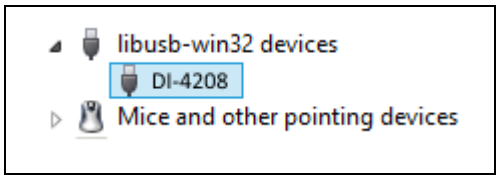
DI-4108 and DI-4208 USB Data Acquisition (DAQ) System Communication Protocol

DATAQ Instruments

Although DATAQ Instruments provides ready-to-run WinDaq software with its Data Acquisition products, programmers will want the flexibility to integrate either the DI-4108 or DI-4208 (hence forth DI-4x08) in the context of their own application. To do so they want complete control over DI-4x08 hardware, which can be accomplished by using the device at the protocol level. This white paper describes how protocol-level programming of the DI-4x08 is implemented. We'll define the DI-4x08's command set and scan list architecture and finish with a description of the DI-4x08's binary response format. References to model DI-4x08 apply equally to both the DI-4108 and DI-4208 unless otherwise noted.

Device Access

The DI-4x08 can be accessed using the Libusb open source library using end point 1 to control data transfers to and from the instrument via its USB interface in both Windows and non-Windows implementations. When a DI-4x08 is connected to a PC in a Windows implementation the instrument appears in the Device Manager as a "DI-4108" or "DI-4208" under the "libusb-win32 devices" tree:



The following constants apply to the DI-4x08 and must be correctly referenced from your program via Libusb:

- PID = 4108_{16} or 4208_{16}
- VID = 0683_{16}

DI-4x08 Command Set Overview

The DI-4x08 employs a simple 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-4x08 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 scanning (never echoes)
`stop`	Stop scanning (always echoes)
`slist arg0 arg1`	Defines scan list configuration
`srate arg0`	Defines scan rate
Filter	
`filter arg0 arg1`	Defines the operating mode (filter, min, max, last point) for the specified channel
`dec arg0`	Defines the filter decimation factor
Rate measurement	
`ffl arg0`	Sets the moving average filter length of the rate measurement digital input channel
LED color	
`led arg0`	Sets the ACTIVE 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-4x08 commands, any descriptions and

examples related to a command echo assume that the DI-4x08 is not actively scanning.

Basic Communication Commands

The DI-4x08 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-4x08 communication attempt. If these commands don't work with a functioning DI-4x08 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 ($0D_{16}$). For example:

```
Command:    info 1        'what model is connected?
Response:   info 1 4x08   'command echo, plus connected model no.
```

DI-4x08 Basic Communication Commands	
ASCII Command	**Action**
`info 0`	Returns "DATAQ"
`info 1`	Returns device name: "4x08"
`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-4x08'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 for the DI-4x08 (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
`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 packet size command defines the number of bytes the DI-4x08 sends with each transmission burst. The larger the packet size the more bytes transmitted per burst. Since in most cases 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. A partial packet may transmit when data acquisition is stopped via the *stop* command. In that case, packet size will vary depending upon defined packet size, the timing of the *stop* command, and the number of channels enabled in the scan list, since a packet terminates only after completing a complete scan of all enabled channels.

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

Response: `ps 1 'command echo`

Note that the packet size command affects only the size of the packet the instrument transmits. Actual packet size may be changed by the USB driver outside of the control of the instrument.

Multi-unit Synchronization Commands

Model DI-4x08 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-4x08 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-4x08 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 = syncget₁ 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 = syncget₁ 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

The DI-4x08 *start* commands support an argument that defines the instrument's scanning mode, and initiates scanning accordingly. Since a *start* command immediately initiates scanning, the command is never echoed. It's important to clarify that issuing the start command commences a continuous stream of returned data, not just one packet of data. Your program needs to be able to accommodate the volume and rate of returned binary data without gaps to ensure that data is acquired reliably and contiguously.

DI-4x08 Start Command Modes	
ASCII Command	**Action**
`start 0`	Normal scanning: The instrument begins scanning the channels enabled in its scan list through the *slist* command at a rate defined by the *srate* command.
`start 1`	Reserved for future use.

```
Command:        start 0    'begin normal 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
```

slist Command

The DI-4x08 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-4x08'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-4x08's scan list in any order that satisfies the requirements of the application. The DI-4x08'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-4x08 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-4x08 Scan List Word Definitions*. For example, the command *slist 6 10* configures the sixth position of the scan list to specify data from the counter. Assuming we wish to sample analog channels 0 and 2 at a range of ±10 and ±100 V respectively (model DI-4208 only), along with the rate channel on its 5 kHz range, the following sequence would work:

slist 0 768

slist 1 2

slist 2 1033

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 2. Further any scan list position (except position 0) may be modified without affecting the contents of the rest of the list.

DI-4x08 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	Unused bits =0				Analog Range (see Analog Measurement Range table)				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	Unused bits =0												1	0	0	0
Rate (DI2)	0	0	0	0	Rate 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

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

The protocol supports access to the analog programmable gain feature of the DI-4x08 via four scan list bits per analog channel that are reserved for that purpose:

DI-4108 Analog Measurement Range Table				
Bit Position				Range
11	10	9	8	(V Full Scale)
0	0	0	0	±10
0	0	0	1	±5
0	0	1	0	±2
0	0	1	1	±1
0	1	0	0	±0.5
0	1	0	1	±0.2

DI-4208 Analog Measurement Range Table				
Bit Position				Range
11	10	9	8	(V Full Scale)
0	0	0	0	±100
0	0	0	1	±50
0	0	1	0	±20
0	0	1	1	±10
0	1	0	0	±5
0	1	0	1	±2

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 DI2 connections)				
Bit Position				Range*
11	10	9	8	(Hz)
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

* 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.

(examples apply to model DI-4208 only)

```
Command:  slist 0 0      'enabled analog channel 0, ±100 V range
Response: slist 0 0      'command echo
Command:  slist 1 515    'enabled analog channel 3, ±20 V range
Response: slist 1 515    'command echo
Command:  slist 2 265    'rate channel enabled, 50 kHz range
Response: slist 2 265    'command echo
```

srate Scan rate Command

Command *srate* defines a sample rate divisor used to determine scan rate, or the rate at which the DI-4x08 scans through items in the scan list that you defined with the *slist* command. *srate* is specified with an integer argument in the range of 375 to 65,535 inclusive, and the resulting scan speed per scan list element is defined by the following equation:

Sample rate per scan list element (Hz) = 60,000,000 ÷ (*srate × dec)*

This approach results in a per channel sample rate ranging from 915.5413 to 160,000 Hz with a decimation factor of 1 (see the *dec* command), and from 1.788 to 312.5 Hz with a decimation factor of 512. The host program may achieve a further reduction in sample rate below 1.788 Hz by using selective sampling methods whereby every nth point is selected as the converted value. For example, a sample rate per scan list element of 0.5 Hz is achieved by applying integer values of 2400 and 500 to *srate* and *dec* respectively, and further selecting every 100th value from the reported data stream. Every 1000th reading is effectively 0.05 Hz. Averaging every n values on each channel is more difficult but recommended since it reduces noise by a factor of the square root of n.

At a given sample rate per element value equal to 20 kHz or lower (*srate* ≥ 3000) analog *or* digital channels in any combination of digital, rate, or count inputs may be added to the scan list without affecting sample rate per element, resulting in a total throughput value of 220 kHz when all scan list positions are active.

When the instrument's filtering feature is used (see the *filter* command) good practice dictates using

decimation to slow sample rate as opposed to applying higher *srate* values, since taking the latter approach reduces the number of values available to the filter.

Note that the numerator (60,000,000) used in the above equation can change between data acquisition products. The command `info 9` can be used to determine the value for each product.

Filter Commands

The DI-4x08 supports a range of acquisition 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 filtered result. The acquisition mode and 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	CIC filter
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 or Minimum is specified.

When the *filter* command defines a CIC filter as the Acquisition Mode, the *dec* command sets the number of samples used to calculate the CIC filter. When arg0 = 1 four stages of 2-sample moving window averages are applied and all filtered values are returned. When arg0 > 1 every arg0 sample is returned. For example if

arg0 is two or four every other, or every fourth sample is returned respectively.

```
dec arg0
```

Where: 1 ≤ *arg0* ≤ 512 sets the number of values used by the Acquisition Mode defined by the *filter* command.

The *filter* command supports a wildcard syntax that uses an asterisk character ("*") to in place of *arg0* to command that all channels be set to the value defined by *arg1*. Sample filter 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
```

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-4x08 has a panel-mounted, multi-color LED labeled as *Active* 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-4x08 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 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-4x08 counter

```
Command:    reset 1   'reset the counter
Response:   reset 1   'command echo
```

Binary Stream Output Format

The DI-4x08'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. Analog values are represented as 16-bit quantities.

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-4x08 transmits a 16-bit binary number for every analog channel conversion in the form of a signed, two's complement value:

DI-4x08 ADC Binary Coding																		
A_{15}	A_{14}	A_{13}	A_{12}	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	1	1	1	1	32767	9.9996
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	0.000031
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	-0.000031
. . .																	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	-32767	-9.9996
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-32768	-10.0

* Assuming both the DI-4108 and the -4208 are programmed for their ±10 V full scale range.

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

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

For example, if the ±50 V and ±0.2 V ranges are selected for the DI-4208 and -4108 respectively and each returned ADC count values of 23978, the applied voltage was:

For the DI-4208:

$$36.5875 = \mathbf{50} \times \frac{23978}{32768}$$

For the DI-4108:

$$0.14635 = \mathbf{0.2} \times \frac{23978}{32768}$$

Rate and Count Channel Binary Coding

If enabled the DI-4x08 delivers 16-bit count and rate data. Meaningful information is extracted from the DI-4x08 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-4x08 for the indicated measurement
range is the selected rate measurement range in Hz (see Rate Range Table)

Control

Revision	Date	Description
1.0	Jul 28, 2017	Original release level
1.1	December 1, 2017	Fixed errors in the *DI-4x08 ADC Binary Coding* table