Data Acquisition Instrument Command Protocol

Note: This document assumes that the latest firmware updates have been applied to the device intended to be used with it. If you are unsure if your instrument is current, please follow this link for model DI-2008, or this link for all other models.

Revised April 7, 2021

Table of Contents

Overview 5

Supported Instruments 5

USB Interface Support (all devices) 7

USB Interface Restrictions with the Stand-alone Option 8

Ethernet Command/Response Interface 9

Ethernet Command Format (type DQCommand, 0x31415926) 9

Command Response (type DQResponse, 0x21712818) 10

Invoking Shared Commands Using Ethernet 11

Other Ethernet Response Types 12

Ethernet Device Discovery Response 12

Active WinDaq File header Response (type WdqHeader, 0x05772156) 13

ADC Data Stream Response (type DQAdcData, 0x14142135) 13

Read USB Drive File Response: (type UsbDriveData, 0x17320508) for data, or (UsbDriveEOF, 0x22360679) for end of file 14

Shared and USB-specific Protocol Command Usage 16

Protocol Command Set 17

Typical Program Flow 19

Basic Communication Commands 20

Info Commands 20

Multi-unit Synchronization Commands (USB) 22

syncget, syncset, syncstart Commands 22

Ethernet Interface Configuration 24

Connect (10) 24

Disconnect (11) 24

KeepAlive (12) 25

SetWdqHeader (21) 25

ipaddr command 26

netmask command 26

gateway command 27

keepalive command 28

port command 29

Multi-unit Synchronization Commands (Ethernet) 30

SlaveIp (5) 30

SyncStart (1) 30

SyncStop (6) 31

Sample Pseudo Code for Ethernet Device Synchronization 31

Date and Time Commands 32

ymd command 32

hms command 32

USB Drive Commands 33

UsbDrive (22) 33

ud_start 34

ud_stop 34

ud_ls <arg0> <arg1> 34

ud_del <arg0> 34

ud_read <arg0> <arg1> <arg2> 34

ud_status <arg0> <arg1> 35

ud_power <arg0> 35

ud_stream <arg0> 35

ud_cfgwr <arg0> <arg1> 36

ud_cfgrd <arg0> 36

Measurement Configuration Commands 37

encode Command 37

eol Command 37

slist Command 38

DI-1100 Scan List Definition 39

DI-2008 Scan List Definition 41

DI-2108 and DI-1110 Scan List Definition 43

DI-2108P, 4108, 4208 and 4730 Scan List Definition 44

DI-4718B Scan List Definition 45

Rate Range Table for Supported Instruments 46

Scanning Commands 47

srate Scan rate Command 47

srate Variable Table 47

start Command 48

stop Command 48

Oversampling Mode Commands 50

filter command 50

dec Command 51

deca Command (firmware version 1.21 and higher) 51

cjcdelta Command 53

Rate Measurement Moving Average 54

LED Color Command 55

Digital I/O Commands 56

endo command 56

dout command 57

din command 57

Reset Command 58

Binary Stream Output Formats 59

Binary Coding Notes 59

DI-1100 Binary Stream Output Format and ADC Coding 60

DI-1110 Binary Stream Output Format and ADC Coding 62

DI-1120 Binary Stream Output Format and ADC Coding 64

DI-2008, -2108, -2108P, -4108, -4208, -4730 Binary Stream Output Format and ADC Coding 66

DI-2008 Thermocouple Temperature Coding 68

DI-4718B Binary Stream Output Format and ADC Coding 68

ASCII Stream Output Format 71

Document Revisions 72

Overview

Although DATAQ Instruments provides ready-to-run WinDaq software with its data acquisition instruments, programmers will want the flexibility to integrate them in the context of their own application. To do so they want complete control over instruments hardware, which is accomplished by using the device at the protocol level. This white paper describes how protocol-level programming is implemented across the Windows and Linux operating systems. We'll define the protocol command set and scan list architecture and finish with a description of the instrument ASCII and binary response formats. Please note that .Net classes have been released for the instruments supported by this document, which allows programming at a much higher and convenient level than at the protocol-level under Windows.

Supported Instruments

This protocol document supports the following data acquisition hardware models:

DI-1100	DI-1110	DI-1120	DI-2008	DI-2108
DI-2108P	DI-4108	DI-4208	DI-4718B	DI-4730

Please note that not all commands described in this protocol support all product models. For any particular command, this document will indicate the specific models that the command supports. Further, instruments can be purchased with only a USB interface, a USB and Ethernet interface, and with or without a stand-alone option in the form of USB drive support. All product combinations (with or without USB drive support) are supported by this protocol with commands that are shared between USB and Ethernet interfaces, and others that apply only to Ethernet. Each command in this document is prefaced by a graphic that defines the products supported by the command, and whether that support is for Ethernet-enabled interfaces only ("E"), USB interface only ("U"), shared between Ethernet and USB interfaces ("S"), or not supported by the model (blank.) For example:

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
U	U	U	U	U	U	U	U	U	U

Command supports USB for the indicated models

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
						S	S	S	S

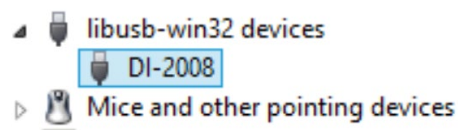
Command supports USB and Ethernet for the indicated models

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
						E	E	E	E

Command supports Ethernet for the indicated models

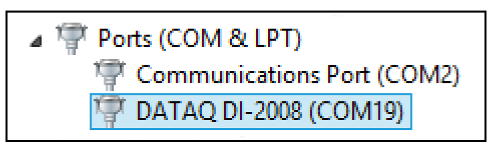
USB Interface Support (all devices)

All device models described in the protocol support a USB interface, accessed using the Libusb open source library, to control data transfers to and from the instrument via its USB interface in both Windows and non-Windows implementations. When an instrument that supports USB is connected to a PC in a Windows implementation the instrument's model number appears in the Device Manager under the "libusb-win32 devices" tree. For example, a connected model DI-2008 appears like this:



Most instruments may also be placed in the CDC mode (Communication Device Class) where they can hook a COM port. This approach is perhaps the most universal and easiest to manage. [Follow this link for details](). Since most instrument command and responses are ASCII, a standard terminal emulator (e.g. PuTTY, Tera Term) can be used to experiment with the device.

The same DI-2008 when placed in its CDC mode appears under the "Ports" tree in Windows' Device Manager like this:



The following constants for *product ID* (PID) and *vendor ID* (VID) apply, dependent upon instrument model and interface mode (LibUSB or CDC), and must be correctly referenced from your program:

Instrument Model	PID (LibUSB)	PID (CDC)	VID
DI-1100	0x1100	0x1101	0x0683 (all product models)
DI-1110	0x1110	0x1111	
DI-1120	0x1120	0x1121	
DI-2008	0x2008	0x2009	
DI-2108	0x2108	0x2107	
DI-2108P	0x2109	(unsupported)	
DI-4108	0x4108	0x4109	
DI-4208	0x4208	0x4209	
DI-4718B	0x4718	0x4719	
DI-4718B	0x4730	0x4731	

USB Interface Restrictions with the Stand-alone Option

When the stand-alone option is present on a device in the form of a USB Type A receptacle that accepts a USB drive, the port for both the USB interface and the drive is shared. This means than any command that is specific to the USB drive may not be delivered through the USB interface and must be delivered via the Ethernet interface, if present. While it is possible to configure and activate stand-alone or real time PC-connected data acquisition operations using the USB interface, direct access to the USB drive is possible only using the Ethernet interface. For this reason, commands that deal with the USB drive are reserved as Ethernet-only, and data can be read from a USB drive only by physically connecting it to a PC if the instrument is not fitted with the Ethernet option.

Ethernet Command/Response Interface

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
						E	E	E	E

Some models can be purchased with an Ethernet interface option (in addition to the standard USB interface.) Ethernet interface implementations support UDP communication over specific IP address ports:

UDP Port Number	Function
1235 (fixed)	Device's discovery receiving port
1234 (programmable)	PC's default discovery receiving port.
51235 (fixed)	Device's command receiving port
1234 (programmable)	PC's default status/data receiving port. Programmable via the PORT command.

Ethernet Command Format (type DQCommand, 0x31415926)

Commands can best be visualized as a packet structure whose various elements define the command type, its arguments, group id, and payload (if necessary):

```
struct Command {
   uint32 TYPE;
   uint32 GroupID;
   uint32 Command;
   uint32 arg0;
   uint32 arg1;
   uint32 arg2;
   char PayLoad[];
};
```

Ethernet Command Structure Elements	
Element	**Definition**
Type	Defines the command type. Currently there is only one supported command type: DQCommand with value 0x31415926
GroupID	GroupID is assigned by the Connect command (value = 10) and acts as a security measure to prevent unintended device access by unrelated Ethernet traffic. Assigned to all devices in a group, GroupID also allows configuration of all devices with the same GroupID using a single broadcasted UDP command. • GroupID = 0 indicates an idle (available) device.

Ethernet Command Structure Elements	
Element	**Definition**
	• When the internal GroupID is zero, the only command accepted is Connect (value = 10), which will assign a non-zero GroupID to the device • Once the GroupID is acquired by the device, it is used to compare with the GroupID in the command. It may be a good practice to change the GroupID every time acquisition is reconfigured, to prevent confusing data from a previous session with data from the current session.
`Command`	A command value that defines a specific action. Refer to both the Ethernet-specific, USB-specific, and Shared Protocol Command Set sections of this document for individual command values. All Shared protocol commands have a value of 13. Ethernet-specific command values are defined by the specific command.
`arg1-3`	Up to three parameters as required by the specific command
`PayLoad`	A null-terminated ASCII string or binary image as needed by the specific command.

Command Response (type DQResponse, 0x21712818)

When a Shared or Ethernet-specific command is issued with command type DQCommand (value 0x31415926 as defined above), responses are captured using the Command Response type:

```
struct Response{
      uint32 TYPE;
      uint32 GroupID;
      uint32 Order;
      uint32 PayLoadLength;
      char PayLoad [PayLoadLength];
};
```

Command Response Elements	
Element	**Definition**
`Type`	DQResponse with value 0x21712818
`GroupID`	See above Ethernet Command Format (type DQCommand, 0x31415926)
`Order`	Order of the instrument when used as a member of a sync group
`PayLoadLength`	The number of returned ASCII character in the payload
`PayLoad`	A null-terminated ASCII string represented by the command response

Invoking Shared Commands Using Ethernet

As mentioned, the Ethernet command set provides a mechanism for invoking commands that are shared between both USB and Ethernet interfaces of any given instrument. Here, we'll provide an example of basic shared command *info* to query a device for its model number using the Ethernet interface.

First, the command packet is constructed using the Ethernet command structure discussed above. The command `info 1` instructs the instrument to respond with its model number (we'll assume we're working with a DI-2108 in this example.)

```
struct Command {
     uint32 0x31415926;      'generic command type
     uint32 1234;            'arbitrary key value
     uint32 13;              'value of 13 denotes a shared command
     uint32;                 'ignored
     uint32;                 'ignored
     uint32;                 'ignored
     char "info 1";          'shared command to query model number
};
```

Transmission of the above packet triggers a response packet from the device:

```
struct Response{
       uint32 0x21712818;      'generic response type
       uint32 1234;            'arbitrary key value
       uint32;                 'ignored
       uint32 11;              'number of characters in the response
       char "info 1 2108";     'instrument's text response
};
```

Other Ethernet Response Types

In addition to the Command response, Ethernet devices support several other structured responses to convey data and configuration information.

Ethernet Device Discovery Response

Ethernet instruments are discovered by the host PC through use of a discovery query command consisting of a UDP broadcast:

```
dataq_instruments XXXX
```

where XXXX is optional and, if included, defines the PC port number to which the instrument will reply. If no port is specified, the instrument replies to default port number 1234.

The instrument's response to a discovery command is defined as follows:

```
<IP> <MAC> <SoftwareRev> <DeviceModel> <ADCRunning> <Reserved>
<LengthOfDescription>  <DeviceDescription> <SerialNumber> <GroupID>
<OrderInGroup> <Master/Slave> <ThumbRecording*> <TriggerCount*>
<RemainingDriveCapacity*>
```

* These fields are omitted if no USB drive is present

A sample response might be:

```
192.168.0.29 00:1B:81:77:42:34 117 4208 0 0 4 Dev0 4D5B903E 0 0 2 0 0 7773856
```

Detailed response definitions:

Detailed Ethernet Device Discovery Response	
Element	**Definition**
IP	The device's IPv4 address
MAC	The device's MAC address
SoftwareRev	The device's firmware revision, 2 hex bytes (e.g. $117_{16} = 279_{10}$ for firmware revision 2.79)
DeviceModel	Model number of the responding device without "DI-" prefix (e.g. DI-4208 replies as "4208")
ADCRunning	Returned as "0" (ADC not running) or "1" (ADC running)
Reserved	0
lengthOfDescription	The number of characters in <DeviceDescription>
DeviceDescription	The device's description as defined using the *info 5* command
SerialNumber	Returns the instrument's serial number (left-most 8 digits only; right-most two digits are for internal use)
GroupID	The group id assigned to the device

OrderInGroup	When operating with multiple, synchronized devices returns the device's order in the chain
Master/Slave	When operating with multiple, synchronized devices: "0" device master is false; "1" device master is true; "2" is for a device by itself (no sync).
ThumbRecording	USB Drive recording state: "0" recording is false; "1" recording is true
TriggerCount	Number of times the device has triggered (2^{16} max)
RemainingDriveCapacity	Returns the number of unused bytes in the attached USB drive

Active WinDaq File header Response (type WdqHeader, 0x05772156)

Instruments with the stand-alone option can record data to a connected USB drive in either a high resolution WinDaq format (WDH), or in a circular format that allows continuous recording regardless of file or USB drive size (WHC.) In such configurations, and in response to the "ud_status" command , this structure returns the embedded Windaq file header (WDH or WHC).

```
struct WDQHeader {
      uint32 TYPE;
      uint32 GroupID;
      uint32 Mode;
      uint32 PayLoadSize;
      char WdhHeader [PayLoadSize];
};
```

Ethernet Active WinDaq File Header Response Elements	
Element	**Definition**
Type	WdqHeader with value 0x05772156
GroupID	See above Ethernet Command Format (type DQCommand, 0x31415926)
Mode	0 for data from 0-1155 bytes, common between WDH and WHC header styles 1 for data > 1155 bytes to the end and specific to WHC header style
PayLoadSize	The number of returned bytes in the PayLoad
PayLoad	The returned WinDaq header with the number of bytes equal to PayLoadSize

ADC Data Stream Response (type DQAdcData, 0x14142135)

While an instrument is scanning it is possible to obtain the real time data stream from the device on a continuous basis. This response follows a SyncStart command issued via the Ethernet Command (type DQCommand, 0x31415926.)

```
struct ADCData {
      uint32 TYPE;
      uint32 GroupID;
```

```
      uint32 Order,
      uint32 CumulativeCount;
      uint32 PayLoadSamples;
      short ADCData [PayLoadSamples];
};
```

ADC Data Stream Response Elements	
Element	**Definition**
Type	DQAdcData with value 0x14142135
GroupID	See above Ethernet Command Format (type DQCommand, 0x31415926)
Order	Assigned order for the device sending out this packet of ADCData
CumulativeCount	The total number of samples delivered over successive responses during the current scanning operation. May be used with PayLoadSamples to detect data gaps where previous CumulativeCount plus current PayLoadSamples should equal current CululativeCount if no data gaps were experienced.
PayLoadSamples	The number of returned samples
AdcData	A number of Adc data values equal to PayLoadSamples.

Read USB Drive File Response:

(type UsbDriveData, 0x17320508) for data, or

(UsbDriveEOF, 0x22360679) for end of file

Read USB Drive File responds to command "ud_read" issued via the Ethernet Command (type DQCommand, 0x31415926.)

```
struct UsbDrive { //Retrieve content of a file
      uint32 Type;
      uint32 GroupID;
      uint32 Offset,
      uint32 PayLoadSize
      char   DriveData[PayLoadSize];
};
```

Read USB Drive File Response Elements	
Element	**Definition**
Type	UsbDriveData, 0x17320508 for data UsbDriveEOF, 0x22360679 for end of file
GroupID	See above Ethernet Command Format (type DQCommand, 0x31415926)
Offset	File offset

PayLoadSize	The number of returned bytes
DriveData	A number of byte values equal to PayLoadSize.

Shared and USB-specific Protocol Command Usage

The protocol employs an ASCII character command set that allows complete control of the instrument. All of the commands in the following table must be terminated with a carriage return character (0xD) to be recognized by the instrument. Command arguments (if any) are also ASCII, and the command and each argument must be separated by a space character (0x20). All commands echo if the instrument is not scanning. Command arguments and responses are always in decimal unless an instrument is instructed to begin scanning in its binary output mode. Finally, all commands are not supported by all instrument models. The instruments that support any specific command group are defined in the detailed documentation for that group.

All commands echo if the instrument is not scanning. Commands will not echo while scanning is active to prevent an interruption of the data stream. In this sense, the *start* command never echoes, and the *stop* command always echoes. In all the following descriptions and examples, references to command echoes assume that the instrument is not actively scanning.

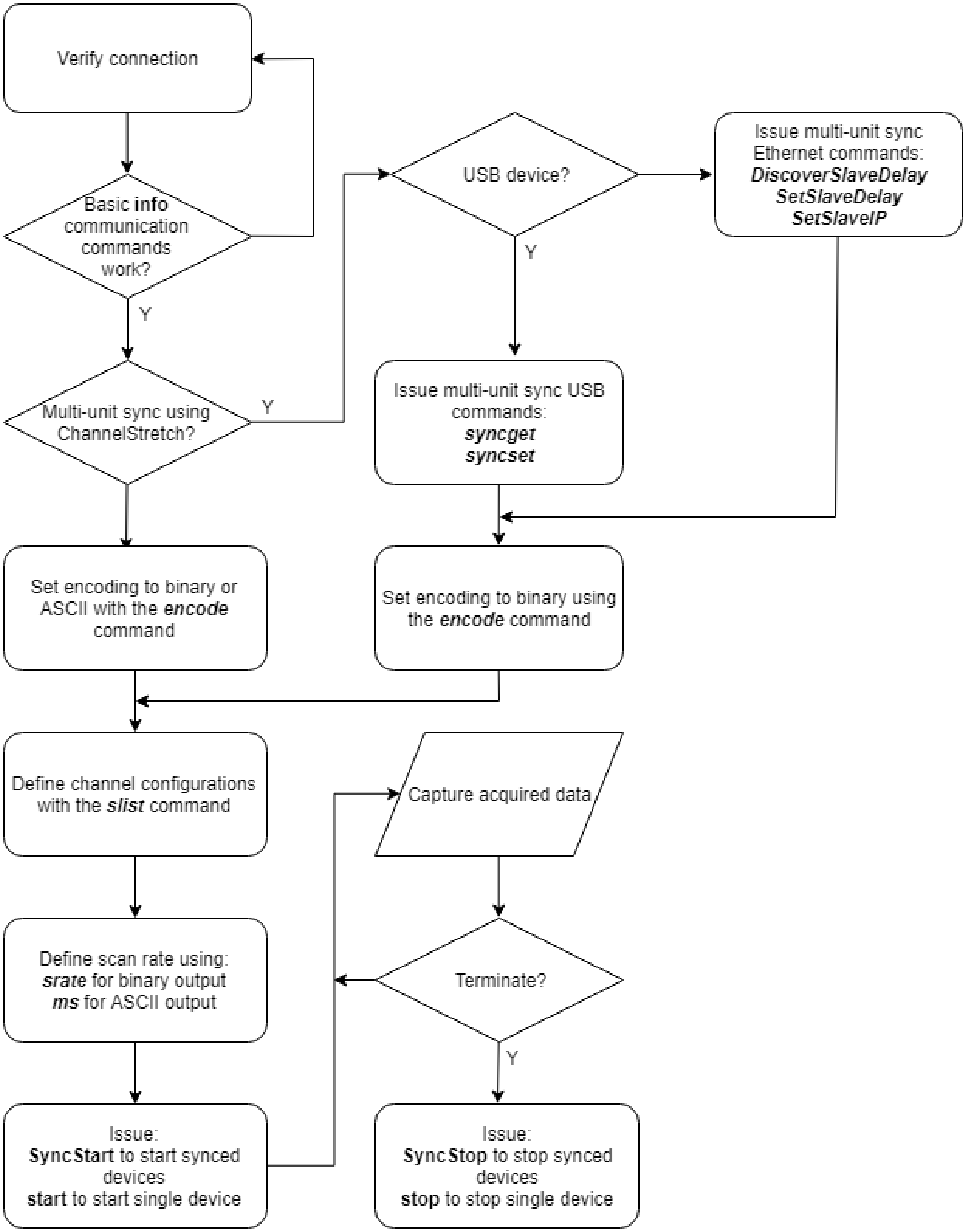
Please note that any given instrument's command buffer is very small. To prevent command buffer overruns ensure that you do not send a new command without first receiving the previous command's echo.

Protocol Command Set

Shared and USB-specific Command Set	
Command Groups	**Action**
Basic communication	
`*IDN?`	VISA-compatible device ID command
`info arg0`	Echoes the command and argument with additional information as defined by the argument
`ps arg0`	Defines communication packet size
USB Unit Synchronization	
`syncget arg0`	Sets and retrieves various synchronization timing parameters
`syncset arg0`	Sets the synchronization timing constant for the device
`syncstart arg0`	Starts multi-unit synchronized scanning (see the *start* command to start scanning with a single device)
Ethernet Interface Communication Configuration	
`Connect`	Establishes a connection with a data acquisition device
`Disconnect`	Breaks the connection with a data acquisition device
`KeepDeviceAlive`	Keeps device alive
`SetWdqHeader`	Updates the active WinDaq header stored in the device
`ipaddr`	Defines the instrument's IP address
`netmask`	Defines the instrument's IP address subnet mask
`gateway`	Defines the gateway IP address that the instrument will use
`keepalive`	Sets the device KeepDeviceAlive timeout value
`port`	Defines the UDP port number used by the PC.
Ethernet Unit Synchronization	
`SlaveIP`	Registers a given slave's IP address with the master
`SyncStart`	Starts a synchronized device acquisition session
`SyncStop`	Stops a synchronized device acquisition session
Date and time operations (for devices with the stand-alone option)	
`ymd`	Sets date of internal real time clock
`hms`	Sets time of day of internal real time clock
USB Drive Commands	
`ud_start`	Starts a USB drive recording session
`ud_stop`	Stops a USB drive recording session
`ud_ls`	Lists all files in the USB drive root directory
`ud_del`	Deletes a specified file from the USB drive root directory
`ud_read`	Retrieves data from a file on the USB drive

`ud_status`	Returns either the latest recorded scan from the USB drive during a recording session, or the current WinDaq header being used for the session
`ud_power`	Used to cycle power to the USB drive port
`ud_stream`	Streams data recorded to a USB drive in real time during a recording session
`ud_cfgwr`	Write WinDaq header, one byte at a time
`ud_cfgrd`	Read WinDaq header, one byte at a time
Measurement Configuration	
`encode arg0`	Defines device output coding
`eol arg0`	Defines EOL termination when outputting ASCII data
`slist arg0 arg1`	Defines scan list configuration
Scanning	
`srate arg0`	Defines scan rate
`start arg0`	Start single unit scanning (never echoes). See command *syncstart* for multi-unit, synchronized acquisition
`stop`	Stop scanning (always echoes)
Oversampling Mode Commands	
`filter arg0 arg1`	Defines one of several oversampling methods per channel
`dec arg0`	Defines the number of samples evaluated for the oversampling method
`deca argo`	A multiplier for the *dec* command
CJC Commands	
`cjcdelta`	Adjusts and reports CJC sensor offset
Rate measurement	
`ffl arg0`	Sets the digital filter length of the rate measurement digital input channel
LED color	
`led arg0`	Sets the LED to a specified color
Digital I/O	
`endo arg0`	Enables defined ports as inputs or outputs
`dout arg0`	Outputs the specified data to the digital output port
`din`	Returns the value of each digital port that is configured as an input
Reset	
`reset arg0`	Performs various reset operations

Typical Program Flow



Basic Communication Commands

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
S	S	S	S	S	S	S	S	S	S

Info Commands

Basic Communication Commands	
ASCII Command	Action
`info 0`	Returns "info 0 DATAQ"
`info 1`	Returns device model number (e.g."info 1 2008")
`info 2`	Returns firmware revision, 2 hex bytes (e.g. 65_{16} = 101_{10} for firmware revision 1.01)
`info 3 to info 4`	Proprietary internal use for initial system verification
`info 5 <arg0>`	Sets device description string arg0 (31 characters followed by null). Issued without <arg0> returns the previously defined device description string. arg0 may be contained within quotation marks to included spaces in device description. Model DI-2008 does not support info 5.
`info 6`	Returns the instrument's serial number (left-most 8 digits only)
`info 7 to info 8`	Proprietary internal use for initial system verification
`info 9`	Returns the sample rate dividend value based upon current settings (see the *srate* command for details)
`ps 0`	Make packet size 16 bytes (DEFAULT)
`ps 1`	Make packet size 32 bytes
`ps 2`	Make packet size 64 bytes
`ps 3`	Make packet size 128 bytes
`ps 4`	Make packet size 256 bytes
`ps 5`	Make packet size 512 bytes
`ps 6`	Make packet size 1024 bytes
`ps 7`	Make packet size 2048 bytes

The protocol supports a number of basic command/response items that provide a simple means to ensure the integrity of the communication link between a program and the instrument. These commands elicit simple, yet useful responses from the instrument and should be employed as the programmer's first communication attempt. If these commands don't work with a functioning instrument then a problem exists in the communication chain and further programming efforts will be futile until resolved.

Responses to this set of commands include echoing the command, followed by a space (20_{16}), followed by the response, and ending with a carriage return (D_{16}). For example:

```
Command:   info 1            'what model is connected?
Response:  info 1 2008       'command echo, plus connected model number

Command:   info 5 Dev0       'define device name as "Dev0"
Response:  info 5 Dev0       'command echo
Command:   info 5            'command without device name
Response:  info 5 Dev0       'echoes device name as previously defined

Command:   info 5 "Dev 0"    'define device name as "Dev 0"
Response:  info 5 "Dev 0"    'command echo
Command:   info 5            'command without device name
Response:  info 5 Dev 0      'echoes device name as previously defined
```

The packet size command defines the number of bytes the instrument sends with each transmission burst. The larger the packet size the more bytes transmitted per burst. Since a packet will not transmit until it is full, you should adjust packet size as a function of both sampling rate and the number of enabled channels to minimize latency when channel count and sample rate are low, and avoid a buffer overflow when sampling rate and channel count are high. In Ethernet mode, only `ps 6` or lower is accepted.

```
Command:   ps 1     'make packet size 32 bytes
Response:  ps 1     'command echo
Command:   ps       'query current packet size
Response:  ps 32    'packet size is set to 32 bytes
```

Multi-unit Synchronization Commands (USB)

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
		U	U		U	U	U	U	U

Many instruments support ChannelStretch™, synchronized data acquisition across multiple units of the same or sometimes different model. The commands in this group manage various aspects of the synchronization process when used with a USB interface.

syncget, syncset, syncstart Commands

These commands in combination manage synchronized sampling across multiple instruments, the feature colloquially referred to as *ChannelStretch* in of product literature. Each supports a 16-bit, unsigned number (in string format and in the range of "0" to "65535") as an argument, a returned value, or both as indicated. There is much that goes on in firmware to provide cross-unit synchronization, and a detailed treatment of that process is beyond the scope of this protocol. To simplify the functional application of synchronization we offer only a brief description of each synchronization command, and then pseudocode to show how they are applied.

Synchronization Command Modes*	
ASCII Command	**Action**
`syncget 0`	Returns the preferred synchronization timing constant of the device as an unsigned, 16-bit constant (0 to 65535)
`syncget 1`	Forces the device to re-evaluate the preferred synchronization timing constant, returns the resulting 16-bit, unsigned timing constant for the device, and sets a new value returned by the *syncget 0* command. This procedure takes two seconds to complete and is required when the device sends a *stop 03* error string in the returned data.
`syncget 2`	Returns the time parameter for the device, which is used by the *syncstart* command
`syncget 3`	Returns the active synchronization time constant of the device. If this value is equal for all synced devices, the *syncset* command is not required. Otherwise an averaged value is used (see pseudocode example.)
`syncset arg0`	Sets the synchronization timing constant for the device represented by *arg0* as an unsigned, 16-bit constant. it takes one parameter, which is the average of <x>s returned in *syncget 0* command from all devices involved in synchronization operation. Ensure that all synchronized devices must have the SAME *syncset* value.
`syncstart arg0`	Starts synchronized scanning. arg0 is the value returned by the *syncget 2* command with bit 10 inverted. The result must be ≥ 1.

* Output coding must be set to binary to synchronize devices. The feature is not supported in the ASCII output mode.

Typical Synchronization Procedure Pseudocode

Set up

Pseudocode for two-device, synchronized data acquisition. Command subscripts denote the target device for the command. It is assumed that both devices are connected and communicating. The delay between program line "$F = syncget_1\ 2$" and the last *syncstart* command must be less than 200 mS.

```
A = syncget₁ 0
B = syncget₂ 0
C = (A+B)/2
D = syncget₁ 3
E = syncget₂ 3
if not(D = E = C)
    syncset₁ C
    syncset₂ C
    delay 1 second
end if
F = syncget₁ 2
G = (F) XOR (0x0400)
if G = 0 then G = 1
syncstart₁ G
syncstart₂ G
```

Error handling

Pseudocode example to recover when odd-byte packet (indicating an error state) is received and the data stream has stopped and assuming we have two synchronized devices. In the pseudocode below error$ is the last seven bytes in the buffer concatenated into a string. The delay between program line "$F = syncget_1\ 2$" and the last *syncstart* command must be less than 200 mS.

```
if (error$ == "stop 03")
     A = syncget₁ 1
     B = syncget₂ 1
     C = (A+B)/2
     D = syncset₁ C
     E = syncset₂ C
     delay 1 second
end if
F = syncget₁ 2
G = (F) XOR (0x0400)
if G = 0 then G = 1
syncstart₁ G
syncstart₂ G
```

Ethernet Interface Configuration

For instruments equipped with an Ethernet, these commands allow interface configuration.

Connect (10)

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
						E	E	E	E

Connect Command Elements	
Parameters	Action
`Command value`	10 (sent using Ethernet Command Format, type DQCommand, 0x31415926)
`Response`	"connected" as retrieved using the Command Response (type DQResponse, 0x21712818)
`GroupID`	The group id to be assigned to the device (32-bit unsigned integer)
`arg0`	Lower word contains the desired PC UDP port. If arg0 = 0, default port of 1234 is used
`arg1`	arg1 = 1; This device is a master arg1 = 0; This device is a slave arg1 = 2; This device is used alone without synchronization with other devices
`arg2`	If this device is used in a synchronized group, this value is its order in the group ranging from 0 to the maximum number of devices. Otherwise, the value is ignored.

Disconnect (11)

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
						E	E	E	E

Disconnect Command Elements	
Parameters	Action
`Command value`	11 (sent using Ethernet Command Format, type DQCommand, 0x31415926)
`Response`	"disconnected" as retrieved using the Command Response (type DQResponse, 0x21712818)
`GroupID`	The device's group id assigned by the CONNECT command

KeepAlive (12)

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
						E	E	E	E

If any enabled device is not issued a KeepAlive command for more than 8 seconds it will drop its session and GroupID, and fall back to an idle state. Any GroupID matching the command serves as KeepAlive to the receiving device. Once scanning starts, only the master needs to receive the KeepAlive command to keep the sync group alive in a synchronized device setting.

This command does not generate a response.

SetWdqHeader (21)

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
						E	E	E	E

Updates the active WinDaq file header (WDH or WHC format) in the instrument, which defines default data acquisition parameters. Complete details for both WDH and WHC formats are published here. Alternatively,

SetWdqHeader Command Elements	
Parameters	**Action**
`Command value`	21 (sent using Ethernet Command Format, type DQCommand, 0x31415926)
`Response`	None
`arg0`	Offset of the header (in terms of bytes, but must be 4-byte aligned). If Offset is 1536, the header will be written to EEPROM
`arg1`	Length (1...1024) in terms of bytes, but must be in multiple of 4-byte
`PayLoad`	binary image of <length> to be written

ipaddr command

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
						S	S	S	S

Sets the IPv4 address of the device (IPv6 is not supported). Can be sent in one of two methods:

- As a text command via the USB interface
- Via Ethernet as a packet command using Ethernet Command Format, type DQCommand, 0x31415926 and command value 13. The response is retrieved using the Ethernet Command Response (type DQResponse, 0x21712818)

```
ipaddr arg0
```

Where: arg0 is the IP address in 'A.B.C.D' format

If arg0 is not supplied instrument returns its currently configured IP address

If arg0 = 0 instrument is set for DHCP

Command: `ipaddr 192.168.3.28` `'sets ip address`

Response: `ipaddr 192.168.3.28` `'command echo`

Command: `ipaddr` `'what is the current ip address?`

Response: `192.168.3.28` `'returns current ip address`

netmask command

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
						S	S	S	S

Sets the subnet mask (IPv4) for the device (IPv6 is not supported). Can be sent in one of two methods:

- As a text command via the USB interface
- Via Ethernet as a packet command using Ethernet Command Format, type DQCommand, 0x31415926 and command value 13. The response is retrieved using the Ethernet Command Response (type DQResponse, 0x21712818)

```
netmask arg0
```

Where: arg0 is the subnet mask in 'A.B.C.D' format

If arg0 is not supplied instrument returns its currently configured netmask address

Command: `netmask 255.255.0.0    'sets subnet mask`

Response: `netmask 255.255.0.0    'command echo`

Command: `netmask                'what is the current subnet mask?`

Response: `255.255.0.0            'returns current subnet mask`

gateway command

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
						S	S	S	S

Sets the gateway IP address (IPv4) for the device to use (IPv6 is not supported). Can be sent in one of two methods:

- As a text command via the USB interface
- Via Ethernet as a packet command using Ethernet Command Format, type DQCommand, 0x31415926 and command value 13. The response is retrieved using the Ethernet Command Response (type DQResponse, 0x21712818)

```
gateway arg0
```

Where: arg0 is the IP address in 'A.B.C.D' format

If arg0 is not supplied instrument returns its currently configured gateway address

Command: `gateway 192.168.3.1     'sets gateway IP address`

Response:	`gateway 192.168.3.1`	`'command echo`
Command:	`gateway`	`'what is the current gateway IP?`
Response:	`192.168.3.1`	`'returns current gateway ip`

keepalive command

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
						S	S	S	S

Sets the device keepalive timeout value. Can be sent in one of two methods:

- As a text command via the USB interface
- Via Ethernet as a packet command using Ethernet Command Format, type DQCommand, 0x31415926 and command value 13. The response is retrieved using the Ethernet Command Response (type DQResponse, 0x21712818)

```
keepalive arg0
```

Where: $0 \le agr0 \le 65535$ and represents the keepalive timeout to the nearest 10 mS. If zero, keepalive is disabled.
If arg0 is not supplied instrument returns its current keepalive value.

Command:	`keepalive 8000`	`'sets keepalive to 8 seconds`
Response:	`keepalive 8000`	`'command echo`
Command:	`keepalive`	`'what is the current keepalive value?`
Response:	`8000`	`'returns current keepalive value`

port command

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
						S	S	S	S

Sets the UDP the device will use to communicate with a PC. Can be sent in one of two methods:

- As a text command via the USB interface
- Via Ethernet as a packet command using Ethernet Command Format, type DQCommand, 0x31415926 and command value 13. The response is retrieved using the Ethernet Command Response (type DQResponse, 0x21712818)

```
port arg0
```

Where: $0 \le$ agr0 ≤ 65535 and represents the port number on the PC the device will use for UDP communication. Default is port 1234.
If arg0 is not supplied instrument returns the current port value.

Command:	`port 1272`	`'sets port to 1272`
Response:	`port 1272`	`'command echo`
Command:	`port`	`'what is the current PC UDP port?`
Response:	`1272`	`'returns current PC UDP port`

Multi-unit Synchronization Commands (Ethernet)

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
						E	E	E	E

Many instruments support ChannelStretch™, synchronized data acquisition across multiple units of the same or sometimes different model. The commands in this group manage various aspects of the synchronization process when used with an Ethernet interface.

SlaveIp (5)

Tells the master device in a chain about the slave devices it needs to sync with.

SlaveIp Command Elements	
Parameters	**Action**
`Command value`	5
`Response`	Slave IP address as retrieved using the Command Response (type DQResponse, 0x21712818)
`arg0`	The total number of slave devices in the group. If zero, no IP address is assigned. If negative, the device will not try to sync with other devices. Maximum number of IPs to register is 16.
`arg1`	Slave index for this device (zero-based). Value is 0-15.
`arg2`	Timing adjustment value. Always 2. Payload is the IP of a slave device

SyncStart (1)

Starts a synchronized data acquisition session within a group with common GroupID. If SYNCSTOP is issued, a delay of two seconds is required before the next SYNCSTART to ensure internal timer/housekeeping is performed. If the master contains no enabled channels, serving only as a sync facilitator, Payload should contain "mastersynconly." If WiFi connected devices are involved, unicast is necessary. If no WiFi is involved, then broadcast is preferred so that PC can send SYNCSTART to all devices in a more responsive manner.

SyncStart Command Elements	
Parameters	**Action**
Command value	1
Response	None
PayLoad	If the master contains no enabled channels, serving only as a sync facilitator, Payload should contain "mastersynconly". Otherwise, PayLoad should be empty.

SyncStop (6)

Ends a synchronized data acquisition session within a group with common GroupID. Payload should contain normal USB command "stop." If WiFi connected devices are involved, unicast is necessary, see above discussion. If no WiFi is involved, broadcast is preferred

SyncStop Command Elements	
Parameters	**Action**
Command value	6
Response	None
PayLoad	"stop"

Sample Pseudo Code for Ethernet Device Synchronization

The following pseudo code provides a general framework for how a synchronized data acquisition session between Ethernet-enabled devices is achieved. Use the GroupID attribute to broadcast time-critical, simultaneous commands SyncStart and SyncStop.

```
for i=0 to N
'use Ethernet Connect command to define group ids, master, slave
    Connect device(i) with same group id
    if i=0 then
        device(0)=master
    else
        device(i)=slave
    endif
next

'tell master the IP address of each slave
for i = 1 to n
    SlaveIp (device0, device(i))
next

for i = 0 to n
'define data acquisition settings for all devices
next
```

```
'begin acquiring data
SyncStart device(all) 'use GroupID to broadcast to all devices at once

do
    read data from device(all) 'use ADC Data Stream Reponse type 0x14142135

    if data stream has gap then
        fill buffer with NaN
    endif

    send keepalive to device(0) before programmed timeout

loop until stop

SyncStop device(all) 'use GroupID to broadcast to all devices at once
'delay at least two seconds before issuing another SyncStart
```

Date and Time Commands

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
				S		S	S	S	S

Date and time operations allow configuration of the internal real time clock present in all instruments with the stand-alone option that allow recording to a USB drive without a connected PC. We recommend defining time as UTC to support data portability across time zones.

ymd command

Sets real time clock date

```
ymd arg0
```

Where: arg0 is the date in 'yyyy/mm/dd' format

Command: `ymd 2018/05/15` `'sets date to May 15, 2018`
Response: `ymd 2018/05/15` `'command echo`

Command: `ymd` `'what is the date?`
Response: `ymd 2018/05/15` `'returns current date`

hms command

Sets real time clock time-of-day

```
hms arg0
```

Where: arg0 is the time in 'hh:mm:ss' format (using a 24-hour clock format)

Command:	`hms 15:13:25`	`'sets time to 15:13:25 (03:13:25 pm)`
Response:	`hms 15:13:25`	`'command echo`

Command:	`hms`	`'what is the date?`
Response:	`hms 15:13:25`	`'returns current date`

USB Drive Commands

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
						E	E	E	E

UsbDrive (22)

Provides access to a connected USB drive for instruments purchased with the stand-alone option. Responses are handled via the Read USB Drive File Response types (UsbDriveData, 0x17320508 for data, and UsbDriveEOF, 0x22360679 for end of file)

UsbDrive Command Elements	
Parameters	Action
`Command value`	22
`Response`	Via USB Drive File Response types UsbDriveData, 0x17320508 for data, and UsbDriveEOF, 0x22360679 for end of file
`PayLoad`	USB drive command in text format as defined immediately below. All commands and responses are strings. Command or response string values are in base 10.

The following USB drive commands are supported:

ud_start

ud_start Command	
Command	**Action**
`ud_start`	Initiates a USB drive recording session

ud_stop

ud_stop Command	
Command	**Action**
`ud_stop`	Stops a USB drive recording session

ud_ls <arg0> <arg1>

list Command	
Command	**Action**
`ud_ls`	Returns a list of all files in the USB drive. Response can be terminated by sending PayLoad "ctrlc" from the UsbDrive(22) command. <arg 0> specifies the offset of the list, where "0" points to the beginning. <arg 1> specifies the number of lines after the offset, where "0" means all lines.

ud_del <arg0>

del Command	
Command	**Action**
`ud_del`	Deletes the file named by <arg0> from the root directory of the USB drive
`arg0`	The text filename to be deleted. If agr0 = "*.*" all files are deleted from the root directory of the USB drive

ud_read <arg0> <arg1> <arg2>

del Command	
Command	**Action**
`ud_read`	Retrieves data from specified file using Read USB Drive File Response types (type UsbDriveData, 0x17320508 for data, or UsbDriveEOF, 0x22360679 for end of file)
`arg0`	The name of the file to be retrieved

arg1	The first byte to be returned as an offset into the file
arg2	The number of bytes to be retrieved

ud_status <arg0> <arg1>

status Command	
Command	**Action**
ud_status	Performs one of two operations: Returns the latest readings for all enabled channels from the current USB drive recording session; Returns the active Windaq header for USB drive operation. Response can be terminated by sending PayLoad "ctrlc" from the UsbDrive(22) command.
arg0	"0" = Return the latest readings "1" = Return active WinDaq header
arg1	Used if arg0 = "1" "0" = Returning the active WDH header "1" = Returning the segment beyond WDH to WHC

ud_power <arg0>

ud_power Command	
Command	**Action**
ud_power	Controls power on the USB port
arg0	"0" = Turns USB port power off "1" = Turns USB port power on Power cycling the USB port forces a USB drive reboot

ud_stream <arg0>

ud_stream Command	
Command	**Action**
ud_stream	Controls whether data will be streamed to the PC while recording to the USB drive.
arg0	"0" = No real time streaming > "0" = Real time streaming enabled. adc data is streamed to the PC that sent the command while recording to USB drive. arg0 value serves as the preferred transmission size in terms of scans. Note the max UDP packet size supported by the device is 1432 bytes. Gaps may occur if sample rate is set too high. When a gap occurs the next transmit will begin with the latest data point with Realigned flag set (see Real Time USB Drive Data Response type RTUsbDriveData, 0x16180339)

ud_cfgwr <arg0> <arg1>

ud_cfgwr Command	
Command	**Action**
`ud_cfgwr`	Used to define re-trigger mode and to write a WinDaq configuration header to the instrument to be used in stand-alone applications. All arguments are string values, and when numerical, are expressed in base 10.
`arg0`	If arg0 = "rearm", arg1 is a string value interpreted as the trigger mode. Otherwise, arg0 is a string value representing a byte index for writing WinDaq header values.

`arg1`	If arg0 = "rearm", arg1 = "1" for trigger auto-rearm; arg1 = "0" for single-shot trigger operation.
	If arg0 <> "rearm ", arg0 is the byte index for writing a WinDaq file header ranging from 0 to 1535; arg1 is the byte value to be written. When arg0 = "1536" all header values are written to the instrument's EEPROM. For example:
	`ud_cfgwr rearm 1     'enable trigger auto re-arm mode`
	`ud_cfgwr 0 18        'write bit pattern 00010010 to header position 0`
	`ud_cfgwr 1536        'write WinDaq header to EEPROM`
	Refer to this online resource for details related to WinDaq file header construction.

ud_cfgrd <arg0>

ud_cfgrd Command	
Command	**Action**
`ud_cfgrd`	Used to read re-trigger mode and a WinDaq file configuration header from the instrument. All arguments and responses are string values, and when numerical, are expressed in base 10.

`arg0`	If arg0 = "rearm ", returned value is either "1" for auto re-arm enabled, or "0" for single-shot.
	If arg0 <> "rearm" arg0 is treated as a byte index to the WinDaq header defined using the ud_cfgwr command and the instruction returns the string representation for that index. When arg0 <> "rearm" the first index read must always be index 0. Examples:
	`ud_cfgrd rearm       'what is the devices rearm mode?`
	`ud_cfgrd rearm 1     'rearm mode is auto re-arm`
	`ud_cfgrd 0           'what value is at WinDaq header index 0?`
	`ud_cfgrd 0 117       ' WinDaq header index 0 contains bit pattern 01001011`
	`ud_cfgrd 157         'what value is at WinDaq header index 157?`
	`ud_cfgrd 157 87      ' WinDaq header index 0 contains bit pattern 01010111`

Measurement Configuration Commands

encode Command

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
S	S	S		S		S	S	S	S

The *encode* command defines the instrument's output coding format:

encode Command*	
ASCII Command	**Action**
`encode 0`	Enables the binary output mode (DEFAULT)
`encode 1`	Enables the ASCII, floating-point output mode. Supports analog input channels only, and the device must be configured for CDC mode of operation. Libusb is not supported, nor is multi-unit synchronization using ChannelStretch™. Measurement range commands using slist are ignored while the instrument is in the ASCII mode.
`encode 129`	Same as encode 1 but with serial date number in the first column.

*Refer to the binary and ASCII stream data format sections in this document for output formatting details

Command: `encode 0 'enable binary output mode`
Response: `encode 0 'always echoes`

Command: `encode 1 'enable ASCII floating-point output mode`
Response: `encode 1 'always echoes`

eol Command

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
S	S	S		S		S	S		S

The *eol* command defines a termination character the device uses for ASCII mode data output (see the *encode* command.) Instrument models that do not support the *eol* command terminate with a carriage return character (D_{16}.) The *eol* command is valid only while in the ASCII data output mode (*encode* = 1.) Attempting to issue the command while in the binary data output mode results in a *command not found* response.

eol Command	
ASCII Command	**Action**
`eol 0`	Terminate ASCII data output with a carriage return (D_{16}) character (DEFAULT)
`eol 1`	Terminate ASCII mode output with a line feed character (A_{16}.)
`eol 2`	Terminate ASCII mode output with carriage return and line feed ($D_{16}A_{16}$) characters.

slist Command

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
S	S	S	S	S	S	S	S	S	S

Instruments employ a scan list approach to data acquisition. A scan list is an internal schedule (or list) of channels to be sampled in a defined order. It is important to note that a scan list defines only the type and order in which data is to be sampled, not the sampled data itself. Since instruments support different measurement characteristics, their scan list architectures will vary. Be certain to refer to the scan list architecture for your specific instrument.

During general-purpose use each entry in the scan list is represented by a 16-bit number, which is defined in detail in the *Scan List Word Definitions* tables below for each instrument model. Writing any value to the first position of the scan list automatically resets the slist member count to 1. This count increases by 1 each time a new member is added to the list, which must be filled from lowest to highest positions. The first item in the scan list initializes to 0 (typically analog input channel 0) upon power up. Therefore, upon power up, and assuming that no changes are applied to the scan list, only analog input channel 0 is sampled when scanning is set to active by the start command.

The *slist* command along with two arguments separated by a space character is used to configure the scan list:

slist offset config

offset defines the index within the scan list and can range from 0 to 10 to address up to eleven possible positions depending upon instrument model. *config* is the 16-bit configuration parameter as defined in table *Scan List Word Definitions tables* for each instrument model .

For example, the command *slist 5 10* sent to model DI-2008 configures the sixth position of the scan list to specify data from the counter. Continuing with the DI-2008 and assuming that we wish to sample analog channels 2 and 4 (on their ±10 V scale), and 6 (on its ±2.5 V scale) , and the rate, counter, and digital inputs, the following scan list configuration would work:

```
slist 0 2562
slist 1 2564
slist 2 3078
slist 3 9
slist 4 10
slist 5 8
```

Note that since the act of writing to scan list position 0 resets the slist member counter, the above configuration is complete upon writing scan list position 5. Other considerations:

- Any scan list position (except position 0) may be modified without affecting the contents of the rest of the list.
- Channel type definitions may be placed in the scan list in any order that satisfies the requirements of the application.
- Any analog, digital input, rate, or counter channel may appear in the scan list only once.
- *slist* positions must be defined sequentially beginning with position 0.
- To be consistent with general programming standards, analog channel numbers begin with 0 instead of 1 as indicated the product labels.

DI-1100 Scan List Definition

DI-1100 Scan List Word Definitions																
	Bit Position															
Function	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Analog In, Channel 0	All Unused Bits = 0												0	0	0	0
Analog In, Channel 1													0	0	0	1
Analog In, Channel 2													0	0	1	0
Analog In, Channel 3													0	0	1	1
Ignore	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

DI-1120 Scan List Definition

<table>
<tr><th colspan="17">DI-1120 Scan List Word Definitions</th></tr>
<tr><th rowspan="2">Function</th><th colspan="16">Bit Position</th></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>Analog In, Channel 0</td><td colspan="4" rowspan="4">Unused bits =0</td><td colspan="4" rowspan="4">Analog Range (see Analog Measurement Range table)</td><td colspan="4" rowspan="4">Unused bits =0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>Analog In, Channel 1</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>Analog In, Channel 2</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>Analog In, Channel 3</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>Digital In</td><td colspan="12">Unused bits =0</td><td>1</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>Rate (DI2)</td><td>0</td><td>0</td><td>0</td><td>0</td><td colspan="4">Rate Range (see Rate Range table)</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>Count (DI3)</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>Ignore</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
</table>

DI-2008 Scan List Definition

<table>
<tr><th colspan="17">DI-2008 Scan List Word Definitions</th></tr>
<tr><th rowspan="2">Function</th><th colspan="16">Bit Position</th></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>Analog In, Channel 0</td><td colspan="3" rowspan="8">Unused bits = 0</td><td rowspan="8">Mode (see DI-2008 Measurement Table)</td><td rowspan="8">Range (see DI-2008 Measurement Table)</td><td colspan="3" rowspan="8">± Full Scale value or TC type
(see DI-2008 Measurement Table)</td><td colspan="4" rowspan="8">Unused bits = 0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>Analog In, Channel 1</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>Analog In, Channel 2</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>Analog In, Channel 3</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>Analog In, Channel 4</td><td>0</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>Analog In, Channel 5</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>Analog In, Channel 6</td><td>0</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>Analog In, Channel 7</td><td>0</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>Digital In</td><td colspan="12">Unused bits = 0</td><td>1</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>Rate (DI2)</td><td>0</td><td>0</td><td>0</td><td>0</td><td colspan="4">See DI-2008 Rate Measurement</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>Count (DI3)</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>Ignore</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
</table>

DI-2008 Measurement Table					
Scan List Bit Position					
12 and 11			10	9	8
Range = 0 Mode = 0	Range = 1 Mode = 0	Range = don't care Mode = 1			
±500 mV	±50 V	B thermocouple	0	0	0
±250 mV	±25 V	E thermocouple	0	0	1
±100 mV	±10 V	J thermocouple	0	1	0
±50 mV	±5 V	K thermocouple	0	1	1
±25 mV	±2.5 V	N thermocouple	1	0	0
±10 mV	±1 V	R thermocouple	1	0	1
n/a	n/a	S thermocouple	1	1	0
n/a	n/a	T thermocouple	1	1	1

DI-2108 and DI-1110 Scan List Definition

DI-2108 and DI-1110 Scan List Word Definitions																
Function	Bit Position															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Analog In, Channel 0	All Unused Bits = 0												0	0	0	0
Analog In, Channel 1													0	0	0	1
Analog In, Channel 2													0	0	1	0
Analog In, Channel 3													0	0	1	1
Analog In, Channel 4													0	1	0	0
Analog In, Channel 5													0	1	0	1
Analog In, Channel 6													0	1	1	0
Analog In, Channel 7													0	1	1	1
Digital In													1	0	0	0
Rate (DI2)	0	0	0	0	Range (see Rate Range table)				0	0	0	0	1	0	0	1
Count (DI3)	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
Ignore	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

DI-2108P, 4108, 4208, and 4730 Scan List Definition

<table>
<tr><th colspan="17">DI-2108P, DI-4108, DI-4208, and DI-4730 Scan List Word Definitions</th></tr>
<tr><th rowspan="2">Function</th><th colspan="16">Bit Position</th></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>Analog In, Channel 0</td><td colspan="4" rowspan="8">Unused bits =0</td><td colspan="4" rowspan="8">Analog Range (see Analog Measurement Range table)</td><td colspan="4" rowspan="8">Unused bits =0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>Analog In, Channel 1</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>Analog In, Channel 2</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>Analog In, Channel 3</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>Analog In, Channel 4</td><td>0</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>Analog In, Channel 5</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>Analog In, Channel 6</td><td>0</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>Analog In, Channel 7</td><td>0</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td>Digital In</td><td colspan="12">Unused bits =0</td><td>1</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>Rate (DI2)</td><td>0</td><td>0</td><td>0</td><td>0</td><td colspan="4">Rate Range (see Rate Range table)</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>Count (DI3)</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>Ignore</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
</table>

DI-1120, DI-2108P, DI-4108 and DI-4208 Analog Measurement Range Table							
Bit Position					Range (Volts full scale)		
11	10	9	8	DI-4108	DI-1120 DI-4208	DI-2108P	DI-4730
0	0	0	0	±10	±100	±10	±1000
0	0	0	1	±5	±50	±5	±100
0	0	1	0	±2	±20	±2.5	±10
0	0	1	1	±1	±10	0 to 10	±1
0	1	0	0	±0.5	±5	0 to 5	±0.1
0	1	0	1	±0.2	±2	(undefined)	±0.01

DI-4718B Scan List Definition

DI-4718B Scan List Word Definitions																
Function	Bit Position															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Analog In, Channel 0	All Unused Bits = 0												0	0	0	0
Analog In, Channel 1													0	0	0	1
Analog In, Channel 2													0	0	1	0
Analog In, Channel 3													0	0	1	1
Analog In, Channel 4													0	1	0	0
Analog In, Channel 5													0	1	0	1
Analog In, Channel 6													0	1	1	0
Analog In, Channel 7													0	1	1	1
Digital In													1	0	0	0

Rate Range Table for Supported Instruments

The protocol also supports a range setting for rate measurements where a count value may be converted to a frequency in Hertz by applying the following formula:

$$rate = \frac{counts + 32768}{65536} \times range$$

"Range" is defined in the following table. Refer to the instrument's specifications for the maximum measurable rate as a function of burst rate.

Rate Range Table for Supported Instruments				
Bit Position				Range* (Hz)
11	10	9	8	
0	0	0	1	50,000
0	0	1	0	20,000
0	0	1	1	10,000
0	1	0	0	5,000
0	1	0	1	2,000
0	1	1	0	1,000
0	1	1	1	500
1	0	0	0	200
1	0	0	1	100
1	0	1	0	50
1	0	1	1	20
1	1	0	0	10

* For all products other than the DI-2008 (see below) maximum measurable frequency is a function of *srate* (see *srate* Scan Rate Command) and duty cycle of the applied signal: *srate* < 60,000,000 × ((duty cyle) ÷ 50%) ÷ (Range × 2), where srate ≥ 500 (burst rate ≤ 160,000 Hz) with one channel enabled, and duty cycle is the percentage of the cycle for the shorter input state.

Rate measurement using Model DI-2008 is not restricted by either duty cycle or *srate*. Minimum pulse width is 500 nS when making rate measurements using the DI-2008.

Scanning Commands

srate Scan rate Command

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
S	S	S	S	S	S	S	S	S	S

Command *srate* defines the value of a sample rate divisor used to determine scan rate, or the rate at which the instrument scans channels that you enabled with the *slist* command. The various instruments supported by this protocol can behave differently as a function of a number of variables. The general form of the equation that defines sample rate is:

Sample rate type (Hz) = (dividend) ÷ (srate × dec × deca)

The values assumed by all variables in the above equation are described in the following table as a function of instrument model number:

srate Variable Table

srate Variable Values By Model							
	DI-1100*	DI-1110	DI-1120	DI-2008*	DI-2108	DI-2108P	DI-4108, DI-4208, DI-4718B, DI-4730
Sample rate type	per channel	per channel	per channel	throughput	per channel	throughput	per channel
srate min	1 achn: 1500 2 achn: 2000 3 achn: 2500 4 achn: 3000	375	375	4	375	750	375
srate max	65535	65535	65535	2232	65535	65535	65535
dec support	no	no	yes	yes	yes	yes	yes
dec min**	1 (fixed)	1 (fixed)	1	1	1	1	1
dec max**	1 (fixed)	1 (fixed)	512	32767	512	512	512
*deca min**	1	1	1	n/a***	1	1	1
*deca max***	40000	40000	40000	n/a***	40000	40000	40000
dividend	60,000,000	60,000,000	60,000,000	1 achn: 8000 2-8 achn: 800	60,000,000	120,000,000	60,000,000

* "achn" is the number of enabled analog channels on the indicated device

** The device's supported decimation factors. Refer to the *filter* command for details.

*** Model DI-2008 does not support the deca command

Sample rate is defined as either a per channel or throughput rate. For models that support throughput values, the sample rate per channel is throughput divided by either the number of enabled analog channels, or the number of analog and digital channels depending upon the model.

For the DI-2008, sample rate per channel is the throughput rate divided by the number of enabled analog channels only regardless of the number of enabled digital channels.

For the DI-2108P, sample rate per channel is the throughput rate divided by the total number of analog and digital channels. In this case total channels can be as high as 11 (eight analog, one for all digital input ports, one for the counter, and one for the rate.)

start Command

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
S	S	S	S	S	S	S	S	S	S

Since the *start* command immediately initiates scanning, the command is never echoed:

Start Command Modes	
ASCII Command	**Action**
`start 0`	Begin scanning: The instrument begins scanning the channels enabled in its scan list through the *slist* command at a rate defined by either the *srate* or *ms* commands depending upon output coding defined by the *encode* command.

```
Command:    start 0   'begin scanning
Response:             'never echoes
```

stop Command

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
S	S	S	S	S	S	S	S	S	S

The protocol's *stop* command terminates scanning. Since the *stop* command terminates scanning, it is always echoed.

Command:	`stop`	`'stop scanning`
Response:	`stop`	`'always echoes`

The instrument has the ability to detect that its internal 1024-sample buffer has overflowed. Should this error condition occur the instrument will stop scanning and place *stop 01* in the last seven bytes of its final response. Buffer overflows can be greatly minimized by ensuring that the instrument buffer is flushed continuously and frequently in a high-priority routine.

Command:	`(none)`	`'scanning unexpectedly stops`
Response:	`stop 01`	`'1024-sample buffer has overflowed`

Oversampling Mode Commands

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
S	S	S	S	S	S	S	S	S	S

Most instruments support a range of oversampling modes that are selectable per channel:

Supported Oversampling Modes by Model (DI-)										
Oversampling Mode	1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
Last point	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CIC filter	*	*	✓		✓		✓	✓	✓	✓
Average	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Maximum	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Minimum	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

* CIC filter mode is active during acquisition only if the instrument is in the ASCII mode as defined by the *encode* command.

filter command

All instruments that support oversampling modes can be programmed to report the last point that was acquired, and the maximum or the minimum of a range of values. Filtered results are possible as either average or CIC filtered values depending upon model. Oversampling mode is programmed on a per channel basis using the *filter* command, which command accepts two arguments of the form:

```
filter arg0 arg1
```

Where: 0 ≤ *arg0* ≤ 7 and is equal to a specific analog channel number.

0 ≤ *arg1* ≤ 3:

arg1	
Value	Oversampling Mode
0	Last Point
1	Average or CIC filter depending upon instrument model
2	Maximum
3	Minimum

dec Command

A decimation factor (*dec*) must be applied to define the number of samples used by each acquisition mode (except Last Point.) For example, if *dec* has a value of 100 and the *filter* command defines the maximum oversampling mode, one value is reported for every 100 that are acquired, the maximum of the 100 samples. The next acquired 100 values are evaluated and the maximum value is reported, and so on. Setting *dec* to a value of 1 essentially forces the filter's last point mode even if maximum, minimum, or average (CIC filter) is specified.

```
dec arg0
```

Refer to the *Sample Rate Variable Values By Model* table above for valid *arg0* values for each model. Sample oversampling and decimation commands and responses:

Command:	`filter 1 2`	`'set analog channel 1 to maximum oversampling mode`
Response:	`filter 1 2`	`'analog channel 1 set to maximum oversampling mode`
Command:	`dec 128`	`'set the decimation factor to 128`
Response:	`dec 128`	`'the current decimation factor is 128`

deca Command (firmware version 1.21 and higher)

A decimation factor defined by the *dec* command may be multiplied to include more samples through use of the *deca* command. For example, if dec is set to the value of 232 and *deca* is set to the value 10, the total number of samples evaluated for the applied filter mode is 2320 (232 × 10.) To take complete advantage of CIC filters for the products that support them, *dec* should always be defined to be as large as possible. For example, if a total decimation factor of 1024 is desired, *dec* and *deca* should be defined as 512 and 2 respectively. For all desired decimation factors less than or equal to 512, *deca* should be set to 1 (i.e. providing no multiplier effect.)

```
deca arg0
```

Where: $1 \le arg0 \le 40000$. *deca* default value is 1.

Refer to the *Sample Rate Variable Values By Model* table above for valid *arg0* values for each model. Sample oversampling and decimation commands and responses:

Command:	`filter 1 2`	`'set analog channel 1 to maximum oversampling mode`
Response:	`filter 1 2`	`'analog channel 1 set to maximum oversampling mode`
Command:	`dec 512`	`'set the decimation factor to 512`
Response:	`dec 512`	`'the current decimation factor is 512`
Command:	`deca 3`	`'set the decimation multiplier factor to 3` `'overall decimation factor is 1536`
Response:	`deca 3`	`'the current decimation multiplier factor is 3`

cjcdelta Command

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
			U						

The *cjcdelta* command reads or applies CJC (cold junction compensation) offsets per channel to ensure measurement accuracy when using thermocouples. Since accurate thermocouple measurements depend upon an equally accurate measurement of junction temperature (where the thermocouple connects to the instrument), the *cjcdelta* command exists to ensure accurate junction temperature readings. Adjustments using *cjcdelta* should be applied only on a channel with a connected NIST-traceable thermocouple whose junction is held in an ice bath. This allows *cjcdelta* to adjust the measured temperature to 0°C, ± 0.0625°C.

The command syntax of *cjcdelta* consists of the command with at least one, but no more than two ASCII integer arguments that are separated by a space character and terminated with a carriage return character. General forms of the command follow:

`cjcdelta -1`

Reads CJC offsets from all channels and returns eight values separated by a space in the order of channel 0 to channel 7. Each value will fall in the range of -100 to +100. An offset may be calculated by multiplying the value returned per channel by 0.0625°C.

`cjcdelta arg0 arg1`

Sets a defined CJC offset for a defined channel,

where:
0 ≤ arg0 ≤ 7 and represents the channel number
-100 ≤ arg1 ≤ 100 and represents the offset multiplier (j * 0.0625°C)

`Cjcdelta -2`

Writes CJC offsets to the DI-2008's flash memory.

Note that CJC offsets are inversely proportional to temperature. Higher offset values decrease temperature readings, and lower values increase temperature readings.

Rate Measurement Moving Average

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
	S	S	S	S	S	S	S		S

When the rate channel is enabled in the instrument's scan list using the *slist* command, a moving average filter may be applied to smooth readings. The moving average factor is defined by the command:

ffl arg0

where 1 ≤ *arg0* ≤ 64 and the default value is 32.

Command: `ffl 20 'set the MA factor to 20`

Response: `ffl 20 'the current MA factor is 20`

LED Color Command

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
S	S	S	S	S	S	S	S	S	S

Instruments have a panel-mounted, multi-color LED for general-purpose use. The *led* command accepts one argument that defines the color of the LED, and takes the following form:

`led arg0`

Where:

arg0	Color	arg0	Color
0	Black	4	Red
1	Blue	5	Magenta
2	Green	6	Yellow
3	Cyan	7	White

Command: `led 1 'set the led color to blue`

Response: `led 1 'the led color is blue`

Digital I/O Commands

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
	S	S	S	S	S	S	S	S*	S

*D0 and D1 only. Fixed as inputs

The protocol supports three commands for digital I/O, and instruments provide seven ports for that purpose. Each port can be programmed as either an input or an output, with input as the default state for all. A port configured as an output is really a switch that is either on or off to control an external load.

One command (*endo*) defines configuration on a per port basis, input or switch. A second command (*dout*) defines the state of a port's switch if the port is configured as an output. The third command (*din*) reads the state of all ports regardless of I/O configuration.

endo command

```
endo arg0
```

Where: $0 \leq arg0 \leq 127_{10}$ and maps input/switch configuration to each of seven digital ports where bit *n* represents D*n*. A value of one written to a port configures it as a switch. A value of zero configures the port as an input.

Command:
```
endo 20   'ports D0,D1,D3,D5,D6 as inputs
          'ports D2 and D4 as switches
```
Response:
```
endo 20   'command echo
```

All digital ports are re-initialized when endo is issued.

dout command

```
dout arg0
```

Where: $0 \leq arg0 \leq 127_{10}$ ($0 \leq arg0 \leq$ 0x7F) and defines the bit state of the 7-bit output port where bit *n* represents D*n*.

Command:	`endo 20`	`'ports D0,D1,D3,D5,D6 as inputs`
		`'ports D2 and D4 as switches`
Response:	`endo 20`	`'command echo`

Command:	`dout 4`	`'set D2 switch is on. D4 switch is off`
Response:	`dout 4`	`'command echo`

din command

```
din
```

Command:	`din`	`'read all port states`
Response:	`din 20`	`'ports D2 and D4 are set. Others are clear`

din does not discriminate between ports configured as inputs or as switches. The command simply returns the state of all ports as a 7-bit value where bit *n* represents D*n*. A port configured as a switch returns the state of the switch. One configured as a digital input returns the applied state.

Reset Command

Supported Instrument Models (DI-)									
1100	1110	1120	2008	2108	2108P	4108	4208	4718B	4730
	S	S	S	S	S	S	S		S

The reset command resets accumulated counts to zero for any instruments that support a counter input

```
reset arg0
```

Where: arg0 = 1 to reset the counter to a count of zero

Command: `reset 1 'reset the counter`

Response: `reset 1 'command echo`

Binary Stream Output Formats

All products that are a part of this protocol document can output a binary response, which is the preferred format for implementations that require a fast sample rate. When the response mode is binary and the *start* command is received data begins issuing from the device in an order and rate defined by the previously configured scan list and sample rate. When the end of data defined by the last scan list element is reached the instrument wraps back to data represented by the first scan list element and repeats. The result is an instrument that sends a continuous binary data stream that stops only after receiving the *stop* command. For this reason, properly interpreting data in real time delivered from any instrument configured for a binary output format can be challenging for even the most experienced programmer.

Binary data output format changes depending upon the instrument to accommodate the devices unique measurement characteristics. In each situation, nomenclature will remain consistent: A_x values denote analog channel ADC values, and D_x, R_x and C_x are digital, rate, and counter value inputs respectively. In all cases the 'x' subscript represents the bit position of the binary value.

Binary Coding Notes

Binary data output for ADC values varies by instrument. However, for instruments that support counter and rate inputs, meaningful values are extracted across all product models by applying the following formulas:

$$counter\ value = counts + 32768$$

$$rate = \frac{rate\ value + 32768}{65536} \times range$$

Where: *counts* is the 16-bit value provided by the instrument for count.
rate value is the 16-bit value provided by the instrument for rate.
range is the selected rate measurement range in Hz (see Rate Range Table in the slist section for your specific instrument)

DI-1100 Binary Stream Output Format and ADC Coding

DI-1100 Binary Data Stream Example (assumes all functions and channels enabled in order in the scan list)										
Scan list position (measurement	Word Count	Byte Count	B7	B6	B5	B4	B3	B2	B1	B0
0 (Analog in 0)	1	1	A3	A2	A1	A0	0	0	D1	D0
		2	A11	A10	A9	A8	A7	A6	A5	A4
1 (Analog in 1)	2	3	A3	A2	A1	A0	0	0	0	0
		4	A11	A10	A9	A8	A7	A6	A5	A4
2 (Analog in 2)	3	5	A3	A2	A1	A0	0	0	0	0
		6	A11	A10	A9	A8	A7	A6	A5	A4
3 (Analog in 3)	4	7	A3	A2	A1	A0	0	0	0	0
		8	A11	A10	A9	A8	A7	A6	A5	A4

The DI-1100 transmits a 12-bit binary number for every analog channel conversion in the form of a signed, 12-bit Two's complement value:

DI-1100 ADC Binary Coding													
A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	Counts	Voltage
0	1	1	1	1	1	1	1	1	1	1	1	2047	9.995
0	1	1	1	1	1	1	1	1	1	1	0	2046	9.990
. . .													
0	0	0	0	0	0	0	0	0	0	0	1	1	0.0048
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	-1	-0.0048
. . .													
1	0	0	0	0	0	0	0	0	0	0	1	-2047	-9.995
1	0	0	0	0	0	0	0	0	0	0	0	-2048	-10.0

Applied voltage as a function of ADC counts has the following relationship:

$$volts = 10 \times \frac{counts}{2048}$$

DI-1110 Binary Stream Output Format and ADC Coding

<table>
<tr><th colspan="11">DI-1110 Binary Data Stream Example
(assumes all functions and channels enabled in order in the scan list)</th></tr>
<tr><th>Scan list position
(measurement</th><th>Word Count</th><th>Byte Count</th><th>B7</th><th>B6</th><th>B5</th><th>B4</th><th>B3</th><th>B2</th><th>B1</th><th>B0</th></tr>
<tr><td rowspan="2">0
(Analog in 0)</td><td rowspan="2">1</td><td>1</td><td>A3</td><td>A2</td><td>A1</td><td>A0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>2</td><td>A11</td><td>A10</td><td>A9</td><td>A8</td><td>A7</td><td>A6</td><td>A5</td><td>A4</td></tr>
<tr><td rowspan="2">1
(Analog in 1)</td><td rowspan="2">2</td><td>3</td><td colspan="8" rowspan="14">Same as analog in 0</td></tr>
<tr><td>4</td></tr>
<tr><td rowspan="2">2
(Analog in 2)</td><td rowspan="2">3</td><td>5</td></tr>
<tr><td>6</td></tr>
<tr><td rowspan="2">3
(Analog in 3)</td><td rowspan="2">4</td><td>7</td></tr>
<tr><td>8</td></tr>
<tr><td rowspan="2">4
(Analog in 4)</td><td rowspan="2">5</td><td>9</td></tr>
<tr><td>10</td></tr>
<tr><td rowspan="2">5
(Analog in 5)</td><td rowspan="2">6</td><td>11</td></tr>
<tr><td>12</td></tr>
<tr><td rowspan="2">6
(Analog in 6)</td><td rowspan="2">7</td><td>13</td></tr>
<tr><td>14</td></tr>
<tr><td rowspan="2">7
(Analog in 7)</td><td rowspan="2">8</td><td>15</td></tr>
<tr><td>16</td></tr>
<tr><td rowspan="2">8
(Digital in)</td><td rowspan="2">9</td><td>17</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>$\overline{D1}$</td><td>$\overline{D0}$</td></tr>
<tr><td>18</td><td>0</td><td>D6</td><td>D5</td><td>D4</td><td>D3</td><td>D2</td><td>D1</td><td>D0</td></tr>
<tr><td rowspan="2">9
(Rate in)</td><td rowspan="2">10</td><td>19</td><td>R7</td><td>R6</td><td>R5</td><td>R4</td><td>R3</td><td>R2</td><td>R1</td><td>R0</td></tr>
<tr><td>20</td><td>R15</td><td>R14</td><td>R13</td><td>R12</td><td>R11</td><td>R10</td><td>R9</td><td>R8</td></tr>
<tr><td rowspan="2">10
(Counter in)</td><td rowspan="2">11</td><td>21</td><td>C7</td><td>C6</td><td>C5</td><td>C4</td><td>C3</td><td>C2</td><td>C1</td><td>C0</td></tr>
<tr><td>22</td><td>C15</td><td>C14</td><td>C13</td><td>C12</td><td>C11</td><td>C10</td><td>C9</td><td>C8</td></tr>
</table>

The DI-1110 transmits a 12-bit binary number for every analog channel conversion in the form of a signed, 12-bit Two's complement value:

DI-1110 ADC Binary Coding													
D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Counts	Voltage
0	1	1	1	1	1	1	1	1	1	1	1	2047	9.995
0	1	1	1	1	1	1	1	1	1	1	0	2046	9.990
. . .													
0	0	0	0	0	0	0	0	0	0	0	1	1	0.0048
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	-1	-0.0048
. . .													
1	0	0	0	0	0	0	0	0	0	0	1	-2047	-9.995
1	0	0	0	0	0	0	0	0	0	0	0	-2048	-10.0

Applied voltage as a function of ADC counts has the following relationship:

$$volts = 10 \times \frac{counts}{2048}$$

DI-1120 Binary Stream Output Format and ADC Coding

DI-1120 Binary Data Stream Example (assumes all functions and channels enabled in order in the scan list)										
Scan list position (measurement	Word Count	Byte Count	B7	B6	B5	B4	B3	B2	B1	B0
0 (Analog in 0)	1	1	A5	A4	A3	A2	A1	A0	0	0
		2	A13	A12	A11	A10	A9	A8	A7	A6
1 (Analog in 1)	2	3	Same as analog in 0							
		4								
2 (Analog in 2)	3	5								
		6								
3 (Analog in 3)	4	7								
		8								
8 (Digital in)	9	17	0	0	0	0	0	0	$\overline{D1}$	$\overline{D0}$
		18	0	D6	D5	D4	D3	D2	D1	D0
9 (Rate in)	10	19	R7	R6	R5	R4	R3	R2	R1	R0
		20	R15	R14	R13	R12	R11	R10	R9	R8
10 (Counter in)	11	21	C7	C6	C5	C4	C3	C2	C1	C0
		22	C15	C14	C13	C12	C11	C10	C9	C8

The DI-1120 transmits a 14-bit binary number for every analog channel conversion in the form of a signed, 12-bit Two's complement value:

DI-1120 ADC Binary Coding															
D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Counts	Voltage*
0	1	1	1	1	1	1	1	1	1	1	1	1	1	8191	9.9988
0	1	1	1	1	1	1	1	1	1	1	1	1	0	8190	9.9976
. . .															
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0.0012
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1	-0.0012
. . .															

DI-1120 ADC Binary Coding															
D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Counts	Voltage*
1	0	0	0	0	0	0	0	0	0	0	0	0	1	-8191	-9.9988
1	0	0	0	0	0	0	0	0	0	0	0	0	0	-8192	-10.0

* Assuming the DI-1120 is programmed for the ±10 V full scale range.

Applied voltage as a function of ADC counts and measurement range has the following relationship for the DI-1120:

$$volts = full\ scale\ range \times \frac{counts}{8192}$$

DI-2008, -2108, -2108P, -4108, -4208, -4730 Binary Stream Output Format and ADC Coding

DI-2008, -2108, -2108P, -4108, -4208, -4730 Binary Data Stream Example (assumes all functions and channels enabled in order in the scan list)										
Scan list position (measurement	Word Count	Byte Count	B7	B6	B5	B4	B3	B2	B1	B0
0 (Analog in 0)	1	1	A7	A6	A5	A4	A3	A2	A1	A0
		2	A15	A14	A13	A12	A11	A10	A9	A8
1 (Analog in 1)	2	3	Same as analog in 0							
		4								
2 (Analog in 2)	3	5								
		6								
3 (Analog in 3)	4	7								
		8								
4 (Analog in 4)	5	9								
		10								
5 (Analog in 5)	6	11								
		12								
6 (Analog in 6)	7	13								
		14								
7 (Analog in 7)	8	15								
		16								
8 (Digital in)	9	17	0	0	0	0	0	0	$\overline{D1}$	$\overline{D0}$
		18	0	D6	D5	D4	D3	D2	D1	D0
9 (Rate in)	10	19	R7	R6	R5	R4	R3	R2	R1	R0
		20	R15	R14	R13	R12	R11	R10	R9	R8
10 (Counter in)	11	21	C7	C6	C5	C4	C3	C2	C1	C0
		22	C15	C14	C13	C12	C11	C10	C9	C8

All instruments transmit a 16-bit binary number for every analog channel conversion in the form of a signed, 16-bit Two's complement value:

DI-2008, -2108, -2108P, -4108, -4208, -4730 ADC Binary Coding																	
D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Counts	Voltage*
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	32767	9.9997
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	32766	9.9994
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0.0003
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1	-0.0003
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-32767	-9.9997
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-32768	-10.0

* Assuming the instrument is programmed for the ±10 V full scale range.

Applied voltage as a function of ADC counts has the following relationship for all instruments in this category, and for the DI-2108P when in a bipolar measurement range:

$$volts = full\ scale\ range \times \frac{counts}{32768}$$

When the DI-2108P is in a unipolar range the following applies:

$$volts = full\ scale\ range \times \frac{counts + 32768}{65536}$$

For example, the Binary Coding for 0-10V Range is as follows:

DI-2108-P ADC Binary Coding for 0-10V Range																	
D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Counts	Voltage*
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	32767	9.99992
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	32766	9.99985
...																	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	5.00015
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5.00000
...																	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-32767	0.00015
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-32768	0.00000

$$volts = full\ scale\ range \times \frac{counts}{65536}$$

DI-2008 Thermocouple Temperature Coding

Channels configured as a thermocouple input (DI-2008 only) borrow two ADC counts from the measurement range to indicate error conditions.

ADC counts = +32767 indicates an unrecoverable CJC error. The DI-2008's processor cannot communicate with the CJC temperature sensor, or the reading is outside the CJC sensor's measurement range.

ADC counts = -32768 indicates a TC burnout (open) condition.

An applied temperature is derived from ADC counts (A) according to the following equation, where m and b are determined by TC type:

$$°C = mA + b$$

TC type	J	K	T	B	R/S	E	N
m	0.021515	0.023987	0.009155	0.023956	0.02774	0.018311	0.022888
b	495	586	100	1035	859	400	550

DI-4718B Binary Stream Output Format and ADC Coding

<table>
<tr><th colspan="11">DI-4718B Binary Data Stream Example
(assumes all functions and channels enabled in order in the scan list)</th></tr>
<tr><th>Scan list position
(measurement</th><th>Word
Count</th><th>Byte
Count</th><th>B7</th><th>B6</th><th>B5</th><th>B4</th><th>B3</th><th>B2</th><th>B1</th><th>B0</th></tr>
<tr><td rowspan="2">0
(Analog in 0)</td><td rowspan="2">1</td><td>1</td><td>A7</td><td>A6</td><td>A5</td><td>A4</td><td>A3</td><td>A2</td><td>A1</td><td>A0</td></tr>
<tr><td>2</td><td>A15</td><td>A14</td><td>A13</td><td>A12</td><td>A11</td><td>A10</td><td>A9</td><td>A8</td></tr>
<tr><td rowspan="2">1
(Analog in 1)</td><td rowspan="2">2</td><td>3</td><td colspan="8" rowspan="3">Same as analog in 0</td></tr>
<tr><td>4</td></tr>
<tr><td>2</td><td>3</td><td>5</td></tr>
</table>

(Analog in 2)		6								
3 (Analog in 3)	4	7								
		8								
4 (Analog in 4)	5	9								
		10								
5 (Analog in 5)	6	11								
		12								
6 (Analog in 6)	7	13								
		14								
7 (Analog in 7)	8	15								
		16								
8 (Digital in)	9	17	0	0	0	0	0	0	$\overline{\mathbf{D1}}$	$\overline{D0}$
		18	0	0	0	0	0	0	D1	D0

Transmits a 16-bit binary number for every analog channel conversion in the form of a signed, 16-bit Two's complement value:

DI-4718B ADC Binary Coding																	
D_{15}	D_{14}	D_{13}	D_{12}	D_{11}	D_{10}	D_9	D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Counts	Voltage
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	32767	4.9997
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	32766	4.9994
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0.0003
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1	-0.0003
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-32767	-4.9997
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-32768	-5.0

Applied voltage as a function of ADC counts has the following relationship for the DI-4718B:

$$volts = 10 \times \frac{counts}{32768}$$

ASCII Stream Output Format

Products that support an ASCII output format for analog channels, when enabled with the *encode* commands, generate a comma-separated value result with each line of the output terminated with either a carriage return or line feed character as defined by the *eol* command. Channel output order is defined by the scan list, and values are presented as floating point numbers scaled in volts. For example, if the following two assignments are made in the scan list using the *slist* command and *eol* command value = 1:

Position 0 = analog channel 5

Position 1 = analog channel 4

this CSV output results after issuing the *start* command:

```
(analog channel 5 value),(analog channel 4 value)<cr>
(analog channel 5 value),(analog channel 4 value)<cr>
(analog channel 5 value),(analog channel 4 value)<cr>
                                 .
                                 .
                                 .
(analog channel 5 value),(analog channel 4 value)<cr>
```

Document Revisions

Revision	Date	Description
		Pre-release revision, Aug 23, 2018
		Pre-release revision, Aug 20, 2018
		Pre-release revision, Aug 7, 2018
		Pre-release revision, Aug 3, 2018
		Pre-release revision, Sep 18, 2018
		Pre-release revision, Oct 3, 2018
		Pre-release revision, Oct 15, 2018
		Pre-release revision, Dec 10, 2018
		Pre-release revision, Dec 12, 2018
		Pre-release revision, Feb 14, 2019
		Pre-release revision, May 3, 2019
		Pre-release revision, August 1, 2019
		Pre-release revision, April 17, 2020
		Pre-release revision, May 28, 2020
		Pre-release revision, June 1, 2020
		Pre-release revision, April 7, 2021