIXed]?

This command returns the current frequency of the signal on the clock input. This measurement is independent of the accumulation period.

This command is superseded by SENSe2:FREQuency[:CW|:FIXed]?. It is also retained for backwards compatibility with 71600B systems.

PLUGin Subsystem

The PLUGin subsystem allows you to query data, or pass commands and optional data to the instrument's plug-ins.

Refer to "Error Analysis Plug-in Commands" on page 5-1 for a list of commands available with the Error Analysis plug-in, and to "Introduction" on page 5-2 for details on how to use the Error Analysis plug-in commands.

The PLUGin Subsystem

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
PLUGin			
:CATalog?		<string>	
:GET			
:BINary?	<string command>	<block_data>	
:STRing?	<string command>	<string>	
:HELP			
:HEADers?	[string name]	<block_data>	
:IDENtity?	<string name>	<string>	
:PUT			
:BINary	<string command>, <block_data>		
:STRing	<string command> [, string data]		

PLUGin:CATalog?

This query command retrieves a list of plug-in's in the system. The response is a list of comma-separated strings.

PLUGin:GET:BINary? <string command>

This command retrieves binary data from a plug-in. The response is returned in a definite length block.

PLUGin:GET:STRing? <string command>

This command retrieves ASCII data from a plug-in. The response is returned as ASCII data.

PLUGin:HELP:HEADers? [string name]

This query command retrieves a list of commands and queries implemented by all plug-ins in the system, or the specific plug-in specified. The response is a definite length block consisting of full path items separated by linefeeds. If an item is query only, "/qonly/" will be appended at the end. If an item is command only, "/nquery/" will be appended at the end.

PLUGin:IDENtity? <string name>

This query command retrieves identity information about the specified plug-in. The response is ASCII data consisting of two fields delimited by a comma. The field format is "plugin name,revision".

PLUGin:PUT:BINary <string command> [block_data]

This command sends binary data to a plug-in. The data passed to the plug-in consists of a plug-in command and definite length block data, which is then processed by the plug-in. Command strings are specific to a particular plug-in.

PLUGin:PUT:STRing <string command> [, string data]

This command sends quoted string data to a specific plug-in. The data passed to the plug-in consists of command and optional data. The optional parameter [, string data] implementation depends on the plug-in you are sending the commands to.

SENSe Commands

The SENSe commands control the error detector's Data and Clock inputs. The SENSe commands also control the pattern generator's Clock In port.

SENSe1: The Data Sense

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
SENSe[1]			
:EYE			
:ACENter	ONCE | 0 | 1 | OFF | ON		overlapped command
:ACENter?		0 | 1	
:ALIGN			
:AUTO	ONCE | 0 | 1 | OFF | ON		new for 86130A/ overlapped command
:AUTO?		0 | 1	
:HEIGht?		<NR3>	
:QUICk			
:ALIGN			
:AUTO	ONCE | 0 | 1 | OFF | ON		new for 86130A/ overlapped command

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
:AUTO?		0 | 1	
:QUICk:ACENter	ONCE | 0 | 1 | OFF | ON		new for 86130A/ overlapped command
:QUICk:ACENter?		0 | 1	
:QUICk:TCENter	ONCE | 0 | 1 | OFF | ON		new for 86130A/ overlapped command
:QUICk:TCENter?		0 | 1	
:TCENter	ONCE | 0 | 1 | OFF | ON		over-lapped command
:TCENter?		0 | 1	
:THReshold	<numeric value>		
:THReshold?		<NR3>	
:WIDTh?		<NR3>	
:GATE			
:BURSt	0 | 1 | OFF | ON		
:BURSt?		0 | 1	
:MANNer	TIME | ERRors | BITS		
:MANNer?		TIME | ERR | BITS	
:MODE	MANual | SINGle | REPetitive		
:MODE?		MAN | SING | REP	
:PERiod			

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
:BITS	<numeric value>		
:BITS?		<NR3>	
:ERRors	<numeric value>		
:ERRors?		<NR3>	
[:TIME]	<numeric value>		
[:TIME]?		<NR3>	
[:STATe]	0 | 1 | OFF | ON		overlapped command
[:STATe]?		0 | 1	
:LOGGing	0 | 1 | OFF | ON		changed from 716xxB
:LOGGing?		0 | 1	
:FILename	<string>		new for 86130A
:FILename?		<string>	
:PATTern			
:FORMat			
[:DATA]	PACKed,<numeric value>		
[:DATA]?		<NR1>	
:MDENsity			
[:DENSity]	<numeric value>		
[:DENSity]?		<NR3>	
[:SELect]	PRBS<n> | PRBN<n> | ZSUBstitut<n> | MDENsity<n> | UPATtern<n> | FILENAME, <string>		changed from 716xxB

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
[:SELect]?		PRBS<n> | PRBN<n> | ZSUB<n> | MDEN<n> | UPAT	
:TRACk	0 | 1 | OFF | ON		new for 86130A
:TRACk?		0 | 1	
:UPATtern<n>			
:DATA	[A|B,]<block_data>		
:DATA?	[A|B]	<block_data>	
:IDATa	[A|B,]<start_bit>, <length_in_bits>, <block_data>		
:IDATa?	[A|B,]<start_bit>, <length_in_bits>	<block_data>	
:LABel	<string>		
:LABel?		<string>	
:USE	STRaight | APATtern		
:USE?		STR | APAT	
[:LENGth]	<numeric value>		
[:LENGth]?		<NR1>	
:UFILe			
:DATA	[A|B,]<filename>, <block_data>		new for 86130A
:DATA?	[A|B,]<filename>	<block_data>	
:IDATa	[A|B,]<filename>, <start_bit>, <length_in_bits>, <block_data>		new for 86130A

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
:IDATa?	[A|B,]<filename>, <start_bit>, <length_in_bits>	<block_data>	
:LABel	<filename>, <string>		new for 86130A
:LABel?	<filename>	<string>	
:NAME?		<string>	new for 86130A
:USE	<filename> STRaight | APATtern		new for 86130A
:USE?	<filename>	STR | APAT	
[:LENGth]	<filename>, <numeric_value>		new for 86130A
[:LENGth]?	<filename>	<NR3>	
:SYNChronizat	ONCE | 0 | 1 | OFF | ON		
:SYNChronizat?		0 | 1	
:THReshold	<numeric value>		
:THReshold?		<NR3>	
:VOLTage			
:ZOTHreshold	<numeric value>		
:ZOTHreshold?		<NR3>	
:AUTO	0 | 1 | OFF | ON		
:AUTO?		0 | 1	

SENSe[1]:EYE:ACENter ONCE | 0 | 1 | OFF | ON SENSe[1]:EYE:ACENter?

This command initiates a search for the 0/1 threshold voltage midway between the two 0/1 threshold voltages with a measured BER just in excess of the BER configured by the `EYE:THReshold` command. If successful, the command leaves the zero-one-threshold at this value, and the center of the eye can be found by querying the zero-one-threshold value. If unsuccessful, the `EYE:HEIGht?` will return 9.91E+37 (Not-A-Number, NAN). The command `:ACENter|:ACENtre OFF` aborts a previously started search. When this command is in execution, SENSe[1]:VOLTage:ZOTHreshold:AUTO is set to off.

The command :ACENter may also be spelled :ACENtre

The response form returns the current value of the 0/1 threshold voltage.

NOTE

The command `:ACENter` is an overlapped command. For more information, refer to "Overlapped and Sequential Commands" on page 2-13

SENSe[1]:EYE:ALIGN:AUTO ONCE | 0 | 1 | OFF | ON SENSe[1]:EYE:ALIGN:AUTO?

This command turns on or off autoalign. When this command is in execution, SENSe[1]:VOLTage:ZOTHreshold:AUTO is set to off.

The query form returns the state of autoalign.

SENSe[1]:EYE:HEIGht?

This is a query command that searches for the value of data amplitude that puts the zero-to-one threshold level midway between the upper and lower bounds at which the error ratio exceeds the threshold value set by the `:EYE:THReshold` command.

If the result is not available or the search was unsuccessful, then the number 9.91E+37 (Not-A-Number, NAN) will be returned.

SENSe[1]:EYE:QUICk:ALIGN:AUTO ONCE | 0 | 1 | OFF | ON
SENSe[1]:EYE:QUICk:ALIGN:AUTO?

This command is only available for instrument setups that include the following:

- BER Threshold <= 1.0E-2
- PRBS patterns (2^n-1)

This command turns on or off quick autoalign.

NOTE This command is new for Agilent 86130A.

SENSe[1]:EYE:QUICk:ACENter ONCE | 0 | 1 | OFF | ON
SENSe[1]:EYE:QUICk:ACENter?

This command is only available for instrument setups that include the following:

- BER Threshold <= 1.0E-2
- PRBS patterns (2^n-1)

This command initiates a search for the 0/1 threshold voltage midway between the two 0/1 threshold voltages with a measured BER just in excess of the BER configured by the `EYE:THReshold` command. If successful, the command leaves the zero-one-threshold at this value, and the center of the eye can be found by querying the zero-one-threshold value. If unsuccessful, the `EYE:HEIGht?` will return 9.91E+37 (Not-A-Number, NAN). The command `:ACENter|:ACENtre OFF` aborts a previously started search.

The command :ACENter may also be spelled :ACENtre

The response form returns the current value of the 0/1 threshold voltage.

NOTE The command `:ACENter` is an overlapped command. For more information, refer to "Overlapped and Sequential Commands" on page 2-13

SENSe[1]:EYE:QUICk:TCENter ONCE | 0 | 1 | OFF | ON SENSe[1]:EYE:QUICk:TCENter?

This command is only available for instrument setups that include the following:

- BER Threshold <= 1.0E-2
- PRBS patterns (2^n-1)

This command initiates a search for the value of data/clock delay that puts the active clock edge in the center of the data eye, midway between the two relative delay points with a measured BER just in excess of the BER configured by the `EYE:THReshold` command. If successful, the command leaves the data/clock delay at this value and the center of the eye can be found by querying the data delay value. If unsuccessful, the `EYE:WIDth?` will return 9.91E+37 (Not-A-Number, NAN). The command `:TCENter OFF` aborts a previously started search.

Note

The clock/data align feature (used to center the sampling point in the data input eye) uses information derived from the input clock frequency.

For the clock/data align feature to work properly, the input frequency must be stable during the measurement. The frequencies at the start and end of the measurement are compared, and if they differ by more than 10%, the measurement fails.

When a source clocking the instrument changes frequency, it will take time for the instrument to sense the change and adjust its configuration. Refer to the sections dealing with clock stabilization to ensure that the instrument's configuration has stabilized following any change of frequency prior to performing a clock to data alignment.

There is no need to alter the sync-mode before or after a clock to data alignment procedure, as AUTO sync-mode is automatically configured for the duration of the procedure.

The command :TCENter may also be spelled :TCENtre

The response form returns the current value of the data/clock delay.

NOTE

The command `:TCENter` is an overlapped command. For more information, refer to "Overlapped and Sequential Commands" on page 2-13.

SENSe[1]:EYE:TCENter ONCE | 0 | 1 | OFF | ON
SENSe[1]:EYE:TCENter?

This command initiates a search for the value of data/clock delay that puts the active clock edge in the center of the data eye, midway between the two relative delay points with a measured BER just in excess of the BER configured by the `EYE:THReshold` command. If successful, the command leaves the data/clock delay at this value and the center of the eye can be found by querying the data delay value. If unsuccessful, the `EYE:WIDth?` will return 9.91E+37 (Not-A-Number, NAN). The command `:TCENter OFF` aborts a previously started search.

Note

The clock/data align feature (used to center the sampling point in the data input eye) uses information derived from the input clock frequency.

For the clock/data align feature to work properly, the input frequency must be stable during the measurement. The frequencies at the start and end of the measurement are compared, and if they differ by more than 10%, the measurement fails.

When a source clocking the instrument changes frequency, it will take time for the instrument to sense the change and adjust its configuration. Refer to the sections dealing with clock stabilization to ensure that the instrument's configuration has stabilized following any change of frequency prior to performing a clock to data alignment.

There is no need to alter the sync-mode before or after a clock to data alignment procedure, as AUTO sync-mode is automatically configured for the duration of the procedure.

The command :TCENter may also be spelled :TCENtre

The response form returns the current value of the data/clock delay.

NOTE The command `:TCENter` is an overlapped command. For more information, refer to "Overlapped and Sequential Commands" on page 2-13.

SENSe[1]:EYE:THReshold <numeric value> SENSe[1]:EYE:THReshold?

This command sets the BER threshold to be used in the determination of the edges of the eye.

The query form returns the current BER threshold value.

SENSe[1]:EYE:WIDTh?

This is a query command that interrogates the eye width found by the most recent search for the value of data/clock delay that put the active edge in the center of the data eye.

If the result is not available or the search was unsuccessful, then the number 9.91E+37 (Not-A-Number, NAN) will be returned.

SENSe[1]:GATE:BURSt 0|1|OFF|ON

This command turns Burst Gating OFF or ON.

SENSe[1]:GATE:BURSt?

This query returns the Burst Gating state of 0 or 1. 0 indicates OFF and 1 indicates ON.

SENSe[1]:GATE:MANNer TIME | ERRors | BITS SENSe[1]:GATE:MANNer?

This command sets the manner in which the accumulation period is controlled.

When TIME is selected, the error detector performs SINGLE and REPETITIVE accumulation periods that are controlled by elapsed time.

When ERRors is selected, the error detector performs SINGLE and REPETITIVE accumulation periods that are controlled by the accumulation of bit errors.

When BITS is selected, the error detector performs SINGLE and REPETITIVE accumulation periods that are controlled by the accumulation of clock bits.

The query form returns the current manner of accumulation.

SENSe[1]:GATE:MODE MANual | SINGle | REPetitive SENSe[1]:GATE:MODE?

This command sets the accumulation period mode to either Manual, Single, or Repetitive.

This command causes all past results to be labelled as invalid.

The query form returns the current selection of the accumulation mode.

SENSe[1]:GATE:PERiod:BITS <numeric value> SENSe[1]:GATE:PERiod:BITS?

When GATE:MANNer is set to BITS, the duration of the accumulation period is set in clock bits (or periods). Values of 1e7 through 1e15 in decade steps are permitted.

This command causes all past results to be labelled as invalid.

The query form returns the number of bits to which the gate period is set.

SENSe[1]:GATE:PERiod:ERRors <numeric value> SENSe[1]:GATE:PERiod:ERRors?

When GATE:MANNer is set to ERRors, this sets the duration of the accumulation period in bit errors. Values of 10, 100 and 1000 are permitted.

This command causes all past results to be labelled as invalid.

The query form returns the number of errors to which the gate period is set.

SENSe[1]:GATE:PERiod[:TIME] <numeric value> SENSe[1]:GATE:PERiod[:TIME]?

When GATE:MANNer is set to TIME, the duration of the accumulation period is set in seconds. Neither a value less than 1 second nor greater than 99 days, 23 hours, 59 minutes and 59 seconds is permitted.

This command causes all past results to be labelled as invalid.

The query form returns the time to which the gate period is set.

SENSe[1]:GATE [:STATe] 0 | 1 | OFF | ON
SENSe[1]:GATE [:STATe]?

This command turns accumulation on or off.

Note

Previous commands that have altered the configuration of the instrument might not have settled. In order to ensure that the GATE ON command is not executed until conditions have settled, it is strongly recommended that the frequency be allowed to stabilize prior to the GATE ON command, and then be followed by a synchronization search. Refer to "Pattern Changes and Settling Time" on page 2-15

NOTE When `GATE:MODE SINGle is executed,` the `GATE[:STATe]ON` command is an overlapped command. For more information, refer to "Overlapped and Sequential Commands" on page 2-13

The query form returns the current state of the accumulation gating.

SENSe[1]:LOGGing 0 | 1 | OFF | ON
SENSe[1]:LOGGing?

This command allows you to save accumulated results in a file for future analysis.

The query form returns whether or not the accumulated results will be saved in a log file.

NOTE This command is *not* precisely the same as 716xxB command. The ONCE parameter is not supported.

SENSe[1]:LOGGing:FILename <string>
SENSe[1]:LOGGing:FILename?

This command enables you to send accumulated data to a specified filename.

The query form returns the filename to which the logged data is sent.

NOTE This command is new for 86130A.

SENSe[1]:PATTern:FORMat[:DATA] PACKed, <numeric value> SENSe[1]:PATTern:FORMat[:DATA]?

The command form controls the format of data transfer for the :PATTern:UPATtern<n>:DATA and :PATTern:UPATtern<n>:IDATa commands. The command permits the packing of bits within a byte to be set. The data may be sent 1 bit/byte or 8 bits/byte. If 1 bit/byte is selected, numeric values of either binary 1 or binary 0 are allowed. If 8 bits/byte is selected, the left-most bit of the first byte received becomes the first bit of the pattern.

The query form returns the current value to which the data is packed.

SENSe[1]:PATTern:MDENsity[:DENSity] <numeric value> SENSe[1]:PATTern:MDENsity[:DENSity]?

The command form sets the mark density of the pattern at the error detector.

The query form returns the current mark density of the pattern at the error detector.

SENSe[1]:PATTern[:SELect] PRBS<n> | PRBN<n> | ZSUBstitut<n> | MDENsity<n> | UPATtern<n> | FILENAME,<string> SENSe[1]:PATTern[:SELect]?

This command defines the type of reference pattern being used by the error detector. The <character data> parameter is retained for backwards compatibility and may be one of the following:

`PRBS<n>`	*<n> = 7, 10, 15, 23, or 31*
`PRBN<n>`	*<n> = 7, 10, 15, or 23*
`ZSUBstitut<n>`	*<n> = 7, 10, 15, or 23*
`UPATtern<n>`	*<n> = 0 through 12*
`MDENsity<n>`	*<n> = 7, 10, 15, and 23*
`FILENAME`	*<string>*

ZSUBstitut is a contraction of the phrase `Zero SUBstitution` and is used for defining patterns in which a block of bits is replaced by a block of zeros. Zero SUBstitution is not implemented in 86130A.

MDENsity is a contraction of the phrase `Mark DENsity` and is used for defining a pattern in which the user may set the density of marks.

UPATtern<n> is a contraction of the phrase `User PATtern` and is used to define the contents of a pattern store. For the 86130A, <n> is defined as follows:

<n> = 0	current pattern
<n> = 1 through 12	disc storage

FILENAME is a parameter that allows the remote user to load a user pattern from the instrument disk drive.

Note

If the PG and ED are coupled, setting the pattern by using the SOURce1:PATTern:SELect command will cause the pattern to be set in both the PG and the ED. If the PG and ED are uncoupled, then the ED pattern must be selected using the SENSe1:PATTern:SELect command.

The query form returns the patterns types in short form.

Note

If a user pattern is selected and the [:SELECT]? command is used, then the response is “UPAT”. The particular value of <n> or the name of the file specified in the command form is not returned.

If you wish to get the full path in a response, you may use the following commands:

```
[SOURce[1]:]PATTern:UFILe:NAME?
SENSe[1]:PATTern:UFILe:NAME?
```

NOTE This command is *not* precisely the same as 716xxB command. PRBN and FILENAME are available.

NOTE This command is order-sensitive. For more information, refer to “User Pattern Edits” on page 2-16

SENSe[1]:PATTern:TRACk 0 | 1 | OFF | ON SENSe[1]:PATTern:TRACk?

This command form turns on the error detector pattern tracking; the error detector and pattern generator patterns are coupled.

The query form returns the current state of the pattern track setting.

NOTE This command is new for 86130A.

SENSe[1]:PATTern:UPATtern<n>:DATA [A|B,] <block_data> SENSe[1]:PATTern:UPATtern<n>:DATA? [A|B,]

The command form sets the bits of the user pattern. The bits are sent as an arbitrary block diagram data element. The <block_data> may be packed using the FORMat[:DATA]PACKed command.

If ":USE APATtern" is selected, then the first parameter indicates which half pattern is to receive the data. If "USE STRaight" is selected, either "A" or no first parameter is acceptable.

The length of the <block_data> embedded in the header always refers to the length of the data in bytes. It should not be confused with the packing value contained in the SENSe[1]:PATTern:FORMat[:DATA]PACKed <numeric value> command.

For example, consider the following header:

```
#19<data>
```

#	This indicates the start of the header.
1	The number of decimal digits to follow form the length.
9	This is the length of the data block (in bytes) that follow.
<data>	For data packed with a value of 1, each bit is displayed. For data packed with a value of 8, each digit represents 8 packed bits.

For the 86130A, this command *can change* the pattern length. If the pattern length is extended, the additional length is filled with the incoming data. If fewer bits than specified by the "LENGth" command are sent, the data that resides beyond the block length is truncated.

The query form returns the <block_data>.

SENSe[1]:PATTern:UPATtern<n>:IDATa [A|B,] <start_bit>, <length_in_bits>, <block_data> SENSe[1]:PATTern:UPATtern<n>:IDATa? [A|B,] <start_bit>, <length_in_bits>

The command form is similar to the :DATA command and is used to set the bits in a user pattern. The :IDATa command downloads a part of the user pattern. The data may be packed using the FORMat[:DATA]PACKed command. The parameters <start_bit> and <length_in_bits> define the boundaries of the part.

For example, consider the following header:

```
3,9,#19<data>
```

3	This indicates the start bit.
9	This is the number of bits.
#	This indicates the start of the header.
1	This is the number of decimal digits to follows that form the length.
9	This is the length of the data block that follows.
<data>	For data packed with a value of 1, each bit is displayed. For data packed with a value of 8, each digit represents 8 packed bits.

The query form returns the <block_data>.

SENSe[1]:PATTern:UPATtern<n>:LABel <string> SENSe[1]:PATTern:UPATtern<n>:LABel?

The command form sets the label for the pattern at the Data input of the error detector.

The query form returns the label for the current pattern at the Data input of the error detector.

SENSe[1]:PATTern:UPATtern<n>:USE STRaight | APATtern SENSe[1]:PATTern:UPATtern<n>:USE?

The command form sets the pattern mode at the Data input of the error detector to standard or alternate pattern.

The query form returns the pattern mode at the Data input of the error detector.

SENSe[1]:PATTern:UPATtern<n>[:LENGth] <numeric value> SENSe[1]:PATTern:UPATtern<n>[:LENGth]?

The command form sets the length of the user pattern at the Data input of the error detector.

The query form returns the current length of the user pattern at the Data input of the error detector.

SENSe[1]:PATTern:UFILe:DATA [A | B,] <filename>, <block_data> SENSe[1]:PATTern:UFILe:DATA? [A|B,] <filename>

This command and query allows the use of the full path to specify the user pattern.

The command form sets the bits of the user pattern. The bits are sent as an arbitrary block diagram data element. The <block_data> may be packed using the FORMat[:DATA]PACKed command.

If ":USE APATtern" is selected, then the first parameter indicates which half pattern is to receive the data. If "USE STRaight" is selected, either "A" or no first parameter is acceptable.

The length of the <block_data> embedded in the header always refers to the length of the data in bytes. It should not be confused with the packing value contained in the SENSe[1]:PATTern:FORMat[:DATA]PACKed <numeric value> command.

For example, consider the following header:

#19<data>

#	This indicates the start of the header.
1	The number of decimal digits to follow form the length.
9	This is the length of the data block (in bytes) that follow.
<data>	For data packed with a value of 1, each bit is displayed. For data packed with a value of 8, each digit represents 8 packed bits.

For the 86130A, this command *can change* the pattern length. If the pattern length is extended, the additional length is filled with the incoming data. If fewer bits than specified by the "LENGth" command are sent, the data that resides beyond the block length is truncated.

The query form returns the <block_data>

NOTE This command is order-sensitive. For more information, refer to “User Pattern Edits” on page 2-16

NOTE This command is new for 86130A.

SENSe[1]:PATTern:UFILe:IDATa [A | B,] <filename>, <start_bit>, <length_in_bits>, <block_data> SENSe[1]:PATTern:UFILe:IDATa? [A|B,] <filename>, <start_bit>, <length_in_bits>

This command and query allows the use of the full path to specify the user pattern.

The command form is similar to the :DATA command. The IDATa command is a contraction of the phrase `Incremental Data` and is used to download part of a user-defined pattern.

If “:USE APATtern” is selected, then the first parameter indicates which half pattern is to receive the data. If “USE STRaight” is selected, either “A” or no first parameter are acceptable.

The length of the <block data> embedded in the header refers always to the length of the data in bytes.

The first parameter defines the starting position within the overall pattern of the first bit of the transmitted pattern. The first bit is counted as bit zero. The second parameter defines how many bits are to be transmitted and the third parameter provides the data itself.

For example, consider the following header:

```
3,9,#19<data>
```

3	This indicates the start bit.
9	This is the number of bits.
#	This indicates the start of the header.
1	This is the number of decimal digits to follows that form the length.
9	This is the length of the data block that follows.
<data>	For data packed with a value of 1, each bit is displayed. For data packed with a value of 8, each digit represents 8 packed bits.

The response form returns <block_data>.

NOTE This command is order-sensitive. For more information, refer to “User Pattern Edits” on page 2-16

NOTE This command is new for 86130A.

SENSe[1]:PATTern:UFILe:NAME?

This query returns a full path for the current user pattern.

NOTE This command is new for 86130A.

SENSe[1]:PATTern:UFILe:USE <filename>, STRaight | APATtern SENSe[1]:PATTern:UFILe:USE? <filename>

This command and query allows the use of the full path to specify the user pattern.

When USE STRaight is selected the whole of the pattern is repeatedly output.

When USE APATtern is selected, the pattern is considered to be composed of two halves. The store is set to have a length of 128 bits for each half pattern; all bits are set to zero and the trigger is set to occur on the A/B changeover.

NOTE This command is order-sensitive. For more information, refer to “Alternate Pattern Setup” on page 2-17

NOTE This command is new for 86130A.

SENSe[1]:PATTern:UFILe[:LENGth] <filename>, <numeric_value> SENSe[1]:PATTern:UFILe[:LENGth]? <filename>

This command and query allows the use of the full path to specify the user pattern.

When USE STRaight is selected, the command form sets the length of the user pattern.

If an alternate pattern is selected (USE APATtern), the LENGth command sets length of each half of the pattern.

The response form returns the entire length of a STRaight pattern and the length of each half of an APATern pattern.

NOTE This command is new for 86130A.

SENSe[1]:PATTern:UFILe:LABel <filename>, <string>
SENSe[1]:PATTern:UFILe:LABel? <filename>

This command and query allows the use of the full path to specify the user pattern.

The command form gives the user pattern a user-specified label.

The response form returns the user pattern's label.

NOTE This command is new for 86130A.

SENSe[1]:SYNChronizat ONCE | 0 | 1 | OFF | ON
SENSe[1]:SYNChronizat?

These commands configure the settings that control synchronization of the reference pattern to the incoming pattern. SENSe[1]:SYNChronizat ON turns automatic re synchronization on. SENSe[1]:SYNChronizat OFF turns automatic resynchronization off. SENSe[1]:SYNChronizat ONCE initiates a resynchronization attempt.

The query form returns the current selection of the pattern synchronization.

The sub-command SYNChronizat may also be spelled SYNChronisat

SENSe[1]:SYNChronizat:THReshold <numeric value>
SENSe[1]:SYNChronizat:THReshold?

This command sets the threshold level of error ratio at which synchronization is successful.

Note

The valid values are 1e-01 thru 1e-08 in decade steps.

The query form returns the threshold level of error ratio at which synchronization is set.

The sub-command SYNChronizat may also be spelled SYNChronisat

SENSe[1]:VOLTage:ZOTHreshold <numeric value> SENSe[1]:VOLTage:ZOTHreshold?

This command allows the level at which the error detector discriminates between a zero and a one to be configured.

A numeric value parameter sets the level to a given value in Volts. It also sets `:ZOTHreshold:AUTO OFF`.

When in `:ZOTHreshold:AUTO OFF`, the query form of the `:ZOTHreshold` command returns the last user-entered value.

NOTE This command is order-sensitive. For more information, refer to "Output Level Adjustments" on page 2-20

SENSe[1]:VOLTage:ZOTHreshold:AUTO 0 | 1 | OFF | ON SENSe[1]:VOLTage:ZOTHreshold:AUTO?

This command enables an automatic mode in which the 0/1 threshold level is set to the mean of the input signal.

The query form of this command returns the current setting of the hardware discrimination circuit.

When in `:ZOTHreshold:AUTO ON`, the query form returns the value automatically determined by the hardware.

SENSe2: The Clock Sense

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
SENSe2			
:FREQuency			
[:CW|:FIXed]?		<NR3>	
:VOLTage			
:EDGE	NEGative | POSitive		
:EDGE?		NEG | POS	

SENSe2:FREQuency [:CW | :FIXed]?

This query returns the frequency of the signal at the error detector clock input. You may also use the following forms of this query:

```
SENSe2:FREQ?
SENSe2:FREQ:CW?
SENSe2:FREQ:FIXed?
```

SENSe2:VOLTage:EDGE NEGative | POSitive
SENSe2:VOLTage:EDGE?

This command sets the active edge of the clock input. The rising edge (POSitve) or falling edge (NEGative) may start the period in which the input data is sampled.

The response form returns the current setting of the error detector's clock edge.

SENSe6: The Clock Sense

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
SENSe6			
:FREQuency			
[:CW|:FIXed]?		<NR3>	

SENSe6:FREQuency [:CW | :FIXed]?

This query returns the frequency of the signal at the pattern generator external clock input. You may also use the following forms of this query:

```
SENSe6:FREQ?
SENSe6:FREQ:CW?
SENSe6:FREQ:FIXed?
```

This command supersedes the following 716xxB command,
SOURce2:FREQuency[:CW | :FIXed]?<numeric value>

NOTE This command is new for 86130A.

The following SENSe commands are not supported:

SENSe[1]:BLOCk 0|1|OFF|ON
SENSe[1]:BLOCk?
SENSe[1]:BLOCk:BLENgth <numeric value>
SENSe[1]:BLOCk:BLENgth?
SENSe[1]:BLOCk:BSTart <numeric value>
SENSe[1]:BLOCk:BSTart?
SENSe[1]:ELOCation?
SENSe[1]:ELOCation:BEADdress <numeric value>
SENSe[1]:ELOCation:BEADdress?
SENSe[1]:ELOCation ONCE
SENSe[1]:LOGGing:ALARms 0|1|OFF|ON
SENSe[1]:LOGGing:ALARms?
SENSe[1]:LOGGing:BRATe?

SENSe[1]:LOGGing:BRATe <numeric value>
SENSe[1]:LOGGing:DURing[:EVENt] NEVer|ESECond|ERGThrshld
SENSe[1]:LOGGing:DURing[:EVENt]?
SENSe[1]:LOGGing:END[:EVENt] NEVer|ALWays|ESECond|ERGThrshld
SENSe[1]:LOGGing:END[:EVENt]?
SENSe[1]:LOGGing:END:REPort FULL|UREP
SENSe[1]:LOGGing:END:REPort?
SENSe[1]:LOGGing:PORT RS232|ECONtroller
SENSe[1]:LOGGing:PORT?
SENSe[1]:LOGGing:SQUelch 0|1|OFF|ON
SENSe[1]:LOGGing:THReshold <numeric parm>
SENSe[1]:LOGGing:THReshold?
SENSe[1]:LOGGing:SQUelch?
SENSe[1]:PATTern:ZSUBstitut[:ZRUN] <numeric value>
SENSe[1]:PATTern:ZSUBstitut[:ZRUN]?
SENSe[1]:SEEK ONCE|0|1|OFF|ON
SENSe[1]:SEEK?
SENSe[1]:SEEK:PATTern 0|1|OFF|ON
SENSe[1]:SEEK:PATTern?
SENSe2:BANDswitch?
SENSe6:FREQuency:BANDswitch?

SOURce Commands

The SOURce commands control the pattern generator’s Data, $\overline{\text{Data}}$, Clock, $\overline{\text{Clock}}$, and Trigger outputs. They also control the error detector’s Trigger output port.

SOURce1: The Data Source

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
[SOURce[1]:]			
PATTern			
:APCHange			
:IBHalf	ONCE		only if INT & ONES
:MODE	ALTernate | ONEShot | LLEVel | REDGe		changed from 716xxB
:MODE?		ALT | ONES | LLEV | REDG	
:SELect	AHALf | BHALf | ABHAlf		only if INT & ALT/ changed from 716xxB
:SELect?		AHAL | BHAL | ABHA	only if INT & ALT
:SOURce	EXTernal | INTernal		
:SOURce?		EXT | INT	
:EADDition	ONCE | 0 | 1 | OFF | ON		
:EADDition?		0 | 1	
:RATE	<numeric value>		
:RATE?		<NR3>	
:SOURce	EXTernal | FIXed		
:SOURce?		EXT | FIX	
:FORMat			
[:DATA]	PACKed,<numeric value>		

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
[:DATA]?		<NR1>	
:MDENsity			
[:DENSity]	<numeric value>		
[:DENSity]?		<NR3>	
[:SELect]	PRBS<n> | PRBN<n> | ZSUBstitut<n> | MDENsity<n> | UPATtern<n> | FILENAME, <string>		
[:SELect]?		PRBS<n> | PRBN<n> | ZSUB<n> | MDEN<n> | UPAT	
:UFILe			
:DATA	[A|B,]<filename>, <block_data>		new for 86130A
:DATA?	[A|B,]<filename>	<block_data>	
:IDATa	[A|B,]<filename>, <start_bit>, <length_in_bits>, <block_data>		new for 86130A
:IDATa?	[A|B,]<filename>, <start_bit>, <length_in_bits>	<block_data>	
[:LENGth]	<filename>, <numeric_value>		new for 86130A
[:LENGth]?	<filename>	<NR3>	
:LABel	<filename>, <string>		new for 86130A
:LABel?	<filename>	<string>	

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
:NAME?		<string>	new for 86130A
:USE	<filename>, STRaight | APATtern		new for 86130A
:USE?	<filename>	STR | APAT	
:UPATtern<n>			
[:LENGth]	<numeric value>		
[:LENGth]?		<string>	
:LABel	<string>		
:LABel?		<string>	
:USE	STRaight | APATtern		
:USE?		STR | APAT	
:DATA	[A|B,]<block data>		
:DATA?	[A|B]	<block_data>	
:IDATa	[A|B,]<start bit>, <length_in_bits>, <block_data>		
:IDATa?	[A|B,]<start bit>, <length_in_bits>	<block_data>	
VOLTage			
:ECL			event; no query
[:LEVel]			
[:IMMediate]			
[:AMPLitude]	<numeric value>		
[:AMPLitude]?		<NR3>	
:HIGH	<numeric value>		

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
:HIGH?		<NR3>	
:LOW	<numeric value>		new for 86130A
:LOW?		<NR3>	
:OFFSet	<numeric value>		new for 86130A
:OFFSet?		<NR3>	
:LLEVel	ECL | SCFL | LVPECL | CUSTOM | LVTTL		new for 86130A
:LLEVel?		ECL | SCFL| LVPECL | LVTTL | CUSTOM	

[SOURce[1]:]PATTern:APCHange:IBHalf ONCE

This command is short for *Insert B Half*. It causes the single insertion of a number of instances of half B of the alternate pattern to be inserted. It is valid only when :APCHange:SOURce is set to INTernal and :APCHange:MODE is set to ONSHot. It is an event command, and as such has no query form. The number of half 'B' insertions is equal to the smallest integral multiple of the pattern length that divides exactly by 128.

[SOURce[1]:]PATTern:APCHange:MODE ALTernate | ONEShot | LLEVel | REDGe
[SOURce[1]:]PATTern:APCHange:MODE?

This command controls the mode of operation of the alternate pattern output. If ALTernate is selected and the source is set to EXTernal, then the polarity of the signal at the **Aux In** connector governs which half of the pattern is output. If the source is set to INTernal, then the :APCHange:SELect command controls which half of the pattern is output.

If the MODE is set to ONEShot and the source is set to EXTernal, then a single insertion of a number of instances of half B of the pattern is output for each rising edge of the Auxiliary Input. If the source is set to INTernal, then the :APCHange:IBHalf command is used to insert one instance of half B of the pattern. The number of half B instances is equal to the smallest integral multiple of the pattern length that divides exactly by 256. The selections LLEVel and REDGe are new for 86130A.

The response form is the alternate pattern output control mode.The *RST selection is “ALTernate”.

NOTE This command is *not* precisely the same as 716xxB command. The parameters LLEVel and REDGe are available.

NOTE This command is order-sensitive. For more information, refer to “Alternate Pattern Setup” on page 2-17.

[SOURce[1]:]PATTern :APCHange:SELect AHALf | BHALf | AB-HAlf [SOURce[1]:]PATTern :APCHange:SELect?

This command controls whether half A, half B, or both A/B halves of the alternate pattern are output. It is valid only when :APCHange:SOURce is set to INTernal and :APCHange:MODE is set to ALTernate. The selection, ABHAlf, is new for 86130A.

The response form is the output of the alternate pattern parts.

The *RST selection is “AHALf”.

NOTE This command is *not* precisely the same as 716xxB command. The parameter ABHAlf is available.

NOTE This command is order-sensitive. For more information, refer to “Alternate Pattern Setup” on page 2-17.

[SOURce[1]:]PATTern :APCHange:SOURce EXTernal | INTernal [SOURce[1]:]PATTern :APCHange:SOURce?

This command controls the source of control for the alternate pattern output. When EXTernal is selected, the pattern is controlled by the signal at the **Aux In** connector. When INTernal is selected, the pattern is controlled by the user, either from the front-panel or from the GPIB using other commands from within this group.

The response form returns the current control of the alternate patters; external or internal.

The *RST selection is “EXTernal”

NOTE This command is order-sensitive. For more information, refer to “Alternate Pattern Setup” on page 2-17

[SOURce[1]:]PATTern:EADDition ONCE | 0 | 1 | OFF | ON [SOURce[1]:]PATTern:EADDition?

This command is a contraction of the phrase `Error ADDition` and is used to control the addition of errors into the generated pattern. The parameter ONCE causes a single bit error to be added to the pattern. It also turns off the constant rate error addition.

The query form returns the current state of error add.

[SOURce[1]:]PATTern:EADDition:RATE <numeric value> [SOURce[1]:]PATTern:EADDition:RATE?

This command controls the rate of internal, fixed error addition. Values between 1e-3 and 1e-9 in decade steps are permitted.

The query form returns the current error add rate.

[SOURce[1]:]PATTern:EADDition:SOURce EXTernal | FIXed [SOURce[1]:]PATTern:EADDition:SOURce?

This command controls the source of injected errors. When set to EXTernal (and :EADDition[:STATe] is ON), each pulse at the External Errors socket causes an error to be added to the data stream. When set to FIXed (and :EADDition[:STATe] is ON), repetitive errors are internally added to the data stream. The rate of error addition is controlled by the :EADDition:RATE command.

The query form returns the current error add mode; EXT or FIX.

[SOURce[1]:]PATTern:FORMat[:DATA] PACKed,<numeric value> [SOURce[1]:]PATTern:FORMat[:DATA]?

The command form controls the format of data transfer for the :PATTern:UPATtern<n>:DATA and :PATTern:UPATtern<n>:IDATa commands. The PACKed argument permits the packing of bits within a byte to be set. The <numeric value> may be set to either 1 or 8.

The response form returns the current value of the data pack.

[SOURce[1]:]PATTern:MDENsity[:DENSity] <numeric value> [SOURce[1]:]PATTern:MDENsity[:DENSity]?

This command sets the ratio of high bits to the total number of bits in the pattern. The ratio may be varied in eighths, from one to seven (eighths), but excluding three and five (eighths).

The query form returns the mark density in eighths as described above.

[SOURce[1]:]PATTern[:SELect] PRBS<n> | PRBN<n> | ZSUBstitut<n> | MDENsity<n> | UPATtern<n> | FILENAME,<string> [SOURce[1]:]PATTern[:SELect]?

This command defines the type of pattern being generated. The <character data> parameter is retained for backwards compatibility and may be one of the following:

`PRBS<n>`	*<n> = 7, 10, 15, 23, or 31*
`PRBN<n>`	*<n> = 7, 10, 15, or 23*
`ZSUBstitut<n>`	*<n> = 7, 10, 15, or 23*
`UPATtern<n>`	*<n> = 0 through 12*
`MDENsity<n>`	*<n> = 7, 10, 15, and 23*
`FILENAME,`	*<string>*

ZSUBstitut is a contraction of the phrase `Zero SUBstitution` and is used for defining patterns in which a block of bits is replaced by a block of zeros. Zero SUBstitution is not implemented in 86130A.

MDENsity is a contraction of the phrase `Mark DENsity` and is used for defining a pattern in which the user may set the density of marks.

UPATtern<n> is a contraction of the phrase `User PATtern` and is used to define the contents of a pattern store. For the 86130A, <n> is defined as follows:

<n> = 0	current pattern
<n> = 1 through 12	disc storage

FILENAME is a parameter that allows the remote user to load a user pattern from the instrument disk drive.

This is the preferred mechanism for loading user patterns in the 86130A.

Note

If the PG and ED are coupled, setting the pattern by using the SOURce1:PATTern:SELect command will cause the pattern to be set in both the PG and the ED. If the PG and ED are uncoupled, then the ED pattern must be selected using the SENSe1:PATTern:SELect command.

The query form returns the patterns types in short form.

Note

If a user pattern is selected and the [:SELECT]? command is used, then the response is “UPAT”. The particular value of <n> or the name of the file specified in the command form is not returned.

If you wish to get the full path in a response, you may use the following commands:

```
[SOURce[1]:]PATTern:UFILe:NAME?
SENSe[1]:PATTern:UFILe:NAME?
```

[SOURce[1]:]PATTern:UFILe:DATA [A | B,] <filename>, <block_data> [SOURce[1]:]PATTern:UFILe:DATA? [A|B,] <filename>

This command and query allows the use of the full path to specify the user pattern.

The command form sets the bits of the pattern. The bits are sent as an arbitrary block diagram data element. The data may be sent 1 bit/byte or 8 bits/byte, under the control of the :FORMat[:DATA] command. If 1 bit/byte is selected, only numeric values of either binary 1 or binary 0 are allowed. If 8 bits/byte is selected, the most significant bit of the first 8-bit byte received forms the first bit of the pattern.

If ":USE APATtern" is selected, then the first parameter indicates which half pattern is to receive the data. If "USE STRaight" is selected, either "A" or no first parameter are acceptable.

The length of the <block data> embedded in the header refers always to the length in bytes irrespective of the current setting of the [:DATA]PACKed, <numeric value> command.

For example, consider the following header:

#19<data>

#	This indicates the start of the header.
1	The number of decimal digits to follow form the length.
9	This is the length of the data block (in bytes) that follow.
<data>	For data packed with a value of 1, each bit is displayed. For data packed with a value of 8, each digit represents 8 packed bits.

For the 86130A, this command *can change* the pattern length. If the pattern length is extended, the additional length is filled with the incoming data. If fewer bits than specified by the "LENGth" command are sent, the data that resides beyond the block length is truncated.

NOTE This command is order-sensitive. For more information, refer to "User Pattern Edits" on page 2-16

NOTE This command is new for 86130A.

[SOURce[1]:]PATTern:UFILe:IDATa [A | B,] <filename>, <start_bit>, <length_in_bits>, <block_data> [SOURce[1]:]PATTern:UFILe:IDATa? [A|B,] <filename>, <start_bit>, <length_in_bits>

This command and query allows the use of the full path to specify the user pattern.

The command form is similar to the :DATA command. The IDATa command is a contraction of the phrase `Incremental Data` and is used to download part of a user-defined pattern.

If ":USE APATtern" is selected, then the first parameter indicates which half pattern is to receive the data. If "USE STRaight" is selected, either "A" or no first parameter are acceptable.

The length of the <block data> embedded in the header refers always to the length of the data in bytes.

The first parameter defines the starting position within the overall pattern of the first bit of the transmitted pattern. The first bit is counted as bit zero. The second parameter defines how many bits are to be transmitted and the third parameter provides the data itself.

For example, consider the following header:

3,9,#19<data>

3	This indicates the start bit.
9	This is the number of bits.
#	This indicates the start of the header.
1	This is the number of decimal digits to follows that form the length.
9	This is the length of the data block that follows.
<data>	For data packed with a value of 1, each bit is displayed. For data packed with a value of 8, each digit represents 8 packed bits.

The response form returns <block_data>.

NOTE This command is order-sensitive. For more information, refer to "User Pattern Edits" on page 2-16

NOTE This command is new for 86130A.

[SOURce[1]:]PATTern:UFILe[:LENGth] <filename>, <numeric_value> [SOURce[1]:]PATTern:UFILe[:LENGth]? <filename>

This command and query allows the use of the full path to specify the user pattern.

When USE STRaight is selected, the command form sets the length of the user pattern.

If an alternate pattern is selected (USE APATtern), the LENGth command sets length of each half of the pattern.

The response form returns the entire length of a STRaight pattern and the length of each half of an APATern pattern.

NOTE This command is new for 86130A.

[SOURce[1]:]PATTern:UFILe:LABel <filename>, <string> [SOURce[1]:]PATTern:UFILe:LABel? <filename>

This command and query allows the use of the full path to specify the user pattern.

The command form gives the user pattern a label.

The response form returns the user pattern's label.

NOTE This command is new for 86130A.

[SOURce[1]:]PATTern:UFILe:NAME?

This query returns a full path of the current user pattern.

NOTE This command is new for 86130A.

[SOURce[1]:]PATTern:UFILe:USE <filename>, STRaight | APATtern [SOURce[1]:]PATTern:UFILe:USE? <filename>

This command and query allows the use of the full path to specify the user pattern.

When USE STRaight is selected the whole of the pattern is repeatedly output.

When USE APATtern is selected, the pattern is considered to be composed of two halves. The store is set to have a length of 128 bits for each half pattern; all bits are set to zero and the trigger is set to occur on the A/B changeover.

NOTE This command is order-sensitive. For more information, refer to “Alternate Pattern Setup” on page 2-17

NOTE This command is new for 86130A.

[SOURce[1]:]PATTern:UPATtern<n>[:LENGth] <numeric value> [SOURce[1]:]PATTern:UPATtern<n>[:LENGth]?

When USE STRaight is selected, the command form sets the length of the user pattern.

If an alternate pattern is selected (USE APATtern), the LENGth command sets length of each half of the pattern.

The response form returns the entire length of a STRaight pattern and the length of each half of an APATern pattern.

NOTE This command is order-sensitive. For more information, refer to “User Pattern Edits” on page 2-16.

[SOURce[1]:]PATTern:UPATtern<n>:LABel <string> [SOURce[1]:]PATTern:UPATtern<n>:LABel?

The command form gives the user pattern a label.

The response form returns the user pattern’s label.

[SOURce[1]:]PATTern:UPATtern<n>:USE STRaight | APATtern [SOURce[1]:]PATTern:UPATtern<n>:USE?

When USE STRaight is selected the whole of the pattern is repeatedly output.

When USE APATtern is selected, the pattern is considered to be composed of two halves. The store is set to have a length of 128 bits for each half pattern; all bits are set to zero and the trigger is set to occur on the A/B changeover.

NOTE This command is order-sensitive. For more information, refer to “Alternate Pattern Setup” on page 2-17

[SOURce[1]:]PATTern:UPATtern<n>:DATA [A|B,] <block data> [SOURce[1]:]PATTern:UPATtern<n>:DATA? [A|B,]

This command form sets the bits of the pattern. The bits are sent as an arbitrary block diagram data element. The data may be sent 1 bit/byte or 8 bits/byte, under the control of the :FORMat[:DATA] command. If 1 bit/byte is selected, only numeric values of either binary 1 or binary 0 are allowed. If 8 bits/byte is selected, the most significant bit of the first 8-bit byte received forms the first bit of the pattern.

If ":USE APATtern" is selected, then the first parameter indicates which half pattern is to receive the data. If "USE STRaight" is selected, either "A" or no first parameter are acceptable.

The length of the <block data> embedded in the header refers always to the length in bytes irrespective of the current setting of the [:DATA]PACKed, <numeric value> command.

For example, consider the following header:

```
#19<data>
```

#	This indicates the start of the header.
1	The number of decimal digits to follow form the length.
9	This is the length of the data block (in bytes) that follow.
<data>	For data packed with a value of 1, each bit is displayed. For data packed with a value of 8, each digit represents 8 packed bits.

For the 86130A, this command *can change* the pattern length. If the pattern length is extended, the additional length is filled with the incoming data. If fewer bits than specified by the "LENGth" command are sent, the data that resides beyond the block length is truncated.

NOTE This command is order-sensitive. For more information, refer to "User Pattern Edits" on page 2-16

[SOURce[1]:]PATTern:UPATtern<n>:IDATa [A|B,] <start_bit>,<length_in_bits>,<block_data> [SOURce[1]:]PATTern:UPATtern<n>:IDATa? [A|B,] <start_bit>,<length_in_bits>

The command form is similar to the :DATA command. The IDATa command is a contraction of the phrase `Incremental Data` and is used to download part of a user-defined pattern.

If ":USE APATtern" is selected, then the first parameter indicates which half pattern is to receive the data. If "USE STRaight" is selected, either "A" or no first parameter are acceptable.

The length of the <block data> embedded in the header refers always to the length of the data in bytes.

The first parameter defines the starting position within the overall pattern of the first bit of the transmitted pattern. The first bit is counted as bit zero. The second parameter defines how many bits are to be transmitted and the third parameter provides the data itself.

For example, consider the following header:

3,9,#19<data>

3	This indicates the start bit.
9	This is the number of bits.
#	This indicates the start of the header.
1	This is the number of decimal digits to follows that form the length.
9	This is the length of the data block that follows.
<data>	For data packed with a value of 1, each bit is displayed. For data packed with a value of 8, each digit represents 8 packed bits.

The response form returns <block_data>.

NOTE This command is order-sensitive. For more information, refer to "User Pattern Edits" on page 2-16

[SOURce[1]:]VOLTage:ECL

This command sets the data output values to those used for the ECL family. Retained for backwards compatibility. Superseded by SOURce1:VOLTage:LLEVel.

[SOURce[1]:]VOLTage[:LEVel][:IMMediate][:AMPLitude] <numeric value> [SOURce[1]:]VOLTage[:LEVel][:IMMediate][:AMPLitude]?

The command form sets the peak to peak value of the data signal in units of Volts.

The query form returns the peak to peak value of the data signal in units of Volts.

NOTE This command is order-sensitive. For more information, refer to "Output Level Adjustments" on page 2-20

[SOURce[1]:]VOLTage[:LEVel][:IMMediate]:HIGH <numeric value> [SOURce[1]:]VOLTage[:LEVel][:IMMediate]:HIGH?

The command form is used to set the DC high output level in units of Volts.

The query form returns the DC high output level in units of Volts.

NOTE This command is order-sensitive. For more information, refer to "Output Level Adjustments" on page 2-20

[SOURce[1]:]VOLTage[:LEVel][:IMMediate]:LOW <numeric value> [SOURce[1]:]VOLTage[:LEVel][:IMMediate]:LOW?

The command form is used to set the DC low output level in units of Volts.

The query form returns the DC low output level in units of Volts.

NOTE This command is order-sensitive. For more information, refer to "Output Level Adjustments" on page 2-20

NOTE This command is new for 86130A.

[SOURce[1]:]VOLTage[:LEVel][:IMMediate]:OFFSet <numeric value>
[SOURce[1]:]VOLTage[:LEVel][:IMMediate]:OFFSet?

This command form is used to set the mean of the high and low DC output level in units of Volts.

The query form returns the mean of the high and low DC output level in units of Volts.

NOTE This command is order-sensitive. For more information, refer to "Output Level Adjustments" on page 2-20

NOTE This command is new for 86130A.

[SOURce[1]:]VOLTage:LLEVel ECL | LVPECL | SCFL | LVTTL | CUSTOM
[SOURce[1]:]VOLTage:LLEVel?

The command form sets the output level appropriate for the specified logic level. The query form returns the output level for the specified logic level.

NOTE This command is new for 86130A.

SOURce2: The Clock Source

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
SOURce2			
:FREQuency			changed from 716xxB
[:CW|:FIXed]?	[MINimum | MAXimum]	<NR3>	query only
:VOLTage			
:ECL			event; no query
[:LEVel]			
[:IMMediate]			
[:AMPLitude]	<numeric value>		
[:AMPLitude]?		<NR3>	
:HIGH	<numeric value>		
:HIGH?		<NR3>	
:LOW	<numeric value>		new for 86130A
:LOW?		<NR3>	
:OFFSet	<numeric value>		new for 86130A
:OFFSet?		<NR3>	
:LLEVel	ECL | SCFL | LVPECL | LVTTL | CUSTOM		new for 86130A
:LLEVel?		ECL | SCFL | LVPECL | LVTTL | CUSTOM	

SOURce2:FREQuency[:CW | :FIXed]? [MINimum | MAXimum]

This query returns the bit rate of the measured frequency from internal or external clock. This command is superseded by SENSe6:FREQuency [:CW|:FIXed]?

This command is retained only for backwards compatibility with 716xxB systems.

NOTE This command is *not* precisely the same as 716xxB command. The numeric value parameter is no longer supported.

SOURce2:VOLTage:ECL

Sets the output “AMPLitude” and “HIGH” values to those used for the ECL family. There is no query form for this command. This command is provided for backwards compatibility only and is superseded by SOURce2:VOLTage:LLEVel.

SOURce2:VOLTage [:LEVel][:IMMediate][:AMPLitude] <numeric value>
SOURce2:VOLTage [:LEVel][:IMMediate][:AMPLitude]?

The command form sets the peak to peak value of the Clock signal in units of Volts.

The query form returns the peak to peak value of the Clock signal in units of Volts.

NOTE This command is order-sensitive. For more information, refer to “Output Level Adjustments” on page 2-20

SOURce2:VOLTage[LEVel][:IMMediate]:HIGH <numeric value>
SOURce2:VOLTage[LEVel][:IMMediate]:HIGH?

The command form sets the DC high output level of the Clock output in Volts.

The query form returns the DC high output level of the Clock output in Volts.

NOTE This command is order-sensitive. For more information, refer to “Output Level Adjustments” on page 2-20

SOURce2:VOLTage[LEVel][:IMMediate]:LOW <numeric value> SOURce2:VOLTage[LEVel][:IMMediate]:LOW?

The command form sets the DC low output level of the Clock output in Volts.

The query form returns the DC low output level of the Clock output in Volts.

NOTE This command is order-sensitive. For more information, refer to "Output Level Adjustments" on page 2-20

NOTE This command is new for 86130A.

SOURce2:VOLTage[:LEVel][:IMMediate]:OFFSet <numeric value> SOURce2:VOLTage[:LEVel][:IMMediate]:OFFSet?

The command form sets the offset value of the of the Clock output in Volts.

The query form returns the offset value of the of the Clock output in Volts.

NOTE This command is order-sensitive. For more information, refer to "Output Level Adjustments" on page 2-20

NOTE This command is new for 86130A.

SOURce2:VOLTage:LLEVel ECL | LVPECL | SCFL | LVTTL| CUS-TOM SOURce2:VOLTage:LLEVel?

The command form sets the output levels appropriate for the specified logic level.

The query form returns the output levels for the specified logic level.

NOTE This command is new for 86130A.

SOURce3: The Trigger Source

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
SOURce3			
:TRIGger			
:APATtern<n>	ABCHange | SOPattern		
:APATtern<n>?		ABCH | SOP	
:MDENsity<n>	<numeric value>		
:MDENsity<n>?		<NR1>	
[:MODE]	DCLock | PATTern		
[:MODE]?		DCL | PATT	
:PRBN<n>	<numeric value>		
:PRBN<n>?		<NR1>	
:PRBS<n>	<0 | 1 | OFF | ON> {,<0 | 1 | OFF | ON>}		
:PRBS<n>?		<0 | 1> {,<0 | 1>}	
:UPATtern<n>	<numeric value>		
:UPATtern<n>?		<NR1>	

SOURce3:TRIGger:APATtern<n> ABCHange | SOPattern SOURce3:TRIGger:APATtern<n>?

For alternate patterns only

For the parameter ABChange, the pattern generator trigger output is synchronized to the voltage level at the **Aux In** port.

For the parameter SOPattern, the pattern generator trigger output is synchronized to the start of pattern A or pattern B.

The query form returns the current state of the alternate pattern trigger mode.

NOTE This command is order-sensitive. For more information, refer to “User Pattern Edits” on page 2-16

SOURce3:TRIGger:MDENsity<n> <numeric value> SOURce3:TRIGger:MDENsity<n>?

This command selects the position within the PRBS at which the trigger pulse is to be output whenever a mark density PRBS is selected. The number ‘n’ must be one of 7, 10, 15, and 23. The parameter must be in the range 0 through pattern length - 1.

The query form returns the position within the PRBS at which the trigger pulse is to be output whenever a mark density PRBS is selected.

SOURce3:TRIGger[:MODE] DCLock | PATTern SOURce3:TRIGger[:MODE]?

The command form sets the pattern generator trigger output to pattern or divided clock mode

The query form returns the current pattern generator trigger output mode.

SOURce3:TRIGger:PRBN<n> <numeric value> SOURce3:TRIGger:PRBN<n>?

The command form sets a bit position within the pattern. This bit position is used to synchronize the pattern generator trigger output.

The query form returns the bit position within the pattern.

SOURce3:TRIGger:PRBS<n> <0 | 1 | OFF | ON>{,<0 | 1 | OFF | ON>} SOURce3:TRIGger:PRBS<n>?

This command sets the pattern, the occurrence of which causes a trigger pulse to be output. The number ‘n’ must one of 7, 10, 15, 23 or 31. The number of parameters depends on the pattern length, and is the minimum that can define a unique place in the overall pattern, for example a pattern of length 2^{n-1} the number of parameters is `n`. The parameter values are either 1 or 0. An *all-ones* pattern is disallowed.

The query form returns the state of the N-bit trigger pattern function for the pattern generator trigger output.

SOURce3:TRIGger:UPATtern<n> <numeric value> SOURce3:TRIGger:UPATtern<n>?

This command form selects a position within the user pattern at which the trigger pulse is to be output. The parameter must be in the range of 0 through pattern length.

The response form returns the current bit position within the user pattern at which the trigger pulse is output.

SOURce7: The Trigger Source

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
SOURce7			
:TRIGger			
[:MODE]	PATTern | DCLock		
[:MODE]?		PATT | DCL	

SOURce7:TRIGger[:MODE] DCLock | PATTern
SOURce7:TRIGger[:MODE]?

This command configures the TRIGGER OUT port from the error detector to be either a divided clock mode (a square wave at clock rate/8) or pattern mode (a pulse synchronized to repetitions of the pattern).

The query form returns current mode for the trigger output of the error detector.

SOURce9: The Clock Source

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
SOURce9			
:FREQuency			
[:CW |:FIXed]	<numeric value>		
[:CW |:FIXed]?	[MINimum | MAXimum]	<NR3>	
:OUTPut			
[:STATe]	0 | 1 | OFF | ON		
[:STATe]?		0 | 1	

SOURce9:FREQuency[:CW|:FIXed] <numeric value> SOURce9:FREQuency[:CW|:FIXed]? [MINimum | MAXimum]

This command may be used to configure the internal clock source frequency. You may also use any of the forms listed below:

SOURce9:FREQuency
SOURce9:FREQuency:CW
SOURce9:FREQuency:FIXed

The response form returns the current internal clock source frequency.

SOURce9:OUTPut[:STATe] 0 | 1 | OFF | ON SOURce9:OUTPut[:STATe]?

This command switches the internal clock source off or on.

The response form returns whether or not the internal clock source is currently on or off.

NOTE

To setup the instrument to accept an external clock input, send SOURce9:OUTPut[:STATe] OFF.

SOURce10: The Data Source

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
SOURce10			
:VOLTage			
:ECL			
[:LEVel]			
[:IMMediate]			
[:AMPLitude]	<numeric value>		
[:AMPLitude]?		<NR3>	
:HIGH	<numeric value>		
:HIGH?		<NR3>	
:LOW	<numeric value>		new for 86130A
:LOW?		<NR3>	
:OFFSet	<numeric value>		new for 86130A
:OFFSet?		<NR3>	
:LLEVel	ECL | SCFL | LVPECL | LVTTL | CUSTOM		new for 86130A
:LLEVel?		ECL | SCFL | LVPECL | LVTTL | CUSTOM	
:TRACk	0 | 1 | OFF | ON		
:TRACk?		0 | 1	

SOURce10:VOLTage:ECL

This command sets the $\overline{\text{Data}}$ output to track the ECL family levels.

SOURce10:VOLTage[:LEVel][:IMMediate][:AMPLitude] <numeric value> SOURce10:VOLTage[:LEVel][:IMMediate][:AMPLitude]?

This command sets the peak to peak value of the $\overline{\text{Data}}$ signal, in units of millivolts. This command applies when data tracking is disabled.

The query form returns the current peak-to-peak value of the $\overline{\text{Data}}$ signal.

SOURce10:VOLTage [:LEVel][:IMMediate]:HIGH <numeric value> SOURce10:VOLTage [:LEVel][:IMMediate]:HIGH?

This command is used to set the high output level, in units of millivolts. This command applies when data tracking is disabled.

The query form returns the current high level of the $\overline{\text{Data}}$ output.

SOURce10:VOLTage [:LEVel][:IMMediate]:LOW <numeric value> SOURce10:VOLTage [:LEVel][:IMMediate]:LOW?

This command is used to set the low output level, in units of millivolts. This command applies when data tracking is disabled.

The query form returns the current low level of the $\overline{\text{Data}}$ output.

NOTE This command is new for 86130A.

SOURce10:VOLTage[:LEVel][:IMMediate]:OFFSet <numeric value> SOURce10:VOLTage[:LEVel][:IMMediate]:OFFSet?

This command sets the offset value of the data bar signal, in units of millivolts. This command applies when data tracking is disabled.

The query form returns the current offset of the $\overline{\text{Data}}$ output.

NOTE This command is new for 86130A.

SOURce10:VOLTage:LLEVel ECL | SCFL | LVPECL | LVTTL | CUSTOM SOURce10:TRACK:LLEVel?

This command sets the $\overline{\text{Data}}$ output to one of the following logic families:

ECL, SCFL, LVPECL, LVTTL, or CUSTOM.

The query form returns the current logic family to which $\overline{\text{Data}}$ is set.

NOTE This command is new for 86130A.

SOURce10:VOLTage:TRACk 0 | 1 | OFF | ON SOURce10:VOLTage:TRACk?

The command form enables the $\overline{\text{Data}}$ output settings to track the Data output settings.

The query form returns the current state of data tracking.

SOURce11: The $\overline{\text{Clock}}$ Source

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
SOURce11			
:VOLTage			
:ECL			new for 86130A
[:LEVel]			
[:IMMediate]			
[:AMPLitude]	<numeric value>		
[:AMPLitude]?		<NR3>	
:HIGH	<numeric value>		
:HIGH?		<NR3>	
:LOW	<numeric value>		new for 86130A
:LOW?		<NR3>	
:OFFSet	<numeric value>		new for 86130A
:OFFSet?		<NR3>	
:LLEVel	ECL | SCFL | LVPECL | LVTTL | CUSTOM		new for 86130A
:LLEVel?		ECL | SCFL | LVPECL | LVTTL | CUSTOM	
:TRACk	0 | 1 | OFF | ON		
:TRACk?		0 | 1	

SOURce11:VOLTage:ECL

This command sets the $\overline{\text{Clock}}$ output to the ECL family values.

NOTE This command is new for 86130A.

SOURce11:VOLTage[:LEVel][:IMMediate][:AMPLitude] <numeric value> SOURce11:VOLTage[:LEVel][:IMMediate][:AMPLitude]?

The command form sets the peak to peak value of the $\overline{\text{Clock}}$ output, in units of volts.

The query form returns the peak to peak value of the $\overline{\text{Clock}}$ output, in units of volts.

This command applies when clock tracking is disabled.

SOURce11:VOLTage[:LEVel][:IMMediate]:HIGH <numeric value> SOURce11:VOLTage[:LEVel][:IMMediate]:HIGH?

The command form sets the high output level, in units of millivolts.

The query form returns the high output level, in units of millivolts.

This command applies when clock tracking is disabled.

SOURce11:VOLTage[:LEVel][:IMMediate]:LOW <numeric value> SOURce11:VOLTage[:LEVel][:IMMediate]:LOW?

The command form sets the low output level, in units of millivolts.

The query form returns the low output level, in units of millivolts.

This command applies when clock tracking is disabled.

NOTE This command is new for 86130A.

SOURce11:VOLTage[:LEVel][:IMMediate]:OFFSet <numeric value> SOURce11:VOLTage[:LEVel][:IMMediate]:OFFSet?

The command form sets the offset level, in units of millivolts.

The query form returns the offset level, in units of millivolts.

This command applies when clock tracking is disabled.

NOTE This command is new for 86130A.

SOURce11:VOLTage:LLEVel ECL | SCFL | LVPECL | LVTTL | CUSTOM SOURce11:VOLTage:LLEVel?

The command form sets the $\overline{\text{Clock}}$ Output to the to one of the following logic families:
ECL, SCFL, LVPECL, LVTTL, or CUSTOM.

The query form returns the logic family of the $\overline{\text{Clock}}$ Output.

NOTE This command is new for 86130A.

SOURce11:VOLTage:TRACk 0 | 1 | OFF | ON SOURce11:VOLTage:TRACK?

The command form enables the $\overline{\text{clock}}$ output settings to track the clock output settings.

The query form returns the current status of clock tracking.

The following SOURce commands are not supported:

[SOURce[1]:]PATTern:ZSUBstitut[:ZRUN] <numeric value>
[SOURce[1]:]PATTern:ZSUBstitut[:ZRUN]?
[SOURce[1]:]PATTern:UPATtern<n>:LMODified?
[SOURce[1]:]PATTern:AWORd:DATA<n>
[SOURce[1]:]PATTern:AWORd:DATA<n>?
SOURce3:TRIGger:DCDRatio <numeric value>

SOURce3:TRIGger:DCDRatio?
SOURce3:TRIGger:CTDRatio <numeric value>
SOURce3:TRIGger:CTDRatio?
SOURce4:VOLTage[:LEVel][:IMMediate][:AMPLitude] <numeric value>
SOURce4:VOLTage[:LEVel][:IMMediate][:AMPLitude]?
SOURce4:VOLTage[:LEVel][:IMMediate]:HIGH <numeric value>
SOURce4:VOLTage[:LEVel][:IMMediate]:HIGH?
SOURce4:VOLTage:ECL
SOURce5:VOLTage[:LEVel][:IMMediate][:AMPLitude] <numeric value>
SOURce5:VOLTage[:LEVel][:IMMediate][:AMPLitude]?
SOURce5:VOLTage[:LEVel][:IMMediate]:HIGH <numeric value>
SOURce5:VOLTage[:LEVel][:IMMediate]:HIGH?
SOURce5:VOLTage:ECL
SOURce9:FREQuency[:CW|:FIXed]:STEP[:INCRement] <numeric value>
SOURce9:FREQuency[:CW|:FIXed]:STEP[:INCRement]?
SOURce9:POWer[:LEVel][:IMMediate][:AMPLitude] <numeric value>
SOURce9:POWer[:LEVel][:IMMediate][:AMPLitude]?
SOURce9:IDN?

STATus Subsystem

The STATus commands control the SCPI-compatible status reporting structures.

The STATus Subsystem

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
STATus			
:CLOSs			
:CONDition?		<NR1>	query only
:ENABle	<numeric value>		
:ENABle?		<NR1>	
[:EVENt]?		<NR1>	query only
:NTRansition	<numeric value>		
:NTRansition?		<NR1>	
:PTRansition	<numeric value>		
:PTRansition?		<NR1>	
:FAILure			
:CALibration			
:EVENt?		<NR1>	query only
:EVENt?		<NR1>	query only
:RAM			
:EVENt?		<NR1>	query only

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
:ROM			
:EVENt?		<NR1>	query only
:SOFTware			
:EVENt?		<NR1>	query only
:TEMPerature			
:EVENt?		<NR1>	query only
:OPERation			
:CONDition?		<NR1>	query only
:ENABle	<numeric value>		
:ENABle?		<NR1>	
[:EVENt]?		<NR1>	query only
:NTRansition	<numeric value>		
:NTRansition?		<NR1>	
:PTRansition	<numeric value>		
:PTRansition?		<NR1>	
:PRESet			command only
:QUEStionable			
:CONDition?		<NR1>	query only
:ENABle	<numeric value>		
:ENABle?		<NR1>	
[:EVENt]?		<NR1>	query only
:NTRansition	<numeric value>		
:NTRansition?		<NR1>	
:PTRansition	<numeric value>		
:PTRansition?		<NR1>	

STATus:CLOSs:CONDition?

This query only returns the contents of the condition register in the Clock Loss Status Register.

STATus:CLOSs:ENABle <numeric value>
STATus:CLOSs:ENABle?

The command form sets the enable mask in the Clock Loss Register, which allows true conditions in the event register to be reported in the summary bit.

The query form returns the weighted value of the bits that are set in the enable register.

STATus:CLOSs[:EVENt]?

The bits in this register indicate pattern generator and error detector clock loss.

Refer to Table 3-3 for a definition of the bits within the clock loss register group.

This query returns whether the pattern generator or error detector has experienced the clock loss.

STATus:CLOSs:NTRansition <numeric value>
STATus:CLOSs:NTRansition?

This command sets the transition filter state in the Clock Loss Register. When this mask is set to “1”, negative (logic 1 changing to logic 0) transitions are allowed to pass.

STATus:CLOSs:PTRansition <numeric value>
STATus:CLOSs:PTRansition?

This command sets the transition filter state in the Clock Loss Register. When this mask is set to “1”, positive transitions (logic 0 changing to logic 1) are allowed to pass. This is the default setting of the instrument.

The query form returns the weighted value of the bits that are set to pass positive transitions in the transition filter.

The query form returns the weighted value of the bits that are set to pass negative transitions in the transition filter.

STATus:FAILure:CALibration:EVENt?

The bits in this register indicate that a calibration element has failed. No capability is provided to query the condition register, setup the enable register, nor setup the positive or negative transition filters. This is because failures within this category are non-recoverable, and as such the enable registers are pre-defined.

STATus:FAILure:EVENt?

The bits in this register indicate that a major hardware or software element of the instrument has failed. No capability is provided to query the condition of the register, setup the enable register, nor setup the positive or negative trasition filters. This is because failures within this category are non-recoverable, and as such the enable registers are pre-defined.

STATus:FAILure:RAM:EVENt?

The bits in this register indicate that a RAM element has failed. No capability is provided to query the condition register, setup the enable register, nor setup the positive or negative transition filters. This is because failures within this category are non-recoverable, and as such the enable registers are pre-defined.

STATus:FAILure:ROM:EVENt?

The bits in this register indicate that a flash ROM element has failed. No capability is provided to query the condition register, setup the enable register, nor setup the positive or negative transition filters. This is because failures within this category are non-recoverable, and as such the enable registers are pre-defined.

STATus:FAILure:SOFTware:EVENt?

The bits in this register indicate that a software element has failed. No capability is provided to query the condition register, setup the enable register, nor setup the positive or negative transition filters. This is because failures within this category are non-recoverable, and as such the enable registers are pre-defined.

STATus:FAILure:TEMPerature:EVENt?

The bits in this register indicate that a temperature element has failed. No capability is provided to query the condition register, setup the enable register, nor setup the positive or negative transition filters. This is because failures within this category are non-recoverable, and as such the enable registers are pre-defined.

STATus:OPERation:CONDition?

This query only returns the contents of the condition register in the Operation Status Register.

STATus:OPERation:ENABle <numeric value>
STATus:OPERation:ENABle?

The command form sets the enable mask in the Operation Status Register, which allows true conditions in the event register to be reported in the summary bit.

The query form returns the weighted value of the bits that are set in the enable register.

STATus:OPERation[:EVENt]?

In the Operation Status Register group, this query form returns the contents of the event register. For more information, refer to "Operation Status Register Group" on page 3-22.

STATus:OPERation:NTRansition <numeric value>
STATus:OPERation:NTRansition?

This command sets the transition filter state in the Operation Status Register. When this mask is set to “1”, negative (logic 1 changing to logic 0) transitions are allowed to pass.

The query form returns the weighted value of the bits that are set to pass negative transitions in the transition filter.

STATus:OPERation:PTRansition <numeric value>
STATus:OPERation:PTRansition?

This command sets the transition filter state in the Operation Status Register. When this mask is set to “1”, positive transitions (logic 0 changing to logic 1) are allowed to pass. This is the default setting of the instrument.

The query form returns the weighted value of the bits that are set to pass positive transitions in the transition filter.

STATus:PRESet

The PRESet command is an event that configures the SCPI and device dependent status data structures, such that the device dependent events are reported at a higher level through the mandatory part of the status reporting structures.

The PRESet command affects only the enable register and the transition filter registers for the SCPI mandated and device dependent status data structures. PRESet does not affect either the “status byte” or the “standard event status” as defined by IEEE 488.2. PRESet does not clear any of the event registers. The *CLS command is used to clear all event registers in the device status reporting mechanism.

From the device dependent status data structures, the PRESet command sets the enable register to all one’s and the transition filter to recognize both positive and negative transitions. For the SCPI mandatory status data structures, the PRESet command sets the transition filter registers to recognize only positive transitions and sets the enable register to zero.

STATus:QUEStionable:CONDition?

This query only returns the contents of the condition register in the Questionable Status Register.

STATus:QUEStionable:ENABle <numeric value>
STATus:QUEStionable:ENABle?

The command form sets the enable mask in the Questionable Status Register, which allows true conditions in the event register to be reported in the summary bit.

The query form returns the weighted value of the bits that are set in the enable register.

STATus:QUEStionable[:EVENt]?

In the Questionable Status Register group, this query form returns the contents of the event register.

For more information, refer to "Questionable Status Register Group" on page 3-17

STATus:QUEStionable:NTRansition <numeric value>
STATus:QUEStionable:NTRansition?

This command sets the transition filter state in the Questionable Status Register. When this mask is set to "1", negative (logic 1 changing to logic 0) transitions are allowed to pass.

The query form returns the weighted value of the bits that are set to pass negative transitions in the transition filter.

STATus:QUEStionable:PTRansition <numeric value>
STATus:QUEStionable:PTRansition?

This command sets the transition filter state in the Questionable Status Register. When this mask is set to "1", positive transitions (logic 0 changing to logic 1) are allowed to pass. This is the default setting of the instrument.

The query form returns the weighted value of the bits that are set to pass positive transitions in the transition filter.

SYSTem Subsystem

The SYSTem commands control functions such as general housekeeping and global configurations.

The SYSTem Subsystem

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
SYSTem			
:BEEPer			
:MODE	BERalarm|TONes		new for 86130A
:MODE?		BER | TON	
:STATe	0 | 1 | OFF | ON		
:STATe?		0 | 1	
:THReshold	<numeric value>		new for 86130A
:THReshold?		<NR3>	
:VOLume	<numeric value>		
:VOLume?		<NR1>	
:ERRor			
[:NEXT]?		<NR1>, <string>	query only
:HELP			
:HEADers?		<string> [,<string]	new for 86130A

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
:PTHRough	<string>		not supported commands
:PTHRough?	<string>	<string>	
:VERSion?		<string>	query only

SYSTem:BEEPer:MODE BERalarm | TONes SYSTem:BEEPer:MODE?

The command form sets the instrument's audible beeper to trigger on a specific BER level (BERalarm) or on any occurrence of errors (TONes).

The response form returns the current mode setting of the instrument's audible beeper.

NOTE This command is new for 86130A.

SYSTem:BEEPer:STATe 0 | 1 | OFF | ON SYSTem:BEEPer:STATe?

The command form turns on and off the instrument's audible beeper.

The response form returns whether or not the instrument's audible beeper is turned on.

SYSTem:BEEPer:THReshold <numeric value> SYSTem:BEEPer:THREshold?

The command form sets the BER threshold value at which the instrument's audible beeper will produce sounds.

The response form returns the current setting of the BER threshold at which the instrument's audible beeper will produce sounds.

SYSTem:BEEPer:VOLume <numeric value>
SYSTem:BEEPer:VOLume?

The command form controls the volume of the instrument's audible beeper.

The response form returns the current volume of the instrument's audible beeper.

SYSTem:ERRor[:NEXT]?

This query-only command will pull the next error from the error queue, and return the error number and a string describing the error. The error queue has a depth of twenty.

Note

SCPI-defined errors are all negative. The positive error numbers are specific to the Error Performance Analyzer. The SCPI Messages section at the rear of this manual contains a list of error numbers.

SYSTem:HELP:HEADers?

This query returns the complete list of instrument commands. Not all of the commands are implemented, however. For more information, refer to the specific command groups in this guide.

NOTE This command is new for 86130A.

SYSTem:PTHRough <string>
SYSTem:PTHRough? <string>

The Pass-Through command allows a remote programming command to be passed through an MMS master module to a slave module.

NOTE For the 86130A, this is an unsupported command. It is functional, however, and is included here for your reference.

SYSTem:VERSion?

This query returns the version of the SCPI programming language, which supports the GPIB commands.

NOTE This command is *not* recommended for software releases prior to A.01.03.

NOTE This command is new for 86130A.

The following SYSTem commands are not supported:

SYSTem:BEEPer[:IMMediate] [<freq>[,<time>[,<vol>]]]
SYSTem:DATE
SYSTem:DATE?
SYSTem:FREVision[:CPRocessor][:APPLication]?
SYSTem:FREVision[:CPRocessor]:BOOT?
SYSTem:FREVision[:MPRocessor][:APPLication]?
SYSTem:FREVision[:MPRocessor]:BOOT?
SYSTem:FUPDate CAPPlication|MAPPlication
SYSTem:KLOCk
SYSTem:KLOCk?
SYSTem:PRESet
SYSTem:TIME
SYSTem:TIME?

TEST Subsystem

Self Test verifies specific hardware components for basic functionality.

NOTE This command is similar to the *TST? command.

The TEST Subsystem

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
TEST			
:EXECute?	[SelfTest_value] {,<SelfTest_value>}	1 | 0	

TEST:EXECute? [SelfTest_value] {,<SelfTest_value>}

This command runs user-specified self tests. If no parameter is specified, only the PG Tray Presence and ED Tray Presence tests are run.

Successful completion of a self test returns 0. If a self test fails, 1 is returned.

NOTE The results of the current and previous self tests are stored in E:\Bitalyzer\SelfTest Logs.

SelfTest_value can be one of the parameters listed below.

PARAMETER	DESCRIPTION	COMMENTS
BTPCicard	PCI Card Ready Test	
EDCal	ED Calibration Status	
EDEelb	ED EELB Presence	
EDFLash	ED FlashROM Contents	

PARAMETER	DESCRIPTION	COMMENTS
EDFPga	ED FPGA Configuration Test	
EDOVertemp	ED Over Temperature Test	
EDRam	ED RAM RW Test	
EDTRay	ED Tray Presence	
PCGPibresp	GPIB Response Test	
PCGuiresp	GUI Response Test	
PCHDisk	Free Hard Disk Space Test	
PCRam	Free Memory Space Test	
PGCal	PG Calibration Status	
PGFlash	PG FlashROM Contents	
PGFPga	PG FPGA Configuration Test	
PGOVertemp	PG Over Temperature Test	
PGRam	PG RAM RW Test	
PGTRay	PG Tray Presence	
STBert		parameter runs all BERT self tests
STED		parameter runs all ED self tests
STPC		parameter runs all PC self tests
STPG		parameter runs all PG self tests

Front Panel Functions to Remote Commands

This is a table of the front-panel functions of the 86130A and the corresponding remote commands. Refer to "Front Panel Functions to Error Analysis Remote Commands" on page 5-88 for front-panel to remote Error Analysis commands.

Table 4-2. Front Panel Function to Remote Command for the Agilent 86130A

Front Panel Function	Remote Command
PRESET	SYSTem:PRESet | :PRESet<n>
ED Setup	
Audio	
Audio on	SYSTem:BEEPer:STATe 0 | 1 | OFF | ON
Main Volume	SYSTem:BEEPer:VOLume <numeric value>
Audio on BER Alarm	SYSTem:BEEPer:MODE BERalarm
Tones on All Error Rates	SYSTem:BEEPer:MODE TONes
BER Alarm Threshold	SYSTem:BEEPer:THReshold <numeric value>
Sampling Point Setup	
0/1 Threshold	SENSe[1]:VOLTage:ZOTHreshold <numeric value>
0/1 Threshold Center	SENSe[1]:EYE:ACENter ONCE | 0 | 1 | OFF | ON SENSe[1]:EYE:HEIGht?
Quick 0/1 Threshold Center	SENSe[1]:EYE:QUICk:ACENter ONCE | 0 | 1 | OFF | ON
Auto Align	SENSe[1]:EYE:ALIGN:AUTO 0 | 1 | OFF | ON
Quick Auto Align	SENSe[1]:EYE:QUICk:ALIGN:AUTO 0 | 1 | OFF | ON

Table 4-2. Front Panel Function to Remote Command for the Agilent 86130A

Front Panel Function	**Remote Command**
Avg. 0/1 Threshold	SENSe[1]:VOLTage:ZOTHreshold:AUTO 0 | 1 | OFF | ON
Clock/Data Center	SENSe[1]:EYE:TCENter ONCE | 0 | 1 | OFF | ON SENSe[1]:EYE:WIDTh?
Quick Clock/Data Center	SENSe[1]:EYE:QUICk:TCENter ONCE | 0 | 1 | OFF | ON
Clock Inverted Edge	SENSe2:VOLTage:EDGE NEGative | POSitive
Clock Termination	INPut2:TERMination 0 | -2 | 1.3
Data Input Delay	INPut1:DELay <numeric value>
Data Inverted	INPut1:POLarity NORMal | INVerted
Data Termination	INPut1:TERMination 0 | -2 | 1.3
BER Threshold	SENSe[1]:EYE:THReshold <numeric value>
Return to Results	Not available at this time.
Trigger Output	
Clock Divided by 8	SOURce7:TRIGger[:MODE] DCLock
Pattern	SOURce7:TRIGger[:MODE] PATTern
Pattern Sync	
Sync Type:	
Automatic Sync	SENSe[1]:SYNChronizat | 1 | ON
Manual	SENSe[1]:SYNChronizat | 0 | OFF
Sync Now	SENSe[1]:SYNChronizat ONCE
Sync Threshold	SENSe[1]:SYNChronizat:THReshold <numeric value>
Accumulation Setup	
Start/Stop Accumulation	SENSe[1]:GATE[:STATe] 0 | 1 | OFF | ON
Activation Mode	
Manual	SENSe[1]:GATE:MODE MANual

Table 4-2. Front Panel Function to Remote Command for the Agilent 86130A

Front Panel Function	**Remote Command**
Repeat	SENSe[1]:GATE:MODE REPetitive
Single	SENSe[1]:GATE:MODE SINGle
Measurement Log	
Prompt for File Name	Not available at this time.
File Name	SENSe[1]:LOGGing:FILename <string>
File Name Prefix	Not available at this time.
No Logging	SENSe[1]:LOGGing 0 | 1 | OFF | ON
Period	
Bits	SENSe[1]:GATE:PERiod:BITS <numeric value>
Number of Errors	SENSe[1]:GATE:PERiod:ERRors <numeric value>
Time	SENSe[1]:GATE:PERiod[:TIME] <numeric value>
Pattern	
Pattern Select Form	
2^n-1	[SOURce[1]:]PATTern[:SELect] PRBS<n> SENSe[1]:PATTern[:SELect] PRBS<n> (not required if SENSe[1]:PATtern:TRACk 1 | ON)
2^n	[SOURce[1]:]PATTern[:SELect] PRBN<n> SENSe[1]:PATTern[:SELect] PRBN<n> (not required if SENSe[1]:PATtern:TRACk 1 | ON)
Mark Density	[SOURce[1]:]PATTern[:SELect] MDENsity<n> SENSe[1]:PATTern[:SELect] MDENsity<n> (not required if SENSe[1]:PATtern:TRACk 1 | ON)
User Pattern	[SOURce[1]:]PATTern[:SELect] FILENAME,<character data> SENSe[1]:PATTern[:SELect] FILENAME,<character data> (not required if SENSe[1]:PATtern:TRACk 1 | ON) [SOURce[1]:]PATTern:UPATtern<n>:USE STRaight|APATtern SENSe[1]:PATTern:UPATtern<n>:USE STRaight | APATtern (not required if SENSe[1]:PATtern:TRACk 1 | ON)

Table 4-2. Front Panel Function to Remote Command for the Agilent 86130A

Front Panel Function	**Remote Command**
PG Trigger	
Alt Pattern Trigger Level	SOURce3:TRIGger:APATtern<n> ABCHange
Alt Pattern Trigger Pulse	SOURce3:TRIGger:APATtern<n> SOPattern
Clock Divided by 8	SOURce3:TRIGger:[:MODE] DCLock
Pat Trigger Position	SOURce3:TRIGger:[:MODE] PATTern
N-bit Trigger Pattern	SOURce3:TRIGger:PRBS<n> <0 | 1 | OFF | ON>{,<0 | 1 | OFF | ON>}
Bit Position	SOURce3:TRIGger:PRBN<n> <numeric value> SOURce3:TRIGger:MDENsity<n> <numeric value> SOURce3:TRIGger:UPATtern<n> <numeric value>
Edit Pattern on File	[SOURce[1]:]PATTern:UPATtern<n>[:LENGth] <numeric value> [SOURce[1]:]PATTern:UPATtern<n>:LABel <string> [SOURce[1]:]PATTern:UPATtern<n>:DATA [A|B,] <block data> [SOURce[1]:]PATTern:UPATtern<n>:IDATa [A|B,] <block data>,<start_bit>,<length_in_bits> [SOURce[1]:]PATTern:UPATtern<n>:IDATa [A|B,] <start_bit>,<length_in_bits>,<block data> SENSe[1]:PATTern:UPATtern<n>[:LENGth] <numeric value> SENSe[1]:PATTern:UPATtern<n>:LABel <string> SENSe[1]:PATTern:UPATtern<n>:DATA [A|B,] <block data> SENSe[1]:PATTern:UPATtern<n>:IDATa [A|B,] <start_bit>,<length_in_bits>,<block_data> For your convenience, you may also use commands that specify the entire path. For more information, refer to the UFILe commands starting with SENSe[1]:PATTern:UFILe:NAME? 4-59 and [SOURce[1]:]PATTern:UFILe:NAME? 4-76.
ED Tracks PG	SENSe[1]:PATTern:TRACk 0 | 1 | OFF | ON
PG Setup	
Alternate Pattern Setup	
Alternate AB	[SOURce[1]:]PATTern:APCHange:SOURce INTernal [SOURce[1]:]PATTern:APCHange:SELect ABHAlf

Table 4-2. Front Panel Function to Remote Command for the Agilent 86130A

Front Panel Function	Remote Command
Aux In	[SOURce[1]:]PATTern:APCHange:SOURce EXTernal [SOURce[1]:]PATTern:APCHange:MODE LLEVel | REDGe
Continuous A	[SOURce[1]:]PATTern:APCHange:SOURce INTernal [SOURce[1]:]PATTern:APCHange:SELect AHALf
Continuous B	[SOURce[1]:]PATTern:APCHange:SOURce INTernal [SOURce[1]:]PATTern:APCHange:SELect BHALf
Single Shot B	[SOURce[1]:]PATTern:APCHange:SOURce INTernal [SOURce[1]:]PATTern:APCHange:MODE ONEShot
PG Output Setup	
Clock Logic Level	SOURce2:VOLTage :ECL SOURce2:VOLTage:LLEVel ECL | LVPECL | SCFL | LVTTL| CUSTOM
Clock Output Off	OUTPut2:STATe 0 | OFF
Clock Output On	OUTPut2:STATe 1 | ON
Clock Termination	OUTPut2:TERMination 0 | -2 | 1.3
Clock Tracking Off	SOURce11:TRACk 0 | OFF
Clock Tracking On	SOURce11:TRACk 1 | ON
Data	
Vhi	[SOURce[1]:]VOLTage[:LEVel][:IMMediate]:HIGH <numeric value>
Vlo	[SOURce[1]:]VOLTage[:LEVel][:IMMediate]:LOW <numeric value>
Vofst	[SOURce[1]:]VOLTage[:LEVel][:IMMediate]:OFFSet <numeric value>
Vamptd	[SOURce[1]:]VOLTage[:LEVel][:IMMediate][:AMPLitude] <numeric value>
Xover	OUTPut[1]:DATA:XOVer <numeric value>
Delay	OUTPut[1]:DELay <numeric value>
Clock	
Vhi	SOURce2:VOLTage[LEVel][:IMMediate]:HIGH <numeric value>

Table 4-2. Front Panel Function to Remote Command for the Agilent 86130A

Front Panel Function	**Remote Command**
Vlo	SOURce2:VOLTage[LEVel][:IMMediate]:LOW <numeric value>
Vofst	SOURce2:VOLTage[:LEVel][:IMMediate]:OFFSet <numeric value>
Vamptd	SOURce2:VOLTage [:LEVel][:IMMediate][:AMPLitude] <numeric value>
$\overline{\text{Data}}$	
Vhi	SOURce10:VOLTage[:LEVel][:IMMediate]:HIGH <numeric value>
Vlo	SOURce10:VOLTage[:LEVel][:IMMediate]:LOW <numeric value>
Vofst	SOURce10:VOLTage[:LEVel][:IMMediate]:OFFSet <numeric value>
Vamptd	SOURce10:VOLTage[:LEVel][:IMMediate][:AMPLitude] <numeric value>
Xover	OUTPut10:DATA:XOVer <numeric value>
$\overline{\text{Clock}}$	
Vhi	SOURce11:VOLTage[:LEVel][:IMMediate]:HIGH <numeric value>
Vlo	SOURce11:VOLTage[:LEVel][:IMMediate]:LOW <numeric value>
Vofst	SOURce11:VOLTage[:LEVel][:IMMediate]:OFFSet <numeric value>
Vamptd	SOURce11:VOLTage[:LEVel][:IMMediate][:AMPLitude] <numeric value>
Data Inverted	OUTPut[1]:POLarity NORMal | INVerted
Data Logic Level	[SOURce[1]:]VOLTage:ECL [SOURce[1]:]VOLTage:LLEVel ECL | LVPECL | SCFL | LVTTL | CUSTOM
Data Output Coupling	OUTPut[1]:COUPling AC | DC
Data Output Off	OUTPut[1][:STATe] 0 | OFF
Data Output On	OUTPut[1][:STATe] 1 | ON
Data Termination	OUTPut[1]:TERMination 0 | -2 | 1.3
Data Tracking On	SOURce10:VOLTage:TRACk 0 | 1 | OFF | ON
Bit Rate Setup	
External Clock Input	SOURce9:OUTPut[:STATe] 0 | OFF
Internal Clock Input	SOURce9:OUTPut[:STATe] 1 | ON

Table 4-2. Front Panel Function to Remote Command for the Agilent 86130A

Front Panel Function	**Remote Command**
Clock Rate	SOURce9:FREQuency <numeric value>
Trigger Output	
Alternate Pattern Trigger Level	SOURce3:TRIGger:APATtern<n> ABCHange
Alternate Pattern Trigger Pulse	SOURce3:TRIGger:APATtern<n> SOPattern
Clock Divided by 8	SOURce3:TRIGger[:MODE] DCLock
Pattern Trigger Position	SOURce3:TRIGger:MODE PATTern
Bit Position	SOURce3:TRIGger:PRBN<n> <numeric values> SOURce3:TRIGger:MDENsity<n><numeric value> SOURce3:TRIGger:UPATtern<n><numeric value>
N-bit Trigger Pattern	SOURce3:TRIGger:PRBS<n> <0|1|OFF|ON>{,<0|1|OFF|ON>}
Error Add Setup	
Error Insertion Mode	
Enter add rate	[SOURce[1]:]PATTern:EADDition:RATE <numeric value>
Choose preset add rate	Not available at this time.
Bits Between Errors	Not available at this time.
External/Internal	[SOURce[1]:]PATTern:EADDition:SOURce EXTernal | FIXed
Off	[SOURce[1]:]PATTern:EADDition ONCE | 0 | 1 | OFF | ON
Utility	
Set Date and Time	Not available at this time.
Windows Explorer	Not available at this time.
Event Viewer	Not available at this time.
Touchscreen Off	Not available at this time.
Change GPIB Address	Not available at this time.

Table 4-2. Front Panel Function to Remote Command for the Agilent 86130A

Front Panel Function	**Remote Command**
Display Utility	Not available at this time.
On Screen Keyboard	Not available at this time.
Software Upgrade	Not available at this time.
Brightness Control	Not available at this time.
Self Test	
PG	
Tray Presence	TEST:EXECute? PGTRay
RAM RW Test	TEST:EXECute? PGRam
FlashROM Content	TEST:EXECute? PGFLash
FPGA Config Test	TEST:EXECute? PGFPga
Calibration Status Test	TEST:EXECute? PGCal
OverTemperature Test	TEST:EXECute? PGOVertemp
ED	
Tray Presence	TEST:EXECute? EDTRay
RAM RW Test	TEST:EXECute? EDRam
FlashROM Content	TEST:EXECute? EDFLash
FPGA Config Test	TEST:EXECute? EDFPga
Calibration Status Test	TEST:EXECute? EDCal
OverTemperature Test	TEST:EXECute? EDOVertemp

Table 4-2. Front Panel Function to Remote Command for the Agilent 86130A

Front Panel Function	**Remote Command**
EELB Presence Test	TEST:EXECute?EDEelb
BitAlyzer	
PCI Card Ready Test	TEST:EXECute? BTPCicard
PC	
Free Memory Space	TEST:EXECute? PCRam
GUI Response Test	TEST:EXECute? PCGuiresp
Free Hard Disk Space	TEST:EXECute? PCHDisk
GPIB Response Test	TEST:EXECute? PCGPibresp
View Results	
Main Results	
Delta Error Ratio (Current Period)	FETCh[:SENSe[1]:]:ERATio[:ALL][:FULL]:DELTa?
Delta Error Count (Current Period)	FETCh[:SENSe[1]:]:ECOUnt[:ALL][:FULL]:DELTa?
Delta Error Ratio (Previous Period)	PFETch[:SENSe[1]:]:ERATio[:ALL][:FULL]:DELTa?
Delta Error Count (Previous Period)	PFETch[:SENSe[1]:]:ECOUnt[:ALL][:FULL]:DELTa?
PG Measured Bit Rate	SENSe6:FREQuency?
ED Measured Bit Rate	SENSe2:FREQuency?
PG Pattern	[SOURce[1]:]PATTern[:SELect]?
ED Pattern	SENSe[1]:PATTern[:SELect]?

Table 4-2. Front Panel Function to Remote Command for the Agilent 86130A

Front Panel Function	**Remote Command**
Accumulated Results	
Current Period	
Bit Count	FETCh:SENSe2:BCOunt?
Error Count	FETCh[:SENSe[1]]:ECOunt[:ALL][:FULL][:TOTal]?
Errored 0's Count	FETCh[:SENSe[1]]:ECOunt:ZASone[:TOTal]?
Errored 1's Count	FETCh[:SENSe[1]]:ECOunt:OASZero[:TOTal]?
Error Ratio	FETCh[:SENSe[1]]:ERATio[:ALL][:FULL][:TOTal]?
Errored 0's Ratio	FETCh[:SENSe[1]]:ERATio:ZASone[:TOTal]?
Errored 1's Ratio	FETCh[:SENSe[1]]:ERATio:OASZero[:TOTal]?
Previous Period	
Bit Count	PFETch:SENSe2:BCOunt?
Error Count	PFETch[:SENSe[1]]:ECOunt[:ALL][:FULL][:TOTal]?
Errored 0's Count	PFETch[:SENSe[1]]:ECOunt:ZASone[:TOTal]?
Errored 1's Count	PFETch[:SENSe[1]]:ECOunt:OASZero[:TOTal]?
Error Ratio	PFETch[:SENSe[1]]:ERATio[:ALL][:FULL]:DELTa?
Errored 0's Ratio	PFETch[:SENSe[1]]:ERATio:ZASone[:TOTal]?
Errored 1's Ratio	PFETch[:SENSe[1]]:ERATio:OASZero[:TOTal]?
Accumulated Parameter s	
Accumulation Mode	
Manual	SENSe1:GATE:MODE: MANual
Repetitive	SENSe1:GATE:MODE REPetitive
Single	SENSe1:GATE:MODE SINGle
Accumulation Period	
Bits	SENSe1:GATE:PERiod:BITS <numeric value>

Table 4-2. Front Panel Function to Remote Command for the Agilent 86130A

Front Panel Function	Remote Command
Errors	SENSe1:GATE:PERiod:ERRors <numeric value>
Time	SENSe1:GATE:PERiod[:TIME] <numeric value>
Elapsed Time (Current Period)	FETCh[:SENSe[1]]:GATE:ELAPsed?
Elapsed Time (Previous Period)	PFETch[:SENSe[1]]:GATE:ELAPsed?
Interval Results (Current Period)	
Error Free Seconds	FETCh[:SENSe[1]]:EFINterval:SEConds?
Error Free Deciseconds	FETCh[:SENSe[1]]:EFINterval:DSEConds?
Errored Seconds	FETCh[:SENSe[1]]:EINTerval:SEConds?
Errored Deciseconds	FETCh[:SENSe[1]]:EINTerval:DSEConds?
Power Loss Seconds	FETCh[:SENSe[1]]:LOSS:POWer?
Sync Loss Seconds	FETCh[:SENSe[1]]:LOSS:SYNChronizat?
Interval Results (Previous Period)	
Error Free Seconds	PFETch[:SENSe[1]]:EFINterval:SEConds?
Error Free Deciseconds	PFETch[:SENSe[1]]:EFINterval:DSEConds?
Errored Seconds	PFETch[:SENSe[1]]:EINTerval: SEConds?
Errored Deciseconds	PFETch[:SENSe[1]]:EINTerval:DSEConds?
Power Loss Seconds	PFETch[:SENSe[1]]:LOSS:POWer?
Sync Loss Seconds	PFETch[:SENSe[1]]:LOSS:SYNChronizat?
Eye Results	
Delta Error Ratio (Current Period)	FETCh[:SENSe[1]]:ERATio[:ALL][:FULL]:DELta?

Table 4-2. Front Panel Function to Remote Command for the Agilent 86130A

Front Panel Function	Remote Command
Delta Error Ratio (Previous Period)	PFETch[:SENSe[1]]:ERATio[:ALL][:FULL]:DELta?
Eye Data Input Delay	INPut1:DELay <numeric value>
Eye Edge Threshold	SENSe[1]:EYE:THReshold <numeric value>
Eye Height	SENSe[1]:EYE:HEIGht?
Eye Voltage Center	SENSe[1]:EYE:ACENter ONCE | 0 | 1 | OFF | ON
Eye Width	SENSe[1]:EYE:WIDTh?

Commands Not Supported

This list contains the remote commands that are not supported by the 86130A instrument. Most of the commands are not implemented because of the following reasons:

- The instrument does not provide the required functionality.
- The operation does not apply.
- The command(s) have been obsoleted.

There are some commands in this list that will be implemented in future software releases. These commands include *PSC, *PSC?, *TST, and *OPT commands.

*PSC
*PSC?
*OPT?
DISPLay:REPort PREVious|CURRent
DISPLay:UPAGE[:DEFine] <parameter>
DISPLay:UPAGE:CLEar
DISPLay:WINDow
DISPLay:WINDow[:RESults] <paramter>
DISPLay:WINDow:CONFig <parameter>
FETCh[:SENSe[1]]:LTEXt?
MMEMory:CPDisk <store number>
MMEMory:DELete <file name>
MMEMory:ICPDisk <dest. store number>, AHALf|BHALf, <start_bit>, <end_bit>
MMEMory:INITialize
MMEMory:MPResent?
OUTPut4:COUPling AC|DC
OUTPut4:COUPling?
OUTPut4:TERMination -2|0
OUTPut4:TERMination?
OUTPut5:COUPling AC|DC
OUTPut5:COUPling?
OUTPut5:TERMination -2|0
OUTPut5:TERMination?
OUTPut8:PLENgth?

```
OUTPut8:PLENgth RZ|STRetched
SENSe[1]:BLOCk 0|1|OFF|ON
SENSe[1]:BLOCk?
SENSe[1]:BLOCk:BSTart <numeric value>
SENSe[1]:BLOCk:BSTart?
SENSe[1]:BLOCk:BLENgth <numeric value>
SENSe[1]:BLOCk:BLENgth?
SENSe[1]:ELOCation?
SENSe[1]:ELOCation:BEADdress <numeric value>
SENSe[1]:ELOCation:BEADdress?
SENSe[1]:ELOCation ONCE
SENSe[1]:LOGGing:ALARms?
SENSe[1]:LOGGing:ALARms 0|1|OFF|ON
SENSe[1]:LOGGing:BRATe?
SENSe[1]:LOGGing:BRATe <numeric value>
SENSe[1]:LOGGing:DURing[:EVENt]?
SENSe[1]:LOGGing:DURing[:EVENt] NEVer|ESECond|ERGThrshld
SENSe[1]:LOGGing:END[:EVENt]?
SENSe[1]:LOGGing:END[:EVENt] NEVer|ALWays|ESECond|ERGThrshld
SENSe[1]:LOGGing:END:REPort?
SENSe[1]:LOGGing:END:REPort FULL|UREP
SENSe[1]:LOGGing:PORT?
SENSe[1]:LOGGing:PORT RS232|ECONtroller
SENSe[1]:LOGGing:SQUelch?
SENSe[1]:LOGGing:SQUelch 0|1|OFF|ON
SENSe[1]:LOGGing:THReshold?
SENSe[1]:LOGGing:THReshold <numeric parm>
SENSe[1]:PATTern:ZSUBstitut[:ZRUN]?
SENSe[1]:PATTern:ZSUBstitut[:ZRUN] <numeric value>
SENSe[1]:SEEK ONCE|0|1|OFF|ON
SENSe[1]:SEEK?
SENSe[1]:SEEK:PATTern 0|1|OFF|ON
SENSe[1]:SEEK:PATTern?
SENSe2:BANDswitch?
SENSe6:FREQuency:BANDswitch?
[SOURce[1]:]PATTern:AWORd:DATA<n>
[SOURce[1]:]PATTern:AWORd:DATA<n>?
[SOURce[1]:]PATTern:UPATtern<n>:LMODified?
[SOURce[1]:]PATTern:ZSUBstitut[:ZRUN] <numeric value>
[SOURce[1]:]PATTern:ZSUBstitut[:ZRUN]?
SOURce3:TRIGger:CTDRatio <numeric value>
SOURce3:TRIGger:CTDRatio?
```

SOURce3:TRIGger:DCDRatio <numeric value>
SOURce3:TRIGger:DCDRatio?
SOURce4:VOLTage:ECL
SOURce4:VOLTage[:LEVel][:IMMediate][:AMPLitude] <numeric value>
SOURce4:VOLTage[:LEVel][:IMMediate][:AMPLitude]?
SOURce4:VOLTage[:LEVel][:IMMediate]:HIGH <numeric value>
SOURce4:VOLTage[:LEVel][:IMMediate]:HIGH?
SOURce5:VOLTage:ECL
SOURce5:VOLTage[:LEVel][:IMMediate][:AMPLitude] <numeric value>
SOURce5:VOLTage[:LEVel][:IMMediate][:AMPLitude]?
SOURce5:VOLTage[:LEVel][:IMMediate]:HIGH <numeric value>
SOURce5:VOLTage[:LEVel][:IMMediate]:HIGH?
SOURce9:FREQuency[:CW|:FIXed]:STEP[:INCRement] <numeric value>
SOURce9:FREQuency[:CW|:FIXed]:STEP[:INCRement]?
SOURce9:IDN?
SOURce9:POWer[:LEVel][:IMMediate][:AMPLitude] <numeric value>
SOURce9:POWer[:LEVel][:IMMediate][:AMPLitude]?
STATus:FAILure:EVENt?
SYSTem:BEEPer[:IMMediate] [<freq>[,<time>[,<vol>]]]
SYSTem:DATE
SYSTem:DATE?
SYSTem:FREVision[:CPRocessor][:APPLication]?
SYSTem:FREVision[:CPRocessor]:BOOT?
SYSTem:FREVision[:MPRocessor][:APPLication]?
SYSTem:FREVision[:MPRocessor]:BOOT?
SYSTem:FUPDate CAPPlication|MAPPlication
SYSTem:KLOCk
SYSTem:KLOCk?
SYSTem:PRESet
SYSTem:TIME
SYSTem:TIME?

5

Error Analysis Plug-in Commands

Introduction

The 86130A Error Analysis commands are a "plug-in" subsystem of the instrument GPIB command structure. These commands must be queried from, or passed-through to the system level with PLUGin subsystem commands. Refer to “PLUGin Subsystem” on page 4-38 for more information.

The :EAPLUGIN Error Analysis plug-in GPIB node

All Error Analysis commands or queries must be preceded by the :EAPLUGIN GPIB node. This is to insure the commands or queries are sent to the correct GPIB plug-in.

An example of using :EAPLUGIN when sending a command and query to GPRoperties MBLength (minimum burst length) is shown below.

```
:PLUGin:PUT:STRing ":EAPLUGIN:GPRoperties:MBLength 2"
:PLUGin:GET:STRing? ":EAPLUGIN:GPRoperties:MBLength?”
```

Further examples can be found following each command listed in this chapter.

NOTE For additional information on Error Analysis commands, refer to the online Help included with the instrument and the documentation-set CD-ROM.

2DEMap Subsystem

The 2DEMap commands control the 2D Error Map analyzer.

The 2DEMap Subsystem

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
2DEMap			
:ASCRoll	0 | 1 | OFF | ON		Enable auto scroll
:ASCRoll?		Boolean	
:BLENgth	<integer>		Block length [1,2^31-1]
:BLENGth?		value	
:CEXTents	<numeric>, <numeric>, <numeric>, <numeric>		Chart extents
:CEXTents?		Xmin, Xmax, Ymin, Ymax	
:ENABle	0 | 1 | OFF | ON		
:ENABle?		Boolean	Enable/disable 2D Error Map analyzer
:GTBase	0 | 1 | OFF | ON		Global timebase
:GTBase?		Boolean	
:HBURsts	0 | 1 | OFF | ON		Highlight big bursts

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
:HBURsts?		Boolean	
:MODe	QUANtity | SECond | MARKer		Segment mode
:MODe?			
:SEGMents			Number of segments
:STATus?			Error map status
:STIMe	<integer>		Segment time [1,3600]
:STIMe?		value	
:UCURsor	0 | 1 | OFF | ON		Use cursors (both X and Y)
:UCURsor?		Boolean	
:YCPosition	<integer>		Y-cursor position
:YCPosition?		value	

2DEMap:ASCRoll 0 | 1 | OFF | ON
2DEMap:ASCRoll?

This command enables/disables auto-scroll of the Error Map chart as data is rendered. It returns the status using the query form.

Example

:PLUG:PUT:STR "EAPLUGIN:2DEM:ASCR ON"

:PLUG:GET:STR? "EAPLUGIN:2DEM:ASCR?"

2DEMap:BLENgth <integer>
2DEMap:BLENgth?

This command sets and retrieves and block length as a specific number of bits. A block size could represent a packet of information, a disk sector, or a word on a data bus. Often these data blocks have to do with internal processing boundaries or smallest divisible unit.

Range is from one to (2^31)-1.

Example :PLUG:PUT:STR "EAPLUGIN:2DEM:BLEN 10"

:PLUG:GET:STR? "EAPLUGIN:2DEM:BLEN?"

2DEMap:CEXTents <numeric>, <numeric>, <numeric>, <numeric>
2DEMap:CEXTents?

This command sets or retrieves the boundaries of the view, from the minimum to maximum X-values, and minimum to maximum Y-values. The query returns: <Xmin>, <Xmax>, <Ymin>, <Ymax>

Example :PLUGin:PUT:STRing "EAPLUGIN:2DEMap:CEXTents 1, 1000, 1, 1000"

:PLUGin:GET:STRing? "EAPLUGIN:2DEMap:CEXTents?"

2DEMap:ENABle 0 | 1 | OFF | ON
2DEMap:ENABle?

This command enables or disables the 2D Error Map analyzer. It is equivalent to the 2D Error Map checkbox in the properties dialog.

Example :PLUG:PUT:STR "EAPLUGIN:2DEMap:ENABle ON"

:PLUG:GET:STR? "EAPLUGIN:2DEM:ENAB?"

2DEMap:GTBase 0 | 1 | OFF | ON
2DEMap:GTBase?

This command enables/disables the Global Timebase. Enabling Global Timebase shows the bit rate values in Time along the X-axis. Disabling it shows the bit rate values in Bits along the X-axis.

Time is calculated based on a data rate of bits per second. The data rate can be set or queried using the GPROPerties:CTTRate command.

The GTBase command is equivalent to the Global Timebase checkbox in the properties dialog.

Example

```
:PLUG:PUT:STR "EAPLUGIN:2DEM:GTBase ON"
:PLUG:GET:STR? "EAPLUGIN:2DEM:GTBase?"
```

2DEMap:HBURsts 0 | 1 | OFF | ON
2DEMap:HBURsts?

This command enables/disables the Highlight Bursts checkbox, and retrieves the status using the query form. Enabling causes burst errors to be highlighted on the Error Map. Disabling shows bursts with no highlighting.

Example

```
:PLUG:PUT:STR "EAPLUGIN:2DEM:HBUR ON"
:PLUG:GET:STR? "EAPLUGIN:2DEM:HBUR?"
```

2DEMap:MODe QUANtity | SECond | MARKer
2DEMap:MODe?

This command sets and retrieves the segment mode. QUANtity = a specific number of bits (set using the BLENgth command, below); SECond = a specific number of seconds (set using the STIMe command, below); MARKer = indicates the segment boundary by a marker signal.

Example

```
:PLUG:PUT:STR "EAPLUGIN:2DEM:MOD QUANtity"
:PLUG:GET:STR? "EAPLUGIN:2DEM:MOD?"
```

2DEMap:SEGMents?

This query-only command returns the number of segments in the analysis session. Segments are divisions of the error data stream based upon a specific quantity, a specific number of seconds, or a Marker input signal (see the MODe command, below).

Example

```
:PLUG:GET:STR? "EAPLUGIN:2DEM:SEGments?"
```

2DEMap:STATus?

This query-only command returns the status of the Error Map as “Rendering” or “Complete.”

Example

```
:PLUG:GET:STR? "EAPLUGIN:2DEM:STATus?"
```

2DEMap:STIMe <integer>
2DEMap:STIMe?

This command sets and retrieves the segment time as a specific number of seconds. Time is calculated using the Global Timebase, based on a data rate of bits per second. The data rate can be set or queried using the GPROPerties:CTTRate command. The range is from one to 3,600.

Example

```
:PLUG:PUT:STR "EAPLUGIN:2DEM:STIM 1"
:PLUG:GET:STR? "EAPLUGIN:2DEM:STIM?"
```

2DEMap:UCURsor 0 | 1 | OFF | ON
2DEMap:UCURsor?

This command enables/disables both Cursor X1 and Cursor Y1. It performs the same function as the Use Cursors checkbox in the properties dialog.

Example

```
:PLUG:PUT:STR "EAPLUGIN:2DEM:UCUR ON"
:PLUG:GET:STR? "EAPLUGIN:2DEM:UCUR?"
```

2DEMap:YCPosition <value1>
2DEMap:YCPosition?

This command sets or retrieves the position of Cursor Y1 in the view. The response will be: value1.

Example

```
:PLUG:PUT:STR "EAPLUGIN:2DEM:YCPosition 5"
:PLUG:GET:STR? "EAPLUGIN:2DEM:YCPosition?"
```

ACONtrol Subsystem

The ACONtrol commands return statuses related to the Analyzer Control view.

The ACONtrol Subsystem

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
ACONtrol			
:ERATe?		numeric	Event rate
:MRATe?		numeric	Marker rate
:SEVents?		numeric	Squelch events

ACONtrol:ERATe?

This query-only command returns the number of error events per second being analyzed by the BitAlyzer. An error event can define the error location for up to 32 neighboring errors. Error events are also used to define the location of user Marker, Gating and other non-bit error information. This is only valid during live error analysis. After an acquisition session is ended, this value reflects the last updated information.

Example :PLUGin:GET:STRing? "EAPLUGIN:ACONtrol:ERate?"

ACONtrol:MRATe?

This query-only command returns the measured frequency of the TTL-level input Marker signal as received by the error analyzer. This is only valid during live error analysis. After an acquisition session is ended, this value reflects the last session update.

Example :PLUGin:GET:STRing? "EAPLUGIN:ACONtrol:MRate?"

ACONtrol:SEVents?

This query-only command returns the number of Squelch events. Squelch may occur as high error event rates lead to periods when the analysis software is not able to keep up with the number of errors occurring. This is only valid during live error analysis. After an acquisition session is ended, this value reflects the last session update.

Example :PLUGin:GET:STRing? "EAPLUGIN:ACONtrol:SEVents?"

ASTatus: System Query Command

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
ASTatus?		DISABLE | LIVE | PLAY | RECORD | STOP	Symbol Mode

ASTatus?

This query-only command returns the status of the whole Error Analysis module. The result is one of the following: DISABLE, LIVE, PLAY, RECORD, or STOP. This status matches that shown on the Status Bar in the GUI.

Example :PLUGin:GET:STRing? "EAPLUGIN:ASTatus?"

BERRors Subsystem

The BERRors commands control the Block Errors analyzer.

The BERRors Subsystem

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
BERRors			
:AREA?			Area defined by X-cursors
:BCOunt?			Bin Count
:BLMode	GATing | MARKer | QUANtity		Set mode for Block Length boundary
:BLMode?		GATing | MARKer | QUANtity	
:BLQuantity	<numeric>		Block Size
:BLQuantity?		numeric	
:BRESolution?			Query bin resolution
:CEXTents	<Xmin>, <Xmax>, <Ymin>, <Ymax>		Chart Extents
:CEXTents?		Xmin, Xmax, Ymin, Ymax	
:CRANge	<value1>, <value2>		Chart Range
:CRANge?		value1, value2	

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
:DATA?		Binary	All data in bins
:ENABle	0 | 1 | OFF | ON		Enable/disable Block Errors analyzer
:ENABle?		Boolean	
:ESFClose			Close Error Signature File
:ESFOpen	<string>		Open Error Signature File
:ESFSave	<string>		Save Error Signature
:LSCale	0 | 1 | OFF | ON		Log Scale
:LSCale?		Boolean	
:XCData?		value1, value2	Bin Data
:XCENable	0 | 1 | OFF | ON		X-cursor visibility
:XCENable?		Boolean	
:XCNPeak	<index>		Set X1 cursor to n’th highest bin
:YCENable	0 | 1 | OFF | ON		Y-cursor visibility
:YCENable?		Boolean	
:YCPosition	<value1>		Y-cursor position
:YCPosition?		value1	

BERRors:AREA?

This query-only command returns the area defined by Cursor X1 and Cursor X2. The response is the same as the “Cursor Area” displayed under the view.

Example

```
:PLUGin:GET:STRing? "EAPLUGIN:BERRors:AREA?"
```

BERRors:BCOunt?

This query-only command gets the bin count stored in the server. There is no counterpart in the GUI.

Example

```
:PLUGin:GET:STRing? "EAPLUGIN:BERRors:BCOunt?"
```

BERRors:BLMode GATing | MARKer | QUANtity
BERRors:BLMode?

This command sets or retrieves the definition of the boundary for a Block Length. GATing sets the boundary to Gating Mode, MARKer sets Marker Mode, and QUANtity sets the block size to Specified Quantity Mode, a fixed number of bits or symbols, which must be set using the BLQuantity command, below.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:BERRors:BLMode QUANtity"
:PLUGin:GET:STRing? "EAPLUGIN:BERRors:BLMode?"
```

BERRors:BLQuantity <numeric>
BERRors:BLQuantity?

This command sets or retrieves the Block Size. A block size could represent a packet of information, a disk sector, or a word on a data bus. Often these data blocks have to do with internal processing boundaries or smallest divisible unit.

Input should not be less than two. If the value input is less than two, then the Block Size is set to two.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:BERRors:BLQuantity 16"
:PLUGin:GET:STRing? "EAPLUGIN:BERRors:BLQuantity?"
```

BERRors:BRESolution?

This query-only command returns the bin resolution.

Bin resolution is determined by the Chart Range values (see CRANge). The “From” value is inclusive and the “To” value is exclusive. For example, the range can be specified as [0,1000], which establishes 1000 bins, from Bin 0 to Bin 999. The bin resolution is 1. Bin 0 contains data [–inf, 1], and Bin 999 contains [999, +inf].

When the range is greater than or equal to 1001, the bin resolution will be 2 or more, as necessary to include all data within the available bins.

Example :PLUGin:GET:STRing? "EAPLUGIN:BERRors:BRESolution?"

BERRors:CEXTents <Xmin>, <Xmax>, <Ymin>, <Ymax>
BERRors:CEXTents?

This command sets or retrieves the boundaries of the view, from the minimum to maximum X-values, and minimum to maximum Y-values.

If the Chart Range is changed (using the CRANge command, below), a “Fit View” is performed in an attempt to fit the Chart Extents of the view to the defined interval. Hence, the Chart Extent values queried after changing the Chart Range will differ from those previously set. Use the query command CEXTents? to discover the new chart extents.

The same range check as is used by the GUI applies here.

Example :PLUGin:PUT:STRing "EAPLUGIN:BERRors:CEXTents 1, 1000, 1, 1000"
:PLUGin:GET:STRing? "EAPLUGIN:BERRors:CEXTents?"

BERRors:CRANge <value1>, <value2>
BERRors:CRANge?

This command sets or retrieves the settings of “Chart Range” in the properties dialog. The Chart Range defines the range over which you want to collect histogram data, which may differ significantly from the data displayed in the view (limited by the Chart Extents).

When the Chart Range is changed, a “Fit View” is performed in an attempt to fit the Chart Extents of the view to the defined interval. Hence, the Chart Extent values queried after changing the Chart Range will differ from those previously set. Use the query command CEXTents? to discover the new chart extents.

Example :PLUGin:PUT:STRing "EAPLUGIN:BERRors:CRANge 0, 100"

:PLUGin:GET:STRing? "EAPLUGIN:BERRors:CRANge?"

BERRors:DATA?

Query all the data in all the bins. This gives the user all the data in the server's format, not that of the GUI. It is provided for the user to analyze using information from another source, such as the bin count.

Binary query.

Example

:PLUGin:GET:BINary? "EAPLUGIN:BERRors:DATA?"

BERRors:ENABle 0 | 1 | OFF | ON
BERRors:ENABle?

This command enables/disables the Block Errors analyzer. It is equivalent to the Block Errors checkbox in the properties dialog.

Example

:PLUGin:PUT:STRing "EAPLUGIN:BERRors:ENABle ON"

:PLUGin:GET:STRing? "EAPLUGIN:BERRors:ENABle?"

BERRors:ESFClose

This command will close an open Error Signature file. It works just like the Close button in the Error Signature Manager.

Example

:PLUGin:PUT:STRing "EAPLUGIN:BERRors:ESFClose"

BERRors:ESFOpen <filename>

This command opens the Error Signature file named in <filename>. It works just like the Open button in the Error Signature Manager.

Example

:PLUGin:PUT:STRing "EAPLUGIN:FDIRectories:ESFile ""C:\ELA_GPIB\BlockErrorAnalyzer"""

:PLUGin:GET:STRing? "EAPLUGIN:FDIRectories:ESFile?"

:PLUGin:PUT:STRing "EAPLUGIN:BERRors:ESFOpen ""Block Errors Signature.BEH.csv"""

BERRors:ESFSave <string>

This command saves the Error Signature to the file named in the <string>. It works just like the Save button in the Error Signature Manager.

An Error Signature is an ASCII file containing a list of comma-separated values representing error performance characteristics of a particular failure mode. A saved Error Signature can be compared against new results to identify problem types that have occurred before.

Example

:PLUGin:PUT:STRing "EAPLUGIN:FDIRectories:ESFile ""C:\ELA_GPIB\BlockErrorAnalyzer"""

:PLUGin:GET:STRing? "EAPLUGIN:FDIRectories:ESFile?"

:PLUGin:PUT:STRing "EAPLUGIN:BERRors:ESFSave ""Block Errors Signature.BEH.csv"""

BERRors:LSCale 0 | 1 | OFF | ON
BERRors:LSCale?

This command enables/disables use of the Log Scale. Enabling it shows the Y-axis of the histogram as a logarithmic scale. Disabling it shows the Y-axis as a linear scale. This command works like the Log Scale checkbox in the properties dialog.

Example

:PLUGin:PUT:STRing "EAPLUGIN:BERRors:LSCale ON"

:PLUGin:GET:STRing? "EAPLUGIN:BERRors:LSCale?"

BERRors:XCData?

This query-only command gets the data at Cursor X1 and Cursor X2 (smart cursor in the view). The response will be: value1, value2.

Example

:PLUGin:PUT:STRing "EAPLUGIN:BERRors:CRANge 0, 100"

:PLUGin:GET:STRing? "EAPLUGIN:BERRors:CRANge?"

:PLUGin:PUT:STRing "EAPLUGIN:BERRors:XCENable ON"

:PLUGin:GET:STRing? "BERRors:XCData?"

BERRors:XCENable 0 | 1 | OFF | ON
BERRors:XCENable?

This command sets or retrieves the visibility of the X-cursors. It performs the same function as the X-Axis Cursors checkbox in the properties dialog.

Example

:PLUGin:PUT:STRing "EAPLUGIN:BERRors:XCENable ON"

:PLUGin:GET:STRing? "EAPLUGIN:BERRors:XCENable?"

BERRors:XCNPeak <index>

This command sets the X1 cursor to the n'th highest bin.

The last data bin contains data from values outside the range. If this command returns the last bin (and there is data outside the range), it does not mean that the last data bin is actually the greatest.

Example

:PLUGin:PUT:STRing "EAPLUGIN:BERRors:CRANge 1, 100"

:PLUGin:GET:STRing? "EAPLUGIN:BERRors:CRANge?"

:PLUGin:PUT:STRing "EAPLUGIN:BERRors:XCNPeak 1"

:PLUGin:GET:STRing? "EAPLUGIN:BERRors:XCData?"

BERRors:YCENable0 | 1 | OFF | ON
BERRors:YCENable?

This command sets or retrieves the visibility of Cursor Y1. It performs the same function as the Y-Axis Cursor checkbox in the properties dialog.

Example

:PLUGin:PUT:STRing "EAPLUGIN:BERRors:YCENable ON"

:PLUGin:GET:STRing? "EAPLUGIN:BERRors:YCENable?"

BERRors:YCPosition <value1>, <value2>
BERRors:YCPosition?

This command sets or retrieves the position of Cursor Y1 in the view. The response will be: value1.

Example

:PLUGin:PUT:STRing "EAPLUGIN:BERRors:YCPosition 10"

:PLUGin:GET:STRing? "EAPLUGIN:BERRors:YCPosition?"

BLENgth Subsystem

The BLENgth commands control the Burst Length analyzer.

The BLENgth Subsystem

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
BLENgth			
:AREA?			Area defined by X-cursors
:BCOunt?			Bin Count
:BRESolution?			Query bin resolution
:CEXTents	<Xmin>, <Xmax>, <Ymin>, <Ymax>		Chart Extents
:CEXTents?		Xmin, Xmax, Ymin, Ymax	
:CRANge	<value1>, <value2>		Chart Range
:CRANge?		value1, value2	
:DATA?		Binary	All data in bins
:ENABle	0 | 1 | OFF | ON		Enable/disable Burst Length Analyzer
:ENABle?		0 | 1	
:ESFClose			Close Error Signature File

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
:ESFOpen	<string>		Open Error Signature File
:ESFSave	<string>		Save Error Signature
:GTBase	0 | 1 | OFF | ON		Global Timebase
:GTBase?		0 | 1	
:LSCale	0 | 1 | OFF | ON		Log Scale
:LSCale?		0 | 1	
:XCData?		value1, value2	Bin Data
:XCENable	0 | 1 | OFF | ON		X-cursor visibility
:XCENable?		0 | 1	
:XCNPeak	<index>		Set X1 cursor to n’th highest bin
:YCENable	0 | 1 | OFF | ON		Y-cursor visibility
:YCENable?		0 | 1	
:YCPosition	<value1>, <value2>		Y-cursor position
:YCPosition?		value1	

BLENgth:AREA?

This query-only command returns the area defined by the two X-cursors (Cursor X1 and Cursor X2). The response is the same as the “Cursor Area” displayed under the view.

Example :PLUGin:GET:STRing? "EAPLUGIN:BLENgth:AREA?"

BLENgth:BCOunt?

This query-only command gets the bin count stored in the server. There is no counterpart in the GUI.

Example :PLUGin:GET:STRing? "EAPLUGIN:BLENgth:BCOunt?"

BLENgth:BRESolution?

This query-only command returns the bin resolution.

Bin resolution is determined by the Chart Range values (see CRANge). The “From” value is inclusive and the “To” value is exclusive. For example, the range can be specified as [0,1000], which establishes 1000 bins, from Bin 0 to Bin 999. The bin resolution is 1. Bin 0 contains data [–inf, 1], and Bin 999 contains [999, +inf].

When the range is greater than or equal to 1001, the bin resolution will be 2 or more, as necessary to include all data within the available bins.

Example :PLUGin:GET:STRing? "EAPLUGIN:BLENgth:BRESolution?"

BLENgth:CEXTents <Xmin>, <Xmax>, <Ymin>, <Ymax> BLENgth:CEXTents?

This command sets or retrieves the boundaries of the view, from the minimum to maximum X-values, and minimum to maximum Y-values.

If the Chart Range is changed (using the CRANge command, below), a “Fit View” is performed in an attempt to fit the Chart Extents of the view to the defined interval. Hence, the Chart Extent values queried after changing the Chart Range will differ from those previously set. Use the query command CEXTents? to discover the new chart extents.

The same range check as is used by the GUI applies here.

Example :PLUGin:PUT:STRing "EAPLUGIN:BLENgth:CEXTents 1, 100, 1, 1000"

:PLUGin:GET:STRing? "EAPLUGIN:BLENgth:CEXTents?"

BLENgth:CRANge <value1>, <value2>
BLENgth:CRANge?

This command sets or retrieves the settings of “Chart Range” in the properties dialog. The Chart Range defines the range over which you want to collect histogram data, which may differ significantly from the data displayed in the view (limited by the Chart Extents).

When the Chart Range is changed, a “Fit View” is performed in an attempt to fit the Chart Extents of the view to the defined interval. Hence, the Chart Extent values queried after changing the Chart Range will differ from those previously set. Use the query command CEXTents? to discover the new chart extents.

Example

:PLUGin:PUT:STRing "EAPLUGIN:BLENgth:CRANge 1, 100"

:PLUGin:GET:STRing? "EAPLUGIN:BLENgth:CRANge?"

BLENgth:DATA?

Query all the data in all the bins. This gives the user all the data in the server’s format, not that of the GUI. It is provided for the user to analyze using information from another source, such as the bin count.

Binary query.

Example

:PLUGin:GET:BINary? "EAPLUGIN:BLENgth:DATA?"

BLENgth:ENABle 0 | 1 | OFF | ON
BLENgth:ENABle?

This command enables/disables the Burst Length analyzer. It is equivalent to the Burst Length checkbox in the properties dialog.

Example

:PLUGin:PUT:STRing ":EAPLUGIN:BLENgth:ENABle ON"

:PLUGin:GET:STRing? ":EAPLUGIN:BLENgth:ENABle?"

BLENgth:ESFClose

This command will close an open Error Signature file. It works just like the Close button in the Error Signature Manager.

Example

:PLUGin:PUT:STRing "EAPLUGIN:BLENgth:ESFClose"

BLENgth:ESFOpen <string>

This command opens the Error Signature file named in <string>. It works just like the Open button in the Error Signature Manager.

Example

:PLUGin:PUT:STRing "EAPLUGIN:FDIRectories:ESFile ""C:\ELA_GPIB\BurstLengthAnalyzer"""

:PLUGin:GET:STRing? "EAPLUGIN:FDIRectories:ESFile?"

:PLUGin:PUT:STRing "EAPLUGIN:BLENgth:ESFOpen ""Burst Length Signature.BLH.csv"""

BLENgth:ESFSave <string>

This command saves the Error Signature to the file named in the <string>. It works just like the Save button in the Error Signature Manager.

An Error Signature is an ASCII file containing a list of comma-separated values representing error performance characteristics of a particular failure mode. A saved Error Signature can be compared against new results to identify problem types that have occurred before.

Example

:PLUGin:PUT:STRing "EAPLUGIN:FDIRectories:ESFile ""C:\ELA_GPIB\BurstLengthAnalyzer"""

:PLUGin:GET:STRing? "EAPLUGIN:FDIRectories:ESFile?"

:PLUGin:PUT:STRing "EAPLUGIN:BLENgth:ESFSave ""Burst Length Signature.BLH.csv"""

BLENgth:GTBase 0 | 1 | OFF | ON
BLENgth:GTBase?

This command enables/disables the Global Timebase. Enabling Global Timebase shows the bit rate values in Time along the X-axis. Disabling it shows the bit rate values in Bits along the X-axis.

Time is calculated based on a data rate of bits per second. The data rate can be set or queried using the GPROPerties:CTTRate command.

The GTBase command is equivalent to the Global Timebase checkbox in the properties dialog.

Example

:PLUGin:PUT:STRing "EAPLUGIN:BLENgth:GTBase ON"

:PLUGin:GET:STRing? "EAPLUGIN:BLENgth:GTBase?"

BLENgth:LSCale 0 | 1 | OFF | ON
BLENgth:LSCale?

This command enables/disables use of the Log Scale. Enabling it shows the Y-axis of the histogram as a logarithmic scale. Disabling it shows the Y-axis as a linear scale. This command works like the Log Scale checkbox in the properties dialog.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:BLENgth:LSCale ON"
:PLUGin:GET:STRing? "EAPLUGIN:BLENgth:LSCale?"
```

BLENgth:XCData?

This query-only command gets the data at Cursor X1 and Cursor X2 (smart cursor in the view). The response will be: value1, value2.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:BLENgth:CRANge 1, 100"
:PLUGin:GET:STRing? "EAPLUGIN:BLENgth:CRANge?"
:PLUGin:PUT:STRing "EAPLUGIN:BLENgth:XCENable ON"
:PLUGin:GET:STRing? "EAPLUGIN:BLENgth:XCData?"
```

BLENgth:XCENable 0 | 1 | OFF | ON
BLENgth:XCENable?

This command sets or retrieves the visibility of the X-cursors. It performs the same function as the X-Axis Cursors checkbox in the properties dialog to enable or disable display of Cursors X1 and X2.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:BLENgth:XCENable ON"
:PLUGin:GET:STRing? "EAPLUGIN:BLENgth:XCENable?"
```

BLENgth:XCNPeak <index>

This command sets the X1 cursor to the n'th highest bin.

The last data bin contains data from values outside the range. If this command returns the last bin (and there is data outside the range), it does not mean that the last data bin is actually the greatest.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:BLENgth:CRANge 1, 100"
```

```
:PLUGin:GET:STRing? "EAPLUGIN:BLENgth:CRANge?"
:PLUGin:PUT:STRing "EAPLUGIN:BLENgth:XCNPeak 1"
:PLUGin:GET:STRing? "EAPLUGIN:BLENgth:XCData?"
```

BLENgth:YCENable 0 | 1 | OFF | ON
BLENgth:YCENable?

This command sets or retrieves the visibility of the Y-cursor. It performs the same function as the Y-Axis Cursor checkbox in the properties dialog.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:BLENgth:YCENable ON"
:PLUGin:GET:STRing? "EAPLUGIN:BLENgth:YCENable?"
```

BLENgth:YCPosition <value1>
BLENgth:YCPosition?

This command sets or retrieves the position of Cursor Y1 in the view. The response will be: value1.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:BLENgth:YCPosition 500"
:PLUGin:GET:STRing? "EAPLUGIN:BLENgth:YCPosition?"
```

CORRelation Subsystem

The CORRelation commands control the Correlation analyzer.

The CORRelation Subsystem

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
CORRelation			
:AREA?			Area defined by X-cursors
:BCOunt?			Bin Count
:BLMode	GATing | MARKer | QUANtity		Set mode for definition of Correlation Period
:BLMode?		GATing | MARKer | QUANtity	
:BLQuantity	<numeric>		Correlation Period Quantity
:BLQuantity?		numeric	
:BRESolution?			Query bin resolution
:CEXTents	<Xmin>, <Xmax>, <Ymin>, <Ymax>		Chart Extents
:CEXTents?		Xmin, Xmax, Ymin, Ymax	
:CRANge	<value1>, <value2>		Chart Range

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
:CRANge?		value1, value2	
:DATA?		Binary	All data in bins
:ENABle	0 | 1 | OFF | ON		Enable/disable Correlation analyzer
:ENABle?		Boolean	
:ESFClose			Close Error Signature File
:ESFOpen	<string>		Open Error Signature File
:ESFSave	<string>		Save Error Signature
:GTBase	0 | 1 | OFF | ON		Global Timebase
:GTBase?		Boolean	
:LSCale	0 | 1 | OFF | ON		Log Scale
:LSCale?		Boolean	
:XCData?		value1, value2	Bin Data
:XCENable	0 | 1 | OFF | ON		X-cursor visibility
:XCENable?		Boolean	
:XCNPeak	<index>		Set X1 cursor to n'th highest bin
:YCENable	0 | 1 | OFF | ON		Y-cursor visibility
:YCENable?		Boolean	
:YCPosition	<value>		Y-cursor position
:YCPosition?		value	

CORRelation:AREA?

This query-only command returns the area defined by Cursor X1 and Cursor X2. The response is the same as the “Cursor Area” displayed under the view.

Example :PLUGin:GET:STRing? "EAPLUGIN:CORRelation:AREA?"

CORRelation:BCOunt?

This query-only command gets the bin count stored in the server. There is no counterpart in the GUI.

Example :PLUGin:GET:STRing? "EAPLUGIN:CORRelation:BCOunt?"

CORRelation:BLMode GATing | MARKer | PATTern | QUANtity
CORRelation:BLMode?

This command sets or retrieves the definition of the boundary for a Correlation Period. GATing = Gating Signal, MARKer = Marker Signal, PATTern = PRBS pattern size, and QUANtity = Specified number of bits, which must be set using the BLQuantity command, below.

Example :PLUGin:PUT:STRing "EAPLUGIN:CORRelation:BLMode QUANTITY"

:PLUGin:GET:STRing? "EAPLUGIN:CORRelation:BLMode?"

CORRelation:BLQuantity <numeric>
CORRelation:BLQuantity?

This command sets or retrieves the Correlation Period quantity to be used when Correlation is set to “QUANtity”. Input should not be less than two. If the value input is less than two, then the Correlation Period is set to two.

Example :PLUGin:PUT:STRing "EAPLUGIN:CORRelation:BLQuantity 64"

:PLUGin:GET:STRing? "EAPLUGIN:CORRelation:BLQuantity?"

CORRelation:BRESolution?

Bin resolution is determined by the Chart Range values (see CRANge). The “From” value is inclusive and the “To” value is exclusive. For example, the range can be specified as [0,1000], which establishes 1000 bins, from Bin 0 to Bin 999. The bin resolution is 1. Bin 0 contains data [–inf, 1], and Bin 999 contains [999, +inf].

When the range is greater than or equal to 1001, the bin resolution will be 2 or more, as necessary to include all data within the available bins.

Example

```
:PLUGin:GET:STRing? "EAPLUGIN:CORRelation:BRESolution?"
```

CORRelation:CEXTents <Xmin>, <Xmax>, <Ymin>, <Ymax> CORRelation:CEXTents?

This command sets or retrieves the boundaries of the view, from the minimum to maximum X-values, and minimum to maximum Y-values.

If the Chart Range is changed (using the CRANge command, below), a “Fit View” is performed in an attempt to fit the Chart Extents of the view to the defined interval. Hence, the Chart Extent values queried after changing the Chart Range will differ from those previously set. Use the query command CEXTents? to discover the new chart extents.

The same range check as is used by the GUI applies here..

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:CORRelation:CEXTents 0, 100, 1, 1000"
:PLUGin:GET:STRing? "EAPLUGIN:CORRelation:CEXTents?"
```

CORRelation:CRANge <value1>, <value2> CORRelation:CRANge?

This command sets or retrieves the settings of “Chart Range” in the properties dia-log. The Chart Range defines the range over which you want to collect histogram data, which may differ significantly from the data displayed in the view (limited by the Chart Extents).

When the Chart Range is changed, a “Fit View” is performed in an attempt to fit the Chart Extents of the view to the defined interval. Hence, the Chart Extent values queried after changing the Chart Range will differ from those previously set. Use the query command CEXTents? to discover the new chart extents.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:CORRelation:CRANge 1, 100"
:PLUGin:GET:STRing? "EAPLUGIN:CORRelation:CRANge?"
```

CORRelation:DATA?

Query all the data in all the bins. This gives the user all the data in the server's format, not that of the GUI. It is provided for the user to analyze using information from another source, such as the bin count. The query response form is binary.

Example :PLUGin:GET:BINary? "EAPLUGIN:CORRelation:DATA?"

CORRelation:ENABle 0 | 1 | OFF | ON
CORRelation:ENABle?

This command enables/disables the Correlation analyzer. It is equivalent to the Correlation checkbox in the properties dialog.

Example :PLUGin:PUT:STRing "EAPLUGIN:CORRelation:ENABle ON"

:PLUGin:GET:STRing? "EAPLUGIN:CORRelation:ENABle?"

CORRelation:ESFClose

This command will close an open Error Signature file. It works just like the Close button in the Error Signature Manager.

Example :PLUGin:PUT:STRing "EAPLUGIN:CORRelation:ESFClose"

CORRelation:ESFOpen <string>

This command opens the Error Signature file named in <string>. It works just like the Open button in the Error Signature Manager.

Example :PLUGin:PUT:STRing "EAPLUGIN:CORRelation:ESFOpen ""Correlation Signature.ch.csv"""

CORRelation:ESFSave <string>

This command saves the Error Signature to the file named in the <string>. It works just like the Save button in the Error Signature Manager.

An Error Signature is an ASCII file containing a list of comma-separated values representing error performance characteristics of a particular failure mode. A saved Error Signature can be compared against new results to identify problem types that have occurred before.

Example

:PLUGin:PUT:STRing "EAPLUGIN:FDIRectories:ESFile ""C:\ELA_GPIB\CorrelationAnalyzer"""

:PLUGin:GET:STRing? "EAPLUGIN:FDIRectories:ESFile?"

:PLUGin:PUT:STRing "EAPLUGIN:CORRelation:ESFSave ""Correlation Signature.ch.csv"""

CORRelation:GTBase 0 | 1 | OFF | ON CORRelation:GTBase?

This command enables/disables the Global Timebase. Enabling Global Timebase shows the bit rate values in Time along the X-axis. Disabling it shows the bit rate values in Bits along the X-axis.

Time is calculated based on a data rate of bits per second. The data rate can be set or queried using the GPROPerties:CTTRate command.

The GTBase command is equivalent to the Global Timebase checkbox in the properties dialog.

Example

:PLUGin:PUT:STRing "EAPLUGIN:CORRelation:GTBase ON"

:PLUGin:GET:STRing? "EAPLUGIN:CORRelation:GTBase?"

CORRelation:LSCale 0 | 1 | OFF | ON CORRelation:LSCale?

This command enables/disables use of the Log Scale. Enabling it shows the Y-axis of the histogram as a logarithmic scale. Disabling it shows the Y-axis as a linear scale. This command works like the Log Scale checkbox in the properties dialog.

Example

:PLUGin:PUT:STRing "EAPLUGIN:CORRelation:LSCale ON"

:PLUGin:GET:STRing? "EAPLUGIN:CORRelation:LSCale?"

CORRelation:XCData?

This query-only command gets the data at Cursor X1 and Cursor X2 (smart cursor in the view). The response will be: value1, value2.

Example

:PLUGin:PUT:STRing "EAPLUGIN:CORRelation:CRANge 1, 100"

:PLUGin:GET:STRing? "EAPLUGIN:CORRelation:CRANge?"

:PLUGin:PUT:STRing "EAPLUGIN:CORRelation:XCENable ON"

:PLUGin:GET:STRing? "EAPLUGIN:CORRelation:XCData?"

CORRelation:XCENable 0 | 1 | OFF | ON
CORRelation:XCENable?

This command sets or retrieves the visibility of the X-cursors. It performs the same function as the X-Axis Cursors checkbox in the properties dialog to enable or disable display of Cursors X1 and X2.

Example

:PLUGin:PUT:STRing "EAPLUGIN:CORRelation:XCENable ON"

:PLUGin:GET:STRing? "EAPLUGIN:CORRelation:XCENable?"

CORRelation:XCNPeak <index>

This command sets the X1 cursor to the n'th highest bin.

The last data bin contains data from values outside the range. If this command returns the last bin (and there is data outside the range), it does not mean that the last data bin is actually the greatest.

Example

:PLUGin:PUT:STRing "EAPLUGIN:CORRelation:XCNPeak 2"

:PLUGin:GET:STRing? "EAPLUGIN:CORRelation:XCData?"

CORRelation:YCENable 0 | 1 | OFF | ON
CORRelation:YCENable?

This command sets or retrieves the visibility of Cursor Y1. It performs the same function as the Y-Axis Cursor checkbox in the properties dialog.

Example

:PLUGin:PUT:STRing "EAPLUGIN:CORRelation:YCENable ON"

:PLUGin:GET:STRing? "EAPLUGIN:CORRelation:YCENable?"

CORRelation:YCPosition <value> CORRelation:YCPosition?

This command sets or retrieves the position of Cursor Y1 in the view. The response will be: value1.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:CORRelation:YCPosition 100"
:PLUGin:GET:STRing? "EAPLUGIN:CORRelation:YCPosition?"
```

ECCorrection Subsystem

The ECCorrection commands control the ECC analyzer.

The ECCorrection Subsystem

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
ECCorrection			
:AABCount?		value	Total Accumulation After ECC bit count
:AAECount?		value	Total Accumulation After ECC error count
:AAERate?		value	Total Accumulation After ECC BER
:ABBCount?		value	Total Accumulation Before ECC bit count
:ABECount?		value	Total Accumulation Before ECC error count

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
:ABERate?		value	Total Accumulation Before ECC BER
:ADRate?		value	After ECC data rate
:AECorrections?		value	Total Accumulation Erasure corrections
:AEFailures?		value	Total Accumulation Erasure failures
:AICCorrections?		value	Total Accumulation Inner Code corrections
:AICFailures?		value	Total Accumulation Inner Code failures
:AOCCorrections?		value	Total Accumulation Outer code corrections
:AOCFailures?		value	Total Accumulation Outer code failures
:BDRate?		value	Before ECC data rate
:EMODe	DISabled | FULL		Emulation mode
:EMODe?			

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
:ENABle	0 | 1 | OFF | ON		Enable/disable ECC analyzer and ECC filter
:ENABle?		Boolean	
:EOSYmbols	0 | 1 | OFF | ON		Eliminate ECC Overhead symbols
:EOSYmbols?		Boolean	
:ERASure	<integer>		Erasure [0,65536]
:ERASure?		value	
:IABCount?		value	Current Interval After ECC bit count
:IAECount?		value	Current Interval After ECC error count
:IAERate?		value	Current Interval After ECC BER
:IBBCount?		value	Current Interval Before ECC bit count
:IBECount?		value	Current Interval Before ECC error count
:IBERate?		value	Current Interval Before ECC BER

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
:IECorrections?		value	Current Interval Erasure corrections
:IEFailures?		value	Current Interval Erasure failures
:IICCorrections?		value	Current Interval Inner code corrections
:IICFailures?		value	Current Interval Inner code failures
:IMODe	BIT | SYMBol		Interleave mode
:IMODE?			
:IOCCorrections?		value	Current Interval Outer code corrections
:IOCFailures?		value	Current Interval Outer code failures
:KINNer	<integer>		Inner Code “k” value [1,65536]
:KINNer?		value	
:KOUTer	<integer>		Outer Code “k” value [1,65536]
:KOUTer?		value	
:NINNer	<integer>		Inner Code “n” value [1,65536]
:NINNer?		value	

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
:NOUTer	<integer>		Outer Code “n” value [1,65536]
:NOUTer?		value	
:OVERhead?		value	ECC overhead
:RINTerval	<integer>		Reporting interval
:RINTerval?		integer	
:SCSelect	G709 | G975 | 2RSCode | M2Fec | RSCode		Action only
:SSIZe	<integer>		ECC symbol size
:SSIZE?		integer	
:TDFLag	0 | 1 | OFF | ON		Two-Dimensional Table checkbox
:TDFLag?		Boolean	
:TINNer	<integer>		Inner Code “T” value [0,65536]
:TINNer?		value	
:TOUTer	<integer>		Outer Code “T” value [0,65536]
:TOUTer?		value	
:TOVerrun?		value	Tables overrun
:TPRocessed?		value	Tables processed

ECCorrection:AABCount?

This query-only command returns the Total Accumulation After ECC bit count.

This is the number of bits transmitted by the ECC filter since the beginning of the error analysis session. This value is held until the next interval results are formulated.

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:AABCount?"

ECCorrection:AAECount?

This query-only command returns the Total Accumulation After ECC error count.

The number of errors transmitted from the ECC filter subsequent to error correction emulation processing since the beginning of the error analysis session. These results are continuously updating.

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:IEFailures?"

ECCorrection:AAERate?

This query-only command returns the Total Accumulation After ECC BER.

This is the number of errors transmitted from the ECC filter subsequent to error correction emulation processing since the beginning of the error analysis session divided by the number of bits output by the ECC Emulation during the entire error analysis session. The number of bits output from the ECC filter may be less than the number of bits input to the filter, depending on the (n,k) settings and whether ECC codewords are being stripped from the output or not.

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:AAERate?"

ECCorrection:ABBCount?

This query-only command returns the Total Accumulation Before ECC bit count.

This is the number of bits received by the ECC filter since the beginning of the error analysis session. This value is held until the next interval results are formulated.

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:ABBCount?"

ECCorrection:ABECount?

This query-only command returns the Total Accumulation Before ECC error count.

This is the number of errors received by the ECC filter since the beginning of the error analysis session. These results are continuously updating.

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:ABECount?"

ECCorrection:ABERate?

This query-only command returns the Total Accumulation Before ECC BER.

This is the number of errors received by the ECC filter since the beginning of the error analysis session divided by the total number of bits received since the beginning of the session. These results are continuously updating.

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:ABERate?"

ECCorrection:ADRate?

This query-only command returns the After ECC data rate.

This is the serial bit rate of the error detector adjusted by subtraction of the ECC overhead if codewords are being stripped off during the ECC filter processing (see EOSYmbols command).

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:ADRate?"

ECCorrection:AECorrections?

This query-only command returns the number of Total Accumulation Erasure corrections.

This is the number of tables successfully corrected using the erasure phase of error correction emulation during the entire analysis session. These results are continuously updating.

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:AECorrections?"

ECCorrection:AEFailures?

This query-only command returns the number of Total Accumulation Erasure failures.

This is the number of tables not able to be corrected by the erasure phase of error correction emulation during the entire analysis session. These results are continuously updating.

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:AEFailures?"

ECCorrection:AICCorrections?

This query-only command returns the number of Total Accumulation Inner Code corrections.

This is the number of symbols corrected by inner code error correction emulation during the entire error analysis session. These results are continuously updating.

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:AICCorrections ?"

ECCorrection:AICFailures?

This query-only command returns the number of Total Accumulation Inner Code failures.

This is the number of symbol errors not able to be corrected by inner code error correction emulation during the entire error analysis session. These results are continuously updating.

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:AICFailures ?"

ECCorrection:AOCCorrections?

This query-only command returns the number of Total Accumulation Outer Code corrections.

This is the number of symbols corrected by outer code error correction emulation during the entire error analysis session. These results are continuously updating. When Erasure processing is enabled, the outer code statistics refer to corrections that are not made by erasure processing and which are subsequently made by outer code processing.

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:AOCCorrections ?"

ECCorrection:AOCFailures?

This query-only command returns the number of Total Accumulation Outer Code failures.

This is the number of symbol errors not able to be corrected by outer code error correction emulation during the entire error analysis session. These results are continuously updating.

Example

```
:PLUGin:GET:STRing? "EAPLUGIN:ECC:AOCFailures ?"
```

ECCorrection:BDRate?

This query-only command returns the Before ECC data rate.

This is the serial bit rate of the error detector as measured when the error analysis session was started.

Example

```
:PLUGin:GET:STRing? "EAPLUGIN:ECC:BDRate?"
```

ECCorrection:EMODe DISabled | FULL
ECCorrection:EMODe?

This command sets and retrieves the Emulation Mode. The Mode can be set to Disabled, Inner only, Skip Erasure, or Full Emulation.

“Disabled” deactivates correction. The ECC filter is inserted into the error data processing chain, but no errors are removed as a result of correction emulation. If the ECC Overhead codewords are being stripped (see EOSYmbols command, below), then any errors occurring within the overhead portion of the format will be removed, along with the rest of the overhead.

“Full” engages all stages of emulation. Parameters are specified using the INKT, ONKT, and ERAS commands below.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:ECC:EMODe FULL"
:PLUGin:GET:STRing? "EAPLUGIN:ECC:EMODe?"
```

ECCorrection:ENABle 0 | 1 | OFF | ON ECCorrection:ENABle?

This command enables or disables the ECC analyzer and the ECC filter. It is equivalent to the ECC Analyzer checkbox in the properties dialog. The query form of the command returns the state of the analyzer.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:ECC:ENABle ON"
:PLUGin:GET:STRing? "EAPLUGIN:ECC:ENABle?"
```

ECCorrection:EOSYmbols 0 | 1 | OFF | ON ECCorrection:EOSYmbols?

This command sets and retrieves whether ECC overhead codewords are being stripped from the data stream. It is equivalent to the Strip ECC Overhead Codewords checkbox.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:ECC:EOSYmbols ON"
:PLUGin:GET:STRing? "EAPLUGIN:ECC:EOSYmbols?"
```

ECCorrection:ERASure <integer> ECCorrection:ERASure?

This command sets and retrieves the Erasure strength. For forward error correction systems that have erasure flagging, an additional parameter called erasure strength is required.

Range is zero to 65,536.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:ECC:ERASure 0"
:PLUGin:GET:STRing? "EAPLUGIN:ECC:ERASure?"
```

ECCorrection:IABCount?

This query-only command returns the Current Interval After ECC bit count.

This is the number of bits transmitted by the ECC filter during the most recent interval. This value is held until the next interval results are formulated.

Example

```
:PLUGin:GET:STRing? "EAPLUGIN:ECC:IABCount?"
```

ECCorrection:IAECount?

This query-only command returns the Current Interval After ECC error count.

This is the number of errors transmitted from the ECC filter subsequent to error correction emulation processing during the most recent interval. These results are held until the next interval results are formulated

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:IAECount?"

ECCorrection:IAERate?

This query-only command returns the Current Interval After ECC BER.

This is the number of errors transmitted from the ECC filter subsequent to error correction emulation processing during the most-recent interval divided by the number of bits output during that interval. These results are held until the next interval results are formulated. The number of bits output from the ECC filter may be less than the number of bits input to the filter, depending on the (n,k) settings and whether ECC codewords are being stripped from the output or not.

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:IAERate?"

ECCorrection:IBBCount?

This query-only command returns the Current Interval Before ECC bit count.

This is the number of bits received by the ECC filter during the most recent interval. This value is held until the next interval results are formulated and displayed.

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:IBBCount?"

ECCorrection:IBECount?

This query-only command returns the Current Interval Before ECC error count.

This is the number of errors received by the ECC filter during the most recent interval. This value is held until the next interval results are formulated.

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:IBECount?"

ECCorrection:IBERate?

This query-only command returns the Current Interval Before ECC BER.

This is the number of errors received by the ECC filter during the most recent interval, divided by the total number of bits received by the filter during the same interval. This value is held until the next interval results are formulated.

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:IBERate?"

ECCorrection:IECorrections?

This query-only command returns the number of Current Interval erasure corrections.

This is the number of tables successfully corrected using the erasure phase of error correction emulation during the most-recent interval. These results are held until the next interval results are formulated.

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:IECorrections?"

ECCorrection:IEFailures?

This query-only command returns the number of Current Interval erasure failures.

This is the number of tables not able to be corrected by the erasure phase of error correction emulation during the most-recent interval. These results are held until the next interval results are formulated.

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:IEFailures?"

ECCorrection:IICCorrections?

This query-only command returns the number of Current Interval Inner Code corrections.

This is the number of symbols corrected by inner code error correction emulation during the most-recent interval. These results are held until the next interval results are formulated.

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:IICCorrections ?"

ECCorrection:IICFailures?

This query-only command returns the number of Current Interval Inner Code failures.

This is the number of symbol errors not able to be corrected by inner code error correction emulation during the most-recent interval. These results are held until the next interval results are formulated.

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:IICFailures ?"

ECCorrection:IMODe BIT | SYMBol
ECCorrection:IMODe?

This commands sets and retrieves the Interleave mode. The choices are BIT = bit interleave, or SYMBol = symbol interleave.

Example :PLUGin:PUT:STRing "EAPLUGIN:ECC:IMODe Bit"

:PLUGin:GET:STRing? "EAPLUGIN:ECC:IMODe?"

ECCorrection:IOCCorrections?

This query-only command returns the number of Current Interval Outer Code corrections.

This is the number of symbols corrected by outer code error correction emulation during the most-recent interval. These results are held until the next interval results are formulated. When Erasure processing is enabled, the outer code statistics refer to corrections that are not made by erasure processing and which are subsequently made by outer code processing.

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:IOCCorrections ?"

ECCorrection:IOCFailures?

This query-only command returns the number of Current Interval Outer Code failures.

This is the number of symbol errors not able to be corrected by outer code error correction emulation during the most-recent interval. These results are held until the next interval results are formulated.

Example :PLUGin:GET:STRing? "EAPLUGIN:ECC:IOCFailures ?"

ECCorrection:KINNer <integer>
ECCorrection:KINNer?

This command sets and retrieves the Inner Code “k” value. One-dimensional forward error correction is defined by a single inner code input block size “n,” an output block size “k,” and a correction strength, “T” (see also NINNer and TINNer commands).

Range for k is one to 65,536.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:ECC:KINNer 7000"
:PLUGin:GET:STRing? "EAPLUGIN:ECC:KINNer?"
```

ECCorrection:KOUTer <integer>
ECCorrection:KOUTer?

This command sets and retrieves the Outer Code “k” value. Two-dimensional forward error correction systems have, in addition to the inner code definitions (see NINNer, TINNer, KINNer commands), an outer code definition consisting of a single outer code input block size “n,” an output block size “k,” and a correction strength, “T” (see also NOUTer and TOUTer commands).

Range for k is one to 65,536.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:ECC:KOUTer 5"
:PLUGin:GET:STRing? "EAPLUGIN:ECC:KOUTer?"
```

ECCorrection:NINNer <integer>
ECCorrection:NINNer?

This command sets and retrieves the Inner Code “n” value. One-dimensional forward error correction is defined by a single inner code input block size “n,” an output block size “k,” and a correction strength, “T” (see also TINNer and KINNer commands).

Range for n is one to 65,536.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:ECC:NINNer 10000"
:PLUGin:GET:STRing? "EAPLUGIN:ECC:NINNer?"
```

ECCorrection:NOUTer <integer> ECCorrection:NOUTer?

This command sets and retrieves the Outer Code “n” value. Two-dimensional forward error correction systems have, in addition to the inner code definitions (see NINNer, TINNer, KINNer commands), an outer code definition consisting of a single outer code input block size “n,” an output block size “k,” and a correction strength, “T” (see also KOUTer and TOUTer commands).

Range for n is one to 65,536.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:ECC:NOUTer 10"
:PLUGin:GET:STRing? "EAPLUGIN:ECC:NOUTer?"
```

ECCorrection:OVERhead?

This query-only command returns the amount of ECC overhead.

A given ECC Emulation configuration will define the amount of overhead for the forward error correction system. For an inner code defined as RS1(n1,k1) and an outer code defined as RS2(n2,k2), the overhead is calculated as:

Overhead% := { [(n1 * n2) / (k1 * k2)] – 1.0 } * 100.0

This number is expressed as a percentage with two digits beyond the decimal place.

Example

```
:PLUGin:GET:STRing? "EAPLUGIN:ECC:OVERhead?"
```

ECCorrection:RINTerval <int> ECCorrection:RINTerval?

This command sets and retrieves the Reporting Interval. Range is one to 300.

This value determines how often Current Interval statistics are updated. The parameter is set in seconds, as defined by the Global Timebase. To correspond with actual real-time seconds, the data rate and timebase must be equal; this can be assured by using the measured error detector clock rate at the time of data collection as the source for timebase transformation.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:ECC:RINTerval 8"
:PLUGin:GET:STRing? "EAPLUGIN:ECC:RINTerval?"
```

ECCorrection:SCSelect G709 | G975 | 2RSCode | M2Fec | RSCode

This command selects the Standard ECC configuration. Choose from:

G709 = ITU-T G.709
G975 = ITU-T G.975
2RSCode = Two-Dimensional Reed-Solomon Code
M2Fec = MPEG-2 FEC
RSCode = Reed-Solomon RS(255,239) Code

The default configuration will be (Custom).

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:ECC:SCSelect G709"
```

ECCorrection:SSIZe <integer>
ECCorrection:SSIZe?

This command sets and retrieves the ECC symbol size. This indicates the symbol size for ECC analysis and is separate from the Global Symbol Size. Range is one to 16.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:ECC:SSIZe 8"
:PLUGin:GET:STRing? "EAPLUGIN:ECC:SSIZe?"
```

ECCorrection:TDFlag 0 | 1 | OFF | ON
ECCorrection:TDFlag?

This command sets and retrieves the state of the Two-Dimensional Table flag. When enabled, a two-dimensional error correction code uses both the inner and outer code parameters to create a two-dimensional correction table. A one-dimensional code uses only the inner code parameters.

This command corresponds to the Two-Dimensional Table checkbox in the GUI.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:ECC:TDFLag On"
:PLUGin:GET:STRing? "EAPLUGIN:ECC:TDFL?"
```

ECCorrection:TINNer <integer>
ECCorrection:TINNer?

This command sets and retrieves the Inner Code “T” value. One-dimensional forward error correction is defined by a single inner code input block size “n,” an output block size “k,” and a correction strength, “T” (see also NINNer and KINNer commands).

Range for T is zero to 65,536.

Example

:PLUGin:PUT:STRing "EAPLUGIN:ECC:TINNer 0"

:PLUGin:GET:STRing? "EAPLUGIN:ECC:TINNer?"

ECCorrection:TOUTer <integer>
ECCorrection:TOUTer?

This command sets and retrieves the Outer Code “T” value. Two-dimensional forward error correction systems have, in addition to the inner code definitions (see NINNer, TINNer, KINNer commands), an outer code definition consisting of a single outer code input block size “n,” an output block size “k,” and a correction strength, “T” (see also NOUTer and KOUTer commands).

Range for T is zero to 65,536.

Example

:PLUGin:PUT:STRing "EAPLUGIN:ECC:TOUTer 0"

:PLUGin:GET:STRing? "EAPLUGIN:ECC:TOUTer?"

ECCorrection:TOVerrun?

This query-only command returns the number of Tables Overrun.

This is the number of tables during the entire analysis session that were found to have more errors than could be accommodated with the ECC filter’s memory buffers.

Example

:PLUGin:GET:STRing? "EAPLUGIN:ECC:TOVerrun?"

ECCorrection:TPRocessed?

This query-only command returns the number of Tables Processed during the entire anaysis session. This value is continuously updating.

Example

:PLUGin:GET:STRing? "EAPLUGIN:ECC:TPRocessed?"

EDSFile Subsystem

The EDSFile commands control operations with the Error Data Set files.

The EDSFile Subsystem

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
ESDFile			
:PLAYback			
:BEGin	[PROPerties I NOPRoperties]		Start UER Playback
:END			Stop UER Playback
:FNAMe	<string>		Playback File Name
:FNAMe?		string	
:PROGress?			Query progress of playback
:SPRoperties			Save current property settings & associate with UER file
ESDFile			
:RECord			
:FLIMit	10KB I 50KB I 100KB I 500KB I 1MB I 5 MB I NOLimit		File Limit

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
:FLIMit?		10KB | 50KB | 100KB | 500KB | 1MB | 5 MB | NOLimit	
:FNAMe	<string>		Record File Name
:FNAMe?		string	
:OWRite	0 | 1 | OFF | ON		OK to Overwrite
:OWRite?		Boolean	
:PROPerties	0 | 1 | OFF | ON		Save Properties while Recording
:PROPerties?		Boolean	

EDSFile:PLAYback:BEGin [PROPerties | NOPRoperties]

This command will start playback of an error data set, opening the file specified in the UER (Universal Error Record, or error data set) file name (see PLAY-back:FNAMe, below), with the option to restore the associated properties (if there are any) or not.

The error data set is a file that holds the location of errors, not the results of analyz-ing errors. Each time you open a previously recorded error data set, the BitAlyzer-studies the error locations again and produces analysis results. The “Properties” associated with the error data set define the parameter configuration of the analyzer at the time the error data was recorded, such as Block Length, Timebase, or Chart Range. You can use the property settings saved with the error location data, or make changes to analyze the data in a different way.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:EDSFile:PLAYback:FNAMe
""C:\ELA_GPIB\BlockErrorAnalyzer\Block_2048.uer"""
:PLUGin:GET:STRing? "EAPLUGIN:EDSFile:PLAYback:FNAMe?"
:PLUGin:PUT:STRing "EAPLUGIN:EDSFile:PLAYback:BEGin PROP"
*OPC?
```

EDSFile:PLAYback:END

This command is an action only. It stops the error data set playback.

Example :PLUGin:PUT:STRing "EAPLUGIN:EDSFile:PLAYback:END"

EDSFile:PLAYback:FNAMe <string>
EDSFile:PLAYback:FNAMe?

This command sets or retrieves the UER (Universal Error Record, or error data set) Playback file name.

Example :PLUGin:PUT:STRing "EAPLUGIN:EDSFile:PLAYback:FNAMe ""C:\ELA_GPIB\BlockErrorAnalyzer\Block_2048.uer"""

EDSFile:PLAYback:PROGress?

This query-only command returns the progress of the error data set playback.

Example :PLUGin:GET:STRing? "EAPLUGIN:EDSFile:PLAYback:PROGress?"

EDSFile:PLAYback:SPRoperties

This command is an action only. It is equivalent to the Save Properties button, and saves the current property settings to be associated with the current specified error data set playback file.

The “Properties” associated with the error data set define the parameter configuration of the analyzer at the time the error data was recorded, such as Block Length, Timebase, or Chart Range.

Example :PLUGin:PUT:STRing "EAPLUGIN:EDSFile:PLAYback:FNAMe ""C:\ELA_GPIB\BlockErrorAnalyzer\Block_2048.uer"""

:PLUGin:GET:STRing? "EAPLUGIN:EDSFile:PLAYback:FNAMe?"

:PLUGin:PUT:STRing "EAPLUGIN:EDSFile:PLAYback:SPRoperties"

EDSFile:RECord:FLIMit 10KB | 50KB | 100 KB | 500KB | 1MB | 5MB | NOLimit EDSFile:RECord:FLIMit?

This command sets or retrieves the UER error data set file size limit. The most common reason for limiting the size of the logging file would be to keep a file small enough that it is easily moved or analyzed by another program. Choices are 10 kB, 50 kB, 500 kB, 1 MB, 5 MB, or “No Limit”.

Example

:PLUGin:PUT:STRing "EAPLUGIN:EDSFile:RECord:FLIMit NOLIMIT"

:PLUGin:GET:STRing? "EAPLUGIN:EDSFile:RECord:FLIMit?"

EDSFile:RECord:FNAMe <string> EDSFile:RECord:FNAMe?

This command sets or retrieves the error data set UER Record file name. During accumulation, the error data set will be recorded using the file name specified.

Example

:PLUGin:PUT:STRing "EAPLUGIN:FDIRectories:EDSFile ""C:\ErrorDataFolder"""

:PLUGin:GET:STRing? "EAPLUGIN:FDIRectories:EDSFile?"

:PLUGin:PUT:STRing "EAPLUGIN:EDSFile:RECord:FNAMe ""Filename"""

:PLUGin:GET:STRing? "EAPLUGIN:EDSFile:RECord:FNAMe?

EDSFile:RECord:OWRite 0 | 1 | OFF | ON EDSFile:RECord:OWRite?

This command sets or retrieves the status of the OK to Overwrite parameter for the specified error data set Record file (see RECord:FName, below). If you do not allow file overwrite, and you attempt to re-record after having recorded an error data set, the analyzer will go into LIVE mode and will not record new error information to the file.

Example

:PLUGin:PUT:STRing "EAPLUGIN:EDSFile:RECord:OWRite ON"

:PLUGin:GET:STRing? "EAPLUGIN:EDSFile:RECord:OWRite?"

EDSFile:RECord:PROPerties 0 | 1 | OFF | ON EDSFile:RECord:PROPerties?

This command enables/disables saving associated properties while recording an error data set to the specified UER Record file (see RECord:FName, below). It is equivalent to the checkbox on the General Settings page.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:EDSFile:RECord:PROPerties ON"
:PLUGin:GET:STRing? "EAPLUGIN:EDSFile:RECord:PROPerties?"
```

EFINterval Subsystem

The EFINterval commands control the Error Free Interval analyzer.

The EFINterval Subsystem

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
EFINterval			
:AREA?			Area defined by X-cursors
:BCOunt?		numeric	Bin Count
:BRESolution?			Query bin resolution
:CEXTents	<Xmin>, <Xmax>, <Ymin>, <Ymax>		Chart Extents
:CEXTents?		Xmin, Xmax, Ymin, Ymax	
:CRANge	<value1>, <value2>		Chart Range
:CRANge?		value1, value2	
:DATA?		Binary	All data in bins
:ENABle	0 | 1 | OFF | ON		Enable/disable Error Free Interval analyzer
:ENABle?		Boolean	

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
:ESFClose			Close Error Signature File
:ESFOpen	<string>		Open Error Signature File
:ESFSave	<string>		Save Error Signature
:GTBase	0 | 1 | OFF | ON		Global Timebase
:GTBase?		Boolean	
:LSCale	0 | 1 | OFF | ON		Log Scale
:LSCale?		Boolean	
:XCData?		value1, value2	Bin Data
:XCENable	0 | 1 | OFF | ON		X-cursor visibility
:XCENable?		Boolean	
:XCNPeak	<index>		Set X1 cursor to n’th highest bin
:YCENable	0 | 1 | OFF | ON		Y-cursor visibility
:YCENable?		Boolean	
:YCPosition	<value1>		Y-cursor position
:YCPosition?		<value1>	

EFINterval:AREa?

Query only command.

Example :PLUGin:GET:STRing? "EAPLUGIN:EFINterval:AREa?"

EFINterval:BCOunt?

This query-only command gets the bin count stored in the server. There is no counterpart in the GUI.

Example :PLUGin:GET:STRing? "EAPLUGIN:EFINterval:BCOunt?"

EFINterval:BRESolution?

This query-only command returns the bin resolution.

Bin resolution is determined by the Chart Range values (see CRANge). The “From” value is inclusive and the “To” value is exclusive. For example, the range can be specified as [0,1000], which establishes 1000 bins, from Bin 0 to Bin 999. The bin resolution is 1. Bin 0 contains data [–inf, 1], and Bin 999 contains [999, +inf].

When the range is greater than or equal to 1001, the bin resolution will be 2 or more, as necessary to include all data within the available bins.

Example :PLUGin:GET:STRing? "EAPLUGIN:EFINterval:BRESolution?"

EFINterval:CEXTents <Xmin>, <Xmax>, <Ymin>, <Ymax> EFINterval:CEXTents?

This command sets or retrieves the boundaries of the view, from the minimum to maximum X-values, and minimum to maximum Y-values.

If the Chart Range is changed (using the CRANge command, below), a “Fit View” is performed in an attempt to fit the Chart Extents of the view to the defined interval. Hence, the Chart Extent values queried after changing the Chart Range will differ from those previously set. Use the query command CEXTents? to discover the new chart extents.

The same range check as is used by the GUI applies here.

Example :PLUGin:PUT:STRing "EAPLUGIN:EFINterval:CEXTents 1, 20000, 1, 10000"
:PLUGin:GET:STRing? "EAPLUGIN:EFINterval:CEXTents?"

EFINterval:CRANge <value1>, <value2>

EFINterval:CRANge?

This command sets or retrieves the settings of “Chart Range” in the properties dialog. The Chart Range defines the range over which you want to collect histogram data, which may differ significantly from the data displayed in the view (limited by the Chart Extents).

When the Chart Range is changed, a “Fit View” is performed in an attempt to fit the Chart Extents of the view to the defined interval. Hence, the Chart Extent values queried after changing the Chart Range will differ from those previously set. Use the query command CEXTents? to discover the new chart extents.

Example :PLUGin:PUT:STRing "EAPLUGIN:EFINterval:CRANge 1, 18000"

:PLUGin:GET:STRing? "EAPLUGIN:EFINterval:CRANge?"

EFINterval:DATA?

Query all the data in all the bins. This gives the user all the data in the server’s format, not that of the GUI. It is provided for the user to analyze using information from another source, such as the bin count.

Binary query.

Example :PLUGin:GET:BINary? "EAPLUGIN:EFINterval:DATA?"

EFINterval:ENABle 0 | 1 | OFF | ON
EFINterval:ENABle?

This command enables/disables the Error Free Interval analyzer. It is equivalent to the Error Free Interval checkbox in the properties dialog.

Example :PLUGin:PUT:STRing "EAPLUGIN:EFINterval:ENABle On"

:PLUGin:GET:STRing? "EAPLUGIN:EFINterval:ENABle?"

EFINterval:ESFClose

This command will close an open Error Signature file. It works just like the Close button in the Error Signature Manager.

Example :PLUGin:PUT:STRing "EAPLUGIN:EFINterval:ESFClose"

EFINterval:ESFOpen <string>

This command opens the Error Signature file named in <string>. It works just like the Open button in the Error Signature Manager.

Example :PLUGin:PUT:STRing "EAPLUGIN:EFINterval:ESFOpen ""Error Free Interval Signature.EH.csv"""

EFINterval:ESFSave <string>

This command saves the Error Signature to the file named in the <string>. It works just like the Save button in the Error Signature Manager.

An Error Signature is an ASCII file containing a list of comma-separated values representing error performance characteristics of a particular failure mode. A saved Error Signature can be compared against new results to identify problem types that have occurred before.

Example :PLUGin:PUT:STRing "EAPLUGIN:FDIRectories:ESFile ""C:\ELA_GPIB\EFIAnalyzer"""

:PLUGin:GET:STRing? "EAPLUGIN:FDIRectories:ESFile?"

:PLUGin:PUT:STRing "EAPLUGIN:EFINterval:ESFSave ""Error Free Interval Signature.EH.csv"""

EFINterval:GTBase 0 | 1 | OFF | ON
EFINterval:GTBase?

This command enables/disables the Global Timebase. Enabling Global Timebase shows the bit rate values in Time along the X-axis. Disabling it shows the bit rate values in Bits along the X-axis.

Time is calculated based on a data rate of bits per second. The data rate can be set or queried using the GPROPerties:CTTRate command.

The GTBase command is equivalent to the Global Timebase checkbox in the properties dialog.

Example :PLUGin:PUT:STRing "EAPLUGIN:EFINterval:GTBase ON"

:PLUGin:GET:STRing? "EAPLUGIN:EFINterval:GTBase?"

EFINterval:LSCale 0 | 1 | OFF | ON
EFINterval:LSCale?

This command enables/disables use of the Log Scale. Enabling it shows the Y-axis of the histogram as a logarithmic scale. Disabling it shows the Y-axis as a linear scale. This command works like the Log Scale checkbox in the properties dialog.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:EFINterval:LSCale ON"
:PLUGin:GET:STRing? "EAPLUGIN:EFINterval:LSCale?"
```

EFINterval:XCData?

This query-only command gets the data at Cursor X1 and Cursor X2 (smart cursor in the view). The response will be: value1, value2.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:EFINterval:CRANge 0, 100"
:PLUGin:GET:STRing? "EAPLUGIN:EFINterval:CRANge?"
:PLUGin:PUT:STRing "EAPLUGIN:EFINterval:XCENable ON"
:PLUGin:GET:STRing? "EAPLUGIN:EFINterval:XCData?"
```

EFINterval:XCENable 0 | 1 | OFF | ON
EFINterval:XCENable?

This command sets or retrieves the visibility of the X-cursors. It performs the same function as the X-Axis Cursors checkbox in the properties dialog to enable or disable display of Cursors X1 and X2.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:EFINterval:XCENable ON"
:PLUGin:GET:STRing? "EAPLUGIN:EFINterval:XCENable?"
```

EFINterval:XCNPeak <index>

This command sets the X1 cursor to the n'th highest bin.

The last data bin contains data from values outside the range. If this command returns the last bin (and there is data outside the range), it does not mean that the last data bin is actually the greatest.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:EFINterval:XCNPeak 2"
:PLUGin:GET:STRing? "EAPLUGIN:EFINterval:XCData?"
```

EFINterval:YCENable 0 | 1 | OFF | ON
EFINterval:YCENable?

This command sets or retrieves the visibility of Cursor Y1. It performs the same function as the Y-Axis Cursor checkbox in the properties dialog.

Example

:PLUGin:PUT:STRing "EAPLUGIN:EFINterval:YCENable ON"

:PLUGin:GET:STRing? "EAPLUGIN:EFINterval:YCENable?"

EFINterval:YCPosition <value1>
EFINterval:YCPosition?

This command sets or retrieves the position of Cursor Y1 in the view. The response will be: value1.

Example

:PLUGin:PUT:STRing "EAPLUGIN:EFINterval:YCPosition 100"

:PLUGin:GET:STRing? "EAPLUGIN:EFINterval:YCPosition?"

ESTatistics Subsystem

The ESTatistics commands control the Error Statistics analyzer.

The ESTatistics Subsystem

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
ESTatistics			
:BRCount?		numeric	Burst-related errors
:BRRate?		numeric	Burst-related errors
:ENABle	0 | 1 | OFF | ON		Enable/disable Error Statistics Analyzer
:ENABle?		Boolean	
:MCOunt?		numeric	Markers
:MRATe?		numeric	Markers
:NBRCount?		numeric	Non burst-related errors
:NBRRate?		numeric	Non burst-related errors
:RECount?		numeric	Removed Errors
:TCOunt?		numeric	Total Count
:TECount?		numeric	Total Errors
:TERate?		numeric	Total Errors

ESTatistics:BRCount?

This query-only command returns the number of burst related errors. It counts only the actual errors in the burst related error events.

In order to be counted as a burst related error, the errors must occur inside bursts whose length are equal to or exceed the minimum burst length. For example: If there are occurrences of bursts of length 100, but there are only 53 errors inside the burst, then only 53 errors will be added to the burst related errors, as long as the minimum burst length is less than or equal to 100.

Example :PLUGin:GET:STRing? "EAPLUGIN:ESTatistics:BRCount?"

ESTatistics:BRRate?

This query-only command returns the burst related error rate. During live analysis or error data set file playback, the error rate measurement shows the error rate for the interval of the last integration period. When the measurement session is complete, the error rate value is updated to reflect the overall error rate for the entire session.

Example :PLUGin:GET:STRing? "EAPLUGIN:ESTatistics:BRRate?"

ESTatistics:ENABle 0 | 1 | OFF | ON
ESTatistics:ENABle?

This command enables/disables the Error Statistics analyzer. It is equivalent to the Error Statistics checkbox in the properties dialog.

Example :PLUGin:PUT:STRing "EAPLUGIN:ESTatistics:ENABLe ON"

:PLUGin:GET:STRing? "EAPLUGIN:ESTatistics:ENABLe?"

ESTatistics:MCOunt?

This query-only command returns the number of user Marker signals detected during the measurement.

Example :PLUGin:GET:STRing? "EAPLUGIN:ESTatistics:MCOunt?"

ESTatistics:MRATe?

This query-only command returns the Marker rate. The Marker rate value is updated every three seconds, only during live mode operation (e.g., not during error data set file playback). The frequency measurement of a marker is very rough.

Example :PLUGin:GET:STRing? "EAPLUGIN:ESTatistics:MRATe?"

ESTatistics:NBRCount?

This query-only command returns the number of non-burst related errors. It counts only the actual errors in the non-burst related error events.

In order to be counted as a non-burst related error, the errors must occur inside bursts whose lengths are less than the minimum burst length. For example: If there are occurrences of bursts of length seven, but there are only three errors inside the burst, then only three errors will be added to the non-burst related errors, as long as the minimum burst length is greater than seven.

Example :PLUGin:GET:STRing? "EAPLUGIN:ESTatistics:NBRCount?"

ESTatistics:NBRRate?

This query-only command returns the non-burst related error rate. During live analysis or error data set file playback, the error rate measurement shows the error rate for the interval of the last integration period. When the measurement session is complete, the error rate value is updated to reflect the overall error rate for the entire session.

Example :PLUGin:GET:STRing? "EAPLUGIN:ESTatistics:NBRRate?"

ESTatistics:RECount?

This query-only command returns the number of removed errors. When Error Removal is enabled, errors inside events whose length is less than or equal to the lower limit set in Global Error Removal, or greater than or equal to the upper limit set in Global Error Removal, will be removed from the analysis, and will only be counted here.

Example :PLUGin:GET:STRing? "EAPLUGIN:ESTatistics:RECount?"

ESTatistics:TCOunt?

This query-only command returns the total count (bits or symbols). The unit (bits or symbols) is not reported; verify it by checking if the system is in symbol mode.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:ESTatistics:ENABLe ON”
:PLUGin:GET:STRing? "EAPLUGIN:ESTatistics:ENABLe?"
:PLUGin:GET:STRing? "EAPLUGIN:ESTatistics:TCOunt?"
```

ESTatistics:TECount?

This query-only command returns the total errors (bits or symbols) detected. The unit (bits or symbols) is not reported; verify it by checking if the system is in symbol mode.

Example

```
:PLUGin:GET:STRing? "EAPLUGIN:ESTatistics:TECount?"
```

ESTatistics:TERate?

This query-only command returns the Total error rate. During live analysis or error data set playback, the error rate measurement shows the error rate for the interval of the last integration period. When the measurement session is complete, the error rate value is updated to reflect the overall error rate for the entire session.

Example

```
:PLUGin:GET:STRing? "EAPLUGIN:ESTatistics:TERate?"
```

FDIRectories: File Directories Commands

The FDIRectories commands control the Error Signature and Error Data Folders.

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
FDIRectories			
:EDSFile	<string>		Error Data Folder
:EDSFile?		string	
:ESFile	<string>		Error Signature Folder
:ESFile?		string	

FDIRectories:EDSFile <string> FDIRectories:EDSFile?

This command sets or retrieves the location of the Error Data Folder for both record and playback.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:FDIRectories:EDSFile
""C:\ErrorDataFolder"""
```

```
:PLUGin:GET:STRing? "EAPLUGIN:FDIRectories:EDSFile?"
```

FDIRectories:ESFile <string> FDIRectories:ESFile?

This command sets or retrieves the location of the Error Signature Folder for all histogram views.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:FDIRectories:ESFile
""C:\ErrorSignatureFolder"""
```

```
:PLUGin:GET:STRing? "EAPLUGIN:FDIRectories:ESFile?"
```

GPRoperties Subsystem

The GPRoperties commands are used to set or query the selections for the Global Properties.

The GPRoperties Subsystem

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
GPRoperties			
:BEFThreshold	<numeric>		Burst error free threshold
:BEFThreshold?		numeric	
:BPSYmbol	<numeric>		Bits Per Symbol
:BPSYmbol?		numeric	
:BSGate	0 | 1 | OFF | ON		Separate burst errors by Gate signal
:BSGate?		Boolean	
:BSIPeriod	0 | 1 | OFF | ON		Separate burst errors by Integration Period
:BSIPeriod?		Boolean	
:BSMarker	0 | 1 | OFF | ON		Separate burst errors by Marker signal
:BSMarker?		Boolean	
:CTTRate	<numeric>		User data rate

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
:CTTRate?		numeric	
:DNAGating	0 | 1 | OFF | ON		Do not analyze during Gating interval
:DNAGating?		Boolean	
:DNAResync	0 | 1 | OFF | ON		Do not analyze during resynchronization
:DNAResync?		Boolean	
:ERGThreshold	<numeric>		Greater-than threshold
:ERGThreshold?		numeric	
:ERLThreshold	<numeric>		Less-than threshold
:ERLThreshold?		numeric	
:ERMode	GREater | LESS | NONE | RANGe		Error Removal Mode
:ERMode?		GREater | LESS | NONE | RANGe	
:IMENable	0 | 1 | OFF | ON		Invert Marker Signal
:IMENable?		Boolean	
:IPeriod	1E8 | 1E9 | 1E10 | 1E11 | 1E12 | 1E13 | 1E14		Number of bits in Integration Period
:IPeriod?		1E8 | 1E9 | 1E10 | 1E11 | 1E12 | 1E13 | 1E14	
:IPMode	GATing | QUANtity | MARKer		Integration Period Mode
:IPMode?		GATing | QUANtity | MARKer	

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
:MBLength	<numeric>		Minimum burst length
:MBLength?		numeric	
:SMENable?		Boolean	
:SMODe	GATing | MARKer | NONE | PATTern		Set Synchronize Start Mode
:SMODe?		GATing | MARKer | NONE | PATTern	
:TTMode	USER | CLOCk		Global Timebase Transformation Mode
:TTMode?		USER | CLOCk	

GPRoperties:BEFThreshold <numeric> GPRoperties:BEFThreshold?

This command sets or retrieves the burst error free threshold, the allowable error free distance between errors inside an error event.

If the distance between two errors is too great, the two errors represent independent errors and so they would not be grouped together. If the distance is small, the errors may have to do with the same “event,” depending on how close together they are. If the two errors are separated by at least the error free threshold's number of bits or symbols, then they are classified in separate events. When two errors are separated by less than the error free threshold, then they are associated as part of the same event.

The Error Free Threshold range is from one to 100,000. The same range check as is used in the GUI applies here.

Example

```
:PLUG:PUT:STRing ":EAPLUGIN:GPRoperties:BEFThreshold 2"
:PLUG:GET:STRing? ":EAPLUGIN:GPRoperties:BEFThreshold?"
```

GPRoperties:BPSYmbol <numeric> GPRoperties:BPSYmbol?

This command sets or retrieves the number of bits per symbol. The range is two to 12. The same range check as is used in the GUI applies here.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:GPRoperties:BPSYmbol  2"
:PLUGin:GET:STRing? "EAPLUGIN:GPRoperties:BPSYMbol?"
```

GPRoperties:BSGate 0 | 1 | OFF | ON GPRoperties:BSGate?

This command sets or retrieves the Separate Burst Errors by Gate Signal selection.

When two errors are separated by a Gating signal, then the two errors cannot be part of the same error event. Multiple methods for separating bursts can be simultaneously selected.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:GPRoperties:BSGate ON"
:PLUGin:GET:STRing? "EAPLUGIN:GPRoperties:BSGate?"
```

GPRoperties:BSIPeriod 0 | 1 | OFF | ON GPRoperties:BSIPeriod?

This command sets or retrieves the Separate Burst Errors by Integration Period selection.

When two errors are separated by the boundary of an integration period, then the two errors cannot be part of the same error event. Multiple methods for separating bursts can be simultaneously selected.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:GPRoperties:BSIPeriod ON"
:PLUGin:GET:STRing? "EAPLUGIN:GPRoperties:BSIPeriod?"
```

GPRoperties:BSMarker 0 | 1 | OFF | ON GPRoperties:BSMarker?

This command sets or retrieves the Separate Burst Errors by Marker Signal selection.

When two errors are separated by a Marker signal, then the two errors cannot be part of the same error event. Multiple methods for separating bursts can be simultaneously selected.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:GPRoperties:BSMarker ON"
:PLUGin:GET:STRing? "EAPLUGIN:GPRoperties:BSMarker?"
```

GPRoperties:CTTRate <numeric> GPRoperties:CTTRate?

This command sets or retrieves the user data rate. This reflects the numeric entry of “Serial Bits/Sec”. The range is from one to 1E16 bits. The same range check as used in the GUI applies here.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:GPRoperties:CTTRate 1E9"
:PLUGin:GET:STRing? "EAPLUGIN:GPRoperties:CTTRate?"
```

GPRoperties:DNAGating 0 | 1 | OFF | ON GPRoperties:DNAGating?

This command enables/disables analysis during Gating intervals. If enabled, bits that occur during the Gated period are omitted from the total bit count.

When the analyzer does not count bits during the Gating interval, errors that occur at the end of one Gating period “appear” right next to errors at the beginning of the next Gating period in the analysis. This is independent of the length of the disabling Gate input. If analysis is desired where these errors are truly separated by the Gating interval, then the analyzer should be set to analyze during the Gating interval. In cases where Gated intervals are included in the analysis, the bit error rate will be deflated, as these added bits will contribute to the bit error rate denominator.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:GPRoperties:DNAGating OFF"
:PLUGin:GET:STRing? "EAPLUGIN:GPRoperties:DNAGating?"
```

GPRoperties:DNAResync 0 | 1 | OFF | ON GPRoperties:DNAResync?

This command enables/disables analysis during Resynchronization. If enabled, bits that occur during resynchronization are omitted from the total bit count.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:GPRoperties:DNAResync OFF"
:PLUGin:GET:STRing? "EAPLUGIN:GPRoperties:DNAResync?"
```

GPRoperties:ERGThreshold <numeric>
GPRoperties:ERGThreshold?

This command sets or retrieves the greater-than threshold. This reflects the numeric entry under the “Burst Error Length is greater than” checkbox. The range is from two to 100,000. The same range check as is used in the GUI applies here.

Example

```
:PLUGin:PUT:STRing ":EAPLUGIN:GPRoperties:ERGThreshold 2"
:PLUGin:GET:STRing? ":EAPLUGIN:GPRoperties:ERGThreshold?"
```

GPRoperties:ERLThreshold <numeric>
GPRoperties:ERLThreshold?

This command sets or retrieves the less-than threshold. This reflects the numeric entry under the “Burst Error Length is less than” checkbox. The range is from two to 100,000. The same range check as is used in the GUI applies here.

Example

```
:PLUGin:PUT:STRing ":EAPLUGIN:GPRoperties:ERLThreshold 2"
:PLUGin:GET:STRing? ":EAPLUGIN:GPRoperties:ERLThreshold?"
```

GPRoperties:ERMode GREater | LESS | NONE | RANGe
GPRoperties:ERMode?

This command sets or retrieves the Error Removal mode. The choices are GREATER, LESS, NONE, or RANGE. These choices do not directly reflect the GUI but reflect the server’s expression of the mode. GREATER means that “Burst Error Length is greater than” is selected. LESS means “Burst Error Length is less than” is selected. RANGE means both of these are selected, and NONE means that neither is selected.

Example

```
:PLUGin:PUT:STRing ":EAPLUGIN:GPRoperties:ERMode RANGE"
:PLUGin:GET:STRing? ":EAPLUGIN:GPRoperties:ERMode?"
```

GPRoperties:IMENable 0 | 1 | OFF | ON
GPRoperties:IMENable?

This command sets or retrieves the selection for invert Marker signal. The Marker is defined on the rising edge of the TTL-level Marker signal. Use this command to change to the falling edge.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:GPRoperties:IMENable ON"
```

:PLUGin:GET:STRing? "EAPLUGIN:GPRoperties:IMENable?"

GPRoperties:IPeriod 1E8 | 1E9 | 1E10 | 1E11 | 1E12 | 1E13 | 1E14 GPRoperties:IPeriod?

This command sets or retrieves the number of bits per Integration Period when IPMode (see above) is set to QUANtity. Bits or symbols are continually counted as they enter the analyzer. When the user-specified number of bits/symbols boundary is reached, the rate measurements are updated by dividing the errors in each category by the user-specified number of bits/symbols.

Choices range from 1E8 to 1E14 bits. The same range check as is used in the GUI applies here.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:GPRoperties:IPMode QUANTITY"
:PLUGin:GET:STRing? "EAPLUGIN:GPRoperties:IPMode?"
:PLUGin:PUT:STRing "EAPLUGIN:GPRoperties:IPeriod 1E9"
:PLUGin:GET:STRing? "EAPLUGIN:GPRoperties:IPeriod?"
```

GPRoperties:IPMode GATing | QUANtity | MARKer GPRoperties:IPMode?

This command sets or retrieves the definition of the boundary for an Integration Period. GATing selects “Gating Signal” as the boundary, QUANtity selects a “User Specified Quantity” of bits, and MARKer selects “Marker Signal.”

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:GPRoperties:IPMode MARKER"
:PLUGin:GET:STRing? "EAPLUGIN:GPRoperties:IPMode?"
```

GPRoperties:MBLength <numeric> GPRoperties:MBLength?

This command sets or retrieves the minimum burst length. The length of an error grouping must exceed the Minimum Burst Length before it will be counted in the burst error category. If it does not, it will be classified in the non-burst error category.

The Minimum Burst Length range is from two to 100,000. The same range check as is used in the GUI applies here.

Example

```
:PLUGin:PUT:STRing ":EAPLUGIN:GPRoperties:MBLength 2"
:PLUG:GET:STRing? ":EAPLUGIN:GPRoperties:MBLength?"
```

GPRoperties:SMENable?

This query-only command returns the Symbol Mode selection (symbol mode or not). If Symbol Mode is selected, error statistics are maintained on an N-bit symbol basis. The “Bits Per Symbol” must be set (see BPSYmbol, below).

If Symbol Mode is not set, all error statistics are maintained on a bit-by-bit basis.

Example :PLUGin:GET:STRing? "EAPLUGIN:GPRoperties:SMENable?"

GPRoperties:SMODe GATing | MARKer | NONE | PATTern GPRoperties:SMODe?

This command is used to choose the synchronize start mode, which allows error analysis to be synchronized to start after specific triggering events. Using this mechanism, the error analysis is armed and waits for the occurrence of an external Marker Signal rising edge, external Gating Signal rising edge, or internal PRBS pattern period boundary. If NONE is set, error analysis will begin immediately when the analyzer is instructed to “Start Analysis.”

The choices are GATING, MARKER, NONE, or PATTERN.

Example :PLUGin:PUT:STRing "EAPLUGIN:GPRoperties:SMODe PATTERN"

:PLUGin:GET:STRing? "EAPLUGIN:GPRoperties:SMODe?"

GPRoperties:TTMode USER | CLOCk GPRoperties:TTMode?

This command sets or retrieves the Global Timebase transformation mode. The choices are USER or CLOCK.

In order to perform a bit-to-time transformation, the data rate needs to be known. You can manually enter a data rate (USER), or select the current measured error detector data rate (CLOCK) as the source for time transformations. For the transformation to correspond to actual real-time seconds, the data rate and timebase must be equal.

Example :PLUGin:PUT:STRing "EAPLUGIN:GPRoperties:TTMode CLOCK"

:PLUGin:GET:STRing? "EAPLUGIN:GPRoperties:TTMode?"

PSENsitivity Subsystem

The PSENsitivity commands control the Pattern Sensitivity analyzer.

The PSENsitivity Subsystem

KEYWORD	**PARAMETER FORM**	**RESPONSE FORM**	**COMMENTS**
PSENsitivity			
:AREA?			Area defined by X-cursors
:BCOunt?		numeric	Bin Count
:BRESolution?			Query bin resolution
:CEXTents	<Xmin>, <Xmax>, <Ymin>, <Ymax>		Chart Extents
:CEXTents?		Xmin, Xmax, Ymin, Ymax	
:CRANge	<value1>, <value2>		Chart Range
:CRANge?		value1, value2	
:DATA?		Binary	All data in bins
:ENABle	0 | 1 | OFF | ON		
:ENABle?		Boolean	Enable/disable Pattern Sensitivity Analyzer

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
:ESFClose			Close Error Signature File
:ESFOpen	<string>		Open Error Signature File
:ESFSave	<string>		Save Error Signature
:GTBase	0 | 1 | OFF | ON		
:GTBase?		Boolean	Global Timebase
:LSCale	0 | 1 | OFF | ON		
:LSCale?		Boolean	Log Scale
:PLENgth?		number, no unit	Pattern Size
:PTYPe?			Recognized Pattern
:XCData?		value1, value2	Bin Data
:XCENable	0 | 1 | OFF | ON		X-cursor visibility
:XCENable?		Boolean	
:XCNPeak	<index>		Set X1 cursor to n'th highest bin
:YCENable	0 | 1 | OFF | ON		Y-cursor visibility
:YCENable?		Boolean	
:YCPosition	<value1>, <value2>		Y-cursor position
:YCPosition?		value1, value2	

PSENsitivity:AREA?

This query-only command returns the area defined by Cursor X1 and Cursor X2. The response is the same as the Cursor Area displayed under the view.

PSENsitivity:BCOunt?

This query-only command gets the bin count stored in the server. There is no counterpart in the GUI.

Example :PLUGin:GET:STRing? "EAPLUGIN:PSENsitivity:BCOunt?"

PSENsitivity:BRESolution?

This query-only command returns the bin resolution.

Bin resolution is determined by the Chart Range values (see CRANge). The “From” value is inclusive and the “To” value is exclusive. For example, the range can be specified as [0,1000], which establishes 1000 bins, from Bin 0 to Bin 999. The bin resolution is 1. Bin 0 contains data [–inf, 1], and Bin 999 contains [999, +inf].

When the range is greater than or equal to 1001, the bin resolution will be 2 or more, as necessary to include all data within the available bins.

Example :PLUGin:GET:STRing? "EAPLUGIN:PSENsitivity:BRESolution?"

PSENsitivity:CEXTents <Xmin>, <Xmax>, <Ymin>, <Ymax> PSENsitivity:CEXTents?

This command sets or retrieves the boundaries of the view, from the minimum to maximum X-values, and minimum to maximum Y-values.

If the Chart Range is changed (using the CRANge command, below), a “Fit View” is performed in an attempt to fit the Chart Extents of the view to the defined interval. Hence, the Chart Extent values queried after changing the Chart Range will differ from those previously set. Use the query command CEXTents? to discover the new chart extents.

The same range check as is used by the GUI applies here.

Example :PLUGin:PUT:STRing "EAPLUGIN:PSENsitivity:CEXTents 0, 128, 1, 1000"

:PLUGin:GET:STRing? "EAPLUGIN:PSENsitivity:CEXTents?"

PSENsitivity:CRANge <value1>, <value2>
PSENsitivity:CRANge?

This command sets or retrieves the settings of “Chart Range” in the properties dialog. The Chart Range defines the range over which you want to collect histogram data, which may differ significantly from the data displayed in the view (limited by the Chart Extents).

When the Chart Range is changed, a “Fit View” is performed in an attempt to fit the Chart Extents of the view to the defined interval. Hence, the Chart Extent values queried after changing the Chart Range will differ from those previously set. Use the query command CEXTents? to discover the new chart extents.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:PSENsitivity:CRANge 0, 128"
:PLUGin:GET:STRing? "EAPLUGIN:PSENsitivity:CRANge?"
```

PSENsitivity:DATA?

Query all the data in all the bins. This gives the user all the data in the server’s format, not that of the GUI. It is provided for the user to analyze using information from another source, such as the bin count.

Binary query.

Example

```
:PLUGin:GET:BINary? "EAPLUGIN:PSENsitivity:DATA?"
```

PSENsitivity:ENABle 0 | 1 | OFF | ON
PSENsitivity:ENABle?

This command enables or disables the Pattern Sensitivity analyzer. It is equivalent to the Pattern Sensitivity checkbox in the properties dialog.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:PSENsitivity:ENABle ON"
:PLUGin:GET:STRing? "EAPLUGIN:PSENsitivity:ENABle?"
```

PSENsitivity:ESFClose

This command will close an open Error Signature file. It works just like the Close button in the Error Signature Manager.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:PSENsitivity:ESFClose"
```

PSENsitivity:ESFOpen <string>

This command opens the Error Signature file named in <string>. It works just like the Open button in the Error Signature Manager.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:PSENsitivity:ESFOpen ""Pattern Sensitivity Signature.PSH.csv"""
```

PSENsitivity:ESFSave <string>

This command saves the Error Signature to the file named in the <string>. It works just like the Save button in the Error Signature Manager.

An Error Signature is an ASCII file containing a list of comma-separated values representing error performance characteristics of a particular failure mode. A saved Error Signature can be compared against new results to identify problem types that have occurred before.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:FDIRectories:ESFile ""C:\ELA_GPIB\PSAnalyzer"""
:PLUGin:GET:STRing? "EAPLUGIN:FDIRectories:ESFile?"
:PLUGin:PUT:STRing "EAPLUGIN:PSENsitivity:ESFSave ""Pattern Sensitivity Signature.PSH.csv"""
```

PSENsitivity:GTBase 0 | 1 | OFF | ON
PSENsitivity:GTBase?

This command enables/disables the Global Timebase. Enabling Global Timebase shows the bit rate values in Time along the X-axis. Disabling it shows the bit rate values in Bits along the X-axis.

Time is calculated based on a data rate of bits per second. The data rate can be set or queried using the GPROPerties:CTTRate command.

The GTBase command is equivalent to the Global Timebase checkbox in the properties dialog.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:PSENsitivity:GTBase ON"
:PLUGin:GET:STRing? "EAPLUGIN:PSENsitivity:GTBase?"
```

PSENsitivity:LSCale 0 | 1 | OFF | ON
PSENsitivity:LSCale?

This command enables/disables use of the Log Scale. Enabling it shows the Y-axis of the histogram as a logarithmic scale. Disabling it shows the Y-axis as a linear scale. This command works like the Log Scale checkbox in the properties dialog.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:PSENsitivity:LSCale ON"
:PLUGin:GET:STRing? "EAPLUGIN:PSENsitivity:LSCale?"
```

PSENsitivity:PLENgth?

This query-only command returns the data pattern size (the number of bits or symbols in one cycle of the data pattern) as a number only. The unit (bits or symbols) is as set by the user.

Example

```
:PLUGin:GET:STRing? "EAPLUGIN:PSENsitivity:PLENgth?"
```

PSENsitivity:PTYPe?

This query-only command returns the name of the recognized pattern.

Example

```
:PLUGin:GET:STRing? "EAPLUGIN:PSENsitivity:PTYPe?"
```

PSENsitivity:XCData?

This query-only command gets the data at Cursor X1 and Cursor X2 (smart cursor in the view). The response will be: value1, value2.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:PSENsitivity:CRANge 0, 128"
:PLUGin:GET:STRing? "EAPLUGIN:PSENsitivity:CRANge?"
:PLUGin:PUT:STRing "EAPLUGIN:PSENsitivity:XCENable ON"
:PLUGin:GET:STRing? "EAPLUGIN:PSENsitivity:XCData?"
```

PSENsitivity:XCENable 0 | 1 | OFF | ON
PSENsitivity:XCENable?

This command sets or retrieves the visibility of the X-cursors. It performs the same function as the X-Axis Cursors checkbox in the properties dialog to enable or disable display of Cursors X1 and X2.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:PSENsitivity:XCENable ON"
:PLUGin:GET:STRing? "EAPLUGIN:PSENsitivity:XCENable?"
```

PSENsitivity:XCNPeak <index>

This command sets the X1 cursor to the n'th highest bin.

The last data bin contains data from values outside the range. If this command returns the last bin (and there is data outside the range), it does not mean that the last data bin is actually the greatest.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:PSENsitivity:XCNPeak 1"
:PLUGin:GET:STRing? "EAPLUGIN:PSENsitivity:XCData?"
```

PSENsitivity:YCENable 0 | 1 | OFF | ON
PSENsitivity:YCENable?

This command sets or retrieves the visibility of Cursor Y1. It performs the same function as the Y-Axis Cursor checkbox in the properties dialog.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:PSENsitivity:YCENable ON"
:PLUGin:GET:STRing? "EAPLUGIN:PSENsitivity:YCENable?"
```

PSENsitivity:YCPosition <value1>, <value2>
PSENsitivity:YCPosition?

This command sets or retrieves the position of Cursor Y1 in the view. The response will be: value1.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:PSENsitivity:YCPosition 10"
:PLUGin:GET:STRing? "EAPLUGIN:PSENsitivity:YCPosition?"
```

SCHart Subsystem

The SCHart commands control the Strip Chart analyzer.

Note

Sending remote commands to control the Strip Chart may cause the X-Axis annotations to disappear. To restore the X-Axis annotations, click or touch the **Fit Graph** button on the Strip Chart results window.

The SCHart Subsystem

KEYWORD	PARAMETER FORM	RESPONSE FORM	COMMENTS
SCHart			
:CEXTents	<Xmin>, <Xmax>, <Ymin>, <Ymax>		Chart Extents
:CEXTents?		Xmin, Xmax, Ymin, Ymax	
:ENABle	0 | 1 | OFF | ON		Enable/disable the Strip Chart
:GISince?	<time>, <max>	Binary	Get items since <time>
:GTBase	0 | 1 | OFF | ON		Global Timebase
:GTBase?		Boolean	
:LTIMe?			Latest time
:XCENable	0 | 1 | OFF | ON		X-cursor visibility
:XCENable?		Boolean	
:YCENable	0 | 1 | OFF | ON		Y-cursor visibility
:YCENable?		Boolean	
:YCPosition	<value>		Y-cursor position
:YCPosition?		value	

SCHart:CEXTents <Xmin>, <Xmax>, <Ymin>, <Ymax>
SCHart:CEXTents?

This command sets or retrieves the boundaries of the view, from the minimum to maximum X-values, and minimum to maximum Y-values.

If the Chart Range is changed, a “Fit View” is performed in an attempt to fit the Chart Extents of the view to the defined interval. Hence, the Chart Extent values queried after changing the Chart Range will differ from those previously set. Use the query command CEXTents? to discover the new chart extents. The same range check as is used by the GUI applies here.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:SCHart:CEXTents 10, 100, 1, 1000000"
:PLUGin:GET:STRing? "EAPLUGIN:SCHart:CEXTents?"
```

SCHart:ENABle 0 | 1 | OFF | ON
SCHart:ENABle?

This command enables/disables the Strip Chart analyzer. It is equivalent to the Strip Chart checkbox in the properties dialog.

Example

```
:PLUGin:PUT:STRing "EAPLUGIN:SCHart:ENABle ON"
:PLUGin:GET:STRing? "EAPLUGIN:SCHart:ENABle?"
```

SCHart:GISince? <time>, <max>

This query-only command is used to retrieve all the strip chart data since the specified measurement point until the current measurement point. <max> specifies the maximum number of measurement points you intend to receive. Binary query only.

Note

Each measurement point (line segment between tick marks) on the strip chart represents the BER averaged over a specific number of bits. This number of bits is called the integration period, and it affects the total number of points plotted.

The data returned includes bit number, total BER, burst-related BER, and non-burst-related BER. The bit number allows you to associate data with measurement points on the strip chart. The bit number is the number of the last bit in the integration period for a measurement point. Time values will not be returned, even if the global timebase is used.

Example :PLUGin:GET:BINary? "EAPLUGIN:SCHart:GISince? 1, 3"

SCHart:GTBase 0 | 1 | OFF | ON
SCHart:GTBase?

This command enables/disables the Global Timebase. Enabling Global Timebase shows the bit rate values in Time along the X-axis. Disabling it shows the bit rate values in Bits along the X-axis.

Time is calculated based on a data rate of bits per second. The data rate can be set or queried using the GPROPerties:CTTRate command.

The GTBase command is equivalent to the Global Timebase checkbox in the properties dialog.

Example :PLUGin:PUT:STRing "EAPLUGIN:SCHart:GTBase ON"

:PLUGin:GET:STRing? "EAPLUGIN:SCHart:GTBase?"

SCHart:LTIMe?

This query-only command returns the latest time.

Example :PLUGin:GET:STRing? "EAPLUGIN:SCHart:LTIMe?"

SCHart:XCENable 0 | 1 | OFF | ON
SCHart:XCENable?

This command sets or retrieves the visibility of the X-cursor (there is only one X-cursor for this analysis). It performs the same function as the X-Axis Cursor check-box in the properties dialog.

Example

:PLUGin:PUT:STRing "EAPLUGIN:SCHart:XCENable ON"

:PLUGin:GET:STRing? "EAPLUGIN:SCHart:XCENable?"

SCHart:YCENable 0 | 1 | OFF | ON
SCHart:YCENable?

This command sets or retrieves the visibility of the Y-cursor. It performs the same function as the Y-Axis CursoY-Axis Cursor checkbox in the properties dialog

Example

:PLUGin:PUT:STRing "EAPLUGIN:SCHart:YCENable ON"

:PLUGin:GET:STRing? "EAPLUGIN:SCHart:YCENable?"

SCHart:YCPosition <value1>
SCHart:YCPosition?

This command sets or retrieves the Strip Chart's Y-cursor position. There is only one Y-cursor for this analysis. The response will be: value1.

Example

:PLUGin:PUT:STRing "EAPLUGIN:SCHart:YCPosition 0.0001"

:PLUGin:GET:STRing? "EAPLUGIN:SCHart:YCPosition?"

Front Panel Functions to Error Analysis Remote Commands

This is a table of the front-panel functions of 86130A Error Analysis and the corresponding remote commands.

Table 5-1. Front Panel Function to Remote Command for the Agilent 86130A—Error Analysis

Front Panel Function	**Remote Command**
Burst Length Analyzer:	
Data at smart cursor	BLENgth:XCData?
No counterpart	BLENgth:BCOunt?
Cursor Area display	BLENgth:AREA?
No counterpart	BLENgth:DATA?
Chart Extents display	BLENgth:CEXTents?
Set Chart Extents	BLENgth:CEXTents <numeric>, <numeric>, <numeric>, <numeric>
Chart Range display	BLENgth:CRANge?
Set Chart Ranges in Properties	BLENgth:CRANge <numeric>, <numeric>
State of X-Axis cursor checkbox (enabled/disabled)	BLENgth:XCENable?
X-Axis Cursors checkbox	BLENgth:XCENable <bool>
Set X1 cursor to the n’th highest bin	BLENgth:XCNPeak <index>
Bin Resolution display	BLENgth:BRESolution?
State of Y-axis cursor checkbox (enabled/disabled)	BLENgth:YCENable?

Table 5-1. Front Panel Function to Remote Command for the Agilent 86130A—Error Analysis

Front Panel Function	**Remote Command**
Y-axis Cursor checkbox	BLENgth:YCENable <bool>
Find location of Y-axis cursor on view	BLENgth:YCPosition?
Position Y-axis cursor on view	BLENgth:YCPosition <numeric>
State of analyzer checkbox (enabled/disabled)	BLENgth:ENAble?
Enable Burst Length analysis using checkbox in Analyzer Control	BLENgth:ENAble <bool>
State of Log Scale checkbox (enabled/disabled)	BLENgth:LSCale?
Log Scale checkbox in Properties	BLENgth:LSCale <bool>
State of Global Timebase checkbox (enabled/disabled)	BLENgth:GTBase?
Global Timebase checkbox in Properties	BLENgth:GTBase <bool>
Save Error Signature file named in <string>	BLENgth:ESFSave <string>
Open Error Signature file named in <string>	BLENgth:ESFOpen <string>
Close Error Signature file	BLENgth:ESFClose
Error Statistics Analyzer:	
Total Count display	ESTatistics:TCOunt?
Total Error Count display	ESTatistics:TECount?
Non Burst-Related Error Count display	ESTatistics:NBRCount?
Burst-Related Error Count display	ESTatistics:BRCount?

Table 5-1. Front Panel Function to Remote Command for the Agilent 86130A—Error Analysis

Front Panel Function	Remote Command
Marker Count display	ESTatistics:MCOunt?
Removed Errors Count display	ESTatistics:RECount?
Total Error Rate display	ESTatistics:TERate?
Non Burst-Related Error Rate display	ESTatistics:NBRRate?
Burst-Related Error Rate display	ESTatistics:BRRate?
Marker Rate display	ESTatistics:MRATe?
State of analyzer enable checkbox (enabled/disabled)	ESTatistics:ENABle?
Enable Error Statistics analysis using checkbox in Analyzer Control	ESTatistics:ENABle <bool>
Analysis Control:	
Marker Rate display on Analyzer Control	ACONtrol:MRATe?
Event Rate display on Analyzer Control	ACONtrol:ERATe?
Squelch Events display on Analyzer Control	ACONtrol:SEVents?
Global Properties:	
Current selection for Timebase Transformation Mode	GPRoperties:TTMode?
Select Timebase Transformation Mode	GPRoperties:TTMode USER | CLOCK
Current value for User Rate	GPRoperties:CTTRate?
Enter value for User Rate	GPRoperties:CTTRate <numeric>

Table 5-1. Front Panel Function to Remote Command for the Agilent 86130A—Error Analysis

Front Panel Function	**Remote Command**
State of Do Not Analyze During Gating Interval checkbox (enabled/disabled)	GPRoperties:DNAGating?
Do Not Analyze During Gating Interval checkbox	GPRoperties:DNAGating <bool>
State of Do Not Analyze During Resynchronization checkbox (enabled/disabled)	GPRoperties:DNAResync?
Do Not Analyze During Resynchronization checkbox	GPRoperties:DNAResync <bool>
Current value of Minimum Burst Length	GPRoperties:MBLength?
Enter value for Minimum Burst Length	GPRoperties:MBLength <numeric>
Current value of Burst Error Free Threshold	GPRoperties:BEFThreshold?
Enter value for Burst Error Free Threshold	GPRoperties:BEFThreshold <numeric>
State of Separate Burst Errors by Gate Signal checkbox (enabled/disabled)	GPRoperties:BSGate?
Separate Burst Errors by Gate Signal checkbox	GPRoperties:BSGate <bool>
State of Separate Burst Errors by Integration Period checkbox (enabled/disabled)	GPRoperties:BSIPeriod?
Separate Burst Errors by Gate Signal checkbox	GPRoperties:BSIPeriod <bool>
State of Separate Burst Errors by Marker Signal checkbox (enabled/disabled)	GPRoperties:BSMarker?

Table 5-1. Front Panel Function to Remote Command for the Agilent 86130A—Error Analysis

Front Panel Function	Remote Command
Separate Burst Errors by Gate Signal checkbox	GPRoperties:BSMarker <bool>
Error Removal Mode	GPRoperties:ERMode?
Select Error Removal Mode	GPRoperties:ERMode GREater | LESS | NONE | RANGe
Greater-than threshold for Error Removal Filter	GPRoperties:ERGThreshold?
Set greater-than threshold for Error Removal Filter	GPRoperties:ERGThreshold <numeric>
Less-than threshold for Error Removal Filter	GPRoperties:ERLThreshold
Set less-than threshold for Error Removal Filter	GPRoperties:ERLThreshold <numeric>
Mode of boundary definition for Integration Period	GPRoperties:IPMode?
Select mode of boundary definition for Integration Period	GPRoperties:IPMode GATing | QUANtity | MARKer
Number of bits in Integration Period if QUANtity is selected	GPRoperties:IPeriod?
Select number of bits in Integration Period if QUANtity is selected	GPRoperties:IPeriod 1E8 | 1E9 | 1E10 | 1E11 | 1E12 | 1E13 | 1E14
State of Symbol Mode (enabled/disabled)	GPRoperties:SMENable?
Bits Per Symbol display	GPRoperties:BPSYmbol?
Enter Bits Per Symbol	GPRoperties:BPSYmbol <numeric>
Selection for Synchronize Start Mode	GPRoperties:SMODe?
Select Synchronize Start Mode	GPRoperties:SMODe GATing | MARKer | NONE | PATTern

Table 5-1. Front Panel Function to Remote Command for the Agilent 86130A—Error Analysis

Front Panel Function	**Remote Command**
State of Invert Marker Signal checkbox (enabled/disabled)	GPRoperties:IMENable?
Invert Marker Signal Checkbox	GPRoperties:IMENable <bool>
FILE: Error Data Set of UER File:	
Start UER playback, opening the file specified in the UER file name, with option to restore the associated properties.	EDSFile:PLAYback:BEGin [PROPerties | NOPRoperties]
Stop UER playback	EDSFile:PLAYback:END
Playback progress indicator	EDSFile:PLAYback:PROGress?
UER Playback File Name display	EDSFile:PLAYback:FNAMe?
Name (rename) the UER Playback file	EDSFile:PLAYback:FNAMe <string>
Save Properties button. Save the current properties associated with the current UER playback file.	EDSFile:PLAYback:SPRoperties
State of OK to Overwrite checkbox (enabled/disabled)	EDSFile:RECord:OWRite?
OK to Overwrite checkbox	EDSFile:RECord:OWRite <bool>
State of Record Properties checkbox (enabled/disabled) in General Settings	EDSFile:RECord:PROPerties?
Record Properties checkbox in General Settings	EDSFile:RECord:PROPerties <bool>
Record File Name	EDSFile:RECord:FName?
Set Record File Name	EDSFile:RECord:FName <string>
File Limit	EDSFile:RECord:FLIMit?

Table 5-1. Front Panel Function to Remote Command for the Agilent 86130A—Error Analysis

Front Panel Function	**Remote Command**
Select File Limit	EDSFile:RECord:FLIMit 10KB | 50 KB | 100KB | 500KB | 1MB | 5MB | NOLimit
Error Signature Folder	FDIRectories:ESFile?
Set Error Signature Folder	FDIRectories:ESFile <string>
Error Data Folder	FDIRectories:EDSFile?
Set Error Data Folder	FDIRectories:EDSFile <string>
System:	
Status of Error Analysis display at right end of Status Bar	ASTatus?
Block Error Analysis:	
Data at smart cursor	BERRors:XCData?
No counterpart	BERRors:BCOunt?
Cursor Area display	BERRors:AREA?
No counterpart	BERRors:DATA?
Chart Extents display	BERRors:CEXTents?
Set Chart Extents	BERRors:CEXTents <numeric>, <numeric>, <numeric>, <numeric>
Chart Range display	BERRors:CRANge?
Set Chart Ranges in Properties	BERRors:CRANge <numeric>, <numeric>
State of X-Axis Cursor checkbox (enabled/disabled)	BERRors:XCENable?
X-Axis Cursors checkbox	BERRors:XCENable <bool>
Set X1 cursor to the n'th highest bin	BERRors:XCNPeak <index>
Bin Resolution display	BERRors:BRESolution?
State of Y-Axis Cursor checkbox (enabled/disabled)	BERRors:YCENable?

Table 5-1. Front Panel Function to Remote Command for the Agilent 86130A—Error Analysis

Front Panel Function	**Remote Command**
Y-Axis Cursor checkbox	BERRors:YCENable <bool>
Find location of Y-axis cursor	BERRors:YCPosition?
Position Y-axis cursor on view	BERRors:YCPosition <numeric>
State of analyzer checkbox (enabled/disabled)	BERRors:ENAble?
Enable Block Errors analysis using checkbox in Analyzer Control	BERRors:ENAble <bool>
State of Log Scale checkbox (enabled/disabled)	BERRors:LSCale?
Log Scale checkbox in Properties	BERRors:LSCale <bool>
Save Error Signature File named in <string>	BERRors:ESFSave <string>
Open Error Signature File named in <string>	BERRors:ESFOpen <string>
Close Error Signature File	BERRors:ESFClose
Correlation Analysis:	
Data at smart cursor	CORRelation:XCData?
No counterpart	CORRelation:BCOunt?
Cursor Area display	CORRelation:AREA?
No counterpart	CORRelation:DATA?
Chart Extents display	CORRelation:CEXTents?
Set Chart Extents	CORRelation:CEXTents <numeric>, <numeric>, <numeric>, <numeric>
Chart Range display	CORRelation:CRANge?
Set Chart Ranges in Properties	CORRelation:CRANge <numeric>, <numeric>
State of X-Axis Cursor checkbox (enabled/disabled)	CORRelation:XCENable?

Table 5-1. Front Panel Function to Remote Command for the Agilent 86130A—Error Analysis

Front Panel Function	**Remote Command**
Enable X-Axis Cursor checkbox	CORRelation:XCENable <bool>
Set X1 cursor to the n'th highest bin	CORRelation:XCNPeak <index>
Bin Resolution display	CORRelation:BRESolution?
State of Y-Axis Cursor checkbox (enabled/disabled)	CORRelation:YCENable?
Y-Axis Cursor checkbox	CORRelation:YCENable <bool>
Find location of Y-axis cursor	CORRelation:YCPosition?
Position Y-axis cursor on view	CORRelation:YCPosition <numeric>
State of analyzer checkbox (enabled/disabled)	CORRelation:ENAble?
Enable Correlation analysis using checkbox in Analyzer Control	CORRelation:ENAble <bool>
State of Log Scale checkbox (enabled/disabled)	CORRelation:LSCale?
Log Scale checkbox in Properties	CORRelation:LSCale <bool>
State of Global Timebase checkbox (enabled/disabled)	CORRelation:GTBase?
Global Timebase checkbox in Properties	CORRelation:GTBase <bool>
Save Error Signature File named in <string>	CORRelation:ESFSave <string>
Open Error Signature File named in <string>	CORRelation:ESFOpen <string>
Close Error Signature File	CORRelation:ESFClose

Table 5-1. Front Panel Function to Remote Command for the Agilent 86130A—Error Analysis

Front Panel Function	**Remote Command**
Selected Correlation mode as the definition for the boundary of a Correlation Period	CORRelation:BLMode?
Select Correlation mode to be used as the boundary for a Correlation Period	CORRelation:BLMode GATing | MARKer | PATTern | QUANtity
Number of bits specified if QUANtity is selected as Correlation Period mode	CORRelation:BLQuantity?
Enter number of bits to be used if QUANtity is selected as Correlation Period mode	CORRelation:BLQuantity <numeric>
Error Free Interval Analysis:	
Data at smart cursor	EFINterval:XCData?
No counterpart	EFINterval:BCOunt?
Cursor Area display	EFINterval:AREA?
No counterpart	EFINterval:DATA?
Chart Extents display	EFINterval:CEXTents?
Set Chart Extents	EFINterval:CEXTents <numeric>, <numeric>, <numeric>, <numeric>
Chart Range display	EFINterval:CRANge?
Set Chart Ranges in Properties	EFINterval:CRANge <numeric>, <numeric>
State of X-Axis Cursor checkbox (enabled/disabled)	EFINterval:XCENable?
X-Axis Cursors checkbox	EFINterval:XCENable <bool>
Set X1 cursor to the n’th highest bin	EFINterval:XCNPeak <index>
Bin Resolution display	EFINterval:BRESolution?
State of Y-Axis Cursor checkbox (enabled/disabled)	EFINterval:YCENable?

Table 5-1. Front Panel Function to Remote Command for the Agilent 86130A—Error Analysis

Front Panel Function	**Remote Command**
Y-Axis Cursor checkbox	EFINterval:YCENable <bool>
Find location of Y-axis cursor on view	EFINterval:YCPosition?
Position Y-axis cursor on view on view	EFINterval:YCPosition <numeric>
State of analyzer checkbox (enabled/disabled)	EFINterval:ENAble?
Enable Error Free Interval analysis using checkbox in Analyzer Control	EFINterval:ENAble <bool>
State of Log Scale checkbox (enabled/disabled)	EFINterval:LSCale?
Log Scale checkbox in Properties	EFINterval:LSCale <bool>
State of Global Timebase checkbox (enabled/disabled)	EFINterval:GTBase?
Global Timebase checkbox in Properties	EFINterval:GTBase <bool>
Save Error Signature File named in <string>	EFINterval:ESFSave <string>
Open Error Signature File named in <string>	EFINterval:ESFOpen <string>
Close Error Signature File	EFINterval:ESFClose
Pattern Sensitivity Analysis:	
Data at smart cursor in view	PSENsitivity:XCData?
No counterpart	PSENsitivity:BCOunt?
Cursor Area display in view	PSENsitivity:AREA?
No counterpart	PSENsitivity:DATA?

Table 5-1. Front Panel Function to Remote Command for the Agilent 86130A—Error Analysis

Front Panel Function	**Remote Command**
Chart Extents display on Properties page	PSENsitivity:CEXTents?
Set Chart Extents on Properties page	PSENsitivity:CEXTents <numeric>, <numeric>, <numeric>, <numeric>
Chart Range display on Properties page	PSENsitivity:CRANge?
Set Chart Ranges on Properties page	PSENsitivity:CRANge <numeric>, <numeric>
State of X-Axis Cursors checkbox (enabled/disabled) on Properties page	PSENsitivity:XCENable?
X-Axis Cursors checkbox on Properties page	PSENsitivity:XCENable <bool>
Set X1 cursor to the n’th highest bin	PSENsitivity:XCNPeak <index>
Bin Resolution display	PSENsitivity:BRESolution?
State of Y-Axis cursor checkbox (enabled/disabled)	PSENsitivity:YCENable?
Y-Axis Cursor checkbox	PSENsitivity:YCENable <bool>
Find location of Y-axis cursor on view	PSENsitivity:YCPosition?
Position Y-axis cursor on view	PSENsitivity:YCPosition <numeric>
State of analyzer enable checkbox (enabled/disabled) on Analyzer Control page	PSENsitivity:ENAble?
Enable Pattern Sensitivity analysis using checkbox in Analyzer Control	PSENsitivity:ENAble <bool>
State of Log Scale checkbox (enabled/disabled) on Properties page	PSENsitivity:LSCale?

Table 5-1. Front Panel Function to Remote Command for the Agilent 86130A—Error Analysis

Front Panel Function	**Remote Command**
Log Scale checkbox on Properties page	PSENsitivity:LSCale <bool>
State of Global Timebase checkbox (enabled/disabled) on Properties page	PSENsitivity:GTBase?
Global Timebase checkbox on Properties page	PSENsitivity:GTBase <bool>
Save Error Signature File named in <string> on Properties page	PSENsitivity:ESFSave <string>
Open Error Signature File named in <string> on Properties page	PSENsitivity:ESFOpen <string>
Close Error Signature File	PSENsitivity:ESFClose
Recognized pattern type	PSENsitivity:PTYPe?
Pattern size (number only)	PSENsitivity:PLENgth?
Strip Chart Analysis:	
Latest time	SCHart:LTIMe?
Get all items since the time specified. Equivalent to viewing the Strip Chart.	SCHart:GISince? <time>, <max>
Chart Extents display	SCHart:CEXTents?
Set Chart Extents	SCHart:CEXTents <numeric>, <numeric>, <numeric>, <numeric>
State of analyzer enable checkbox (enabled/disabled)	SCHart:ENABle?
Enable Strip Chart analysis using checkbox in Analyzer Control	SCHart:ENABle <bool>
State of X-Axis Cursor checkbox (enabled/disabled) on Properties page	SCHart:XCENable?

Table 5-1. Front Panel Function to Remote Command for the Agilent 86130A—Error Analysis

Front Panel Function	**Remote Command**
X-Axis Cursor checkbox on Properties page	SCHart:XCENable <bool>
Set X1 cursor to the n'th highest bin	SCHart:XCNPeak <index>
Bin Resolution display	SCHart:BRESolution?
State of Y-Axis Cursor checkbox (enabled/disabled) on Properties page	SCHart:YCENable?
Y-Axis Cursor checkbox on Properties page	SCHart:YCENable <bool>
Find location of Y-axis cursor on view	SCHart:YCPosition?
Position Y-axis cursor on view	SCHart:YCPosition <numeric>
State of Global Timebase checkbox (enabled/disabled)	SCHart:GTBase?
Global Timebase checkbox	SCHart:GTBase <bool>
ECC Analysis:	
Enable ECC Analysis using checkbox in Analyzer Control	ECC:ENABle 0 | 1 | OFF | ON
State of analyzer enable checkbox (enabled/disabled)	ECC:ENABle?
View display of Current Interval Before ECC Error Count	ECC:IBECount?
View display of Current Interval Before ECC BER	ECC:IBERate?
View display of Current Interval Before ECC bit count	ECC:IBBCount?
View display of Current Interval After ECC Error Count	ECC:IAECount?

Table 5-1. Front Panel Function to Remote Command for the Agilent 86130A—Error Analysis

Front Panel Function	**Remote Command**
View display of Current Interval After ECC BER	ECC:IAERate?
View display of Current Interval After ECC bit count	ECC:IABCount?
View display of Current Interval Inner Code Corrections	ECC:IICCorrections?
View display of Current Interval Inner Code Failures	ECC:IICFailures?
View display of Current Interval Outer Code Corrections	ECC:IOCCorrections?
View display of Current Interval Outer Code Failures	ECC:IOCFailures?
View display of Current Interval Erasure Corrections	ECC:IECorrections?
View display of Current Interval Erasure Failures	ECC:IEFailures?
View display of Total Accumulation Before ECC Error Count	ECC:ABECount?
View display of Total Accumulation Before ECC BER	ECC:ABERate?
View display of Total Accumulation Before ECC bit count	ECC:ABBCount?
View display of Total Accumulation After ECC Error Count	ECC:AAECount?
View display of Total Accumulation After ECC BER	ECC:AAERate?

Table 5-1. Front Panel Function to Remote Command for the Agilent 86130A—Error Analysis

Front Panel Function	**Remote Command**
View display of Total Accumulation After ECC bit count	ECC:AABCount?
View display of Total Accumulation Inner Code Corrections	ECC:AICCorrections?
View display of Total Accumulation Inner Code Failures	ECC:AICFailures?
View display of Total Accumulation Outer Code Corrections	ECC:AOCCorrections?
View display of Total Accumulation Outer Code Failures	ECC:AOCFailures?
View display of Total Accumulation Erasure Corrections	ECC:AECorrections?
View display of Total Accumulation Erasure Failures	ECC:AEFailures?
View display of ECC Overhead	ECC:OVERhead?
View display of Before ECC data rate	ECC:BDRate?
View display of After ECC data rate	ECC:ADRate?
View display of Tables Processed	ECC:TPRocessed?
View display of Tables Overrun	ECC:TOVerrun?
Configuration filename entry box on Properties page	ECC:CONFiguration <filename>

Table 5-1. Front Panel Function to Remote Command for the Agilent 86130A—Error Analysis

Front Panel Function	Remote Command
Configuration selection	ECC:SCSelect G709 | G975 | 2RSCode | M2Fec | RSCode
Reporting Interval entry box on Properties page	ECC:RINTerval <integer>
Display in Reporting Interval box on Properties page	ECC:RINTerval?
Emulation Mode selection on Global ECC Settings page	ECC:EMODe DISabled | INNer | SKIPerasure | FULL
Emulation Mode Display on Global ECC Settings page	ECC:EMODe?
Strip ECC Overhead Codewords checkbox on Global ECC Settings page	ECC:EOSYmbols 0 | 1 | OFF | ON
State of Strip ECC Overhead Codewords checkbox (enabled/disabled) on Global ECC Settings page	ECC:EOSYmbols?
Interleave Mode selection on Global ECC Settings page	ECC:IMODe BIT | SYMBol
Interleave Mode display on Global ECC Settings page	ECC:IMODE?
ECC Symbol Size on Global ECC Settings page	ECC:SSIZe <integer>
ECC Symbol Size display on Global ECC Settings page	ECC:SSIZe?
Inner Code “n” entry on Global ECC Settings page	ECC:NINNer <integer>
Inner Code “n” setting display on Global ECC Settings page	ECC:NINNer?
Inner Code “T” entry on Global ECC Settings page	ECC:TINNer <integer>

Table 5-1. Front Panel Function to Remote Command for the Agilent 86130A—Error Analysis

Front Panel Function	**Remote Command**
Inner Code “T” setting display on Global ECC Settings page	ECC:TINNer?
Inner Code “k” entry on Global ECC Settings page	ECC:KINNer <integer>
Inner Code “k” setting display on Global ECC Settings page	ECC:KINNer?
Outer Code “n” entry on Global ECC Settings page	ECC:NOUTer <integer>
Outer Code “n” setting display on Global ECC Settings page	ECC:NOUTer?
Outer Code “T” entry on Global ECC Settings page	ECC:TOUTer <integer>
Outer Code “T” setting display on Global ECC Settings page	ECC:TOUTer?
Outer Code “k” entry on Global ECC Settings page	ECC:KOUTer <integer>
Outer Code “k” setting display on Global ECC Settings page	ECC:KOUTer?
Erasure Strength entry on Global ECC Settings page	ECC:ERASure <integer>
Erasure Strength setting display on Global ECC Settings page	ECC:ERASure?
Two-Dimensional Table checkbox on Global ECC Settings page	ECC:TDFLag
State of Two-Dimensional Table checkbox (enabled/ disabled) on Global ECC Settings page	ECC:TDFLag?
2-D Error Map:	

Table 5-1. Front Panel Function to Remote Command for the Agilent 86130A—Error Analysis

Front Panel Function	Remote Command
Enable 2-D Error Map analysis using checkbox in Analyzer Control	2DEMap:ENABle 0 | 1 | OFF | ON
State of analyzer enable checkbox (enabled/disabled)	2DEMap:ENABle?
Segment Mode selection on Properties page	2DEMap:MODe QUANtity | SECond | MARKer
Segment Mode display on Properties page	2DEMap:MODe?
Block Length entry on Properties page	2DEMap:BLENgth <integer>
Block Length display on Properties page, used if Segment Mode is selected to be QUANtity	2DEMap:BLENgth?
Segment Time entry on Properties page, used if Segment Mode is selected to be TIME	2DEMap:STIMe <integer>
Segment Time display on Properties page	2DEMap:STIMe?
Global Timebase checkbox on Properties page	2DEMap:GTBase 0 | 1 | OFF | ON
State of Global Timebase checkbox (enabled/disabled) on Properties page	2DEMap:GTBase?
Use Cursors checkbox on Properties page	2DEMap:UCURsor 0 | 1 | OFF | ON
State of Use Cursors checkbox (enabled/disabled) on Properties page	2DEMap:UCURsor?
Auto-Scroll checkbox on Properties page	2DEMap:ASCRoll 0 | 1 | OFF | ON

Table 5-1. Front Panel Function to Remote Command for the Agilent 86130A—Error Analysis

Front Panel Function	**Remote Command**
State of Auto-Scroll checkbox (enabled/disabled) on Properties page	2DEMap:ASCRoll?
Highlight Bursts checkbox on Properties page	2DEMap:HBURsts 0 | 1 | OFF | ON
State of Highlight Bursts checkbox (enabled/disabled) on Properties page	2DEMap:HBURsts?
Position Y-Cursor on view	2DEMap:YCPosition <integer>
Find location of Y-Cursor on view	2DEMap:YCPosition?
Set Chart Extents	2DEMap:CEXTents <numeric>, <numeric>, <numeric>, <numeric>
Chart extents display on view	2DEMap:CEXTents?
Error Map Status display on view	2DEMap:STATus?
Number of Segments display on view	2DEMap:SEGMents?

SCPI Messages

Introduction

The system-defined error/event numbers are chosen on an enumerated ("1 of N") basis. The SCPI defined error/event numbers and the < error description > portions of the ERRor query response are listed here. The first error/event described in each class (for example, –100, –200, –300, –400) is a "generic" error. In selecting the proper error/event number to report, more specific error/event codes are preferred, and the generic error/event is used only if the others are inappropriate.

No Error

This message indicates that the device has no errors.

0	No Error

The queue is completely empty. Every error/event in the queue has been read or the queue was purposely cleared by power-on, *CLS, etc.

Command Errors [–199, –100]

An < error/event number > in the range [–199, –100] indicates that an *IEEE 488.2* syntax error has been detected by the instrument's parser. The occurrence of any error in this class should cause the command error bit (bit 5) in the event status register (*IEEE 488.2*, section 11.5.1) to be set. One of the following events has occurred:

- An *IEEE 488.2* system error has been detected by the parser. That is, a controller-to-device message was received which is in violation of the *IEEE 488.2* standard. Possible violations include a data element which violates the device listening formats or whose type is unacceptable to the device.
- An unrecognized header was received. Unrecognized headers include incorrect device-specific headers and incorrect or unimplemented *IEEE 488.2* common commands.
- A Group Execute Trigger (GET) was entered into the input buffer inside of an *IEEE 488.2* < PROGRAM MESSAGE >.

Events that generate command errors shall not generate execution errors, device-specific errors, or query errors.

Table 6-1. Command Errors [–199, –100]

–100	Command error	This is the generic syntax error for devices that cannot detect more specific errors. This code indicates only that a Command Error as defined in *IEEE 488.2*, 11.5.1.1.4 has occurred.
–101	Invalid Character	A syntactic element contains a character which is invalid for that type; for example, a header containing an ampersand, SETUP&. This error might be used in place of errors –114, –121, –141, and perhaps some others.
–102	Syntax error	An unrecognized command or data type was encountered; for example, a string was received when the device does not accept strings.
–103	Invalid separator	The parser was expecting a separator and encountered an illegal character; for example, the semicolon was omitted after a program message unit, OUTPut[1]POLarity NORMal

Table 6-1. Command Errors [–199, –100]

–104	Data type error	The parser recognized a data element different than one allowed; for example, numeric or string data was expected but block data was encountered.
–105	GET not allowed	A Group Execute Trigger was received within a program message (see *IEEE 488.2*, 7.7).
–108	Parameter not allowed	More parameters were received than expected for the header; for example, the OUTPut[1][:STATe] 0 | 1 | OFF | ON command only accepts one parameter, so receiving OUTPut[1][:STATe] 0,OFF is not allowed.
–109	Missing parameter	Fewer parameters were received than required for the header; for example, the command OUTPut[1][:STATe] 0 | 1 | OFF | ON requires one parameter, so receiving OUTPut[1][:STATe] is not allowed.
–110	Command header error	An error was detected in the header. This error message should be used when the device cannot detect the more specific errors described for errors –111 through –119.
–111	Header separator error	A character which is not a legal header separator was encountered while parsing the header; for example, no white space followed the header, thus *ESE"5” is an error.
–112	Program mnemonic too long	The header contains more than twelve characters (see *IEEE 488.2*, 7.6.1.4.1).
–113	Undefined header	The header is syntactically correct, but it is undefined by this specific device; for example, *XYZ is not defined for any device.
–114	Header suffix out of range	Indicates that a nonheader character has been encountered in what the parser expects is a header element.
–120	Numeric data error	This error, as well as errors –121 through –128, are generated when parsing a data element which appears to be numeric, including the non-decimal numeric types. This particular error message should be used if the device cannot detect a more specific error.
–121	Invalid character in number	An invalid character for the data type being parsed was encountered; for example, an alpha in a decimal numeric or a "9" in octal data.
–123	Exponent too large	The magnitude of the exponent was larger than 32000 (see *IEEE 488.2*, 7.7.2.4.1).
–124	Too many digits	The mantissa of a decimal numeric data element contained more than 255 digits excluding leading zeros (see *IEEE 488.2*, 7.7.2.4.1).

Table 6-1. Command Errors [–199, –100]

–128	Numeric data not allowed	A legal numeric data element was received, but the device does not accept one in this position for the header.
–130	Suffix error	This error, as well as errors –131 through –139, are generated when parsing a suffix. This particular error message should be used if the device cannot detect a more specific error.
–131	Invalid suffix	The suffix does not follow the syntax described in *IEEE 488.2*, 7.7.3.2, or the suffix is inappropriate for this device.
–134	Suffix too long	The suffix contained more than 12 characters (see *IEEE 488.2*, 7.7.3.4).
–138	Suffix not allowed	A suffix was encountered after a numeric element which does not allow suffixes.
–140	Character data error	This error, as well as errors –141 through –149, are generated when parsing a character data element. This particular error message should be used if the device cannot detect a more specific error.
–141	Invalid character data	Either the character data element contains an invalid character or the particular element received is not valid for the header.
–144	Character data too long	The character data element contains more than twelve characters (see *IEEE 488.2*, 7.7.1.4).
–148	Character data not allowed	A legal character data element was encountered where prohibited by the device.
–150	String data error	This error, as well as errors –151 through –159, are generated when parsing a string data element. This particular error message should be used if the device cannot detect a more specific error.
–151	Invalid string data	A string data element was expected, but was invalid for some reason (see *IEEE 488.2*, 7.7.5.2); for example, an END message was received before the terminal quote character.
–158	String data not allowed	A string data element was encountered but was not allowed by the device at this point in parsing.
–160	Block data error	This error, as well as errors –161 through –169, are generated when parsing a block data element. This particular error message should be used if the device cannot detect a more specific error.
–161	Invalid block data	A block data element was expected, but was invalid for some reason (see *IEEE 488.2*, 7.7.6.2); for example, an END message was received before the length was satisfied.

Table 6-1. Command Errors [–199, –100]

–168	Block data not allowed	A legal block data element was encountered but was not allowed by the device at this point in parsing.
–170	Expression error	This error, as well as errors –171 through –179, are generated when parsing an expression data element. This particular error message should be used if the device cannot detect a more specific error.
–171	Invalid expression	The expression data element was invalid (see *IEEE 488.2*, 7.7.7.2); for example, unmatched parentheses or an illegal character.
–178	Expression data not allowed	A legal expression data was encountered but was not allowed by the device at this point in parsing.
–180	Macro error	This error, as well as errors –181 through –189, are generated when defining a macro or executing a macro. This particular error message should be used if the device cannot detect a more specific error.
–181	Invalid outside macro definition	Indicates that a macro parameter placeholder ($&<number) was encountered outside of a macro definition.
–183	Invalid inside macro definition	Indicates that the program message unit sequence, sent with a *DDT or *DMC command, is syntactically invalid (see 10.7.6.3).
–184	Macro parameter error	Indicates that a command inside the macro definition had the wrong number or type of parameters.

Execution Errors [–299, –200]

An < error/event number > in the range [–299, –200] indicates that an error has been detected by the instrument's execution control block. The occurrence of any error in this class should cause the execution error bit (bit 4) in the event status register (*IEEE 488.2*, section 11.5.1) to be set. One of the following events has occurred:

- A < PROGRAM DATA > element following a header was evaluated by the device as outside of its legal input range or is otherwise inconsistent with the device's capabilities.
- A valid program message could not be properly executed due to some device condition.

Execution errors shall be reported by the device after rounding and expression evaluation operations have taken place. Rounding a numeric data element, for example, shall not be reported as an execution error. Events that generate execution errors shall not generate Command Errors, device-specific errors, or Query Errors.

Table 6-2. Execution Errors [–299, –200]

–200	Execution error	This is the generic syntax error for devices that cannot detect more specific errors. This code indicates only that an Execution Error as defined in *IEEE 488.2*, 11.5.1.1.5 has occurred.
–201	Invalid while in local	Indicates that a command is not executable while the device is in local due to a hard local control (see *IEEE 488.2*, 5.6.1.5); for example, a device with a rotary switch receives a message which would change the switches state, but the device is in local so the message can not be executed.
–202	Settings lost due to rtl	Indicates that a setting associated with a hard local control (see *IEEE 488.2*, 5.6.1.5) was lost when the device changed to LOCS from REMS or to LWLS from RWLS.
–210	Trigger error	

Table 6-2. Execution Errors [–299, –200]

–211	Trigger ignored	Indicates that a GET, *TRG, or triggering signal was received and recognized by the device but was ignored because of device timing considerations; for example, the device was not ready to respond. Note: a DT0 device always ignores GET and treats *TRG as a Command Error.
–212	Arm ignored	Indicates that an arming signal was received and recognized by the device but was ignored.
–213	Init ignored	Indicates that a request for a measurement initiation was ignored as another measurement was already in progress.
–214	Trigger deadlock	Indicates that the trigger source for the initiation of a measurement is set to GET and subsequent measurement query is received. The measurement cannot be started until a GET is received, but the GET would cause an INTERRUPTED error.
–215	Arm deadlock	Indicates that the arm source for the initiation of a measurement is set to GET and subsequent measurement query is received. The measurement cannot be started until a GET is received, but the GET would cause an INTERRUPTED error.
–220	Parameter error	Indicates that a program data element related error occurred. This error message should be used when the device cannot detect the more specific errors described for errors –221 through –229.
–221	Setting conflict	Indicates that a legal program data element was parsed but could not be executed due to the current device state (see *IEEE 488.2*, 6.4.5.3 and 11.5.1.1.5.)
–222	Data out of range	Indicates that a legal program data element was parsed but could not be executed because the interpreted value was outside the legal range as defined by the device (see *IEEE 488.2*, 11.5.1.1.5.)
–223	Too much data	Indicates that a legal program data element of block, expression, or string type was received that contained more data than the device could handle due to memory or related device-specific requirements.
–224	Illegal parameter value	Used where exact value, from a list of possibilities, was expected.
–230	Data corrupt or stale	Possibly invalid data; new reading started but not completed since last access.
–231	Data questionable	Indicates that measurement accuracy is suspect.

Table 6-2. Execution Errors [–299, –200]

–240	Hardware error	Indicates that a legal program command or query could not be executed because of a hardware problem in the device. Definition of what constitutes a hardware problem is completely device-specific. This error message should be used when the device cannot detect the more specific errors described for errors –241 through –249.
–241	Hardware missing	Indicates that a legal program command or query could not be executed because of missing device hardware; for example, an option was not installed. Definition of what constitutes missing hardware is completely device-specific.
–250	Mass storage error	Indicates that a mass storage error occurred. This error message should be used when the device cannot detect the more specific errors described for errors –251 through –259.
–251	Missing mass storage	Indicates that a legal program command or query could not be executed because of missing mass storage; for example, an option that was not installed. Definition of what constitutes missing mass storage is device-specific.
–252	Missing media	Indicates that a legal program command or query could not be executed because of a missing media; for example, no disk. The definition of what constitutes missing media is device-specific.
–253	Corrupt media	Indicates that a legal program command or query could not be executed because of corrupt media; for example, bad disk or wrong format. The definition of what constitutes corrupt media is device-specific.
–254	Media full	Indicates that a legal program command or query could not be executed because the media was full; for example, there is no room on the disk. The definition of what constitutes a full media is device-specific.
–255	Directory full	Indicates that a legal program command or query could not be executed because the media directory was full. The definition of what constitutes a full media directory is device-specific.
–256	File name not found	Indicates that a legal program command or query could not be executed because the file name on the device media was not found; for example, an attempt was made to read or copy a nonexistent file. The definition of what constitutes a file not being found is device-specific.

Table 6-2. Execution Errors [–299, –200]

–257	File name error	Indicates that a legal program command or query could not be executed because the file name on the device media was in error; for example, an attempt was made to copy to a duplicate file name. The definition of what constitutes a file name error is device-specific.
–258	Media protected	Indicates that a legal program command or query could not be executed because the media was protected; for example, the write-protect tab on a disk was present. The definition of what constitutes protected media is device-specific.
–260	Expression error	Indicates that an expression program data element related error occurred. This error message should be used when the device cannot detect the more specific errors described for errors –261 through –269.
–261	Math error in expression	Indicates that a syntactically legal expression program data element could not be executed due to a math error; for example, a divide-by-zero was attempted. The definition of math error is device-specific.
–270	Macro error	Indicates that a macro-related execution error occurred. This error massage should be used when the device cannot detect the more specific errors described for errors –271 through –279.
–271	Macro syntax error	Indicates that a syntactically legal macro program data sequence, according to *IEEE 488.2*, 10.7.2, could not be executed due to a syntax error within the macro definition (see *IEEE 488.2*, 10.7.6.3.)
–272	Macro execution error	Indicates that a syntactically legal macro program data sequence could not be executed due to some error in the macro definition (see *IEEE 488.2*, 10.7.6.3.)
–273	Illegal macro label	Indicates that the macro label defined in the *DMC command was a legal string syntax but could not be accepted by the device (see *IEEE 488.2*, 10.7.3 and 10.7.6.2); for example, the label was too long, the same as a common command header, or contained invalid header syntax.
–274	Macro parameter error	Indicates that the macro definition improperly used a macro parameter placeholder (see *IEEE 488.2*, 10.7.3).
–275	Macro definition too long	Indicates that a syntactically legal macro program data sequence could not be executed because the string or block contents were too long for the device to handle (see *IEEE 488.2*, 10.7.6.1).

Table 6-2. Execution Errors [–299, –200]

–276	Macro recursion error	Indicates that a syntactically legal macro program data sequence could not be executed because the device found it to be recursive (see *IEEE 488.2*, 10.7.6.6).
–277	Macro redefinition not allowed	Indicates that a syntactically legal macro label in the *DMC command could not be executed because the macro label was already defined (see *IEEE 488.2*, 10.7.6.4).
–278	Macro header not found	Indicates that a syntactically legal macro label in the *GMC? query could not be executed because the header was not previously defined.
–280	Program error	Indicates that a downloaded program-related execution error occurred. This error message should be used when the device cannot detect the more specific errors described for errors –281 through –289.

Note

A downloaded program is used to add algorithmic capability to a device. The syntax used in the program and the mechanism for downloading a program is device-specific.

–281	Cannot create program	Indicates that an attempt to create a program was unsuccessful. A reason for the failure might include not enough memory.
–282	Illegal program name	The name used to reference a program was invalid; for example, redefining an existing program, deleting a nonexistent program, or in general, referencing a nonexistent program.
–283	Illegal variable name	An attempt was made to reference a nonexistent variable in a program.
–284	Program currently running	Certain operations dealing with programs may be illegal while the program is running; for example, deleting a running program might not be possible.
–285	Program syntax error	Indicates that a syntax error appears in a downloaded program. The syntax used when parsing the downloaded program is device-specific.
–286	Program runtime error	

Query Errors [–499, –400]

An < error/event number > in the range [–499, –400] indicates that the output queue control of the instrument has detected a problem with the message exchange protocol described in *IEEE 488.2*, chapter 6. The occurrence of any error in this class should cause the query error bit (bit 2) in the event status register (*IEEE 488.2*, section 11.5.1) to be set. These errors correspond to message exchange protocol errors described in *IEEE 488.2*, section 6.5. One of the following is true:

- An attempt is being made to read data from the output queue when no output is either present or pending.
- Data in the output queue has been lost.

Events that generate query errors shall not generate command errors, execution errors, or device-specific errors; see the other error definitions in this section.

Table 6-3. Query Errors [–499, –400]

–400	Query error	This is the general query error for devices that cannot detect more specific errors. This code indicates only that a Query Error as defined in *IEEE 488.2*, 11.5.1.1.7 and 6.3 has occurred.
–410	Query INTERRUPTED	Indicates that a condition causing an INTERRUPTED Query error occurred (see *IEEE 488.2*, 6.3.2.3); for example, a query followed by DAB or GET before a response was completely sent.
–420	Query UNTERMINATED	Indicates that a condition causing an UNTERMINATED Query error occurred (see *IEEE 488.2*, 6.3.2.2); for example, the device was addressed to talk and an incomplete program message was received.
–430	Query DEADLOCKED	Indicates that a condition causing a DEADLOCKED Query error occurred (see *IEEE 488.2*, 6.3.1.7); for example, both input buffer and output buffer are full and the device cannot continue.

Table 6-3. Query Errors [–499, –400]

–440	Query UNTERMINATED after indefinite response	Indicates that a query was received in the same program message after a query requesting an indefinite response was executed (see *IEEE 488.2*, 6.5.7.5.7.)_

Custom Errors [+2, +18]

An < error/event number > in the range [+2, +18] indicates that an error has been detected that is specific to Agilent 86130A. The occurrence of an error in this class causes the DDE error bit (bit 3) in the event status register (IEEE 488.2, section 11.5.1) to be set. One of the following events has occurred:

- A < PROGRAM DATA > element following a header was evaluated by the device as outside of its legal input range or is otherwise inconsistent with the device's capabilities.
- A valid program message could not be properly executed due to some device condition.

Custom errors shall be reported by the device after rounding and expression evaluation operations have taken place. Rounding a numeric data element, for example, shall not be reported as a custom error. Events that generate custom errors shall not generate command, execution, or query errors.

Table 6-4. Custom Errors

List Entry Format		
Error No.	**Standard Error String**	Custom Error String #1
		Custom Error String #2
		......
		......
		Custom Error String #n
------	-------------------------------	--
+2	**Invalid PRBS Trigger Pattern Length**	7 bits only
+3	**Clipped**	PG Data Output Amplitude value clipped
		PG Data Output Offset value clipped
		PG Data Output High Level value clipped

Table 6-4. Custom Errors

		PG Data Output Low Level value clipped
		PG Data Output Delay value clipped
		PG Data Bar Output Amplitude value clipped
		PG Data Bar Output High Level value clipped
		PG Data Bar Output Low Level value clipped
		PG Data Bar Output XOver voltage value clipped
		PG Clock Output Offset value clipped
		PG Clock Output High Level value clipped
		PG Clock Output Low Level value clipped
		PG Clock Bar Output Amplitude value clipped
		PG Clock Bar Output Offset value clipped
		PG Clock Bar Output High Level value clipped
		PG Clock Bar Output Low Level value clipped
		ED Data Input Delay value clipped
		Audio Volume value clipped
+4	**Invalid Termination Value**	<float>
		SM_QUICK
+7	**PlugIn Manager Error**	GPIBPlugInManager is not valid
		<Plugin_name> NULL pointer encountered
		<Plugin_name> Plugin not found
		<Plugin_name> Unable to access storage array
		<Plugin_name> There are no plugins in the system
		<Plugin_name> Unable to allocate element array
+9	**Server Error**	Analysis Engine not available.
		Server fails on COM call. Most probably the server isn’t in right mode for this operation or the file needed is missing.
		Burst Length analyzer not available.
		Burst Length analyzer is not enabled.
		Block Errors analyzer not available.
		Block Errors analyzer is not enabled.
		Correlation analyzer not available.

Table 6-4. Custom Errors

		Correlation analyzer is not enabled.
		Error Free Interval analyzer not available.
		Error Free Interval analyzer is not enabled.
		Pattern Sensitivity analyzer not available.
		Pattern Sensitivity analyzer is not enabled.
		Strip Chart analyzer not available.
		Strip Chart analyzer is not enabled.
		You don’t have the license to run the ECC analyzer.
		ECC analyzer not available.
		ECC analyzer is not enabled.
		You don’t have the license to run the 2-D Error Map analyzer.
		2-D Error Map not available.
		2-D Error Map is not enabled.
		Server not available.
		Error Removal Filter not available.
		Selected filter will remove all errors.
		New setting has no effect right now since the Error Removal Filter is in unsuitable mode.
		Error Removal Filter is in unsuitable mode.
		Symbol Filter not available.
		This setting won’t take effect until you specify the filename.
		The server is confused.
		Error Statistics analyzer not available.
		Error Statistics analyzer is not enabled.
+16	**Invalid Settings**	UPAT<n> is not an APATtern
		User pattern is not set
		PRBS<n> is not set
		PRBN<n> is not set
		MDEN<n> is not set
		Can't Change Voltage levels in AC Termination Mode

Table 6-4. Custom Errors

+17	**Invalid Parameters**	The start_bit = <n> is out of bounds
		The numeric value = <n> is not allowed
+18	**Invalid Pattern Type**	Selected pattern should not be alternate pattern
		Selected pattern is not an alternate pattern

Custom Errors [-321, -103]

An < error/event number > in the range [-321, -103] indicates that an error has been detected that is specific to the Error Analysis capacity of the Agilent 86130A. The occurrence of an error in this class causes the DDE error bit (bit 3) in the event status register (IEEE 488.2, section 11.5.1) to be set. One of the following events has occurred:

- A < PROGRAM DATA > element following a header was evaluated by the device as outside of its legal input range or is otherwise inconsistent with the device's capabilities.
- A valid program message could not be properly executed due to some device condition.

Custom errors shall be reported by the device after rounding and expression evaluation operations have taken place. Rounding a numeric data element, for example, shall not be reported as a custom error. Events that generate custom errors shall not generate command, execution, or query errors.

Table 6-5. Custom Errors

Error No.	**Standard Error String**	Custom Error String #1
		Custom Error String #2
		......
		......
		Custom Error String #n
	-------------------------------	---
–103	**Invalid Separator**	Not a full command.
	Wrong Parameter Type	Not a full query.
		Wrong parameter type/number.
–108	**Too Many Parameters**	(Not used)
–109	**Too Few Parameters**	(Not used)

Table 6-5. Custom Errors

–113	**Wrong Command**	Wrong Command.
–221	**Settings Conflict**	The specified time is later than the latest time.
		Current mapping mode isn’t suitable for this setting.
		The system is using ED clock rate.
		You have to specify the playback file first.
		Currently SegmentQuantity has no effect since the mapping mode isn’t quantity.
		Currently SegmentSecond has no effect since the mapping mode isn’t second.
	Too Much Data	(Not used)
–224	**Illegal Parameter Value**	Settings are adjusted to ...
		Out of range.
		Block length has been adjusted to ...
		X-cursor position has been adjusted to ...
		Y-cursor position has been adjusted to ...
		Chart Extents have been adjusted to ...
		Segment quantity has been adjusted to ...
		Segment second has been adjusted to ...
		User-spec rate has been adjusted to ...
		Minimum burst length has been adjusted to ...
		Burst error-free threshold has been adjusted to ...
		Burst error length lower limit has been adjusted to ...
		Burst error length upper limit has been adjusted to ...
		Symbol size has been adjusted to ...
–254	**Media Full**	(Not used)
–256	**File Not Found**	(Not used)
–258	**Media Protected**	(Not used)
–310	**System Error**	(Not used)
–321	**Out of Memory**	Can’t handle command because we’re out of memory.
		Not enough memory.

Programming Examples

Introduction

This chapter contains example programs for the GPIB programming of 86130A.

NOTE These example programs are for demonstration only. They are provided with no warranty of any kind.

Technical Requirements

Compliler:Microsoft Visual C++ v6.0

Portions of this code are based on the "simple.c" example in the National Instruments 488.2 examples.

These example programs requires the non-standard header file DECL-32.H and the Object file GPIB-32.obj to function. These files (or equivilent files) should be included with GPIB/IEEE 488.2 driver files.

Example Program Organization
Each example program is organized into 4 sections:

1 Communications setup
This section initializes and clears the instrument. It, then, reads and prints the instrument's ID string.

2 Settings and Alignment
This section sets up a bit rate, pattern, and output terminations. It, then, aligns the instrument.

3 Test and results
This section sets up a single time period of accumulation. It monitors the conclusion of the accumulation period with the *OPC command. It, then, queries the bit error ratio.

4 Shut down and clean-up

Accumulation Example

Description

This program is a demonstration of the GPIB commands for 86130A. The program sets up the instrument, then makes an accumulation measurement of 30 seconds.

The following is a list of the commands used in this program. For more information, you may refer to the page listed in this guide.

Command	Description	Page Number
*IDN?	ID Query	Refer to "*IDN?" on page 4-6.
*OPC?	Operation complete querry	Refer to "Mandatory Common Commands and Queries" on page 4-4.
SOURce9:FREQuency 2488000000	Change the data frequency	Refer to "SOURce9: The Clock Source" on page 4-89.
SOURce1:PATTern:SELect PRBS23	Change the data pattern	Refer to "SOURce1: The Data Source" on page 4-66.
SOURce1:VOLTage:LLEVel SCFL	Change the data voltage levels	.

SOURce2:VOLTage:LLEVel SCFL	Change the clock voltage levels	Refer to "SOURce2: The Clock Source" on page 4-82.
OUTPut2:STATe ON	Turn on the clock	Refer to "OUTPut2: The Clock Output" on page 4-26.
OUTPut1:STATe ON	Turn on the data	Refer to "OUTPut1: The Data Output" on page 4-23.
SENSe1:EYE:ALIGn:AUTO 1	Auto Align	Refer to "SENSe1: The Data Sense" on page 4-41.
SENSe1:GATE:MODE SINGle	Set the test mode to single	
SENSe1:GATE:PERiod:TIME 30	Set the test time to thirty seconds	
SENSe1:GATE:STATe ON	Start the test	
FETCh:SENSe1:ERATio?	Query the error ratio measurement	Refer to "The FETCh Measurement Subsystem" on page 4-12.

```
      ===========================================================*/

//header files:
#include <stdio.h>   //standard console i/o header
#include <stdlib.h>  //standard general purpose C++ header
#include <windows.h>//windows header

#include <decl-32.h>//GPIB header

//device parameters (constants)
#define BDINDX              0     // Board Index
#define PRIMARY_ADDR      17    // Primary address of device (default)
#define SECONDARY_ADDR      0     // Secondary address of device
#define TIMEOUT             T30s  // Timeout value = 30 seconds
#define EOTMODE             1     // Enable the END message
#define EOSMODE             0     // Disable the EOS mode

#define TEST_TIME  30// The test time in seconds

//prototypes
int Error_Code(char desc[]);

/*=======================================================
Function:Main
Description:The main function serves to control program flow
Inputs:         (None)
Outputs:int     - error detection and easy function break
                                         (returns zero for no error, 1 for and error)
========================================================*/
int main(){

//declatrations
  int BERT;             //represents the BERT in all GPIB calls
  char buffer[256];//buffer holds the output data from the BERT
  char test_period[30];//used to build the string that sets the test period

//--------------set-up communications with the BERT---------------------------

 BERT = ibdev(BDINDX, PRIMARY_ADDR, SECONDARY_ADDR, TIMEOUT,
EOTMODE, EOSMODE);

 if (ibsta & ERR){  //check for error
      printf("Unable to open device\nibsta = 0x%x iberr = %d\n", ibsta, iberr);
      return 1;
    }//if (ibsta & ERR){

 ibclr (BERT);   //clr the BERT
   if (ibsta & ERR) return Error_Code("unable to clear BERT");

// check for errors

 ibwrt (BERT, "*IDN?\n", 6);//querry for the id number
   if (ibsta & ERR) return Error_Code("id querry (write) failed");

// check for errors
```

```
 ibrd (BERT, buffer, 255);//read back the id
   if (ibsta & ERR) return Error_Code("id querry (read) failed");

// check for errors

 buffer[ibcntl - 1] = '\0'; //terminate the result string with a null character

 printf("Bitalyzer ID: %s\n", buffer);  //print the ID number to the console

//-------------------change the settings on the BERT-------------------------

 ibwrt (BERT, "SOURce9:FREQuency 2488000000\n", 29);
                                             //set the frequency to 2.488GHz (OC-48)
 if (ibsta & ERR) return Error_Code("could not set frequency");

// check for errors

 ibwrt (BERT, "SOURce1:PATTern:SELect PRBS23\n", 30);
                                             //set the pattern to 2^23-1 PRBS
 if (ibsta & ERR) return Error_Code("could not set bit pattern");

// check for errors

 ibwrt (BERT, "SOURce1:VOLTage:LLEVel SCFL\n", 28);
                                             //set the data voltage levels to SCFL
 if (ibsta & ERR) return Error_Code("could not set data levels");

// check for errors

 ibwrt (BERT, "SOURce2:VOLTage:LLEVel SCFL\n", 28);
                                             //set the clock voltage levels to SCFL
 if (ibsta & ERR) return Error_Code("could not set clock levels");

// check for errors

 ibwrt (BERT, "OUTPut2:STATe ON\n", 17);//turn on the clock
 if (ibsta & ERR) return Error_Code("could not turn on clock");

// check for errors

 ibwrt (BERT, "OUTPut1:STATe ON\n", 17);//turn on the data output
 if (ibsta & ERR) return Error_Code("could notturn on data");

// check for errors

 ibwrt (BERT, "SENSe1:EYE:ALIGn:AUTO 1\n", 24); //Auto Align the sample point
 if (ibsta & ERR) return Error_Code("could not start auto align");

// check for errors

 ibwrt (BERT, "*OPC?\n", 6);//querry for the operation complete command
 if (ibsta & ERR) return Error_Code("could not wirte OPC querry");

// check for errors
```

```
 ibrd (BERT, buffer, 255);//read back the operation complete register
   if (ibsta & ERR) return Error_Code("autoalign error, possible timeout");

// check for errors; this second *OPC? below is not needed for software releases after
// A.01.03.

 ibwrt (BERT, "*OPC?\n", 6);//querry for the operation complete command
   if (ibsta & ERR) return Error_Code("could not wirte OPC querry");

// check for errors

 ibrd (BERT, buffer, 255);//read back the operation complete register
   if (ibsta & ERR) return Error_Code("autoalign error, possible timeout");

// check for errors

//------------------------Take the BER measurement-----------------------------

 sprintf(test_period, "%s %d\n", "SENSe1:GATE:PERiod:TIME ", TEST_TIME);
                                      //construct the test period command string

 ibwrt (BERT, test_period, 27);     //set the test period
 if (ibsta & ERR) return Error_Code("could not set test period");

// check for errors

 ibwrt (BERT, "SENSe1:GATE:MODE SINGle\n", 24);
                                                          //set the accumulation to
single mode
 if (ibsta & ERR) return Error_Code("could not set single test mode");

// check for errors

 ibwrt (BERT, "SENSe1:GATE:STATe ON\n", 21); //start the BER measurement
 if (ibsta & ERR) return Error_Code("could not start test");

// check for errors

 Sleep ((TEST_TIME - 5) * 1000); //wait until five seconds before the end
                                      //of the test period (done to aviod timeout prolblems)

 ibwrt (BERT, "*OPC?\n", 6);//querry for the operation complete command
   if (ibsta & ERR) return Error_Code("could not write OPC querry");

// check for errors

 ibrd (BERT, buffer, 255);//read back the operation complete register
   if (ibsta & ERR) return Error_Code("problem with test (possible time out)");

// check for errors

 ibwrt (BERT, "FETCh:SENSe1:ERATio?\n", 21); //retrieve the Error Ratio
 if (ibsta & ERR) return Error_Code("could not wirte Error ratio querry");

// check for errors
```

```
 ibrd (BERT, buffer, 255);//read back the operation complete register
   if (ibsta & ERR) return Error_Code("could not read error ratio");

 buffer[ibcntl - 1] = '\0'; //terminate the result string with a null character

 printf("Error Rate: %s\n", buffer);  //print the ID number to the console

//------------------------shut down and clean up-----------------------------

 return 0;                          //exit the function (no error)

}//Main()

/*=====================================================
Function:Error_Code
Description:The Error_Code function displays an error message if any errors
                        are encountered
Inputs:          char[] desc - a description of the error
Outputs:int     - always returns a 1 (the function is only called when an
                                                error is encountered)
=====================================================*/

int Error_Code(char desc[]){

 //GPIB Error Codes
 char ErrorCodes[21][5] = {"EDVR", "ECIC", "ENOL", "EADR", "EARG",
                          "ESAC", "EABO", "ENEB", "EDMA", "",
                          "EOIP", "ECAP", "EFSO", "", "EBUS",
                          "ESTB", "ESRQ", "", "", "", "ETAB"};

 printf("Error : %s\nibsta = 0x%x iberr = %d (%s)\n",
              desc, ibsta, iberr, ErrorCodes[iberr]);

 return 1;
} //Error_Code()
```

Demonstration Pattern Example

Description

This program is a demonstration of how to open an SONET-OC48 example pattern and set up the instrument. It, then, performs an auto-align, and takes a BER measurement.

GPIB commands and page

Command	Description	Page Number
*IDN?	ID Querry	Refer to "*IDN?" on page 4-6.
*OPC?	Operation complete querry	Refer to "*OPC?" on page 4-7.
SOURce9:FREQuency 2488000000	Change the data frequency	Refer to "SOURce9: The Clock Source" on page 4-89.
SOURce1:PATTern:UPATtern0 STRaight	Repetitive pattern mode	Refer to "SOURce1: The Data Source" on page 4-66.
SOURce1:PATTern:SELect filename,<file>	Change the data pattern	
SOURce1:VOLTage:LLEVel SCFL	Change the data voltage levels	

OUTPut2:STATe ON	Turn on the clock	Refer to "OUTPut2: The Clock Output" on page 4-26.
OUTPut1:STATe ON	Turn on the data	Refer to "OUTPut1: The Data Output" on page 4-23.
SENSe1:EYE:ALIGn:AUTO 1	Auto Align	Refer to "SENSe1: The Data Sense" on page 4-41.
SENSe1:GATE:PERiod:TIME 30	Set the test time to thirty seconds	
SENSe1:GATE:STATe ON	Start the test	
SENSe1:GATE:MODE SINGle	Set the test mode to single	
FETCh:SENSe1:ERATio?	Query the error ratio measurement	Refer to "The FETCh Measurement Subsystem" on page 4-12.

```
==================================================================*/

//header files:
#include <stdio.h>   //standard console i/o header
#include <stdlib.h>  //standard general purpose C++ header
#include <windows.h>//windows header

#include <decl-32.h>//GPIB header

//device parameters (constants)
#define BDINDX              0     // Board Index
#define PRIMARY_ADDR        17    // Primary address of device (default)
#define SECONDARY_ADDR      0     // Secondary address of device
#define TIMEOUT             T30s  // Timeout value = 30 seconds
#define EOTMODE             1     // Enable the END message
#define EOSMODE             0     // Disable the EOS mode
```

```
#define TEST_TIME  30// The test time in seconds

//prototypes
int Error_Code(char desc[]);


/*======================================================
Function:Main
Description:The main function serves to control program flow
Inputs:           (None)
Outputs:int     - error detection and easy function break
                                        (returns zero for no error, 1 for and error)
======================================================*/
int main(){

//declatrations
  int BERT;              //represents the BERT in all GPIB calls
  char buffer[256];//buffer holds the output data from the BERT
  char test_period[30];//used to build the string that sets the test period
  char file_cmd[33],file_name[60],cmd[255];//these strings are used to build the
                                                            //command needed to open the bit pattern
file

//--------------set-up communications with the BERT----------------------------

 BERT = ibdev(BDINDX, PRIMARY_ADDR, SECONDARY_ADDR, TIMEOUT,
EOTMODE, EOSMODE);

 if (ibsta & ERR){   //check for error
      printf("Unable to open device\nibsta = 0x%x iberr = %d\n", ibsta, iberr);
      return 1;
    }//if (ibsta & ERR){

 ibclr (BERT);   //clr the BERT
   if (ibsta & ERR) return Error_Code("unable to clear BERT");

// check for errors

 ibwrt (BERT, "*IDN?\n", 6);//querry for the id number
   if (ibsta & ERR) return Error_Code("id querry (write) failed");

// check for errors

 ibrd (BERT, buffer, 255);//read back the id
   if (ibsta & ERR) return Error_Code("id querry (read) failed");

// check for errors

 buffer[ibcntl - 1] = '\0'; //terminate the result string with a null character

 printf("Bitalyzer ID: %s\n", buffer);  //print the ID number to the console

//--------------------change the settings on the BERT--------------------------

 ibwrt (BERT, "SOURce9:FREQuency 2488000000\n", 29);
                                                 //set the frequency to 2.488GHz (OC-48)
 if (ibsta & ERR) return Error_Code("could not set frequency");

// check for errors

sprintf(file_cmd,"%s","SOURce1:PATTern:SELect Filename,");
 sprintf(file_name,"%s",
  "'e:\\bitalyzer\\patterns\\demos\\sonet sts48\\sonet_sts-48.ptrn'");
 sprintf(cmd, "%s%s\n", file_cmd,file_name);

 ibwrt (BERT, cmd, 92); //set the pattern to a simulated SONET OC48 signal
 if (ibsta & ERR) return Error_Code("could not set bit pattern");

// check for errors


 ibwrt (BERT, "Source1:pattern:Upattern0:USE STRAIGHT\n", 40);
                                         //set the user pattern mode to repetitive
  if (ibsta & ERR) return Error_Code("could not set pattern mode");

// check for errors


 ibwrt (BERT, "SOURce1:VOLTage:LLEVel SCFL\n", 28);
                                      //set the data voltage levels to SCFL
 if (ibsta & ERR) return Error_Code("could not set data levels");

// check for errors

 ibwrt (BERT, "SOURce2:VOLTage:LLEVel SCFL\n", 28);
                                      //set the clock voltage levels to SCFL
 if (ibsta & ERR) return Error_Code("could not set clock levels");

// check for errors

 ibwrt (BERT, "OUTPut2:STATe ON\n", 17);//turn on the clock
 if (ibsta & ERR) return Error_Code("could not turn on clock");

// check for errors

 ibwrt (BERT, "OUTPut1:STATe ON\n", 17);//turn on the data output
 if (ibsta & ERR) return Error_Code("could notturn on data");

// check for errors

 ibwrt (BERT, "SENSe1:EYE:ALIGn:AUTO 1\n", 24); //Auto Align the sample point
 if (ibsta & ERR) return Error_Code("could not start auto align");

// check for errors

 ibwrt (BERT, "*OPC?\n", 6);//querry for the operation complete command
 if (ibsta & ERR) return Error_Code("could not wirte OPC querry");

// check for errors

 ibrd (BERT, buffer, 255);//read back the operation complete register
  if (ibsta & ERR) return Error_Code("autoalign error, possible timeout");
```

```
// check for errors; this second *OPC? below is not needed for software releases after
// A.01.03.

 ibwrt (BERT, "*OPC?\n", 6);//querry for the operation complete command
   if (ibsta & ERR) return Error_Code("could not wirte OPC querry");

// check for errors

 ibrd (BERT, buffer, 255);//read back the operation complete register
   if (ibsta & ERR) return Error_Code("autoalign error, possible timeout");

// check for errors

//-----------------------Take the BER measurement----------------------------

 sprintf(test_period, "%s %d\n", "SENSe1:GATE:PERiod:TIME ", TEST_TIME);
                                        //construct the test period command string

 ibwrt (BERT, test_period, 27);     //set the test period
  if (ibsta & ERR) return Error_Code("could not set test period");

// check for errors

 ibwrt (BERT, "SENSe1:GATE:MODE SINGle\n", 24);
  //set the accumulation to single mode
  if (ibsta & ERR) return Error_Code("could not set single test mode");

// check for errors

 ibwrt (BERT, "SENSe1:GATE:STATe ON\n", 21);
//start the BER measurement
  if (ibsta & ERR) return Error_Code("could not start test");

// check for errors


 ibwrt (BERT, "*OPC?\n", 6);//query for the operation complete command
   if (ibsta & ERR) return Error_Code("could not write OPC querry");
                                                                               //
check for errors

 ibrd (BERT, buffer, 255);//read back the operation complete register
   if (ibsta & ERR) return Error_Code("problem with test (possible time out)");

// check for errors

 ibwrt (BERT, "FETCh:SENSe1:ERATio?\n", 21); //retrieve the Error Ratio
  if (ibsta & ERR) return Error_Code("could not wirte Error ratio querry");

// check for errors

 ibrd (BERT, buffer, 255);//read back the operation complete register
   if (ibsta & ERR) return Error_Code("could not read error ratio");

// check for errors
```

```
 buffer[ibcntl - 1] = '\0'; //terminate the result string with a null character

 printf("Error Rate: %s\n", buffer);  //print the ID number to the console

//------------------------shut down and clean up-----------------------------

 return 0;                          //exit the function (no error)

}//Main()

/*==================================================

Function:Error_Code
Description:The Error_Code function displays an error message if any errors
                          are encountered
Inputs:          char[] desc - a description of the error
Outputs:int     - always returns a 1 (the function is only called when an
                                                  error is encountered)
==================================================*/

int Error_Code(char desc[]){

 //GPIB Error Codes
 char ErrorCodes[21][5] = {"EDVR", "ECIC", "ENOL", "EADR", "EARG",
                         "ESAC", "EABO", "ENEB", "EDMA", "",
                         "EOIP", "ECAP", "EFSO", "", "EBUS",
                         "ESTB", "ESRQ", "", "", "", "ETAB"};

 printf("Error : %s\nibsta = 0x%x iberr = %d (%s)\n",
             desc, ibsta, iberr, ErrorCodes[iberr]);

 return 1;
} //Error_Code()
```

Index