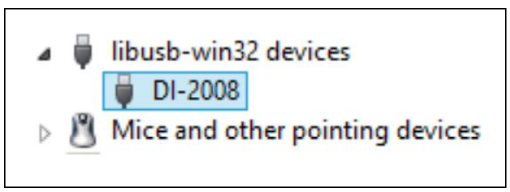
DI-2008 USB Data Acquisition (DAQ) System Communication Protocol

DATAQ Instruments

Although DATAQ Instruments provides ready-to-run WinDaq software with its DI-2008 Data Acquisition instrument, programmers will want the flexibility to integrate the DI-2008 in the context of their own application. To do so they want complete control over DI-2008 hardware, which can be accomplished by using the device at the protocol level. This white paper describes how protocol-level programming of the DI-2008 is implemented across the Windows and Linux operating systems. We'll define the DI-2008's command set and scan list architecture and finish with a description of the DI-2008's binary response format. Please note that a .Net class either has been or will soon be released for the DI-2008, which allows programming the instrument at a much higher level than the protocol-level under Windows.

Device Access

The DI-2008 can be 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 a DI-2008 is connected to a PC in a Windows implementation the instrument appears in the Device Manager as a "DI-2008" under the "libusb-win32 devices" tree:



The following constants apply to the DI-2008 and must be correctly referenced from your program via Libusb:

- PID = 2008_{16}
- VID = 0683_{16}

DI-2008 Command Set Overview

The DI-2008 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 ($0D_{16}$) 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 (20_{16}). All commands echo if the instrument is not scanning. Command arguments and responses are always in decimal.

DI-2008 Command Set	
ASCII Command	**Action**
Basic communication	
`info arg0`	Echoes the command and argument with additional information as defined by the argument
`ps arg0`	Defines communication packet size
Multi-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)
Scanning	
`start arg0`	Start single unit scanning (never echoes). See command *syncstart* for multi-unit, synchronized acquisition
`stop`	Stop scanning (always echoes)
`slist arg0 arg1`	Defines scan list configuration
`srate arg0`	Defines scan rate
Report Modes	
`filter arg0 arg1`	Defines the report mode (average, min, max, last point) for the specified channel
`dec arg0`	Defines the decimation factor applied to the specified report mode
CJC Commands	
`cjcdelta`	Adjusts and reports CJC sensor offset
Rate measurement	
`ffl arg0`	Sets the moving average filter length of the rate measurement digital input channel
LED color	
`led arg0`	Sets the LED to a specified color
Digital I/O	
`dout arg0`	Outputs the specified data to the digital output port
`endo arg0`	Enables defined ports as inputs or outputs
`din`	Returns the value of each digital port that is configured as an input
Reset	
`reset arg0`	Performs various reset operations

Command Echo Protocol

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 of DI-2008 commands, any descriptions and examples related to a command echo assume that the DI-2008 is not actively scanning.

Basic Communication Commands

The DI-2008 command set 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 DI-2008 communication attempt. If these commands don't work with a functioning DI-2008 then a problem exists in the communication chain and further programming efforts will be futile until they are 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 ($0D_{16}$). For example:

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

DI-2008 Basic Communication Commands	
ASCII Command	**Action**
`info 0`	Returns "DATAQ"
`info 1`	Returns device PID: "2008"
`info 2`	Returns firmware revision, 2 hex bytes (e.g. 65_{16} = 101_{10} for firmware revision 1.01)
`info 3 to info 5`	Proprietary internal use for initial system verification
`info 6`	Returns the DI-2008's serial number (left-most 8 digits only; right-most two digital are for internal use)
`info 7 to info 8`	Proprietary internal use for initial system verification
`info 9`	Returns the sample rate divisor value (see the *srate* command for details)*
`ps 0`	Make packet size 16 bytes
`ps 1`	Make packet size 32 bytes
`ps 2`	Make packet size 64 bytes
`ps 3`	Make packet size 128 bytes

* The value returned for this command is a function of the number of enabled analog channels at the time it was invoked, 8000 for a single analog channel and 800 for two or more. The value should be retrieved and used only after channel setup completion.

The packet size command defines the number of bytes the DI-2008 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.

Command: `ps 1` `'make packet size 32 bytes`

Response: `ps 1` `'command echo`

Multi-unit Synchronization Commands

Model DI-2008 supports synchronized data acquisition across multiple units of the same model. The commands in this group manage various aspects of the synchronization process.

syncget, syncset, syncstart Commands

These commands in combination manage synchronized sampling across multiple DI-2008 devices. Each supports a 16-bit, unsigned number (in string format and in the range of "0" to "65535") as either 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.

DI-2008 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.)
`syncget 4`	Returns two, 32-bit integer sync-quality evaluation parameters.
`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.

Typical Synchronization Procedure Using 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
```

syncget 4 Command

Command *syncget 4* can be issued to gain insights to synchronization quality. The command returns two, 32-bit integers:

Command:	`syncget 4`	`'retrieve sync quality`
Response:	`resp1 resp2`	`'two quality measures as 32-bit integers`

In the above example, two quality measures are returned separated by a space character:

`resp1`	*resp1* applies to USB port performance, the higher the number the worse the performance. The best possible measure for *resp1* equals 1. A response greater than 500 means the USB interface is not suitable for synchronization.
`resp2`	*resp2* applies to the tolerance of the sync operation timing, the higher the number the worse the sync timing. A value of 125 or lower is considered very good. A value higher than 10000 indicates very poor sync timing.

Scanning Commands

start Command

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

DI-2008 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 the *srate* command.

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

stop Command

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 prevented by ensuring that the DI-2008 buffer is flushed continuously and frequently. Strategic placement of the LibUSB read command in a high-priority routine will greatly minimize the chance of a buffer overflow.

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

slist Command

The DI-2008 employs 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. The DI-2008's scan list supports four types of inputs: Up to eight analog channels; one counter channel; one rate channel; general-purpose discrete inputs. These type definitions may be placed in the DI-2008's scan list in any order that satisfies the requirements of the application. The DI-2008's scan list is a maximum of 11 elements long, which allows a hardware capacity measurement that's configured to sample all eight analog channels, both the counter and rate channels, and general-purpose digital input ports during one complete scan. Note that 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.

During general-purpose use each entry in the scan list is represented by a 16-bit number, which is defined in detail in the DI-2008 Scan List Word Definitions table below. 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 (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 a total of eleven possible positions. *config* is the 16-bit configuration parameter as defined in table *DI-2008 Scan List Word Definitions*. For example, the command *slist 5 10* configures the sixth position of the scan list to specify data from the counter. Assuming that we wish to sample analog channels 2, 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. Further any scan list position (except position 0) may be modified without affecting the contents of the rest of the list.

<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 Measurement Table)</td><td rowspan="8">Range (see Measurement Table)</td><td colspan="3" rowspan="8">± Full Scale value or TC type (see 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">Range (see Rate 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>

* To be consistent with general programming standards, analog channel numbers begin with 0 instead of 1 as indicated on the product label.

Measurement Table

Scan List Bit Position			Range = 0	Range =1	Range = Don't care
10	9	8	Mode = 0	Mode = 0	Mode = 1
0	0	0	±500 mV	±50 V	B thermocouple
0	0	1	±250 mV	±25 V	E thermocouple
0	1	0	±100 mV	±10 V	J thermocouple
0	1	1	±50 mV	±5 V	K thermocouple
1	0	0	±25 mV	±2.5 V	N thermocouple
1	0	1	±10 mV	±1 V	R thermocouple
1	1	0	n/a	n/a	S thermocouple
1	1	1	n/a	n/a	T thermocouple

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.

Rate Table (for DI2 connections)				
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

*

Command: `slist 0 0` `'enabled analog channel 0`
Response: `slist 0 0` `'command echo`
Command: `slist 1 4` `'enabled analog channel 4`
Response: `slist 1 4` `'command echo`
Command: `slist 2 1033` `'rate channel enabled, 5 kHz range`
Response: `slist 2 1033` `'command echo`

srate Scan rate Command

Command *srate* defines a sample rate divisor used to determine scan rate throughput, or the rate at which the DI-2008 scans through the enabled items in the scan list that you defined with the *slist* command. Note that the sample rate per channel is the throughput rate divided by the number of enabled analog channels. *srate* is specified as an integer argument, along with a second integer argument, *dec* (see Filter Mode Commands.) Integer ranges for both variable are:

$$4 \le srate \le 2232$$

$$1 \le dec \le 32767$$

The formula to calculate sample throughput rate differs by the number of enabled channels. For a single enabled analog channel:

$$\text{Sample rate throughput (Hz)} = 8{,}000 \div (srate \times dec)$$

resulting in a sample throughput range of 2000 Hz at its fastest, and 0.000109 Hz, or 1 sample every 9141.99 seconds.

The formula changes when two or more analog channels are enabled:

$$\text{Sample rate throughput (Hz)} = 800 \div (srate \times dec)$$

resulting in a sample throughput range of 200 Hz at its fastest, and 0.000011 Hz, or 1 sample every 91419.93 seconds.

Only the number of enabled analog channels determines which of the above sample throughput equations to deploy. Digital channels, including rate, count, and discrete inputs have no effect.

Note that the dividend used in the above equation can change between data acquisition products, and as a function of enabled analog channels for the DI-2008. The command `info 9` can be used to determine the value for each case.

Finally, bear in mind that while the DI-2008 is capable of impressively slow sampling rates, their practicality is suspect. Since the instrument's smallest packet size is 16 bytes, one could wait for over a week before the instrument actually delivers a data value.

Report Mode Commands

The DI-2008 supports a range of report modes that are selectable per channel. The instrument can acquire and report the last point that was acquired, the maximum or the minimum of a range of values, or the averaged result. The report mode may be defined on a per channel basis using the *filter* command. The *filter* command accepts two arguments of the form:

```
filter arg0 arg1
```

Where: 0 ≤ *arg0* ≤ 7 and is equal to a specific analog channel number. arg0 can also equal "*" as a shortcut way to reference all channels.

0 ≤ *arg1* ≤ 3:

arg1	
Value	**Acquisition Mode**
0	Last Point
1	Average
2	Maximum
3	Minimum

A decimation factor (*dec*) may be applied to define the number of samples used per channel by each acquisition mode (except Last Point.) For example, if *dec* has a value of 100 and the *filter* command defines an acquisition mode for a channel as Maximum, 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 is specified.

When the *filter* command defines the average, maximum, or minimum Acquisition Modes, the *dec* command sets the number of samples used to calculate the average.

```
dec arg0
```

Where: 1 ≤ *arg0* ≤ 32767 sets the number of values used by the report mode defined by the *filter* command.

Sample report and decimation commands and responses:

Command: `filter * 2     'Set all channels to maximum acquisition mode`
Response: `filter * 2     'Set all channels to maximum acquisition mode`

```
Command:  dec 128      'set the decimation factor to 128
Response: dec 128      'the current decimation factor is 128
```

cjcdelta Command

The *cjcdelta* command 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 DI-2008), the *cjcdelta* command exists to ensure accurate junction temperature readings. Adjustments using *cjcdelta* should be applied only on a channel with a connected 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 (0x20), and terminated with a carriage return character (0x0D). General forms of the command follow:

`cjcdelta(0x20)-1(0x0D)`

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(0x20)i(0x20)j(0x0D)`

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

`cjcdelta(0x20)-2(0x0D)`

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

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

Rate Measurement Commands

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 *ffl arg0* command, 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

The DI-2008 has a panel-mounted, multi-color LED that is available 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

The protocol supports three commands for digital I/O. The DI-2008 provides seven digital ports. Each port can be programmed as either an input or an output. 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: $0 \le arg0 \le 127_{10}$ and maps input/switch configuration to each of seven digital ports. 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
```

dout command

```
dout arg0
```

Where: $0 \le arg0 \le 127_{10}$ ($0 \le arg0 \le 7F_{16}$) and defines the bit state of the 7-bit output port.

```
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. A port configured as a switch returns the state of the switch. One configured as a digital input returns the applied state.

Reset Command

There is only one reset command used to force accumulated counts to zero:

```
reset arg0
```

Where: arg0 = 1 to reset the DI-2008 counter

Command: `reset 1   'reset the counter`

Response: `reset 1   'command echo`

Binary Stream Output Format

The DI-2008's data output format is a binary stream of one 16-bit word per enabled measurement. In the table below A_x values denote analog channel ADC values, and D_x, R_x and C_x are digital, rate, and counter value inputs respectively.

Binary Data Stream Example (all functions and channels enabled in order)										
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

Analog Channel Binary Coding

The DI-2008 transmits a 16-bit binary number for every analog channel conversion in the form of a signed, 16-bit Two's complement value:

DI-2008 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	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 DI-2008 is programmed for the ±10 V full scale range.

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

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

For example, if full scale range is ±25 mV and counts is 25879:

$$19.74\ mV = .025 \times \frac{25879}{32768}$$

If full scale range is ±5 V and counts is 1502:

$$0.2292\ V = 5 \times \frac{1502}{32768}$$

Channels configured as a thermocouple (TC) input 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	m	b
J	0.021515	495
K	0.023987	586
T	0.009155	100
B	0.023956	1035
R/S	0.02774	859
E	0.018311	400
N	0.022888	550

Rate and Count Channel Binary Coding

If enabled the DI-2008 delivers 16-bit count and rate data. Meaningful information is extracted from the DI-2008 for these measurements as follows:

$$counter\ value = counts + 32768$$

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

Where: *counts* is the 16-bit value provided by the DI-2008 for the indicated measurement.
range is the selected rate measurement range in Hz (see Rate Range Table.)

Control

Revision	Date	Description
1.0	March 3, 2017	Original release level
1.01	June 6, 2017	Modification to LibUSB tree graphic to replace DI-2108 reference with DI-2008
1.02	June 28, 2017	Added explanation of *stop 01* response while scanning to the *stop* command section.