Programmer's Reference Guide for 8810A

Table of Contents

1 Introduction

The 8810A Angle Position Indicator Measurement Instrument provides two fully independent channels with 0.0001° resolution and 0.004° (Standard)/ 0.001° (Special) accuracy. The 8810A allows all programming to be done via the touch-screen or mouse interface. In addition, remote operation capabilities are provided via IEEE-488, USB, Ethernet and J1 connection (50 pin DSUB connector in back of the unit).

Reference Documentation

For additional information about this instrument refer to the *Operation Manual for Model 8810A*. For additional information about the Application Programming Interface (API) provided in the API8810ADll refer to the *Function Reference Manual for 8810A.*

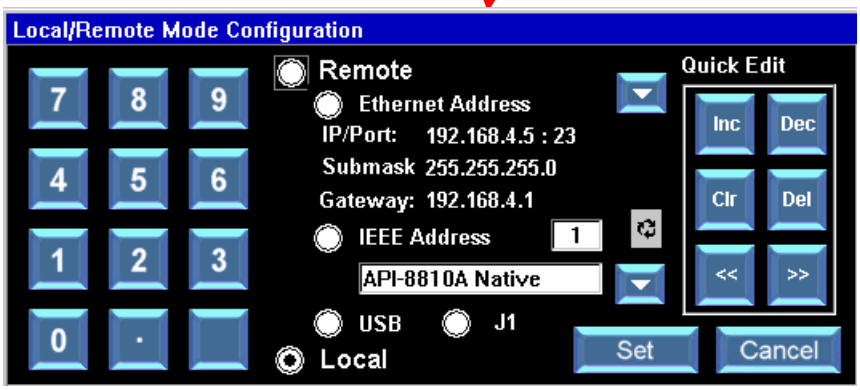
Reference CD

For electronic copies of the 8810A documentation, API-8810A Soft Panel application program, and source code for API-8810ADll and Soft Panel application refer to the 8810A Product CD.

2 Remote Setup

To enable remote operation capabilities via IEEE-488, USB, Ethernet and J1 connection (50 pin DSUB connector in back of the unit), the unit must be configured for remote operation.

Click on the button labeled “Loc/Rem” to view the Local/Remote Configuration screen:

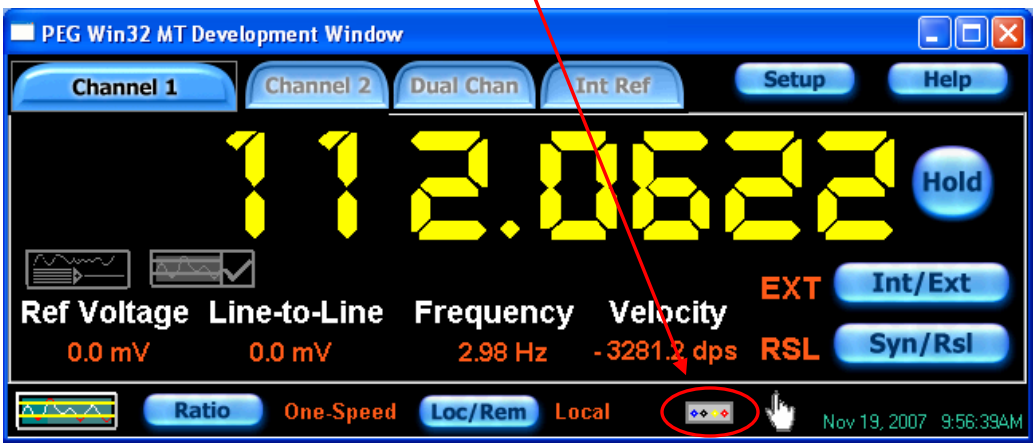


In “Local” mode, the configuration settings can be queried. The unit must be in one of the “Remote” modes (Ethernet, IEEE, USB or J1) before configuration settings can be changed remotely. Note, for remote programming via the IEEE interface, the language type must be selected, refer to section 3 on language support.

2.1 Controlling Channel 1 Signal Input

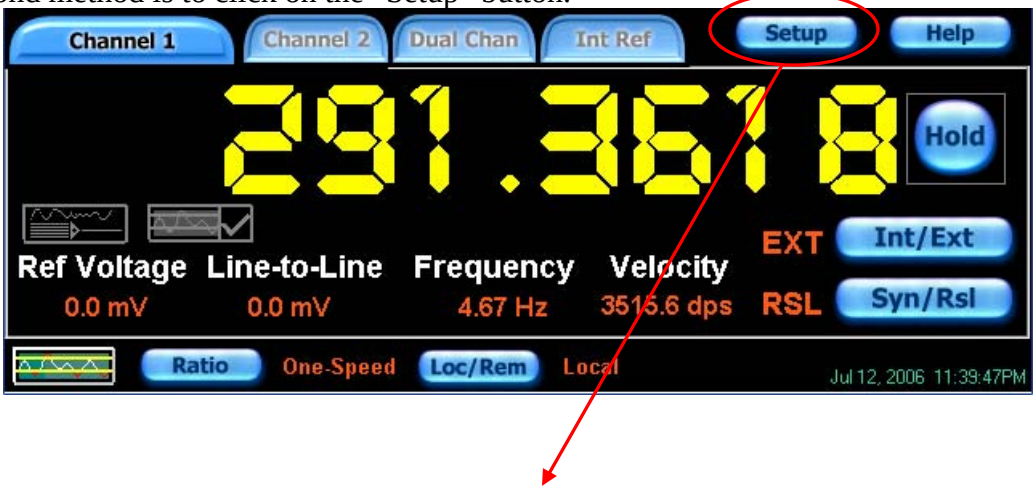
The signal input for channel 1 can be read from the front panel connector or from the J1 connection in the back of the unit. The signal input for channel 2 is read only from the J1 connection.

The configuration for channel 1 is configured two ways:
One method is to click the button shown below:



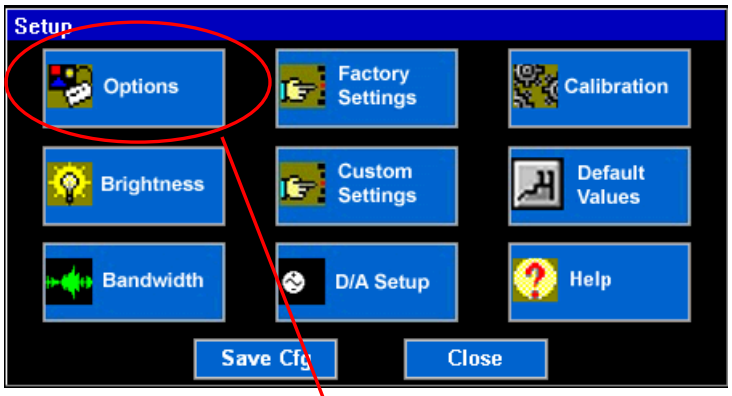
Button configures Chan 1 Input to be read from the Front Connector or the Back (J1) Connector.

The second method is to click on the "Setup" button.

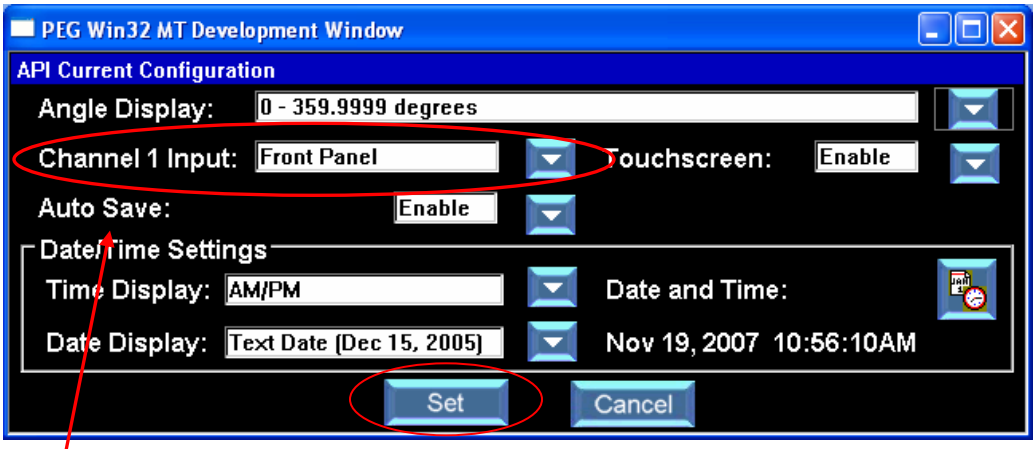


(See next page)

Click the “Options” button.



Select the “Front Panel” or “Back Connector” option for Channel 1 Input. Click on the “Set” button.



Note: the “Auto Save” option is not available on 8810A Revision B instruments.

3 Language Support

The 8810A Unit is a direct replacement for all 8810's. This unit supports the following languages:

	IEEE-488.1	USB	Ethernet
API-8810A Native	**Supported**	**Supported**	**Supported**
API-8810 Native (Legacy)	**Supported**	Not available	Not available
API-8810 SR103 (Legacy)	**Supported**	Not available	Not available
API-8810 HSR202 (Legacy)	**Supported**	Not available	Not available
API-8810 HSR203 (Legacy)	**Supported**	Not available	Not available
API-8810 MATE/CIIL (Legacy)	Please contact manufacturer for MATE/CIIL support	Not available	Not available

The following table lists the applicable IEEE-488 bus commands for the API-8810A.

Mnemonic	ASCII	Hex	Function
GTL	SOH	01	Go To Local - This command instructs the API to go to local mode. All front panel controls are active.
SDC	EOT	04	Selected Device Clear - When the SDC command is received, and if the API is addressed to listen, the API will initialize to the conditions listed under DCL.
DCL	DC4	14	Device Clear - When the API receives the DCL command it is initialized to the following state: SYNCHRO DATA FREEZE - OFF SRQ MODE - OFF GET MODE – OFF
GET	BS	08	Group Execute Trigger - When the GET command is received, and if the API is addressed to listen and has the GET mode switch on, data sent to the API will be applied to the instrument.
LLO	DC1	11	Local Lockout - This command disables the front panel REM switch. It gives the controller complete control over whether the API is in remote or local operation.
SPE	CAN	18	Serial Poll Enable - After receipt of this command the API, when addressed to talk, will transmit the Status Byte.
SPD	EM	19	Serial Poll Disable - This command cancels the SPE command and allows the API, when it is addressed to talk, to send data.
UNL	/	3F	Unlisten - Unaddresses the API listen address.
UNT	-	5F	Untalk - Unaddresses the API talk address.

The following table lists the interface function capability codes for the API-8810A.

Code	Function
AH1	Acceptor handshake - complete capability
SH1	Source handshake - complete capability
T6	Talk capability - all except TON
TEO	Extended Talk capability – none
L4	Listen capability - all except LON
LEO	Extended Listen capability - none
SR1	Service request - complete capability
RL1	Remote/Local - complete capability
PPO	Parallel Poll - no capability
DC1	Device Clear - complete capability
DT1	Device Trigger - complete capability

3.1 Compatibility to 8810 APIs

The 8810A will provide language compatibility to the following 8810 systems:

- API-8810 Native
- API-8810 SR103
- API-8810 HSR202
- API-8810 HSR203

Legacy 8810 Data

When the IEEE language type selected is one of the 8810 legacy languages, the API will send angle data to the controller in the following format:

<DDDDDD<CR><LF>
e.g., <179999<CR><LF>

The standard API data message will always be 7 characters long plus a <cr><lf>. The API when used in the ±180 mode will send angle data to the IEEE controller in the following format:

<SDDDDDD<CR><LF>
e.g., <-149999<CR><LF>

The ±180 degree data message will always be eight characters long plus a <CR><LF>.

Serial Poll

When the IEEE language type selected is one of the 8810 legacy languages, the status byte returned by the API indicates the status of the instrument. The bits of the status byte are defined as:

D7	D6	D5	D4	D3	D2	D1	D0
ERROR	RQS	0	0	0	0	FREEZE	RESOLVER

ERROR -When bit is set the API data is not stable. If FREEZE is programmed, this bit will always be 0.

RQS -When bit is set the API is asserting the SRQ line.

FREEZE -When bit is set the display is frozen.

RESOLVER -When bit is set the API is programmed for RESOLVER mode. When cleared the API is set to SYNCHRO mode.

If the RQS bit is set, the remaining bits indicate the state of the API when the SRQ line was last asserted. If the RQS line is not set then the remaining bits indicate the state of the API at the time the status byte is read.

Service Request The API can be programmed to assert the SRQ line when the display data is stable. Stability is defined as the angle readings being within <TBD> degrees or the FREEZE mode is programmed. The V command instructs the API to assert the SRQ line when stable data is detected. If stability is not detected within 4 seconds, SRQ will be asserted nevertheless and the ERROR bit in the STATUS byte will be set. This command cancels itself once SRQ is asserted and must be reprogrammed for subsequent SRQs. When SRQ is asserted the display data is saved and will be transmitted to the controller (when addressed to talk) regardless of the display value. Once read, the API output data will then agree with the display.

GET Mode When the G command is included in the programming string, the API will hold off applying the programming data until the GET (Group Executive Trigger) bus command is received. GET mode is cancelled once the bus command GET is received and must be reprogrammed if desired again.

3.2 Language Independent Commands

Note the following commands are case-sensitive.

Function	Syntax (commands must be sent with upper-case)	Comments
API COMMANDS		
Identification	*IDN?<cr><lf>	Queries the device for the ID.
Error Reporting	*ERR?<cr><lf>	Queries for any error messages on the error message queue. "No error" is returned when there are no errors on the queue.
Reset	*RST?<cr><lf>	Clears the error message queue and resets the device with power-on or last saved configuration settings.
Language	APICMD<b>LANG?<cr><lf>	Queries the IEEE Language setting. Query returns: '8810ANATIVE', or '8810NATIVE', or '8810SR103', or '8810HSR202', or '8810HSR203', or "8810MATECIIL'
	APICMD<b>LANG<b>< 8810ANATIVE|8810NATIVE| 8810SR103|8810HSR202|8810HSR203|8810MATECIIL> <cr><lf>	Sets the IEEE Language setting.

3.3 API-8810A Native

The API-8810A Native language is support via the IEEE-488.1, USB and Ethernet interfaces. The language provides remote programming access to the features available on the 8810A unit. Note the following commands are case-sensitive.

API CHANNELS		
Function	**Syntax (commands must be sent with upper-case)**	**Comments**
Track/Latch Angle	API<chan><b>UPDATE?<cr><lf>	Queries the track/latch state of the channel. Query returns: 'LATCHED' or 'TRACK'.
	API<chan><b>UPDATE<b><LATCH|TRACK><cr><lf>	Sets the track/latch state for the channel.
Signal Mode	API<chan><b>MODE?<cr><lf>	Queries the mode state of the channel. Query returns: 'RSL' or SYN'.
	API<chan><b>MODE<b><RSL|SYN><cr><lf>	Sets the Rsl/Syn state for the channel.
Reference Mode	API<chan><b>REF_SOURCE?<cr><lf>	Queries the reference source for the channel. Query returns: 'INT' or 'EXT'.
	API<chan><b>REF_SOURCE<b><INT|EXT><cr><lf>	Sets the Internal/External reference source mode for the channel.
Ratio	API<chan><b>RATIO?<cr><lf>	Queries the ratio setting for each channel. Query returns the '1' always for channel 1 and for channel 2, the ratio setting value: Ratio Range = 1 to 255.

	API<chan><b>RATIO<b><value><cr><lf>	Sets the ratio setting for each channel. Channel 1 can only be set to 1. Channel 2 can be set to any value between 1 and 255.
Bandwidth	API<chan><b>BANDWIDTH?<cr><lf>	Queries the bandwidth setting for each channel. Query returns: AUTO bwValue or OVERRIDE bwValue
	API<chan><b>BANDWIDTH<b><AUTO><cr><lf>	Sets the channel to auto-bandwidth mode.
	API<chan><b>BANDWIDTH<b><SET><value><cr><lf>	Sets the channel bandwidth value Range: 6 <= value <= 1200
Angle Averaging	API<chan><b>AVERAGE?<cr><lf>	Queries the averaging setting for the channel. Query returns: ON avgRate or OFF avgRate
	API<chan><b>AVERAGE<b>STATE<b><ON|OFF><cr><lf>	Turns on or off the channel's averaging feature.
	API<chan><b>AVERAGE<b>RATE?<cr><lf>	Queries the averaging setting for the averaging rate in msec for the channel.
	API<chan><b>AVERAGE<b>RATE<b><value><cr><lf>	Set the channel's averaging rate in msec. Range: 10<= value <= 10000.

	API<chan><b>AVG_ANGLE?<cr><lf>	Queries the average angle value for the channel. Query returns angle in degrees: Range: 0.0000 < value < 359.9999 or Bipolar Range: -180.0000 < value < +180.0000
Angle Limit Testing	API<chan><b>ANG_LIMIT?<cr><lf>	Queries the angle limit test settings for the channel. Query returns: 'ON' or 'OFF' to indicate whether limit testing is active or disable, a blank space, 'ABS' or 'ERR' to indicate whether absolute angle or angle error comparison is used for limit testing.
	API<chan><b>ANG_LIMIT<b><ON|OFF><cr><lf>	Turns on or off the channel's angle limit testing feature.
	API<chan><b>ANG_LIMIT<b>CMP?<cr><lf>	Queries the angle limit test comparison settings for the channel. Query returns 'ABS' or 'ERR'.
	API<chan><b>ANG_LIMIT<b>CMP<b><ABS|ERR><cr><lf>	Set the channel's angle limit comparison to Absolute Angle or Angle Error.

	API<chan><b>ANG_LIMIT<b>HI?<cr><lf>	Queries the upper limit test value settings for the channel. Query returns upper limit in degrees: Range: 0.0000 < value < 359.9999 or Bipolar Range: -180.0000 < value < +180.0000
	API<chan><b>ANG_LIMIT<b>HI<b><value><cr><lf>	Sets the upper limit test value in degrees: Range: 0.0000 < value < 359.9999 or Bipolar Range: -180.0000 < value < +180.0000
	API<chan><b>ANG_LIMIT LO?<cr><lf>	Queries the lower limit test value settings for the channel. Query returns lower limit in degrees: Range: 0.0000 < value < 359.9999 or Bipolar Range: -180.0000 < value < +180.0000
	API<chan><b>ANG_LIMIT<b>LO<b><value><cr><lf>	Sets the lower limit test value in degrees: Range: 0.0000 < value < 359.9999 or Bipolar Range: -180.0000 < value < +180.0000

	API<chan><b>ANG_LIMIT STEP?<cr><lf>	Queries the angle step value for limit test value settings for the channel. Query returns step value in degrees: Range: 0.001 < value < 359.999
	API<chan><b>ANG_LIMIT<b>STEP<b><value> <cr><lf>	Sets the angle step test value in degrees: Range: 0.0000 < value < 359.9999 or Bipolar Range: -180.0000 < value < +180.0000
Angle	API<chan><b>ANGLE?<cr><lf>	Queries the angle data for the channel. Query returns angle in degrees: Range: 0.0000 < value < 359.9999 or Bipolar Range: -180.0000 < value < +180.0000
Velocity	API<chan><b>VEL?<cr><lf>	Queries the angle velocity data for the channel. Query returns channel velocity in degrees per second: Range: -32767 < value < +32767.
Line-to-Line Voltage	API<chan><b>LL_VOLT?<cr><lf>	Queries the line-to-line voltage for the channel.
Null Voltage	API<chan><b>NULL_VOLT?<cr><lf>	Queries the null voltage for the channel. Note this feature has not been implemented. The value 0.00 will always be returned.
Reference	API<chan><b>REF_VOLT?<cr><lf>	Queries the reference

Voltage		voltage for the channel.
Reference Frequency	API<chan><b>REF_FREQ?<cr><lf>	Queries the reference frequency for the channel.
Digital-to-Analog Setup	API<chan><b>DA<b>OUTPUT?<cr><lf>	Queries the D/A setup for the data to use for D/A voltage output conversion for the channel. Query returns 'ANG' or 'VEL'.
	API<chan><b>DA<b>OUTPUT<b><ANG|VEL><cr><lf>	Sets the D/A data to use either angle or velocity for D/A voltage output conversion for the channel.
	API<chan><b>DA<b>HIDATA?<cr><lf>	Queries the D/A Upper Limit Data conversion for the channel. Query returns for Angle Data in degrees: Unipolar Range: 0.0000 < value < 359.9999 or Bipolar Range: -180.0000 < value < +180.0000 Query returns for Velocity Data in degrees/sec: Range: -10000<value<10000

	API<chan><b>DA<b>HIDATA<b><value><cr><lf>	Sets the D/A Upper Limit Data conversion for the channel. Angle Data Range in degrees: Unipolar Range: 0.0000 < value < 359.9999 or Bipolar Range: -180.0000 < value < +180.0000 Velocity Data Range in degrees/sec: -10000<value<10000
	API<chan><b>DA<b>HIVOLT?<cr><lf>	Queries the D/A Voltage conversion for the Upper Limit Conversion for the channel. Query returns voltage value in the range: -10.0<value<10.0
	API<chan><b>DA<b>HIVOLT<b><value><cr><lf>	Sets the D/A Voltage conversion for the Upper Limit Conversion for the channel. Voltage range in volts: -10.0<value<10.0

	API<chan><b>DA<b>LODATA?<cr><lf>	Queries the D/A Lower Limit Data conversion for the channel. Query returns for Angle Data in degrees: Unipolar Range: 0.0000 < value < 359.9999 or Bipolar Range: -180.0000 < value < +180.0000 Query returns for Velocity Data in degrees/sec: Range: -10000<value<10000
	API<chan><b>DA<b>LODATA<b><value><cr><lf>	Sets the D/A Lower Limit Data conversion for the channel. Angle Data Range in degrees: Unipolar Range: 0.0000 < value < 359.9999 or Bipolar Range: -180.0000 < value < +180.0000 Velocity Data Range in degrees/sec: -10000<value<10000
	API<chan><b>DA<b>LOVOLT?<cr><lf>	Queries the D/A Voltage conversion for the Lower Limit Conversion for the channel. Query returns voltage value in the range: -10.0<value<10.0

	API<chan><b>DA<b>LOVOLT<b><value><cr><lf>	Sets the D/A Voltage conversion for the Lower Limit Conversion for the channel. Voltage range in volts: -10.0<value<10.0

MULTIPLE CHANNEL QUERIES Note: Channel 1 and Channel 2 data are returned with Channel 1 data first and then Channel 2 data, separated by a comma		
Function	**Syntax (commands must be sent with upper-case)**	**Comments**
Angle	APIALL<b>ANGLE?<cr><lf>	Queries the angle data for BOTH channels. Query returns angle in degrees: Range: 0.0000 < value < 359.9999 or Bipolar Range: -180.0000 < value < +180.0000
Angle Averaging	APIALL<b>AVERAGE?<cr><lf>	Queries the averaging setting for BOTH channels. Query returns: ON avgRate or OFF avgRate
	APIALL<b>AVG_ANGLE?<cr><lf>	Queries the average angle value for BOTH channels. Query returns angle in degrees: Range: 0.0000 < value < 359.9999 or Bipolar Range: -180.0000 < value < +180.0000
Bandwidth	APIALL<b>BANDWIDTH?<cr><lf>	Queries the bandwidth setting for BOTH channels. Query returns: AUTO bwValue or OVERRIDE bwValue
Signal Mode	APIALL<b>MODE?<cr><lf>	Queries the mode state of BOTH channels. Query returns: 'RSL' or

		SYN'.
Ratio	APIALL<b>RATIO?<cr><lf>	Queries the ratio setting for BOTH channels. Query returns '1' for channel 1 and for channel 2, the ratio setting: Ratio Range = 1 to 255.
Reference Mode	APIALL<b>REF_SOURCE?<cr><lf>	Queries the reference source for BOTH channels. Query returns: 'INT' or 'EXT'.
Track/Latch Angle	APIALL<b>UPDATE?<cr><lf>	Queries the track/latch state of BOTH channels. Query returns: 'LATCHED' or 'TRACK'.
Angle Limit Testing	APIALL<b>ANG_LIMIT?<cr><lf>	Queries the angle limit test settings for BOTH channels. Query returns: 'ON' or 'OFF' to indicate whether limit testing is active or disable, a blank space, 'ABS' or 'ERR' to indicate whether absolute angle or angle error comparison is used for limit testing.
Velocity	APIALL<b>VEL?<cr><lf>	Queries the angle velocity data for BOTH channels. Query returns channel velocity in degrees per second: Range: -32767 < value < +32767.
Line-to-Line Voltage	APIALL<b>LL_VOLT?<cr><lf>	Queries the line-to-line voltage for BOTH channels.

Null Voltage	APIALL<b>NULL_VOLT?<cr><lf>	Queries the null voltage for BOTH channels. Note this feature has not been implemented. The value 0.00 will always be returned.
Reference Voltage	APIALL<b>REF_VOLT?<cr><lf>	Queries the reference voltage for BOTH channels.
Reference Frequency	APIALL<b>REF_FREQ?<cr><lf>	Queries the reference frequency for BOTH channels.

INTERNAL REFERENCE GENERATOR		
Function	**Syntax (commands must be sent with upper-case)**	**Comments**
Reference Generator Frequency	REF_GEN<b>FREQ?<cr><lf>	Queries the frequency setting for the internal reference generator.
	REF_GEN<b>FREQ<b><value><cr><lf>	Sets the frequency setting for the internal reference generator. Frequency range is 47.0 to 20000.0 Hz.
Reference Generator Voltage	REF_GEN<b>VOLT?<cr><lf>	Queries the voltage setting for the internal reference generator.
	REF_GEN<b>VOLT<b><value><cr><lf>	Sets the voltage setting for the internal reference generator. Voltage range is 2.0 to 115.0 volts.
Reference Generator Output State	REF_GEN<b>STATE?<cr><lf>	Queries the output state of the internal reference generator. Query returns: 'OPEN', or 'CLOSED'.
	REF_GEN<b>STATE<b><OPEN|CLOSE><cr><lf>	Sets the output state for the internal reference generator. The CLOSE state will out allow reference signals to be available at the output connectors. The OPEN state will prevent the reference signals from being outputted.

API CONFIGURATION		
Function	**Syntax (commands must be sent with upper-case)**	**Comments**
Communi-cation Setting	APICMD <b>COMM?<cr><lf>	Queries the communication settings. Possible results are: Local Mode or Remote IEEE Addr: API-IEEE Language or Remote USB or Remote Ethernet or Remote J1 or Remote with Lockout via IEEE Addr: API-IEEE Language or Remote with Lockout via USB or Remote with Lockout via Ethernet or Remote with Lockout via J1
	APICMD<b>COMM<b><IEEE|USB|ETHERNET|J1><cr><lf>	Sets the communication setting to communicate remotely via IEEE, USB, Ethernet or J1.
Go To Local	APICMD<b>COMM<b><LOCAL><cr><lf>	Sets the device to Local mode.
Local Lockout	APICMD<b>COMM<b><LOCKOUT><cr><lf>	Sets the device to Local Lockout mode.
Angle Display Format	APICMD<b>ANG_FMT?<cr><lf>	Queries the angle display format. Query returns: (‘0 to 360’, or ‘-180 to 180’, or ‘Deg, Min, Sec’)

	APICMD<b>ANG_FMT<b><360|180|MIN><cr><lf>	Sets the angle display format.
Channel 1 Input Connector	APICMD<b>CH1INPUT?<cr><lf>	Queries the channel 1 input connector setting. Query returns: 'FRONT', or 'BACK'.
	APICMD<b>CH1INPUT<b><FRONT|BACK><cr><lf>	Sets the channel 1 input connector setting to either the Front or Back connectors.
Angle Difference	APICMD<b>SHOWDIFF?<cr><lf>	Queries the device to determine if angle difference mode is ON or OFF.
	APICMD<b>SHOWDIFF<b><ON|OFF><cr><lf>	Sets the device to show angle difference or Channel 2 angle value. If angle difference is turned ON, the device will automatically switch the screen to Dual Chan and show the angle difference value in place of Channel 2 angle data.
	APICMD<b>ANGDIFF?<cr><lf>	Queries the device for the angle difference between Channel 1 angle and Channel 2 angle value. Angle Diff Range: -180.0000 < value < +180.0000
Default Values	APICMD<b>RSTFRAM<cr><lf>	Sets the device to the factory default conditions.

API CHARTING AND BUFFERING Note: Retrieval of the buffered data is available only via USB or Ethernet!		
Function	**Syntax (commands must be sent with upper-case)**	**Comments**
Charting and Buffering	APIBUF<b>CNT? <cr><lf>	Queries the device for the number of records in the buffer.
	APIBUF<b>RECORD? <cr><lf>	Queries the device for the recording state. Query returns: 'NOT RECORDING' or 'RECORDING'
	APIBUF<b>RECORD<b><START|STOP|CLEAR> <cr><lf>	Set the device to start or stop recording or to clear the buffer. The device will automatically switch the screen to Charting screen when this command is sent.
	APIBUF<b>SAMPLE_RATE?<cr><lf>	Queries the device for the sample rate for recording data. Query returns the sample interval, followed by a blank, and the sample units ('MSEC', 'SEC' or 'MIN').
	APIBUF<b>SAMPLE_RATE<b><value><b><MSEC|SEC|MIN><cr><lf>	Sets the device to sample at the given rate. Sample Rate range: 100 msec < value < 30 min
	APIBUF<b>SAMPLE_TYPE?<cr><lf>	Queries the device for the type of data to record. Query returns: 'ANG', 'ANGERR' or 'VEL'.
	APIBUF<b>SAMPLE_TYPE<b><ANG|ANGERR|VEL> <cr><lf>	Sets the device to record Angle, Angle Error or Velocity data.

	APIBUF<b>PLOT_CH?<cr><lf>	Queries the device for the channel to plot. Query returns: 'BOTH', 'CH1' or 'CH2'
	APIBUF<b>PLOT_CH<b><BOTH|CH1|CH2><cr><lf>	Sets the device to record BOTH channels, Channel 1 only, or Channel 2 only.
	APIBUF<b>CHAN<chan><b><recstart#><b><recstop#><cr><lf>	Note, retrieval of the buffered data is available only via USB or Ethernet. Via USB, the maximum number of records returned for each call is 5. Via Ethernet, the maximum number of records returned for each call is 150.
Calibration	APICMD<b>CALIBRATE?<cr><lf>	Queries the device for the calibration state. Query returns: 'CAL DONE' or 'CALIBRATING'
	APICMD<b>CALIBRATE<cr><lf>	Calibrates the unit.

3.4 API-8810 Native (Legacy)

The API-8810 Native language is only support via the IEEE-488.1. The language is available to provide backwards compatibility to the 8810 units. Only the features that were available for the 8810 are supported with this language.

Function	Syntax	Comments
Track/Latch (Hold) Angle	Serial Poll Status Byte	Queries the track/latch state of the channel.
	F<cr><lf> or f<cr><lf> or >F<cr><lf> or >f<cr><lf>	Set the latch state for the active channel.
	T<cr><lf> or t<cr><lf> or >T<cr><lf> or >t<cr><lf>	Set the track state for the active channel.
Channel Select	S<cr><lf> or s<cr><lf> or >S<cr><lf> or >s<cr><lf> or 1<cr><lf> or >1<cr><lf>	If Channel 1 Input is set for “Front Panel”, this will set the Signal Mode to SYNCHRO. Channel 1 is set as the active channel. If Channel Input is set for “Back Connector” the Signal Mode configuration is not modified. Channel 1 is set as the active channel.
	R<cr><lf> or r<cr><lf> or >R<cr><lf> or >r<cr><lf> or 2<cr><lf> or >2<cr><lf>	If Channel 1 Input is set for “Front Panel”, this will set the Signal Mode to RESOLVER. Channel 1 is set as the active channel. If Channel Input is set for “Back Connector” the Signal Mode configuration is not modified. Channel 2 is set as the active channel.
Angle Display Format	B<cr><lf> or b<cr><lf> or >B<cr><lf> or >b<cr><lf>	Set the angle format to +/-180 degrees.
	U<cr><lf> or u<cr><lf> or >U<cr><lf> >u<cr><lf>	Set the angle format to 0 to 359.9999 degrees.
GET Command	G<cr><lf> or g<cr><lf> or >G<cr><lf> or >g<cr><lf>	Queues the commands until the GET but command is received.

Assert SRQ	V<cr><lf> or v<cr><lf> or >V<cr><lf> or >v<cr><lf>	Asserts SRQ when data is stable.

3.5 API-8810 SR103 (Legacy)

The API-8810 SR103 language is only support via the IEEE-488.1. The language is available to provide backwards compatibility to the 8810 SR103 units. Only the features that were available for the 8810 SR103 are supported with this language.

Function	Syntax	Comments
Track/Latch (Hold) Angle	Serial Poll Status Byte	Queries the track/latch state of the channel.
	I<cr><lf> or i<cr><lf> or >I<cr><lf> or >i<cr><lf>	Set the latch state for the active channel.
	F<cr><lf> or f<cr><lf> or >F<cr><lf> or >f<cr><lf>	Set the track state for the active channel.
Channel Select	S<cr><lf> or s<cr><lf> or >S<cr><lf> or >s<cr><lf> or 1<cr><lf> or >1<cr><lf>	If Channel 1 Input is set for “Front Panel”, this will set the Signal Mode to SYNCHRO. Channel 1 is set as the active channel. If Channel Input is set for “Back Connector” the Signal Mode configuration is not modified. Channel 1 is set as the active channel.
	R<cr><lf> or r<cr><lf> or >R<cr><lf> or >r<cr><lf> or 2<cr><lf> or >2<cr><lf>	If Channel 1 Input is set for “Front Panel”, this will set the Signal Mode to RESOLVER. Channel 1 is set as the active channel. If Channel Input is set for “Back Connector” the Signal Mode configuration is not modified. Channel 2 is set as the active channel.
Angle Display Format	B<cr><lf> or b<cr><lf> or >B<cr><lf> or >b<cr><lf>	Set the angle format to +/-180 degrees.
	U<cr><lf> or u<cr><lf> or >U<cr><lf> >u<cr><lf>	Set the angle format to 0 to 359.9999 degrees.
Power-Up Or Preset	P<cr><lf> or p<cr><lf> or >P<cr><lf> or >p<cr><lf>	Resets the device to power-up or last saved configuration.

3.7 API-8810 HSR202 (Legacy)

The API-8810 HSR202 language is only support via the IEEE-488.1. The language is available to provide backwards compatibility to the 8810 HSR202 units. Only the features that were available for the 8810 HSR202 are supported with this language.

Function	Syntax	Comments
Track/Latch (Hold) Angle	Serial Poll Status Byte	Queries the track/latch state of the channel.
	I<cr><lf> or i<cr><lf> or >I<cr><lf> or >i<cr><lf>	Set the latch state for the active channel.
	F<cr><lf> or f<cr><lf> or >F<cr><lf> or >f<cr><lf>	Set the track state for the active channel.
Channel Select	S<cr><lf> or s<cr><lf> or >S<cr><lf> or >s<cr><lf>	If Channel 1 Input is set for "Front Panel", this will set the Signal Mode to SYNCHRO. Channel 1 is set as the active channel. If Channel Input is set for "Back Connector" the Signal Mode configuration is not modified. Channel 1 is set as the active channel.
	R<cr><lf> or r<cr><lf> or >R<cr><lf> or >r<cr><lf>	If Channel 1 Input is set for "Front Panel", this will set the Signal Mode to RESOLVER. Channel 1 is set as the active channel. If Channel Input is set for "Back Connector" the Signal Mode configuration is not modified. Channel 2 is set as the active channel.
Power-Up Or Preset	P<cr><lf> or p<cr><lf> or >P<cr><lf> or >p<cr><lf>	Resets the device to power-up or last saved configuration.

3.8 API-8810 HSR203 (Legacy)

The API-8810 HSR203 language is only support via the IEEE-488.1. The language is available to provide backwards compatibility to the 8810 HSR203 units. Only the features that were available for the 8810 HSR203 are supported with this language.

Function	Syntax	Comments
Track/Latch (Hold) Angle	Serial Poll Status Byte	Queries the track/latch state of the channel.
	I<cr><lf> or i<cr><lf> or >I<cr><lf> or >i<cr><lf>	Set the latch state for the active channel.
	F<cr><lf> or f<cr><lf> or >F<cr><lf> or >f<cr><lf>	Set the track state for the active channel.
Channel Select	S<cr><lf> or s<cr><lf> or >S<cr><lf> or >s<cr><lf> or 1<cr><lf> or >1<cr><lf>	If Channel 1 Input is set for “Front Panel”, this will set the Signal Mode to SYNCHRO. Channel 1 is set as the active channel. If Channel Input is set for “Back Connector” the Signal Mode configuration is not modified. Channel 1 is set as the active channel.
	R<cr><lf> or r<cr><lf> or >R<cr><lf> or >r<cr><lf> or 2<cr><lf> or >2<cr><lf>	If Channel 1 Input is set for “Front Panel”, this will set the Signal Mode to RESOLVER. Channel 1 is set as the active channel. If Channel Input is set for “Back Connector” the Signal Mode configuration is not modified. Channel 2 is set as the active channel.
Power-Up Or Preset	P<cr><lf> or p<cr><lf> or >P<cr><lf> or >p<cr><lf>	Resets the device to power-up or last saved configuration.

3.9 API-8810 MATE/CIIL (Legacy)

The API-8810 MATE/CIIL language is only support via the IEEE-488.1. The language is available to provide backwards compatibility to the 8810 units with MATE/CIIL support. Only the features that were available for the 8810 MATE/CIIL are supported with this language. Please contact the manufacture if MATE/CIIL support is needed.

4 API-8810A USB Protocol

The 8810A USB interface supports only the API-8810A Language. Sending commands via the USB interface require the following protocol:

Number of Bytes to be sent (2 bytes)	Command ID (8810 (i.e. 0x226A) (2 bytes)	Data

The following is code snippets from the API8810ADll (USBComm.cpp) that makes calls to the Cypress CyAPI.lib file to sending commands to the 8810A:

```
bool USB_WriteMsg(char* szMsg, bool bExpectReply, char*pszReply)
{
   bool bSuccess = false;
   unsigned short usTotalBytes = (unsigned short)strlen(szMsg) + 4;  // Length of Message
                                                                     // + 2 bytes for Bytes sent
                                                                     // + 2 bytes for Command
   unsigned short usCommand = 0x226A;                                // 8810 (0x226A) Command

   char   szData[256];
   LONG   nDataCnt = 0;
   LONG   BytesToRead = 0;
   byte   loByte, hiByte;

   //****************************************
   // Format data to be sent
   //  Protocol:
   //      (16 bits) Number of bytes to be sent
   //      (16 bits) Command ID (0x226A) for 8810
   //      szMsg - data message
   //****************************************

   // Low byte of Total Bytes to send
   loByte = (byte)(usTotalBytes & 0x00FF);
   // High byte of Total Bytes to send
   hiByte = (byte)(usTotalBytes >> 8);

   szData[nDataCnt++] = loByte;
   szData[nDataCnt++] = hiByte;

   // Low byte of Command
   loByte = (byte)(usCommand & 0x00FF);
   // High byte of Command
   hiByte = (byte)(usCommand >> 8);

   szData[nDataCnt++] = loByte;
   szData[nDataCnt++] = hiByte;

   // Message Data
   for (int i = 0; i < (int)strlen(szMsg); i++)
      szData[nDataCnt++] = szMsg[i];

   if(glb_pUSBDevice)
   {
      if(glb_pUSBDevice->IsOpen())
      {
         short   numOfTries = 0;
         do
         {
            // Write Data Message
            if (glb_pUSBDevice->BulkOutEndPt)
            {
               if (!glb_pUSBDevice->BulkOutEndPt->XferData((PUCHAR)&szData, nDataCnt))
```

```
            {
               ReinitUSB();
               break;
            }
         }

         // Get Reply if one is expected
         if (bExpectReply)
         {
            if (glb_pUSBDevice->BulkInEndPt)
            {
               // Read data (note, max returned from Cypress USB is 64 bytes
               BytesToRead = 64;
              unsigned char   aReceiveBuffer[64];

               for (int i = 0; i < 64; i++)
                  aReceiveBuffer[i] = 0;

               glb_pUSBDevice->BulkInEndPt->TimeOut = 10000;  // 10 second timeout
                bSuccess = glb_pUSBDevice->BulkInEndPt->XferData(aReceiveBuffer, BytesToRead);
                numOfTries++;

                if(!bSuccess)
                {
                   Wait(500);
                }
                else
                {
                   strcpy(pszReply, (char *)aReceiveBuffer);
                }
            }
             else
               bSuccess = true;
           }
           else
              bSuccess = true;
        }while((!bSuccess) && (numOfTries < 2));
     }
     else
     {
        ReinitUSB();
     }
  }

  return bSuccess;
}
```

5 API-8810A Ethernet Protocol

The 8810A Ethernet interface supports only the API-8810A Language. Sending commands via the Ethernet interface requires the creation and connection via a TCP/IP socket.

The following code snippet, `CreateClientSocket()` from the API8810ADll (Ethernet.cpp) makes calls to the Winsock API to create and connect a TCP/IP socket to send commands to the 8810A. Note the code utilizes the `PingHost()` call to make sure that the IP address specified for the 8810A is reachable before attempting to create the socket. This avoids waiting for the socket timeout in the `connect()` call if the device is not reachable. The code snippet, `CloseClientSocket()` closes the socket connection.

```
int CreateClientSocket(char *pszIPAddr, int nPort, SOCKET* s)
{
   WSADATA      wsaData;
   SOCKET        sock;
   SOCKADDR_IN   ServerAddr;

   int result;

   /* Before trying to make a connection to the server, ping it to make sure it's reachable */
   result = PingHost(pszIPAddr);
   if (result != 0)
      return ETHER_CANNOT_ESTABLISH_CONNECTION;

   // Initialize Winsock version 2.2
   WSAStartup(MAKEWORD(2,2), &wsaData);

   // Create a new socket to make a TCP client connection
   sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
   setsockopt( sock, SOL_SOCKET, SO_RCVTIMEO, (char*)&RECEIVE_TIMEOUT, sizeof(int) );
   setsockopt( sock, SOL_SOCKET, SO_SNDTIMEO, (char*)&SEND_TIMEOUT, sizeof(int) );

   // set to no_delay to insure quick ack
   result = setsockopt( sock, IPPROTO_TCP, TCP_NODELAY, (char*)&NO_DELAY, sizeof(int) );

   // Setup a SOCKADDR_IN structure that will be used to connect
   // to the listening server on the Port.
   ServerAddr.sin_family = AF_INET;
   ServerAddr.sin_port = htons(nPort);
   ServerAddr.sin_addr.s_addr = inet_addr(pszIPAddr);

   // Make a connection to the server with socket sock
   connect(sock, (const struct sockaddr *)&ServerAddr, sizeof(ServerAddr));
   *s = sock;
   Socket = sock;  // put it into global socket
   return ETHER_SUCCESS;
}

int CloseClientSocket(SOCKET s)
{
  closesocket(s);
  WSACleanup();
  return ETHER_SUCCESS;
}
```

After a socket connection is made to the 8810A, device log-in is required. 8810A Ethernet login is accomplished by sending **“NAII\r\n”** command via the Ethernet connection to the 8810A.

The following code snippets, `Ethernet_WriteMsg()`, `SendEthernetMsg()`and `ReadEthernetMsg()` from the API8810ADll (Ethernet.cpp) makes calls to the Winsock API to send and receive messages to and from the 8810A.

```
#define MSG_MAX_SIZE         1500      /* Maximum number of bytes to send */
#define RECV_MSG_MAX_SIZE    1500      /* Maximum number of bytes that can be read */

bool Ethernet_WriteMsg(SOCKET s, char* szMsg, bool bExpectReply, char* pszReply)
{
   bool bSuccess = false;
   char aReceiveBuffer[RECV_MSG_MAX_SIZE];
   int nBytesRead = 0;

   if (SendEthernetMsg(s, &szMsg[0], strlen(szMsg)) == ETHER_SEND_ERROR)
      return bSuccess;

   if (bExpectReply)
   {
      if (ReadEthernetMsg(s, RECV_MSG_MAX_SIZE, aReceiveBuffer, &nBytesRead) == ETHER_RECV_ERROR)
         return bSuccess;

      strncpy(pszReply, (char *)aReceiveBuffer, nBytesRead);
   }

   bSuccess = true;
   return bSuccess;
}

int SendEthernetMsg(SOCKET s, char *pszMessage, int nMessageLen)
{
   int ret;
   char sendbuff[MSG_MAX_SIZE];
   int nLeft;
   int nIndex;
   int status = 0;

   // Copy the data to be sent to the buffer
   for (nIndex = 0; nIndex < nMessageLen; nIndex++)
      sendbuff[nIndex] = pszMessage[nIndex];

   nLeft = nMessageLen;
   nIndex = 0;

   while (nLeft > 0)
   {
      ret = send(s, &sendbuff[nIndex], nLeft, 0);
      // It seems we sent some data
      if (ret != SOCKET_ERROR)
      {
         nLeft -= ret;
         nIndex += ret;
      }
      // got SOCKET_ERROR
      else
      {
         status = ETHER_SEND_ERROR;
         break;
      }
   }

   if (nLeft > 0)
      status = ETHER_SEND_ERROR; /* ERROR */
   else
      status = ETHER_SUCCESS;  /* SUCCESS */
   return status;
}
```

```
int ReadEthernetMsg(SOCKET s, int nMessageLenToBeRead, char *pszMessage, int *nMessageLen)
{
   int ret;
   int nLeft;
   int nIndex;
   int status = 0;

   nLeft = nMessageLenToBeRead;
   nIndex = 0;

   while (nLeft > 0)
   {
      ret = recv(s, pszMessage, nLeft, 0);
      // It seems we got some data
      if (ret != SOCKET_ERROR)
      {
         nLeft -= ret;
         nIndex += ret;
         pszMessage += ret;

         // We don't know the exact size of each message
         // for API we know that it won't exceed RECV_MSG_MAX_SIZE bytes
         nMessageLenToBeRead = nLeft;
         nLeft = 0;
      }
      // got SOCKET_ERROR
      else
      {
         status = ETHER_RECV_ERROR;
         break;
       }
    }

   if (nIndex > 0)
   {
      *nMessageLen = nIndex;
      status = ETHER_SUCCESS;  /* SUCCESS */
   }
   else
      status =  ETHER_RECV_ERROR; /* ERROR */
   return status;
}
```

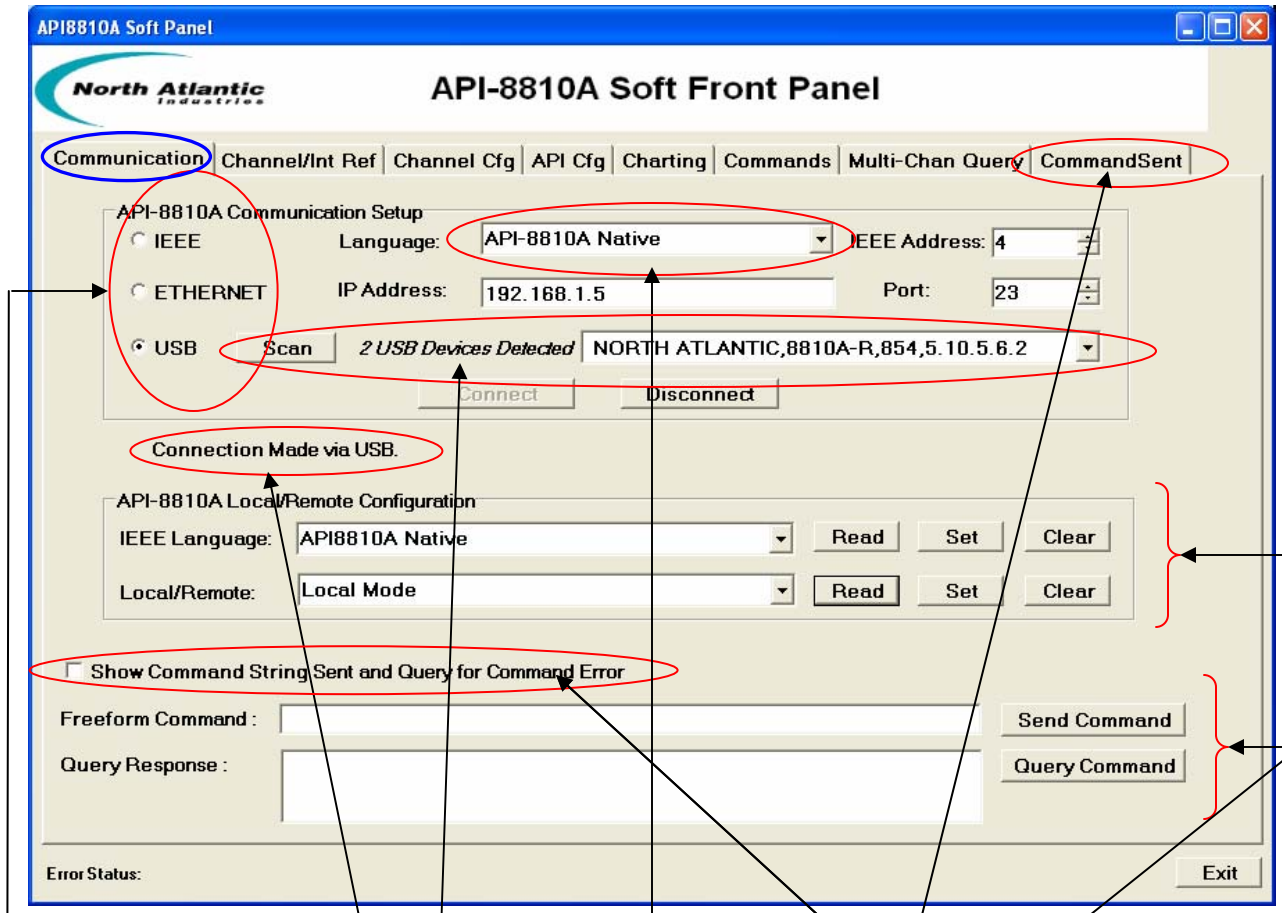
6 API-8810A DLL

A dynamic link library (DLL) written in C, compiled under Microsoft Visual .NET 2003 has been included in the software package to provide a program interface that handles the language syntax to communicate with the unit. The function lists provided in this Dynamic-link library (DLL) is described in *Function Reference Manual for 8810A.*

7 API-8810A Soft Panel Program

A Soft Panel application written in C#, compiled under Microsoft Visual .NET 2003 that invokes the routines in the API-8810A Dll has been included in the software package. Note, the Microsoft .NET Framework 1.1 must be installed on your machine prior to running the Soft Panel application. The .NET Framework Version 1.1 Redistributable Package can be downloaded from the Microsoft Web site:

http://www.microsoft.com/downloads



Choose the communication interface to the API-8810A.

Connection Status

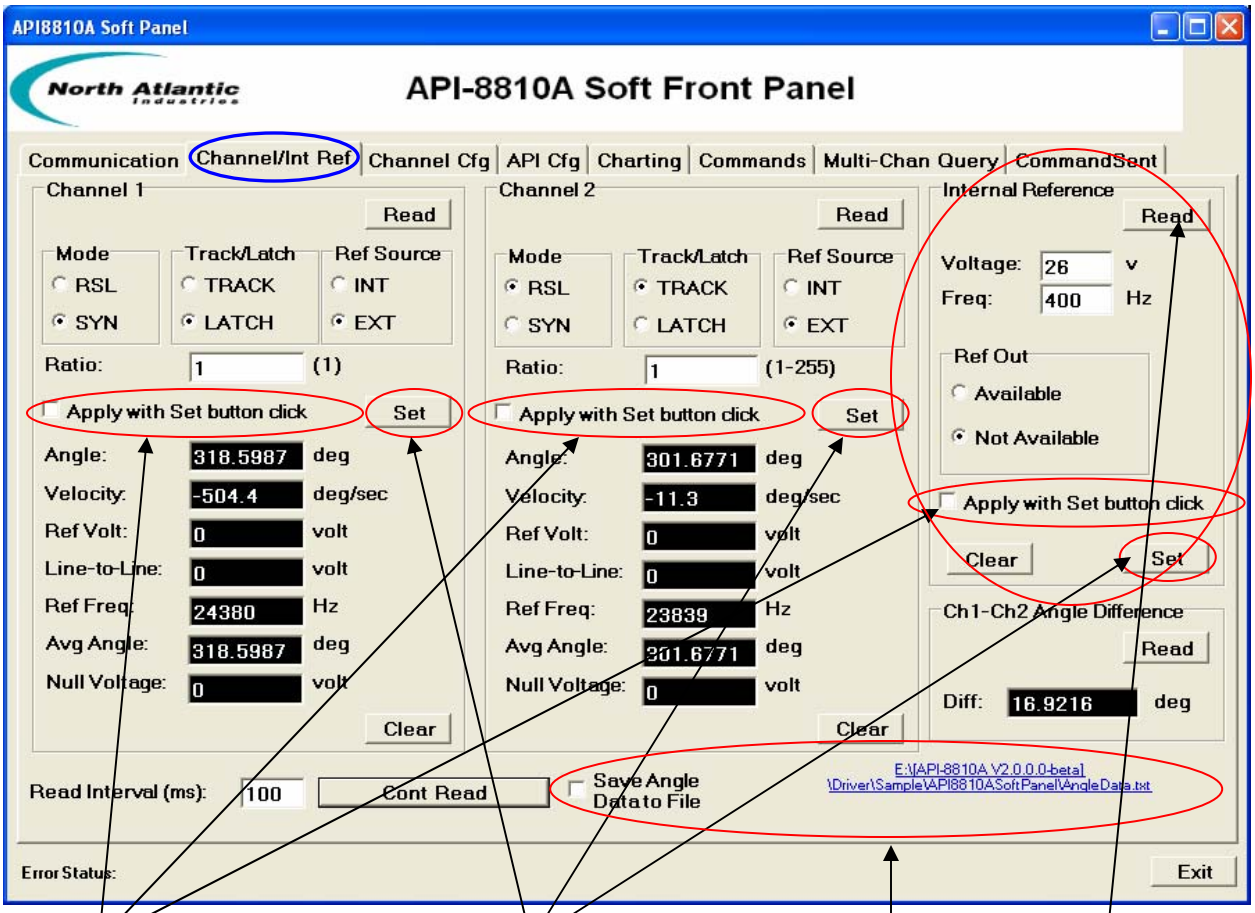
Scans for API-8810A devices connected via USB.

Choose the IEEE Language to communicate with API-8810A.

Option to show the command string and any errors in the tab labeled "Command Sent"

Writes/ Queries Freeform commands.

Gets/Sets the API-8810A Communication Setup. Note, communication to API-8810A must first be established.



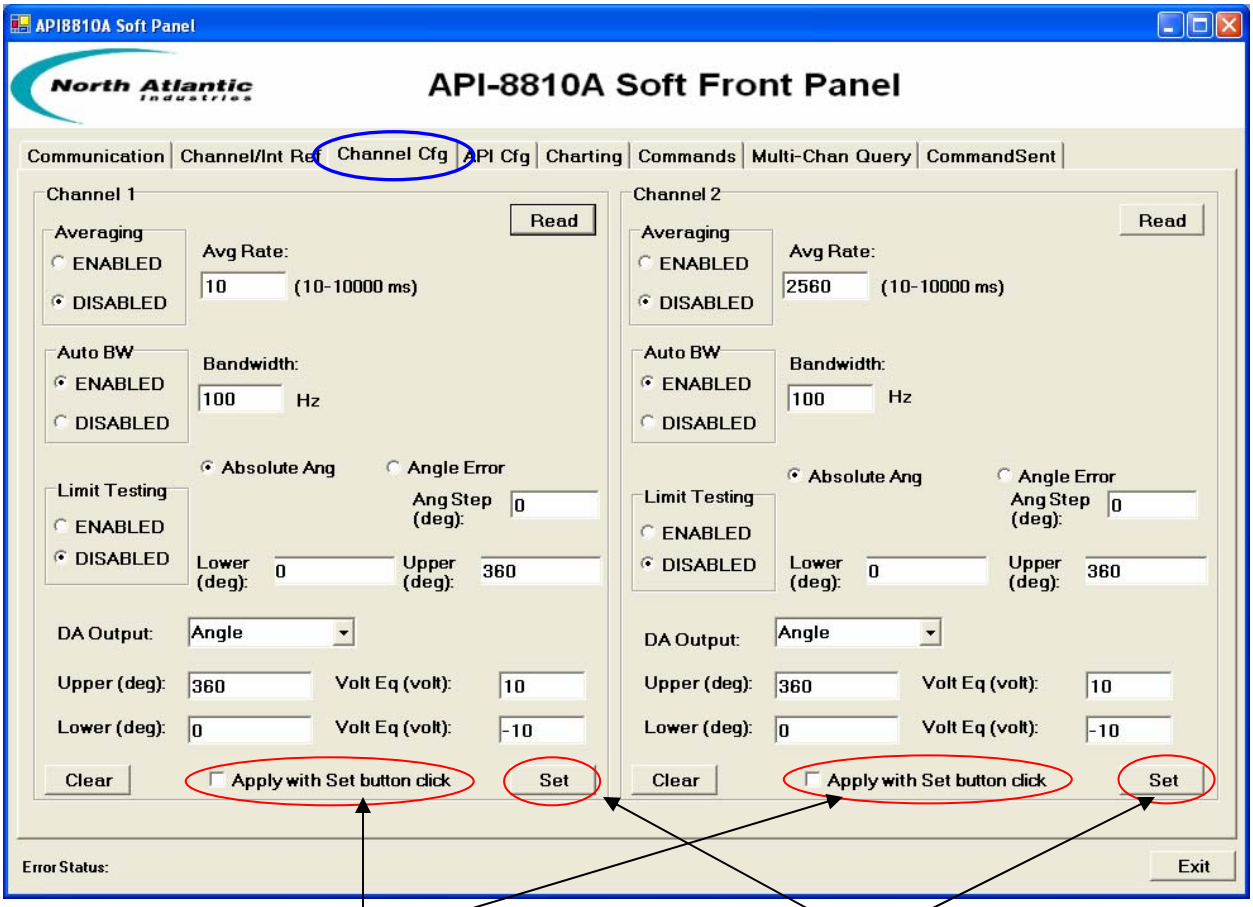
When checked, the changes will apply when the Set button is clicked.

"Sets" are allowed only when the API-8810A unit is set for Remote mode with specified communication interface (IEEE, USB, or Ethernet)

On units where Internal Reference is not available, API-8810A will return the default settings:
Voltage: 26 v
Frequency: 400 Hz
Ref Out: Not Available

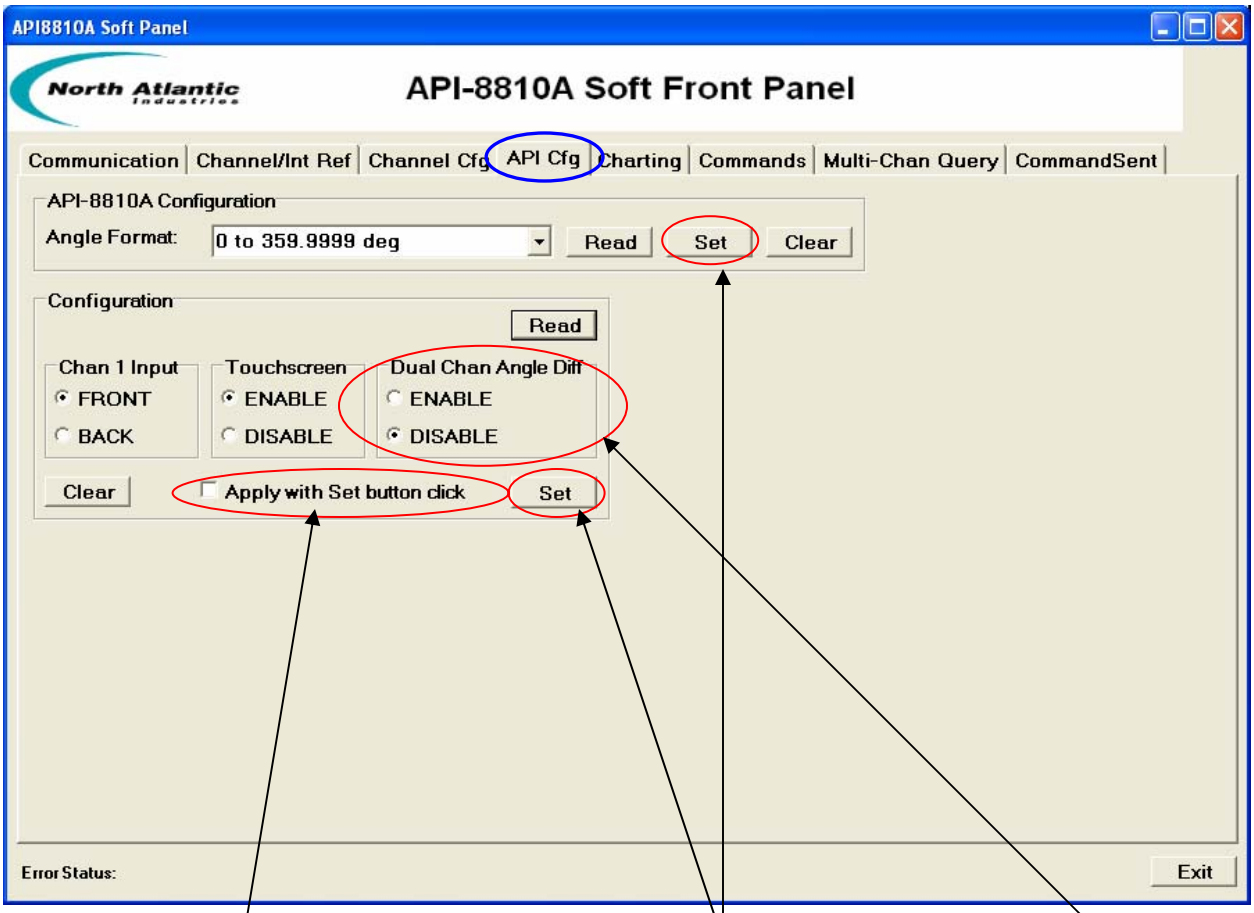
When checked, angle data for Channel 1 and Channel 2 will be written to the file specified during "Cont Read" mode. The data is written as follows (tab-delimited):

1	230.2710	314.4984
2	231.2988	316.3996
3	232.1987	319.0801
4	233.0217	322.1867
5	233.6806	325.3922
6	234.2289	328.4045
7	234.6222	330.6965
8	234.9082	332.2267
9	235.0213	332.8417



When checked, the changes will apply when the Set button is clicked.

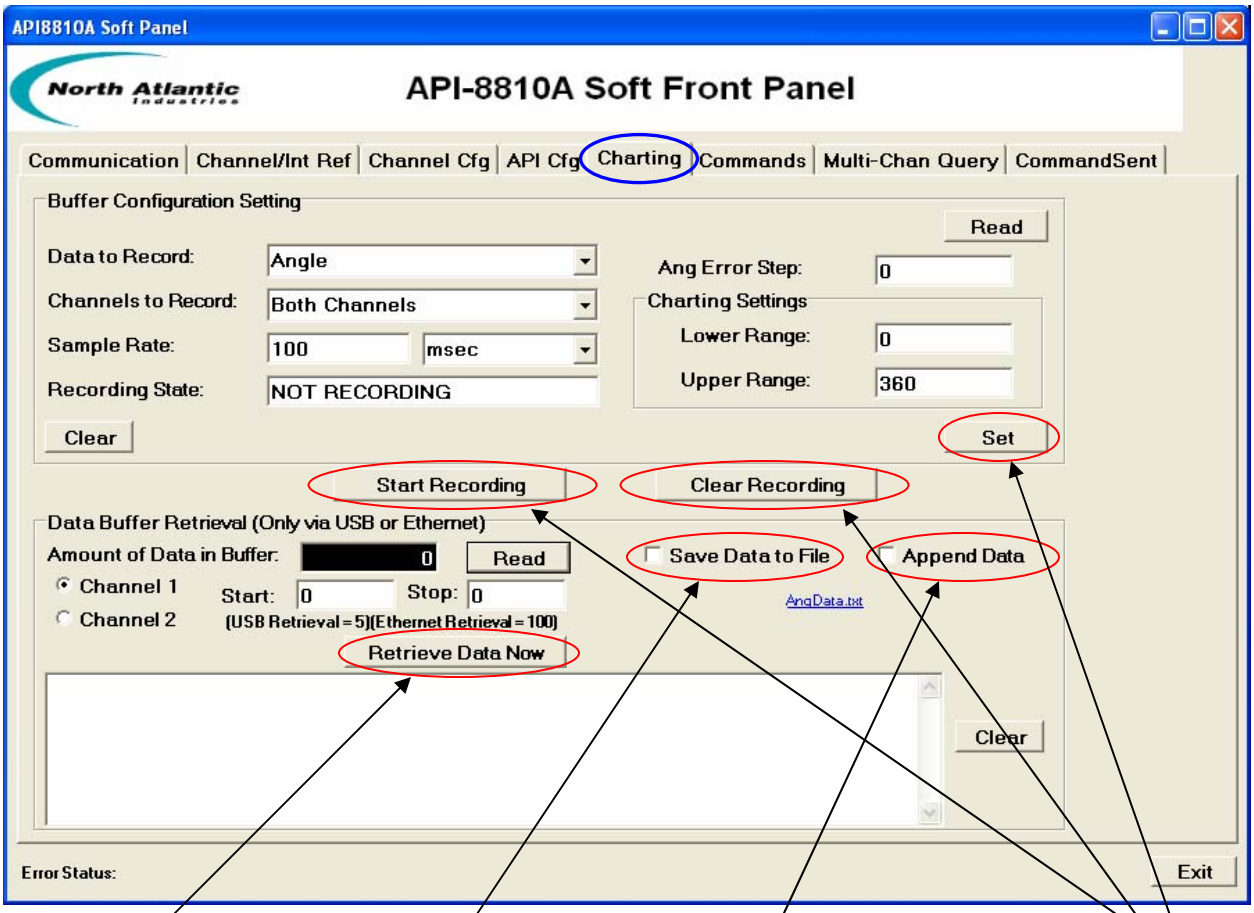
"Sets" are allowed only when the API-8810A unit is set for Remote mode with specified communication interface (IEEE, USB, or Ethernet)



When checked, the changes will apply when the Set button is clicked.

"Sets" are allowed only when the API-8810A unit is set for Remote mode with specified communication interface (IEEE, USB, or Ethernet)

Upon receiving a remote enable of the Dual Chan Angle Diff configuration, the API-8810A unit will automatically show the Dual Channel Screen. The angle value shown for Channel 2 represents the angle difference between the angle readings for Channel 1 and Channel 2.

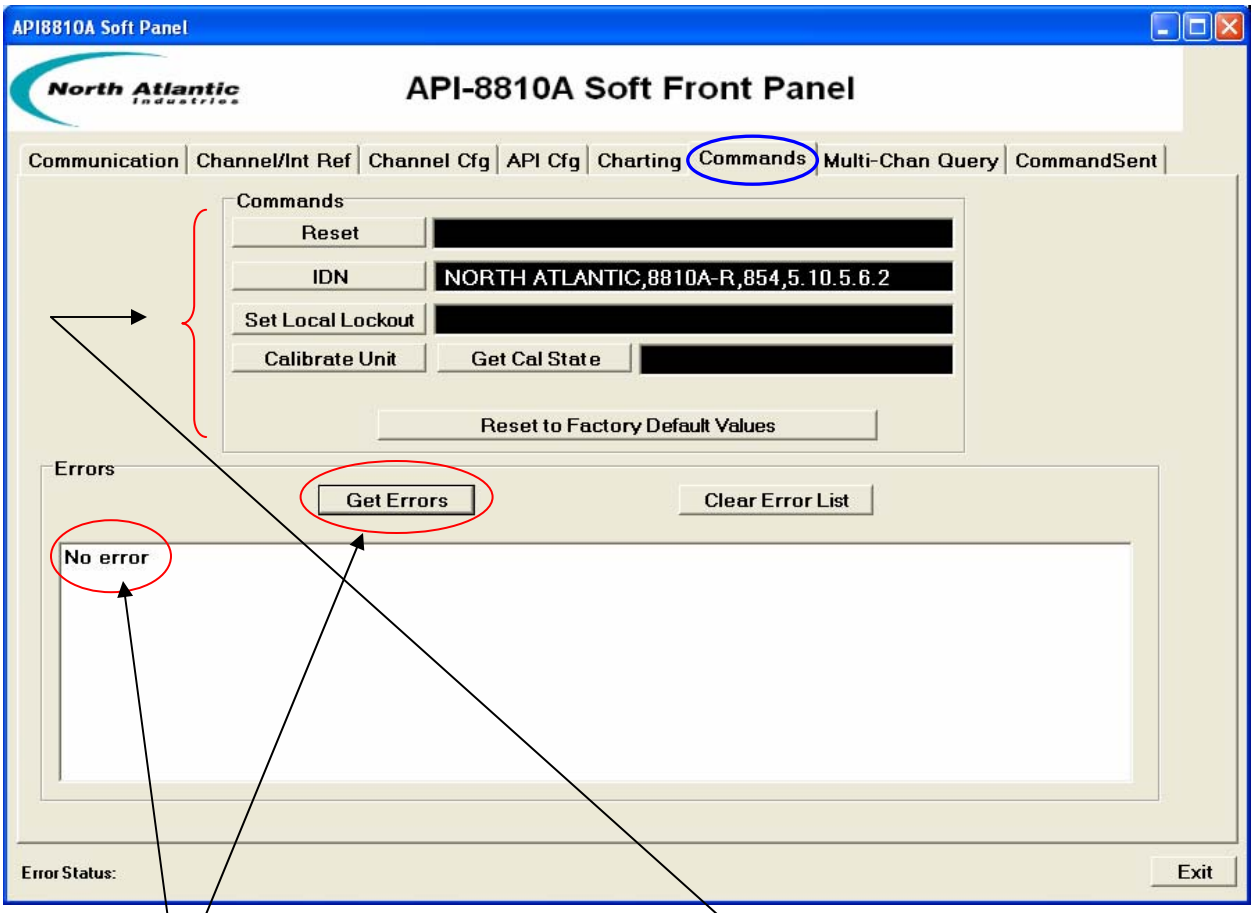


API-8810A Data Buffer Retrieval is allowed only if the remote connection is USB or Ethernet.

If checked, saves the data buffer retrieved from the device to the specified file.

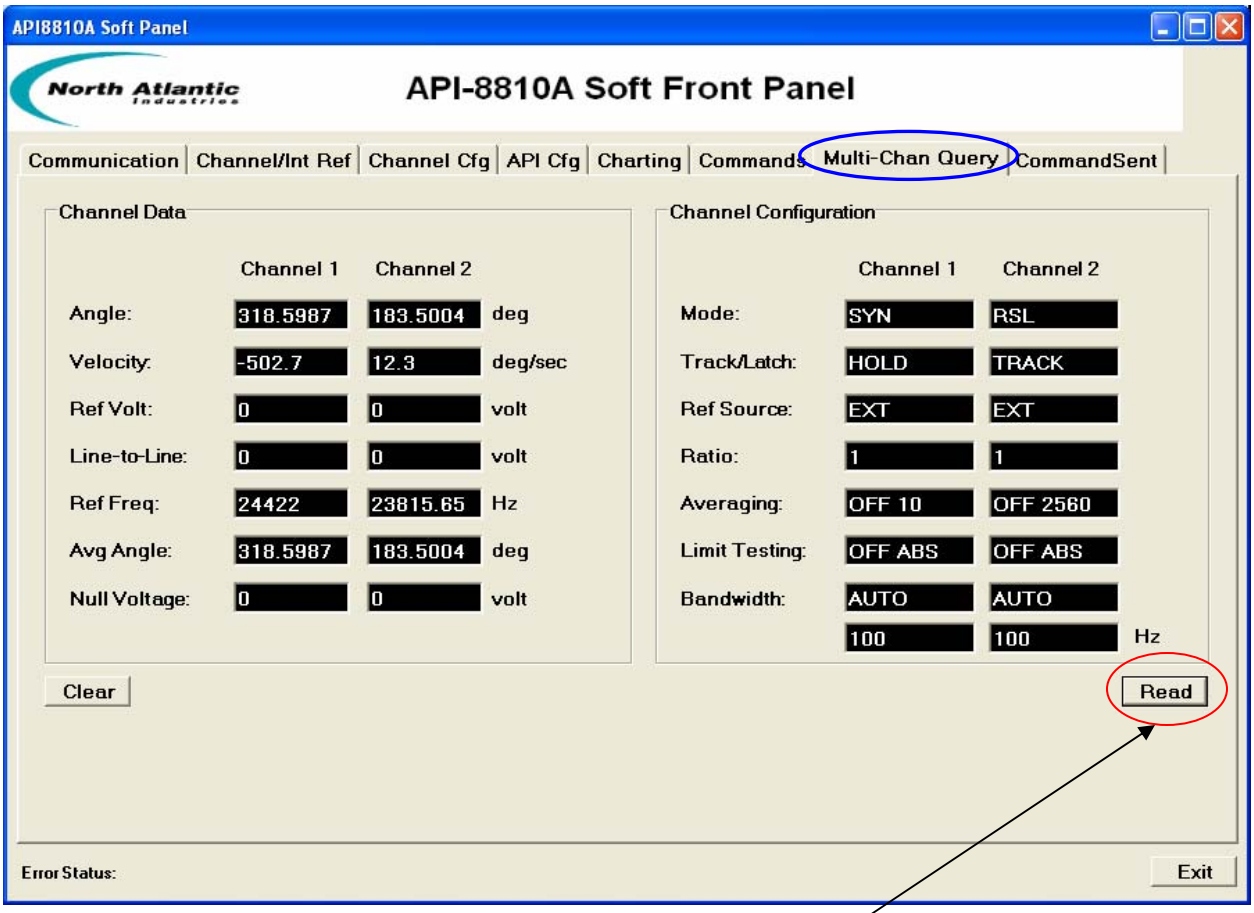
If checked, appends buffered data to the specified file. If not checked, any contents in the specified file are overwritten.

"Set", "Start Recording", and "Clear Recording" are allowed only when the API-8810A unit is set for Remote mode with specified communication interface (IEEE, USB, or Ethernet). Upon receiving a remote "Start" or "Stop" Recording command, the API-8810A unit will automatically show the Chart screen.

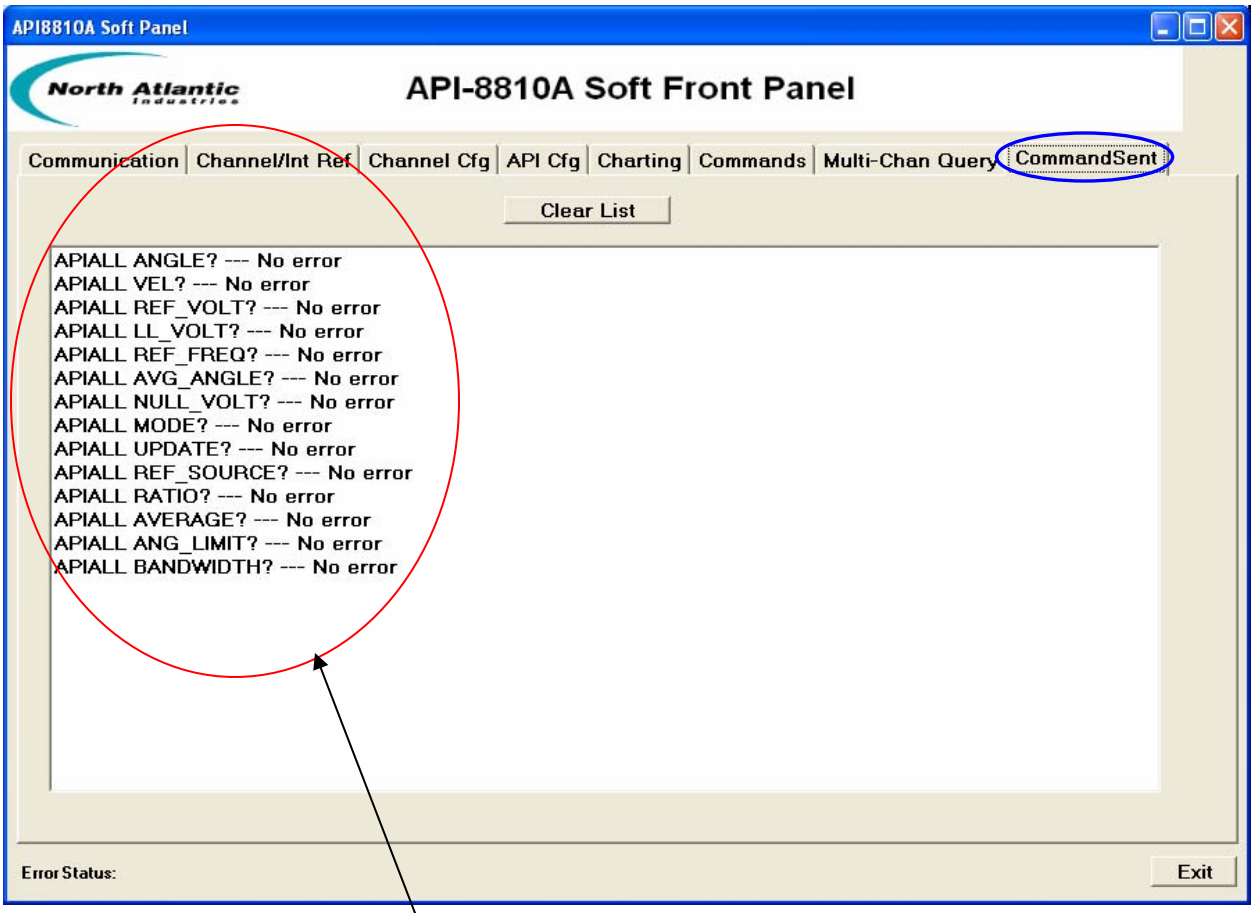


"No error" is returned when there is no error on the Error Queue.

"Reset", "Set Local Lockout", "Calibrate Unit" and "Reset to Factory Default Values" are allowed only when the API-8810A unit is set for Remote mode with specified communication interface (IEEE, USB, or Ethernet)



In this screen, the Read button will invoke the Multiple Channel Query routines to retrieve the data for both channels using the query command that returns data for channels.



Commands sent to the API 8810A as well as the results from performing a call to the API8810A Dll's API8810A_GetErrors() method to retrieve any messages from the Error Queue.

8 Cypress USB Driver Installation for Windows XP

In order to communicate with API 8810A unit via the USB 2.0 interface, the Cypress USB Driver must be installed. *The Cypress USB Driver included on your CD will install only on a machine with Windows XP.*

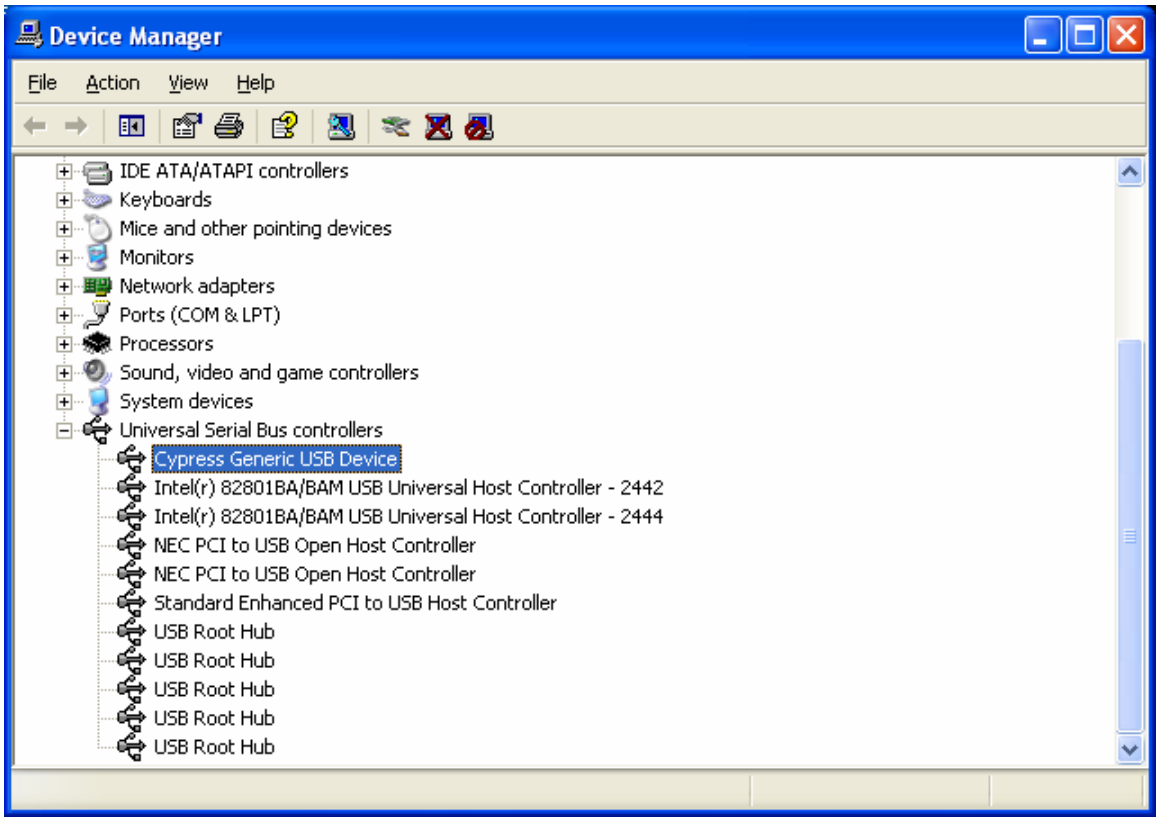
On the CD, the folder labeled, "Cypress USB Driver", copy the following three files:

1. CyUSB.PNF
2. CyUSB.inf
3. CyUSB.sys

to your windows INF folder (in most systems, this is c:\WINDOWS\INF).

From Windows Explorer, find the file CyUSB.inf, with your mouse, select the file, and right-click on this file. You should see "Install" as an option. Click on the "Install" menu.

After installation, power up the API 8810 unit and connect the USB cable from your machine to the unit. Using Device Manager (My Computer->Properties->Hardware->Device Manager), check that the installation is correct by looking at the Universal Serial Bus controllers (you should see an entry labeled "Cypress Generic USB Device") – see figure below:



Revision History

Revision ID	Revision Date	Description	Author
1.0.0.0	Jul 27, 2006	Initial Release	gc
1.0.0.1	Aug 4, 2006	Added Language Independent Commands section (3.2). Added remote programmability support to configure 8810A's Bandwidth, Language, and Channel 1 Front/Back settings. Added USB Protocol definition.	gc
2.0.0.2	Dec 21, 2007	Added remote programmability support for averaging, limit testing, data buffer control, multiple channel data retrieval. Updated screen shots for Soft Panel application. Removed function reference in API-8810A Dll. The functions specified in the Dll have been moved to the Function Reference Manual.	gc