BERTScope™ CR
BERTScope™ CR HS
BERTScope™ CRJ

Clock Recovery Remote Control Guide

Part Number 0130-702.00 Rev B

SYNTHESYS RESEARCH, INC.

3475-D Edison Way
Menlo Park, California U.S.A. 94025

Voice (650) 364-1853 Fax (650) 364-5716

Email tech_support@bertscope.com

http://www.bertscope.com/

This Remote Control Guide is applicable to the following SyntheSys Research, Inc., products:

BERTScope™ CR Clock Recovery Instrument
BERTScope™ CR HS High Sensitivity Clock Recovery Instrument
BERTScope™ CR$_J$ Clock Recovery Instrument

BERTScope™ **CR** Remote Control Guide
Part Number 0130-702.00 Rev B
24 July 2007

Table of Contents

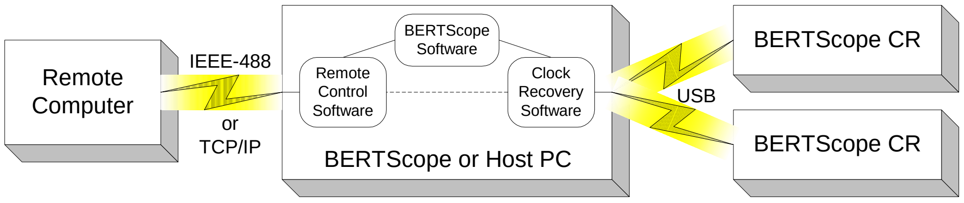
BERTScope™ CR / CR HS / CRJ

Remote Control

Overview

The BERTScope CR can be controlled remotely via either an IEEE-488 or a TCP/IP communications connection. The BERTScope CR remote control software runs on either a BERTScope, or a host PC running Microsoft Windows 2000 or XP operating systems.

The remote control software accepts text-oriented commands from a remote computer via either an IEEE-488 or TCP/IP connection. It then routes the commands to one or more BERTScope CRs, connected to the BERTScope or host PC via USB.



If the local host is a BERTScope, then the BERTScope software gets a chance to handle the command first. BERTScope-specific commands are documented in a separate manual, the *BERTScope Remote Control Guide*, part number 0150-703-06.28.

Any commands the BERTScope software doesn't understand are sent to the Clock Recovery software, which then controls the BERTScope CR. The BERTScope CR-specific commands are documented in this manual.

When a BERTScope CR is connected to a BERTScope, you can inter-mix BERTScope and BERTScope CR commands in the same script, program, or interactive session. If the host is just an ordinary PC, then the commands are routed to the Clock Recovery software directly. In this case, using the BERTScope-specific commands will result in an error.

For most applications, a single BERTScope CR is connected to a BERTScope or host PC. In this case, the BERTScope CR is automatically detected and connected when the remote control software is started, and automatically disconnected when the remote control software is stopped. There is no need in this case to use the **OPEN** or **CLOSE** commands described below.

If more than one BERTScope CR is connected to the BERTScope or host computer, the remote control software will not connect automatically. In this case, the Remote computer must issue a **NAMES?** query to discover the IDs of the connected BERTScope CRs, then **OPEN** the one desired before issuing control commands. If the Remote computer needs to control multiple BERTScope CRs, it would **OPEN**, control, then **CLOSE** one, then **OPEN**, control, and **CLOSE** another. The device that is **OPEN** is referred to as the "current" device throughout this document.

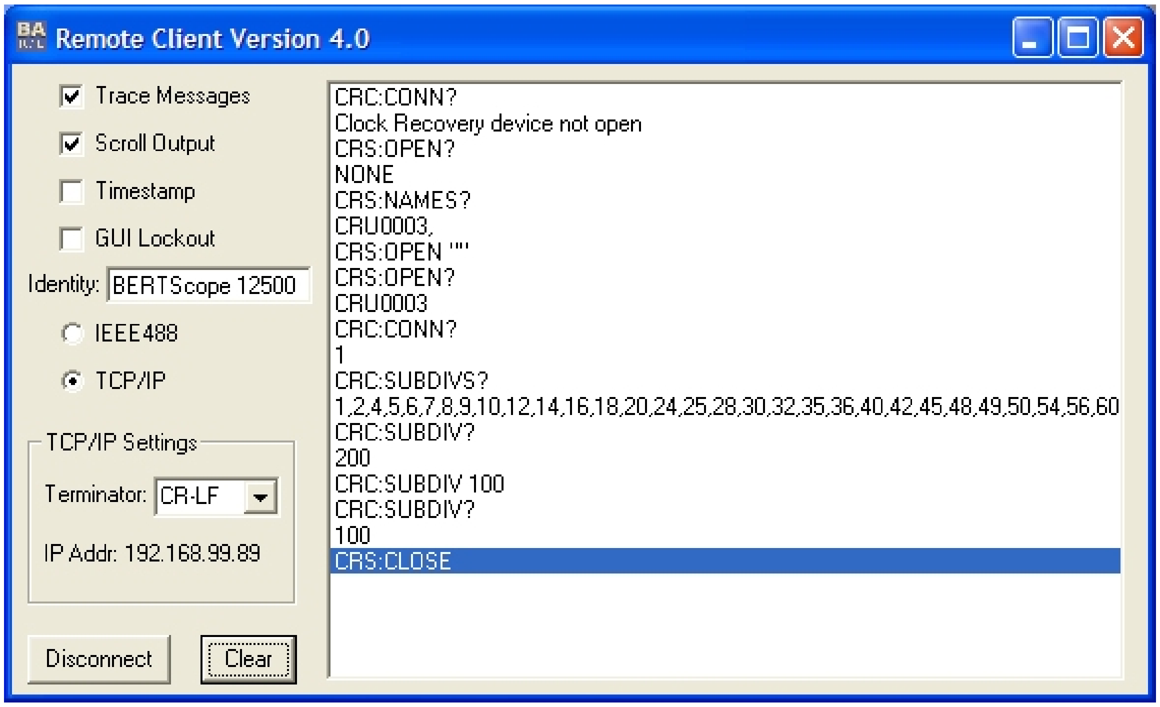
Remote control is implemented using a text-oriented command protocol described in the following pages of this document. These commands enable you to set and query the system parameters of the BERTScope CR, and to retrieve measurements made by the instrument.

Setting Up Remote Control

Before beginning a remote control session, it is necessary to start the Remote Client application, BitAlyzerRemoteClient.exe, on the host computer.

If the host is a BERTScope, select the “Remote” button from the System View’s Tools page.

If the host computer is a normal PC, then just execute the “BitAlyzerRemoteClient.exe” file.



[✓] Trace Messages
[✓] Scroll Output
[ ] Timestamp
[ ] GUI Lockout
Identity: BERTScope 12500
IEEE488 [✓ TCP/IP]
Terminator: CR-LF
IP Addr: 192.168.99.89

```
CRC:CONN?
Clock Recovery device not open
CRS:OPEN?
NONE
CRS:NAMES?
CRU0003,
CRS:OPEN ""
CRS:OPEN?
CRU0003
CRC:CONN?
1
CRC:SUBDIVS?
1,2,4,5,6,7,8,9,10,12,14,16,18,20,24,25,28,30,32,35,36,40,42,45,48,49,50,54,56,60
CRC:SUBDIV?
200
CRC:SUBDIV 100
CRC:SUBDIV?
100
CRS:CLOSE
```

The Remote Client application allows you to select either a TCP/IP or GPIB communication channel.

Internally, the program handles routing of commands between the remote computer and the BERTScope CR.

Select the “Trace Messages” box to monitor the commands sent to the BERTScope CR.

To test the Remote Client connection, use a TCP/IP terminal program, like “telnet.exe”, to interact with the BERTScope CR directly. An example session is shown in the figure above.

Command Syntax

Remote Control Command Lines are defined as ASCII text strings ending with "\r" or "\n".

The commands follow a basic three-part structure, consisting of one word identifying the feature of the BERTScope CR being addressed, another word identifying a specific operation within that feature, and optional parameters. The feature and operation are separated by a colon, and the operation and parameter (if any), are separated by a space.

FEATURE:OPERATION PARAMETER

Commands and parameters are generally *case-insensitive*, with the notable exception of those requiring a *case-sensitive* parameter for the clock recovery **STANDARD** name.

Most commands have a long form and a short form. For example,

CRCONTROL:CONNECTED?

can be typed out exactly as shown, or as

CRC:CONN? Or **crc:conn?**

The short form is convenient during interactive control (using a telnet session, for example). You might want to use the long form in scripts or programs, to increase code clarity.

Features

The BERTScope CR commands are divided into five feature groups:

The **CRS**ervice group (**CRS**, for short) consists of commands that handle device discovery and connection. You can use these commands anytime, even if a 'current' device has not been selected yet (with the **OPEN** command). The **NAMES**, **OPEN**, and **CLOSE** commands are all part of this group.

The **CRC**ontrol group (**CRC**) consists of commands for controlling and monitoring a particular BERTScope CR. The 'current' device must have been specified (using the **CRS:OPEN** command) prior to using any commands in the CRC group.

The **CRL**oop**R**esponse group (**CRLR** or **CRL** for short) consists of commands to measure the actual loop response characteristics of the BERTScope CR The 'current' device must have been specified (using the **CRS:OPEN** command) prior to using any commands in the CRC group.

The **CRSSCW**aveform group (**CRSSCW** or **SSCW** for short) consists of commands to setup and run an Spread Speaktrum Clocking measurement session, The 'current' device must have been specified (using the **CRS:OPEN** command) prior to using any commands in the CRC group.

The **CR**Jitter**S**pectrum group (**CRJS** or **JS** for short) consists of commands to setup and run a Jitter Spectrum session on a BERTScope CRJ, then retrieve up to 3 different integrated jitter measurements. The 'current' device must have been specified (using the **CRS:OPEN** command) prior to using any commands in the CRC group.

Finally, in addition to these feature groups, several miscellaneous features exist to manipulate the host computer display (VIEW, PTFILE, GUILOCKOUT), the run state (RSTATE, RDURATION), and the overall program configuration (SCONFIGUARTION, RCONFIGURATION, ISSTATUS, RSSTATUS). These commands primarily implement the functionality provided by the GUI Console buttons.

Operations

Operations fall into three sub-categories:

Read-Only properties

Query a read-able property by appending a question mark after it.

For example,

CRS:OPEN?

returns the device name of the currently connected BERTScope CR.

Read-Write properties

Query a read-write property by appending a question mark after it.

Set a read-write property by adding an appropriate parameter.

For example,

CRS:OPEN "CR_1234"

selects the BERTScope CR with the device name "CR_1234" as the current device.

Write-Only operation.

Question marks are illegal for write-only operation, and parameters may or may not be required, depending upon the particular operation.

For example,

CRC:RELOCK

requires no parameters, but

CRC:RCONFIGDISK "D:\BitAlyzer\Configurations\mycfg.cru"

requires the complete pathname of the BERTScope CR configuration file.

Parameters

A parameter can be Double or Integer numbers, Boolean (0 or 1), an Enumerated Data Type, or a String.

When a command uses a string as a parameter, the string is enclosed in double-quotes ("string"). Filenames always require the absolute path, enclosed within double-quotes.

Parameters are generally *case-insensitive*, with the notable exception of *case-sensitive* clock recovery Standard names.

For example,

CRC:STANDARD "123abc"

and

CRC:DELETESTANDARD "123ABC"

are specifying different user-defined clock recovery **STANDARD**s.

Command Status

The BERTScope CR receives and operates on each command immediately.

The input values of commands are checked. If an input is received that is outside of the legal range, the input will be clipped and recorded in the status queue.

Command execution sets a status variable that may be queried by the user to determine if the previous command was successful.

You request the status of a command by sending:

SYSTEM:ERROR? or **SYST:ERR?**

This command retrieves items listed in the error status queue in the format:

< n, “event/string” >

The status queue is first in, first out. It can contain up to 100 error messages. If the status queue overflows, the last error/event in the queue is replaced with Error –350, “Queue overflow”. When all errors/events have been read from the queue, further error/event queries will return 0, “NO error”.

To clear the status queue all at once, send:

***CLS**

Command Error Messages and Codes

Error Messages

"TOO MANY PARAMETERS"
"UNRECOGNIZED COMMAND"
"WRONG PARAMETER TYPE"
"TOO LITTLE PARAMETERS"
"WRONG PARAMETER FORMAT"
"WRONG PARAMETER NUMBER"
"WRONG ACTION-ONLY"
"WRONG QUERY-ONLY"
"REQUESTED <QUERY PARAMETER> IS NOT SUPPORTED ON THE PLATFORM"

Error Codes

0 "NO ERROR"
-350 "Queue Overflow"
-10 File transfer error
-20 System error or event
-30 Command error, see list of error messages
-40 No software option

Tips

It is most useful to have an understanding of the basic principles of BERTScope CR clock recovery before undertaking remote control programming. There is a high degree of similarity between the BERTScope's graphical user interface and the remote control command protocol.

For more technical assistance in programming your remote control applications, please contact SyntheSys Research, Inc. We are glad to help.

Technical Support
SyntheSys Research, Inc.
3475-D Edison Way
Menlo Park, CA 94025

Voice: (650) 364-1853
Fax: (650) 364-5716

tech_support@bertscope.com

Command Summary

CRService Commands

(usable before OPEN)

Dynamic Read-Only Properties

TOTALDEVCOUNT **or** TDCT
ATTACHDEVCOUNT **or** ADCT
ATTACHDEVNAMES **or** NAMES

Read-Write Settings

OPEN

Write-Only Operations

CLOSE

CRControl Commands

(requires prior OPEN)

Static Read-Only Identification and Capabilities

SERIALNUM **or** SN
SWREV **or** SW
HWREV **or** HW
FPGAREV **or** FPGA
EXPREV **or** EXP
STANDARDLIST **or** STANDARDS
SUBCLOCKDIVLIST **or** SUBDIVS

Dynamic Read-Only Properties

CONNECTED **or** CONN
BUSY
LOCKSTATE **or** LOCKST
LOCKCOUNT **or** LOCKCT
DATARATE
EDGEDENSITY **or** EDGED
PHASEERRP2P **or** PHERR
PHASEERRRMS **or** PHERMS

Read-Write Settings

DEVICENAME **or** DN
STANDARD
NOMFREQUENCY **or** NOMFREQ
LOCKRANGE **or** RANGE
LOOPBANDWIDTH **or** LOOPBW
PEAKING
LOCKMODE
EDGEDENSITYMODE **or** EDGEDMODE
NOMEDGEDENSITY **or** NOMEDGED
PHASEERRLIMIT **or** PHERRLMT
CLOCKOUTPUT **or** CLKOUT
CLOCKAMPLITUDE **or** CLKAMPL
SUBCLOCKOUTPUT **or** SUBOUT
SUBCLOCKAMPLITUDE **or** SUBAMPL

SUBCLOCKDIV **or** SUBDIV
AUTOSCONFIGDEVICE **or** AUTOSDEV

Write-Only Operations

RELOCK **or** LOCK
RESETLOCKCOUNT **or** RESETLOCKCT
SCONFIGDISK **or** SDISK
RCONFIGDISK **or** RDISK
SCONFIGDEVICE **or** SDEV
RCONFIGDEVICE **or** RDEV
CREATESTANDARD
DELETESTANDARD

CRLoopResponse Commands

(requires prior OPEN)

Read-Write Settings

CHARTTYPE

Dynamic Read-Only Measurements

BANDWIDTH
PEAKING
PEAKFREQ

CRSSCWaveform Commands

(requires prior OPEN)

Read-Write Settings

AVGSETTING **or** AVGS
SHIFT

Dynamic Read-Only Properties and Measurements

AVGCOUNT **or** AVGC
NUMSCANS **or** SCANS
NUMHISTOGRAMS **or** HISTOS
HISTOGRAMPOINTS **or** HISTOPTS
DEVMIN_PPM **or** DEVMIN
DEVMIN_HZ
DEVMAX_PPM **or** DEVMAX
DEVMAX_HZ
MODFREQ
NOMFREQ

Write-Only Operations

SCONFIGURATION **or** SCONFIG
RCONFIGURATION **or** RCONFIG

CRJitterSpectrum Commands (Model CRJ Only)

(requires prior OPEN)

Read-Write Settings

SCANRESMODE **or** SCANRES
CRINITMODE **or** CRINIT
AVGSETTING **or** AVGS
IJUNITS
IJMODE_1
IJMINHZ_1
IJMAXHZ_1
IJMODE_2
IJMINHZ_2
IJMAXHZ_2
IJMODE_3
IJMINHZ_3
IJMAXHZ_3

Dynamic Read-Only Properties and Measurements

AVGCOUNT **or** AVGC
NUMSCANS **or** SCANS
INTEGRATEDJITTER_1 **or** IJ_1
INTEGRATEDJITTER_2 **or** IJ_2
INTEGRATEDJITTER_3 **or** IJ_3

Write-Only Operations

SCONFIGURATION **or** SCONFIG
RCONFIGURATION **or** RCONFIG

Miscellaneous Commands

(usable before OPEN)

Static Read-Only Identification and Capabilities

*IDN
ISSTATUS **or** ISST
RCSTATUS **or** RCST

Dynamic Read-Only Properties

SYSTEM:ERROR **or** SYST:ERR

Read-Write Settings

VIEW
GUILOCKOUT **or** GUIL
RSTATE **or** RST
RDURATION **or** RDUR

Write-Only Operations

*CLS
PTFILE **or** PTF
SCONFIGURATION **or** SCON
RCONFIGURATION **or** RCON

Command Reference

CRService : Clock Recovery Service Commands

CRSERVICE:ATTACHDEVCOUNT?
CRS:ADCT?
Retrieves the number of unique BERTScope CRs (unique serial numbers) that are currently communicating with the Clock Recovery Service, whether or not the remote computer has OPENed a session to any of them.

Returns: < numeric >

CRSERVICE:ATTACHDEVNAMES?
CRS:NAMES?
Retrieves a comma-separated list of all BERTScope CRs that are currently communicating with the Clock Recovery Service, whether or not the remote computer has OPENed a session to any of them.

Each BERTScope CR is identified by its user-assignable device name.

Returns:	<string>	Comma-separated list of BERTScopeCRs found;
		Or “” if none found.

CRSERVICE:CLOSE
CRS:CLOSE
Close the communication connection to the current BERTScopeCR.

Note that closing the “current” BERTScopeCR requires that a new current BERTScopeCR be selected (or opened) prior to issuing any more device-specific commands.

CRSERVICE:OPEN <string>
CRS:OPEN <string>
CRSERVICE:OPEN?
CRS:OPEN?
Selects or Returns the device name of the “current” BERTScopeCR,.

The user-assigned device name or serial number string may be passed to select a particular BERTScopeCR. An empty string may be passed to select the first (usually only) BERTScopeCR found. A BERTScopeCR can be opened by multiple clients. .

Param:	< string >	Serial Number, Device Name, or “”.
Returns:	< string >	Current Device Name, or “NONE”.

CRSERVICE:TOTALDEVCOUNT?
CRS:TDCT?
Retrieves the total number of unique BERTScope CRs (unique serial numbers) that the Clock Recovery Service has encountered since last restarted.

Note that this number may include devices that are no longer connected to the host computer, or are powered off.

Returns: < numeric >

CRControl : Clock Recovery Control Commands

CRCONTROL:AUTOSCONFIGDEVICE <bool>
CRC:AUTOSDEV <bool>
CRCONTROL:AUTOSCONFIGDEVICE?
CRC:AUTOSDEV?
Sets or retrieves the “Setup Auto-Save” flag state.

This flag determines whether or not the BERTScope CR restores its shutdown state at power-on.

Param:	< bool >	0 or 1
Returns:	< 0 >	Off
	< 1 >	On

CRCONTROL:BUSY?
CRC:BUSY?
Retrieves the busy status of the currently selected BERTScopeCR.

Returns:	< 0 >	Device is for available for communication
	< 1 >	Device is temporarily unavailable for communication.

CRCONTROL:CLOCKOUTPUT <bool>
CRC:CLKOUT <bool>
CRCONTROL:CLOCKOUTPUT?
CRC:CLKOUT?
Sets or retrieves whether the clock output is enabled.

Param:	< bool >	0 or 1
Returns:	< 0 >	Disabled
	< 1 >	Enabled

CRCONTROL:CLOCKAMPLITUDE <numeric>
CRC:CLKAMPL <numeric>
CRCONTROL:CLOCKAMPLITUDE?
CRC:CLKAMPL?
Sets or retrieves the clock output amplitude, in mV.

Param:	< numeric >	Range is to 250 mV to 900 mV
Returns:	< numeric >	Current clock output amplitude, in mV

CRCONTROL:CONNECTED?
CRC:CONN?
Retrieves the connection status of the currently selected BERTScopeCR.

Returns:	< 0 >	Device is opened, but not found by the host computer.
	< 1 >	Device is opened and available for communication.

CRCONTROL:CREATESTANDARD <string>
CRC:CREATESTANDARD <string>
Saves a new Clock Recovery Standard to the BERTScope CR’s non-volatile memory, using the current settings for **NOMFREQUENCY**, **LOCKRANGE**, **LOOPBANDWIDTH**, **PEAKING**, and **NOMEDGEDENSITY**.

IMPORTANT NOTE: The Standard name parameter is *case-sensitive*, and may contain space characters.

Param:	< string >	Name of the Standard to select

CRCONTROL:DATARATE?
CRC:DATARATE?
Retrieves the measured data rate, in Hz.

Returns: < numeric > Measured data rate, in Hz

CRCONTROL:DELETESTANDARD <string>
CRC:DELETESTANDARD <string>
Removes a Clock Recovery Standard from the BERTScope CR's non-volatile memory.

IMPORTANT NOTE: The Standard name parameter is *case-sensitive*, and may contain space characters.

IMPORTANT NOTE: The factory-supplied Standards are treated no differently than user-created Standards, and *can be deleted*. If accidentally deleted, a Standard can be manually re-created by pre-setting the desired values for **NOMFREQUENCY**, **LOCKRANGE**, **LOOPBANDWIDTH**, **PEAKING**, and **NOMEDGEDENSITY**, then sending **CRC:CREATESTANDARD** with the desired Standard name.

Param: < string > Name of the Standard to delete

CRCONTROL:DEVICENAME <string>
CRC:DN <string>
CRCONTROL:DEVICENAME?
CRC:DN?
Changes or retrieves the device name of the currently selected BERTScopeCR.

A device name can be from 1 to 11 characters in length.

Note that the BERTScopeCR's device name is saved in non-volatile memory.

Param:	< string >	New Device Name.
Returns:	< string >	Device Name of current device.

CRCONTROL:EDGEDENSITY?
CRC:EDGED?
Retrieves the measured edge density, in percent.

Returns: < numeric > Measured edge density, in %

CRCONTROL:EDGEDENSITYMODE <enum>
CRC:EDGEDMODE <enum>
CRCONTROL:EDGEDENSITYMODE?
CRC:EDGEDMODE?
Sets or retrieves the edge density mode.

Param:	<NOMINAL>	Use **NOMEDGEDENSITY** setting
	<ON_LOCK>	Use edge density found when first locked
Returns:	< NOMINAL > or <ON_LOCK >	

CRCONTROL:EXPREV?
CRC:EXP?
Retrieves the expansion board revision string of the currently selected BERTScopeCR.

Note that If no expansion board is installed in the BERTScopeCR, then "XXXX" is returned.

Returns: < string > Expansion board revision string.

CRCONTROL:FPGAREV?
CRC:FPGA?
Retrieves the FPGA revision string of the currently selected BERTScopeCR.

Returns: < string > FPGA revision string.

CRCONTROL:HWREV?
CRC:HW?
Retrieves the hardware revision string of the currently selected BERTScopeCR.

Returns: < string > Hardware revision string.

CRCONTROL:LOCKCOUNT?
CRC:LOCKCT?
Retrieves the number of times the device has relocked since last reset

Returns:	< numeric >	Number of times the device has relocked

CRCONTROL:LOCKMODE <enum>
CRC:LOCKMODE <enum>
CRCONTROL:LOCKMODE?
CRC:LOCKMODE?
Sets or retrieves the lock mode of the currently selected BERTScopeCR

Param:	<MANUAL>	Mode is manual
	<AUTOMATIC>	Mode is automatic
	<NARROW>	Mode is narrow
Returns:	< MANUAL > or < AUTOMATIC >	

CRCONTROL:LOCKRANGE <numeric>
CRC: RANGE <numeric>
CRCONTROL:LOCKRANGE?
CRC:RANGE?
Sets or retrieves the +/- lock range, in Hz, about the nominal frequency.

Param:	< numeric >	Range is 10e6 Hz to 500e6 Hz
Returns:	< numeric >	Current setting, in Hz

CRCONTROL:LOCKSTATE?
CRC:LOCKST?
Retrieves the lock state of the currently selected BERTScopeCR

Returns:	< UNLOCKED >	Unlocked
	< ACQUIRINGLOCK >	Acquiring lock
	< LOCKED >	Locked
	< HIGHJITTER >	High jitter locked

CRCONTROL:LOOPBANDWIDTH <numeric>
CRC:LOOPBW <numeric>
CRCONTROL:LOOPBANDWIDTH?
CRC:LOOPW?
Sets or retrieves the loop bandwidth in Hz.

Param:	< numeric >	Range is 100e3 Hz to 12e6 Hz
Returns:	< numeric >	Current setting, in Hz

CRCONTROL:NOMEDGEDENSITY<numeric>
CRC:NOMEDGED<numeric>
CRCONTROL:NOMEDGEDENSITY?
CRC:NOMEDGED?
Sets or retrieves the nominal edge density, in percent

Param:	< numeric >	Range is 10 % to 100 %
Returns:	< numeric >	Current setting, in %

CRCONTROL:NOMFREQUENCY <numeric>
CRC:NOMFREQ <numeric>
CRCONTROL:NOMFREQUENCY?
CRC:NOMFREQ?
Sets or retrieves the nominal frequency in Hz.

Param:	< numeric >	Range is 150e6 Hz to 12.5e9 Hz
Returns:	< numeric >	Current setting, in Hz

CRCONTROL:PEAKING <numeric>
CRC:PEAKING <numeric>
CRCONTROL:PEAKING?
CRC:PEAKING?

Sets or retrieves the peaking, in dB.

Param:	< numeric >	Range is to 0 dB to 6 dB
Returns:	< numeric >	Current setting, in dB

CRCONTROL:PHASEERRLIMIT <numeric>
CRC:PHERRLMT <numeric>
CRCONTROL:PHASEERRLIMIT?
CRC:PHERRLMT?

Sets or retrieves the the phase error limit, in percent unit interval.

Param:	< numeric >	Range is 10 %UI to 90 %UI
Returns:	< numeric >	Current setting in %UI

CRCONTROL:PHAERRP2P?
CRC:PHERR?

Retrieves the phase error peak-to-peak measurement, in percent unit interval.

Returns: < numeric >

CRCONTROL:PHAERRRMS?
CRC:PHERMS?

Retrieves the phase error RMS measurement, in percent unit intervals.

Returns: < numeric >

CRCONTROL:RCONFIGDEVICE <enum>
CRC:RDEV <enum>

Restores the BERTScope CR settings from one of the non-volatile Setup locations in the BERTScope CR.

Param:	< POWER_ON >	Configuration used at power-on
	< SETUP_1 >	
	< SETUP_2 >	
	< SETUP_3 >	
	< SETUP_4 >	
	< FACTORY >	Restore to the “factory default” configuration

CRCONTROL:RCONFIGDISK <string>
CRC:RDISK <string>

Restores the BERTScope CR settings from a file. The complete path must be specified in the parameter string. The path must be accessible to the host computer or BERTScope CR running the remote control software. The file extension should be ‘.CRU’.

Example:

CRC:RCONFIGDISK “D:\BitAlyzer\Configurations\mycfg.cru”

Param: < string > Pathname of the configuration file

CRCONTROL:RELOCK>
CRC:LOCK

Initiates a search for the data rate.

IMPORTANT NOTE: This command may take some time to complete. An application program would typically call this, then repeatedly query **LOCKSTATE** until lock is achieved.

CRCONTROL:RESETLOCKCOUNT
CRC:RESETLOCKCT

Resets the lock count to zero.

CRCONTROL:SCONFIGDEVICE <enum
CRC:SDEV <enum>
Saves the current BERTScope CR settings to one of the non-volatile Setup locations in the BERTScope CR.

Param:	< POWER_ON >	Configuration used at power-on
	< SETUP_1 >	
	< SETUP_2 >	
	< SETUP_3 >	
	< SETUP_4 >	

CRCONTROL:SCONFIGDISK <string>
CRC:SDISK <string>
Saves the current BERTScope CR settings to a file. The complete path must be specified in the parameter string. The path must be accessible to the host computer or BERTScope running the remote control software. The file extension should be '.CRU'.

Example:

CRC:SCONFIGDISK "D:\BitAlyzer\Configurations\mycfg.cru"

Param: < string > Pathname of the configuration file

CRCONTROL:SERIALNUM?
CRC:SN?
Retrieves the serial number string of the currently selected BERTScopeCR.

Returns: < string > Serial Number string of current device.

CRCONTROL:STANDARD <string>
CRC:STANDARD <string>
CRCONTROL:STANDARD?
CRC:STANDARD
Changes or retrieves the name of the Clock Recovery Standard.

Selecting a new Standard will change the BERTScope CR's settings for **NOMFREQUENCY**, **LOCKRANGE**, **LOOPBANDWIDTH**, **PEAKING**, and **NOMEDGEDENSITY**.

IMPORTANT NOTE: The Standard name parameter is *case-sensitive*, and may contain space characters.

Param:	< string >	Name of the Standard to select
Returns:	< string >	Currently selected Standard, or "None"

CRCONTROL:STANDARDLIST?
CRC:STANDARDS?
Retrieves a comma-separated list of the available Standards.

IMPORTANT NOTE: The Standard names are *case-sensitive*, and may contain space characters.

Returns: < string > The standards are comma separated in the string.

CRCONTROL:SWREV?
CRC:SW?

Retrieves the firmware revision string of the currently selected BERTScopeCR.

Returns:	< string >	Firmware revision string.

CRCONTROL:SUBCLOCKAMPLITUDE <numeric>
CRC:SUBAMPL <numeric>
CRCONTROL:SUBCLOCKAMPLITUDE?
CRC:SUBAMPL?

Sets or retrieves the sub-rate clock amplitude, in mV.

Param:	< numeric >	Range is 250 mV to 900 mV
Returns:	< numeric >	Current sub-rate clock output amplitude, in mV

CRCONTROL:SUBCLOCKDIV <numeric>
CRC:SUBDIV <numeric>
CRCONTROL:SUBCLOCKDIV?
CRC:SUBDIV?

Sets or retrieves the sub-rate clock divisor.

Param:	< numeric >	The divisor used by the sub-rate clock. (Full rate = 1) 1, 2, 4, 5, 6, 7, 8, 9, 10,12, 14, 16, 18, 20, 24, 25, 28, 30, 32, 35, 36, 40, 42, 45, 48, 49, 50, 54, 56, 60, 63, 64, 70, 72, 80, 81, 90, 100, 108, 112, 120, 126, 128, 140, 144, 160, 162, 168, 180, 192, 196, 200, 216, 224, 240, 252, 256, 280, 288, 320, 324, 336, 360, 384, 392, 432, 448, 504, 512, 576, 648
Returns:	< numeric >	Current sub-rate clock divisor setting

CRCONTROL:SUBCLOCKDIVLIST?
CRC:SUBDIVS?

Retrieves a comma-separated list of available sub-rate clock divisors.

Returns:	< string >	The divisors are comma-separated in the string.

CRCONTROL:SUBCLOCKOUTPUT <bool>
CRC:SUBOUT <bool>
CRCONTROL:SUBCLOCKOUTPUT?
CRC:SUBOUT?

Sets or retrieves sub-rate clock output enable state.

Param:	< bool >	0 or 1
Returns:	< 0 >	Disabled
	< 1 >	Enabled

CRLoopResponse : Clock Recovery Loop Response Commands

CRLOOPRESPONSE:BANDWIDTH?
CRL:BANDWIDTH?

Retrieves the measured Loop Response Bandwidth (Amplitude chart type only).

Returns:	< numeric >	Bandwidth in Hertz.

CRLOOPRESPONSE:CHARTTYPE <enum>
CRL:CHARTTYPE <enum>
CRLOOPRESPONSE: CHARTTYPE?
CRL:CHARTTYPE?

Sets or retrieves the Loop Response Chart Type.

Param:	<CALIBRATION>	Calibration chart type
	<AMPLITUDE>	Amplitude chart type
	<PHASE>	Phase chart type
Returns:	<CALIBRATION>, <AMPLITUDE>, or <PHASE>	

CRLOOPRESPONSE:PEAKFREQ?
CRL:PEAKFREQ?

Retrieves the measured Loop Response Peak Frequency (Amplitude chart type only).

Returns:	< numeric >	Peak Frequency in Hertz.

CRLOOPRESPONSE:PEAKING?
CRL:PEAKING?

Retrieves the measured Loop Response Peaking (Amplitude chart type only).

Returns:	< numeric >	Peaking in dB.

CRSSCWaveform : SSC Waveform Commands

CRSSCWAVEFORM:AVGCOUNT?
SSCW:AVGC?
Retrieves the number of averages completed.

Returns:	< numeric >	Averages completed.

CRSSCWAVEFORM:AVGSETTING <numeric>
SSCW:AVGS <numeric>
CRSSCWAVEFORM:AVGSETTING?
SSCW:AVGS?
Sets or retrieves the number of running averages to perform.

Param:	< numeric >	Range is 1 to 16
Returns:	< numeric >	Current Average Setting

CRSSCWAVEFORM:DEVMAX_HZ?
SSCW:DEVMAX_HZ?
The measured average of maximum frequency relative to the CR Nominal Frequency.

Returns:	< numeric >	Frequency deviation, in Hertz.

CRSSCWAVEFORM:DEVMAX_PPM?
SSCW:DEVMAX?
The measured average of maximum frequency relative to the CR Nominal Frequency.

Returns:	< numeric >	Frequency deviation, in parts per million.

CRSSCWAVEFORM:DEVMIN_HZ?
SSCW:DEVMIN_HZ?
The measured average of minimum frequency relative to the CR Nominal Frequency.

Returns:	< numeric >	Frequency deviation, in Hertz.

CRSSCWAVEFORM:DEVMIN_PPM?
SSCW:DEVMIN?
The measured average of minimum frequency relative to the CR Nominal Frequency.

Returns:	< numeric >	Frequency deviation, in parts per million.

CRSSCWAVEFORM:HISTOGRAMPOINTS?
SSCW:HISTOPTS?
Retrieves the number of points per histogram used when computing the average.

Returns:	< numeric >	Number of points.

CRSSCWAVEFORM:MODFREQ?
SSCW:MODFREQ?
The measured modulation frequency of the SSC waveform.

Returns:	< numeric >	Frequency, in Hertz.

CRSSCWAVEFORM:NOMFREQ?
SSCW:NOMFREQ?
The measured nominal clock frequency, in Hertz.

Returns:	< numeric >	Frequency, in Hertz.

CRSSCWAVEFORM:NUMHISTOGRAMS?
SSCW:HISTOS?
Retrieves the number of histograms to used when computing the average.

Returns:	< numeric >	Number of histograms.

CRSSCWAVEFORM:NUMSCANS?
SSCW:SCANS?
Retrieves the total number of scans completed.

Returns:	< numeric >	Scans completed.

CRSSCWAVEFORM:RCONFIGURATION <string>
SSCW:RCONFIG <string>
Restores the SSC Waveform settings from a file. The complete path must be specified in the parameter string. The path must be accessible to the host computer or BERTScope running the remote control software. The file extension should be '.SSC'.

Example:

SSCW:RCONFIG "D:\BitAlyzer\Configurations\mycfg.ssc"

Param:	< string >	Pathname of the configuration file

CRSSCWAVEFORM:SCONFIGURATION <string>
SSCW:SCONFIG <string>
Saves the current SSC Waveform settings to a file. The complete path must be specified in the parameter string. The path must be accessible to the host computer or BERTScope running the remote control software. The file extension should be '.SSC'.

Example:

SSCW:SCONFIG "D:\BitAlyzer\Configurations\mycfg.ssc"

Param:	< string >	Pathname of the configuration file

CRJitterSpectrum : Jitter Spectrum Commands (Model CRJ Only)

CRJITTERSPECTRUM:AVGCOUNT?
JS:AVGC?
Retrieves the number of averages completed.

Returns:	< numeric >	Averages completed.

CRJITTERSPECTRUM:AVGSETTING <numeric>
JS:AVGS <numeric>
CRJITTERSPECTRUM:AVGSETTING?
JS:AVGS?
Sets or retrieves the number of running averages to perform.

Param:	< numeric >	Range is 1 to 16
Returns:	< numeric >	Current Average Setting

CRJITTERSPECTRUM:CRINITMODE <enum>
JS:CRINIT <enum>
CRJITTERSPECTRUM:CRINITMODE?
JS:CRINIT?
Sets or retrieves the way the BERTScope CR is initialized prior to a Jitter Spectrum session.

Param:	<AUTO>	First tries to use _NO_SSC, then SSC_CAPABLE settings
	<NO_SSC>	Default Settings that for signals without SSC
	<SSC_CAPABLE>	Settings that will accommodate input signals with SSC
Returns:	<AUTO>, <NO_SSC>, or <SSC_CAPABLE>	

CRJITTERSPECTRUM:IJUNITS <enum>
JS:IJUNITS <enum>
CRJITTERSPECTRUM:IJUNITS?
JS:IJUNITS?
Sets or retrieves the units to be used for Jitter Spectrum Amplitude and IJ measurements.

Param:	<PERCENTUI>	Percent of Unit Interval
	<PICOSECS>	Pico-seconds
Returns:	<PERCENTUI>, or <PICOSECS>	

CRJITTERSPECTRUM:IJMAXHZ_1 <double>
JS: IJMAXHZ_1 <double>
CRJITTERSPECTRUM: IJMAXHZ_1?
JS: IJMAXHZ_1?
Sets or retrieves the upper modulation frequency boundary for Integrated Jitter range 1.

Param:	<double>	Upper modulation frequency boundary, in Hertz
Returns:	Current upper modulation frequency boundary, in Hertz	

CRJITTERSPECTRUM:IJMINHZ_1 <double>
JS: IJMINHZ_1 <double>
CRJITTERSPECTRUM: IJMINHZ_1?
JS: IJMINHZ_1?
Sets or retrieves the lower modulation frequency boundary for Integrated Jitter range 1.

Param:	<double>	Lower modulation frequency boundary, in Hertz
Returns:	Current lower modulation frequency boundary, in Hertz	

CRJITTERSPECTRUM:IJMODE_1 <enum>
JS: IJMODE_1 <enum>
CRJITTERSPECTRUM: IJMODE_1?
JS: IJMODE_1?
Sets or retrieves the mode to be used for IJ measurements in IJ range 1.

Param:	<PK-PK>	Peak-to-Peak measurement mode
	<RMS>	Root-Mean-Squared measurement mode
Returns:	<PK-PK>, or <RMS>	

CRJITTERSPECTRUM:IJMAXHZ_2 <double>
JS: IJMAXHZ_2 <double>
CRJITTERSPECTRUM: IJMAXHZ_2?
JS: IJMAXHZ_2?
Sets or retrieves the upper modulation frequency boundary for Integrated Jitter range 2.

Param: <double> Upper modulation frequency boundary, in Hertz
Returns: Current upper modulation frequency boundary, in Hertz

CRJITTERSPECTRUM:IJMINHZ_2 <double>
JS: IJMINHZ_2 <double>
CRJITTERSPECTRUM: IJMINHZ_2?
JS: IJMINHZ_2?
Sets or retrieves the lower modulation frequency boundary for Integrated Jitter range 2.

Param: <double> Lower modulation frequency boundary, in Hertz
Returns: Current lower modulation frequency boundary, in Hertz

CRJITTERSPECTRUM:IJMODE_2 <enum>
JS: IJMODE_2 <enum>
CRJITTERSPECTRUM: IJMODE_2?
JS: IJMODE_2?
Sets or retrieves the measurement mode to be used for IJ measurements in IJ range 2.

Param:	<PK-PK>	Peak-to-Peak measurement mode
	<RMS>	Root-Mean-Squared measurement mode
Returns:	<PK-PK>, or <RMS>	

CRJITTERSPECTRUM:IJMAXHZ_3 <double>
JS: IJMAXHZ_3 <double>
CRJITTERSPECTRUM: IJMAXHZ_3?
JS: IJMAXHZ_3?
Sets or retrieves the upper modulation frequency boundary for Integrated Jitter range 3.

Param: <double> Upper modulation frequency boundary, in Hertz
Returns: Current upper modulation frequency boundary, in Hertz

CRJITTERSPECTRUM:IJMINHZ_3 <double>
JS: IJMINHZ_3 <double>
CRJITTERSPECTRUM: IJMINHZ_3?
JS: IJMINHZ_3?
Sets or retrieves the lower modulation frequency boundary for Integrated Jitter range 3.

Param: <double> Lower modulation frequency boundary, in Hertz
Returns: Current lower modulation frequency boundary, in Hertz

CRJITTERSPECTRUM:IJMODE_3 <enum>
JS: IJMODE_3 <enum>
CRJITTERSPECTRUM: IJMODE_3?
JS: IJMODE_3?
Sets or retrieves the mode to be used for IJ measurements in IJ range 3.

Param:	<PK-PK>	Peak-to-Peak measurement mode
	<RMS>	Root-Mean-Squared measurement mode
Returns:	<PK-PK>, or <RMS>	

CRJITTERSPECTRUM:INTEGRATEDJITTER_1?
JS:IJ_1?
Retrieves the integrated jitter measurement for IJ range 1.

Returns:	< numeric >	Either %UI or ps, depending upon JS:IJUNITS setting Either PK-Pk or RMS, depending on JS:IJMODE_1

CRJITTERSPECTRUM:INTEGRATEDJITTER_2?
JS:IJ_2?
Retrieves the integrated jitter measurement for IJ range 2.

Returns:	< numeric >	Either %UI or ps, depending upon JS:IJUNITS setting Either PK-Pk or RMS, depending on JS:IJMODE_2

CRJITTERSPECTRUM:INTEGRATEDJITTER_3?
JS:IJ_3?
Retrieves the integrated jitter measurement for IJ range 3.

Returns:	< numeric >	Either %UI or ps, depending upon JS:IJUNITS setting Either PK-Pk or RMS, depending on JS:IJMODE_3

CRJITTERSPECTRUM:NUMSCANS?
JS:SCANS?
Retrieves the total number of scans completed.

Returns:	< numeric >	Scans completed.

CRJITTERSPECTRUM:RCONFIGURATION <string>
JS:RCONFIG <string>
Restores the Jitter Spectrum settings from a file. The complete path must be specified in the parameter string. The path must be accessible to the host computer or BERTScope running the remote control software. The file extension should be '.CJS'.

Example:

JS:RCONFIG "D:\BitAlyzer\Configurations\mycfg.cjs"

Param:	< string >	Pathname of the configuration file

CRJITTERSPECTRUM:SCANRESMODE <enum>
JS:SCANRES <enum>
CRJITTERSPECTRUM: SCANRESMODE?
JS:SCANRES?
Sets or retrieves the scan resolution for the Jitter Spectrum session.

Param:	<AUTO>	Chooses the NORMAL res if USB 1.x, HIGHRES if USB 2.0
	<NORMAL>	Suitable for peak detection but not IJ measurement
	< HIGHRES >	4x the resolution of NORMAL mode
Returns:	<AUTO>, <NORMAL>, or <HIGHRES>	

CRJITTERSPECTRUM:SCONFIGURATION <string>
JS:SCONFIG <string>
Saves the current Jitter Spectrum settings to a file. The complete path must be specified in the parameter string. The path must be accessible to the host computer or BERTScope running the remote control software. The file extension should be '.CJS'.

Example:

JS:SCONFIGDISK "D:\BitAlyzer\Configurations\mycfg.cjs"

Param:	< string >	Pathname of the configuration file

Miscellaneous Commands

***CLR>**
Clear the system status queue.

***IDN?**
Retrieves a string identifying the Host computer.

Returns:	< string >	Host computer identity

GUILOCKOUT <bool>
GUIL <bool>
GUILOCKOUT?
GUIL?
Sets or retrieves the “GUI Lock-out” flag state.
The user interface of the BERTScope (CR) software will not response to input if this flag is set On.

Param:	< bool >	0 or 1
Returns:	< 0 >	GUI is not locked
	< 1 >	GUI is locked

ISSTATUS?
ISST?
Retrieves a string identifying the version of the BERTScope CR software running on the Host computer or BERTScope.

Returns:	< string >	“Instrument Server Version MM.mm”, Where MM is the Major version number Where mm is the minor version number

PTFILE <string>
PTF <string>
Save a screen capture of the BERTScope (CR) software to a file. The complete path must be specified in the parameter string. The path must be accessible to the Host computer or BERTScope running the remote control software. The file extension should be ‘.jpg’.

Example:

RCON “C:\mycfg.jpg”

Param:	< string >	Pathname of the image file

RCONFIGURATION <string>
RCON <string>
Restores the complete BERTScope (CR) state from a file. The complete path must be specified in the parameter string. The path must be accessible to the Host computer or BERTScope running the remote control software. The file extension should be ‘.CFG’.

Example:

RCON “D:\BitAlyzer\Configurations\mycfg.cfg”

Param:	< string >	Pathname of the configuration file

RCSTATUS?
RCST?
Retrieves a string identifying the version of the RemoteClient software running on the Host computer or BERTScope.

Returns:	< string >	Remote Client Version MM.mm”, Where MM is the Major version number Where mm is the minor version number

RDURATION <long>
RDUR <long>
RDURATION?
RDUR?
Set or retrieve the run duration of the BERTScope.
Setting to ‘0’ allows the run to go on ‘forever.’

Param:	<long>	Set system run duration, in seconds. Range is [1 to 36E6]. Zero (0) sets an infinite duration.
Returns:	<long >	System run duration, in seconds

RSTATE <bool>
RST <bool>
RSTATE?
RST?
Sets or retrieves the run state of the BERTScope (CR) software.

Param:	< bool >	0 or 1
Returns:	< 0 >	System is not running
	< 1 >	System is running

SCONFIGURATION <string>
SCON <string>
Saves the complete state of the BERTScope (CR) to a file. The complete path must be specified in the parameter string. The path must be accessible to the host computer or BERTScope running the remote control software. The file extension should be ‘.CFG’.

Example:

SCON “D:\BitAlyzer\Configurations\mycfg.cfg”

Param:	< string >	Pathname of the configuration file

SYSTEM:ERROR?
SYST:ERR?
Retrieve errors listed in the system status queue.
This command will not show in the list of trace messages displayed on the remote control client user interface. The status queue is first in, first out. It can contain up to 100 error messages. If the queue overflows, the last error/event in the queue is replaced with Error –350, “Queue overflow”. When all errors/events have been read from the queue, further error/event queries will return 0, “NO error”.

Returns:	< string >	n, “event/string”

VIEW <enum>
VIEW?
Sets or retrieves the current view displayed by the BERTScope (CR) software.
NOTE: the choices below assume the Host computer is NOT a BERTScope.
If the Host computer is a BERTScope, many more views are available.
See the “BERTScope Remote Control Guide” for a complete list.

Param:	< CRCONTROL> or <CRC>	Clock Recovery Control View
	<CRLOOPRESPONSE> or <CRLR>	Clock Recovery Loop Response View
	<CRSSCWAVEFORM> or <SSCW>	SSC Waveform View
	<CRJITTERSPECTRUM> or <JS>	Jitter Spectrum View (Model CRJ Only)
Returns:	< CRCONTROL> or	
	<CRLOOPRESPONSE> or	
	<CRSSCWAVEFORM> or	
	<CRJITTERSPECTRUM> (Model CRJ Only)	

Sample Program

```
// CR_RC.c : Commandline program to execute single command over TCP/IP

#include <windows.h>
#include <winsock.h>
#include <stdio.h>

#define PORTNUM       23           // Port number
#define HOSTNAME      "localhost"  // Server name string

SOCKET ServerSock = INVALID_SOCKET; // Socket bound to the server
SOCKADDR_IN destination_sin;        // Server socket address
PHOSTENT phostent = NULL;           // HOSTENT structure of server
WSADATA WSAData;                    // Contains details of the
                                    // Winsocket implementation

static BOOL OpenSocket();
static BOOL CloseSocket();
static BOOL SendCommand( TCHAR * pszCmd );
static BOOL GetReply( TCHAR * pszReply );

int main( int argc, char* argv[] )
{
    char szCmd[1024];
    char szReply[1024];

    memset( szCmd, 0, sizeof(szCmd) );
    memset( szReply, 0, sizeof(szReply) );

    if ( argc > 1 )
    {
        int i;
        BOOL retval = FALSE;

        for ( i = 1; i < argc; i++ )
        {
            strcat( szCmd, argv[i] );
            if ( i+1 < argc )
                strcat( szCmd, " " );
        }
        if ( !OpenSocket() )
            return FALSE;
        if ( !SendCommand( "*CLS" ) ) // clear error queue
            goto Exit;
        if ( !SendCommand( szCmd ) )
            goto Exit;
        if ( strchr( szCmd, '?' ) != NULL )
        {
            if ( !GetReply( szReply ) )
                goto Exit;
            else
            {
                printf( "%s", szReply );
                retval = TRUE;
            }
        }
        else // not query, so check command status
        {
            BOOL bErrs = FALSE;
            while(1)
            {
                if ( !SendCommand( "system:err?" ) )
                    goto Exit;
                if ( !GetReply( szReply ) )
                    goto Exit;
                else
                {
                    if ( strstr( szReply, "NO ERROR" ) == NULL )
                    {
                        printf( "%s", szReply );
                        bErrs = TRUE;
                    }
                    else // no more errors in queue
                    {
                        retval = !bErrs;
                        break;
                    }
                }
            }
        }
Exit:
        CloseSocket();
        return retval;
    }
    else // help
    {
        printf( "Expecting 'feature:operation parameter'\n" );
        printf( "or        'feature:operation?'\n" );
    }
    return TRUE;
}
```

```
static BOOL OpenSocket()
{
    // Initialize Winsocket.
    if (WSAStartup (MAKEWORD(1,1), &WSAData) != 0)
    {
        printf( "WSAStartup failed. Error: %d\n",
                WSAGetLastError() );
        return FALSE;
    }
    // Create a TCP/IP socket that is bound to the server.
    if ( (ServerSock = socket(AF_INET, SOCK_STREAM, 0) )
         == INVALID_SOCKET )
    {
        printf( "Allocating socket failed. Error: %d\n",
                WSAGetLastError() );
        return FALSE;
    }

    // Fill out the server socket's address information.
    destination_sin.sin_family = AF_INET;

    // Retrieve the host information corresponding to the host name.
    if ((phostent = gethostbyname (HOSTNAME)) == NULL)
    {
        printf( "Unable to get the host name. Error: %d\n",
                WSAGetLastError() );
        closesocket (ServerSock);
        return FALSE;
    }

    // Assign the socket IP address.
    memcpy( (char FAR *)&(destination_sin.sin_addr),
            phostent->h_addr,
            phostent->h_length);

    // Convert to network ordering.
    destination_sin.sin_port = htons (PORTNUM);

    // Establish a connection to the server socket.
    if ( connect( ServerSock,
                  (PSOCKADDR)&destination_sin,
                  sizeof(destination_sin)
                ) == SOCKET_ERROR )
    {
        printf( "Connecting to the server failed. Error: %d\n",
                WSAGetLastError() );
        closesocket (ServerSock);
        return FALSE;
    }
    return TRUE;
}

static BOOL CloseSocket()
{
    shutdown (ServerSock, 0x01); // Disable sending
    shutdown (ServerSock, 0x00); // Disable receiving
    closesocket (ServerSock);
    WSACleanup ();
    return TRUE;
}

static BOOL SendCommand( TCHAR * pszCmd )
{
    // Send a string to the server.
    if ( send( ServerSock, pszCmd, strlen(pszCmd) + 1, 0
             ) == SOCKET_ERROR )
    {
        printf( "Sending data to the server failed. Error: %d\n",
                WSAGetLastError() );
        return FALSE;
    }
    else
        return TRUE;
}

static BOOL GetReply( TCHAR * pszReply )
{
    // Receive data from the server socket.
    int iReturn = recv( ServerSock, pszReply, 1024, 0 );
    if ( iReturn == SOCKET_ERROR )
        sprintf( pszReply, "No data is received, recv failed. Error: %d",
                 WSAGetLastError() );
    return iReturn != SOCKET_ERROR;
}
```

Index