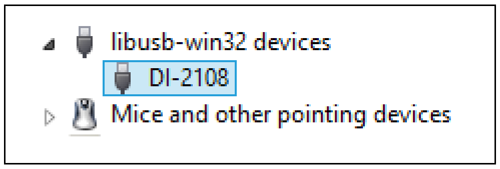
DI-1100 USB Data Acquisition (DAQ) System Communication Protocol

DATAQ Instruments

Although DATAQ Instruments provides ready-to-run WinDaq software with its DI-1100 Data Acquisition Starter Kits, programmers will want the flexibility to integrate the DI-1100 in the context of their own application. To do so they want complete control over DI-1100 hardware, which can be accomplished by using the device at the protocol level. This white paper describes how protocol-level programming of the DI-1100 is implemented across the Windows and Linux operating systems. We'll define the DI-1100's command set and scan list architecture and finish with a description of the DI-1100's binary response format. Windows programmers should consider the using the .Net class and a suitable .Net programming language as a more efficient method for programming the DI-1100 under that operating system.

Device Access

The DI-1100 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-1100 is connected to a PC in a Windows implementation the instrument appears in the Device Manager as a "DI-1100" under the "libusb-win32 devices" tree:



The following constants apply to the DI-1100 and must be correctly referenced from your program via Libusb:

- PID = 1100_{16}
- VID = 0683_{16}

DI-1100 Command Set Overview

The DI-1100 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 as always in decimal.

DI-1100 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
Scanning	
`start arg0`	Start scanning (never echoes)
`stop`	Stop scanning (always echoes)
`slist arg0 arg1`	Defines scan list configuration
`srate arg0`	Defines scan rate
LED color	
`led arg0`	Sets the LED to a specified color

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-1100 commands, any descriptions and examples related to a command echo assume that the DI-1100 is not actively scanning.

Basic Communication Commands

The DI-1100 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-1100 communication attempt. If these commands don't work with a functioning DI-1100 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 1100   'command echo, plus connected model no.
```

DI-1100 Basic Communication Commands	
ASCII Command	**Action**
`info 0`	Returns "DATAQ"
`info 1`	Returns device name: "1100"
`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-1100'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 of 60,000,000 for the DI-1100 (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-1100 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
```

Scanning Commands

start Command

The DI-1100 *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.

DI-1100 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-1100 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-1100's scan list supports up to four analog channels. Analog channels may be placed in the DI-1100's scan list in any order that satisfies the requirements of the application. The DI-1100's scan list is a maximum of four elements long, which allows a hardware capacity measurement to sample all four analog channels. The state of the DI-1100's two digital ports is contained in the returned by the analog channel that appears first in the scan list. Note that any analog 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-1100 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-1100 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 the following scan list configuration would work:

slist 0 2

slist 1 4

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

DI-1100 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
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.

```
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
```

srate Scan rate Command

Command *srate* defines a sample rate divisor used to determine scan rate, or the rate at which the DI-1100 scans through the items in the scan list that you defined with the *slist* command. *srate* is specified with an integer (int) argument (the divisor) within the range of 750 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*

This approach results in a per channel sample rate ranging from 915.5413 to 80,000 Hz. The host program may achieve a further reduction in sample rate below 915.5413 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 10 Hz is achieved by applying an integer value of 60,000 to the *srate* command, and further selecting every 100th value from the reported data stream. Every 1000th reading is effectively 1 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.

Note that the divisor (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.

LED Color Command

The DI-1100 has a panel-mounted, multi-color LED (labeled *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`

Binary Stream Output Format

The DI-1100'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 digital inputs.

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	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

Analog Channel Binary Coding

The DI-1100 transmits a 12-bit binary number for every analog channel conversion in the form of a signed, 16-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	0.004
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	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:

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

Control

Revision	Date	Description
1.0	Mar 20, 2017	Original release level
1.1	Apr 5, 2017	Corrected errors in the ADC Binary Coding Table and formula