Defining the DI-145 Protocol

(Firmware revision 1.07 and up)

DATAQ Instruments

Although DATAQ Instruments provides ready-to-run WinDaq software with its DI-145 Data Acquisition Starter Kits, programmers will want the flexibility to integrate the DI-145 in the context of their own application. To do so they want complete control over DI-145 hardware, which can be accomplished by using the device at the protocol level. This white paper describes how protocol-level programming of the DI-145 is implemented across the Windows and Linux operating systems. First, we'll describe the virtual COM operation of the DI-145's interface and how communicating with the DI-145 is accomplished via a COM port hooked by the operating system. Then we'll define the DI-145's command set and scan list architecture and finish with a description of the DI-145's binary and ASCII response formats. **Note that all of the commands and their arguments described in this protocol are lower case unless otherwise stated.**

Virtual COM Port Operation

Installing the DI-145 driver package and connecting DI-145 hardware to the host computer's USB port results in a COM port being hooked by the operating system and assigned to the DI-145 device. Multiple DI-145 devices may be connected to the same PC without additional driver installations, which results in a unique COM port number assignment to each by the operating system. Hooking a COM port in this manner facilitates ease of programming from any operating system and programming language by simply writing commands to and reading responses from the port, but before any meaningful communication with a connected DI-145 can begin the controlling program must determine the COM port number assigned to the device. The method used for this varies as a function of the host operating system.

Virtual COM Driver (Windows)

DATAQ Instruments provides a minimum installation for Windows that you can download and use at no charge, even for OEM applications. This is a scaled-down version of the standard installation that omits WinDaq software and other utilities that are extraneous in a pure programming environment. The download provides a Microsoft-signed INF file that ensures trouble-free operation with Windows XP (32-bit only), and both 32- and 64-bit Windows Vista and Windows 7. The installation depends upon driver *usbser.sys*, a Windows component located in path *%SystemDrive%\Windows\System32\Drivers*.

COM Port Number Discovery (Windows)

Using the DI-145's vendor and product IDs, Windows' registry can be accessed programmatically to determine the COM port number that the operating system assigned to one or more connected DI-145s. The Vendor and Product ID combination for the DI-145 is: Vid_0683&Pid_1450. With this information and at least one connected DI-145, determining assigned COM port numbers is a two-step process:

1. The registry tree *HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\usbser\Enum\* will contain one Device Instance ID for each DI-145 connected to the PC. The Device Instance ID assumes the following typical string value: *USB\VID_0683&PID_1450\5&18B6E64F&0&8*. The first two sections of this string (*USB\VID_0683&PID_1450\*) are constant for all DI-145s. The second section (*5&18B6E64F&0&8*) will vary depending upon where in the USB port hierarchy the DI-145 is physically connected. Since more than one DI-145 cannot be connected to the same USB port this value will be unique for each concurrently connected DI-145. The entire string value is required for the second step.

2. Registry tree *HKEY_LOCAL_MACHINE\System\CurrentControlSet\Enum\USB\VID_0683&PID_1450\5&18B6E64F&0&8* (continuing with the above example) shows a variable called *FriendlyName* set to string value *DATAQ DI-145 (COMXX)*, where *XX* is the COM port number assigned to the specified DI-145. This string may be parsed to extract the port number assigned to the DI-145. The process may be repeated using other Device Instance IDs determined from step (1) for other connected DI-145 instruments.

COM port number assignments may also be determined manually from Windows' Device Manager in its Ports *(COM & LPT)* section. However, the assigned value will change depending upon the physical USB port connected to the DI-145, any other devices that may hook COM ports, and the apparently arbitrary whims of the Windows operating system.

Virtual COM Driver (Linux)

Linux has two different generic drivers, which are appropriate for a USB to COM port converter. The first is an Abstract Control Model driver designed for modem devices, and is simply named *acm*. The other one is a generic USB to serial driver named *usbserial*.

COM Port Number Discovery (Linux)

If support for the acm driver has been compiled in the kernel, Linux will automatically load it and a new terminal device will be created under */dev/ttyACMx*, where *x* is the COM port number. The path */dev/serial/by-id/usb-0683_1450-** presents links to */dev/ttyACM** device files of currently plugged in DI-145 devices.

The second driver, usbserial, must be loaded manually by using the modprobe command with the vendor ID and product ID values used by the DI-145:

modprobe usbserial vendor=0x0683 product=0x1450

Once the driver is loaded, a new terminal entry appears and should be named */dev/ttyUSBx*, where *x* is the COM port number.

DI-145 Command Set Overview

The DI-145 employs a simple ASCII character command set that allows complete control of the instrument. ASCII comes in handy when a terminal emulator such as HyperTerminal or PuTTY is used to experiment with the device outside of a programming language environment. All of the commands in the following table must be terminated with a carriage return character (x0D).

DI-145 Command Set	
ASCII Command*	**Action**
Command argument formats, where supported*	
`xhhhh`	Defines a 4- to 16-bit hexadecimal number h(hhh) as a command argument where appropriate. This argument format may be used ONLY while the DI-145 is in the ASCII mode (see the asc command.)
`ddddd`	Defines a decimal number in the range of 0 to 65535 as a command argument where appropriate. This argument format is required unless preceded by the asc command, in which case either argument format is allowed.
Basic communication commands	
`info arg`	Echoes the command and argument with additional information as defined by the argument
Scanning commands*	
`start`	Start scanning
`stop`	Stop scanning
`slist arg0 arg1`	Defines scan list configuration
Output format commands	
`asc`	Delimited ASCII ouptut format, base 10
`bin`	Packed binary output format
`float`	Floating point output scaled in volts

* Commands and arguments are separated by a space (0x20) delimiter.

Basic Communication Commands

The DI-145 command set supports a number of basic command/response items that provide a simple means of ensuring the integrity of the communication link between either a terminal emulator or program. These commands elicit simple, yet useful responses from the instrument and should be employed as the programmer's first DI-145 communication attempt. If these commands don't work with a functioning DI-145 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 (0x20), followed by the response, and ending with a carriage return (0x0d). For example, the command "info 1" generates the following response:

```
info 1 1450(0x0D)
```

DI-145 Command Set	
ASCII Command*	**Action**
`info 0`	Returns "DATAQ"
`info 1`	Returns device name: "1450"
`info 2`	Returns firmware revision, 2 hex bytes (e.g. 0x65 = 101_{10} for firmware revision 1.01)
`info 3 to info 5`	Proprietary internal use for initial system verification
`info 6`	Returns the DI-145's serial number (left-most 8 digits only; right-most two digital are for internal use)

Scanning Commands

stop and *start* Commands

Commands *stop* and *start* define the active scanning state. Command *stop* terminates scanning, and command *start* initiates scanning. It is noteworthy that the DI-145 always transmits data at a 240 Hz throughput rate and a mechanism to reduce sample rate is not provided. The host program may achieve a reduction in sample rate using selective sampling methods whereby every nth point is selected as the converted value. For example, assuming a single enabled channel, a sample rate of 120 Hz is achieved by selecting every other value from the reported data stream. Every third reading is effectively 80 Hz, every fourth yields 60 Hz, and so on. Averaging every nth value on each channel is more difficult but recommended because it reduces the noise by a factor of the square root of n.

slist Command

The DI-145 products with firmware revisions greater than 1.07 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. The DI-145's scan list supports two types of inputs: Analog input channels and Digital inputs. These two type definitions may be placed in the DI-145's scan list in any order that satisfies the requirements of the application. The DI-145's scan list is a maximum of 11 elements long, long enough to support a hardware capacity measurement that's configured to sample all four analog channels and the digital input port during one complete scan. Note that although the scan list is 11 positions in length, any analog channel or the digital input port may appear in the scan list only once. Therefore, scan list lengths greater than 5 positions are undefined.

Each entry in the scan list is represented by a 16-bit number, which is defined in detail in the *DI-145 Scan List Word Definitions* table below, and ten out of the eleven elements of the scan list initialize to 0xFFFF upon power-up. 0xFFFF defines the end of the scan list, and the act of writing any value to the first position of the scan list automatically fills the remaining 10 elements with 0xFFFF. The first item in the scan list initializes to 0x0000 (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. Setting scan list position 0 to the value 0xFFFF results in no data being returned when scanning is initiated.

The *slist* command along with two arguments separated by a space character (0x20) is used to configure the scan list: *slist offset config*

offset defines the index within the scan list and can range from 0 to 0xA to address a total of eleven possible positions. *config* is the 16-bit configuration parameter as defined in table *DI-145 Scan List Word Definitions*. For example, the command *slist 4 x0008* configures the fourth position of the scan list to specify data from the digital input port. Other examples follow:

asc (required since we will use the xhhhh format in the commands that follow)

slist 0 x0008 configures the first scan list position to specify data from the digital input port

slist 3 x0000 configures the fourth scan list position to specify data analog channel 0

slist 2 x0002 configures the third scan list position to specify data from analog channel 2

slist 5 xffff terminates the scan list

Assuming that we wish to sample analog channels 0, 2, and 3 along with digital inputs, the following scan list configuration would work:

asc (required since we will use the xhhhh format in the commands that follow)

slist 0 x0000

slist 1 x0002

slist 2 x0003

slist 3 x0008

Note that since the act of writing to scan list position 0 forces the remainder of the list to the value 0xFFFF, the above configuration is complete upon writing scan list position 3. Further any scan list position (except position 0) may be modified without affecting the contents of the rest of the list.

DI-145 Scan List Word Definitions[†]

Function	Bit Position																Argument using xhhhh Format
	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	x0000
Analog In, Channel 1													0	0	0	1	x0001
Analog In, Channel 2													0	0	1	0	x0002
Analog In, Channel 3													0	0	1	1	x0003
Digital In*													1	0	0	0	x0008
Ignore	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	xffff

† To be consistent with general programming standards, analog and counter channel numbers begin with 0 instead of 1 as indicated on the product label.
* Digital inputs using the *slist* command are not supported when the output mode is programmed for binary, since the state of the two digital input ports is embedded in the data returned for all analog input channels while in that mode.

Sidebar: 12-bit ADC Results From A 10-bit ADC Device

During the calibration process performed at DATAQ Instruments and before the DI-145 is shipped to a customer, known voltage values are applied to each of its analog input channels and a connected PC calculates the required scale and offset factors so the instrument meets or exceeds its accuracy specification. Those factors are written to non-volatile memory built into the DI-145 for the instrument's processor to apply going forward. An unavoidable problem is that this procedure consumes a small amount of the DI-145's dynamic ADC range. Since the DI-145 is speced as having ten bits of measurement resolution, and if we were to report only those ten bits, resulting values would show missing counts or codes. If you applied a very precise ramp to an analog channel and slowly changed the voltage between positive and negative full scale you'd see a response that for the most part changed by only one ADC count as the applied voltage passed through the ADC's bit resolution thresholds, but occasionally you'd also see the ramp jump, skipping one count and resulting in a discontinuity called a missing code. Moreover, the places where missing codes exist would be different from unit to unit and even from channel to channel within the same unit. Obviously, this situation is unacceptable.

Fortunately, the DI-145's ADC and protocol are designed for 12 bits, allowing us to use this improved resolution to provide an accurate calibration at nearly the 10-bit level without missing codes, but with the incongruity that we report a 12-bit number for an instrument speced for 10 bits of ADC resolution. The DI-145 is actually not a 12-bit device, since there will be between 993 and 1,024 active counts within the 4,096 possible 12-bit combinations. Usually the DI-145 will register a four-count change for every 19.53 mV change of applied voltage, but sometimes it will register a 5 count change of 24.41 mV. This value is a composite of a four-count change of the 12-bit converter (or a one count change of the 10-bit converter) equal to 19.53 mV, plus a one-count change of the 12-bit converter equal to 4.88 mV. These large changes are approximately evenly distributed, with at least seven 19.53 mV changes between each. The largest weighted average step size is 20.14 mV, yielding an effective average ADC resolution of 9.96 bits. A device with the minimum 993 active counts would have a total of (869) 19.53 mV steps and (124) 24.41 mV steps.

Output Format Commands

The DI-145 offers a selection of three output formats: binary, ASCII (analog channels in ADC counts), and floating point (analog channels scaled into floating point voltages.) These are specified using the DI-145's output format commands. Command *asc* specifies the delimited ASCII output format in base 10 with analog channels dispalyed in ADC counts; command *float* specifies the delimited ASCII output format in base 10 with analog channels dispalyed as floating point voltages; *bin* specifies the binary output format. In all format instances, values for enabled channels (analog and digital) are output in precisely the same order that they were defined in the scan list through use of the *slist* command.

bin Output Format Command

The DI-145's fastest data output format is a compressed binary stream of one 16-bit word per enabled measurement. The least significant bit of the first byte in the binary output stream is always cleared and set in all other response bytes to allow the host program to synchronize with the data stream. The state of the two least significant digital input bits of the DI-145 DO (Remote event) and D1 (Remote stop/start) is embedded in the binary stream of each transmitted analog channel. A logic low applied to either bit on DI-145 hardware results in a '0' value inserted for that bit in the data stream. This default inclusion supports WINDAQ software's real time remote event and remote stop/start features. The stream sequence repeats until data acquisition is halted by the stop command. Note that since the state of the digital input port is automatically folded into the data returned for the analog input channels, the slist command to read the digital input port (*slist scan_list_position x8*) is not supported in the binary output format mode.

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 (sync)**
0 (Analog in 0)	1	1	A4	A3	A2	A1	A0	D1	D0	0
		2	A11	A10	A9	A8	A7	A6	A5	1
1 (Analog in 1)	2	3	A4	A3	A2	A1	A0	D1	D0	1
		4	A11	A10	A9	A8	A7	A6	A5	1
2 (Analog in 2)	3	5	A4	A3	A2	A1	A0	D1	D0	1
		6	A11	A10	A9	A8	A7	A6	A5	1
3 (Analog in 3)	4	7	A4	A3	A2	A1	A0	D1	D0	1
		8	A11	A10	A9	A8	A7	A6	A5	1

Binary Analog Channel Coding

The DI-145 transmits a 12-bit binary number for every analog channel conversion. Meaningful information is extracted from these readings by inverting the most significant bit, and treating the result as a two's complement number. Note that counts will change roughly by four for each 19.5 mV step change in applied voltage to yield approximately 10 bits of resolution (see the *Sidebar: 12-bit ADC Results From A 10-bit ADC Device* section above for a detailed explanation).

Ideal DI-145 ADC Binary Coding											
AD9*	AD8	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	ADC Counts	Applied Voltage
0	1	1	1	1	1	1	1	1	1	2047	+9.995
0	1	1	1	1	1	1	0	1	1	2043	+9.975
...											
0	0	0	0	0	0	1	0	0	0	8	+0.039
0	0	0	0	0	0	0	1	0	0	4	+0.01953
0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	0	-4	-0.01953
1	1	1	1	1	1	1	0	0	0	-8	-0.03906
...											
1	0	0	0	0	0	0	1	0	0	-2044	-9.9805
1	0	0	0	0	0	0	0	0	0	-2048	-10.0000

* ADC counts versus applied voltage may vary depending upon calibration differences between individual DI-145 units.
**After inverting to yield a two's complement number.

asc Output Format Command

Analog and digital input channel data in an ASCII decimal format may be easier to work with in some situations at the application programming level. The asc command instructs the DI-145 to output data using an ASCII decimal format. Each scan (row) of data in an ASCII decimal format consists of a preceding "sc" fixed character pair, then the ASCII decimal representation of the results of active scan elements. Any leading zeros are suppressed for speed, a space delimiter is used between fields, and a carriage return (0x0D) terminates each line.

The *asc* command evoked before other commands also allows command arguments to use the xhhhh hexadecimal format. The *bin* mode may be evoked after the ASCII mode using the xhhhh format for configuration without affecting the predefined setup.

DI-145 ADC channel data is output as a decimal number ranging from -2048 to 2047 as defined in the *Ideal DI-145 ADC Binary Coding* table above. The number should be interpreted as ADC counts, and will change approximately four counts per ADC step. The following is a typical output with four analog channels enabled:

```
sc 12 12 12 12
sc 800 792 796 792
sc 712 708 708 708
sc 4 0 0 -4
sc 796 792 792 792
sc 760 752 756 752
sc 0 -8 -8 -8
sc 544 536 536 532
sc 780 776 776 776
sc -4 -8 -8 -8
sc 240 228 232 228
sc 792 784 788 784
```

The single DI-145 digital input channel is represented as a decimal number ranging from 0 to 3 according to the following table:

Digital Input Representation Using the ASCII Base 10 Formatted Response		
ASCII Base 10 Output	**Digital Inputs**	
	D1	**D0**
0	0	0
1	0	1
2	1	0
3	1	1

If all channels are enabled on the DI-145 in the order of analog input channels 0-3 and the digital input channel, the typical ASCII decimal output may look like this:

```
sc -4 -4 -4 -4 3
sc -4 -4 -4 -4 3
sc -4 -4 -4 -4 3
sc -4 -4 -4 -4 3
sc -4 -4 -4 -4 3
sc -4 -4 -4 -4 3
sc -4 -4 -4 -4 3
sc -4 -4 -4 -4 3
sc -4 -4 -4 -4 3
sc -4 -4 -4 -4 3
sc -4 -4 -4 -4 3
sc -4 -4 -4 -4 3
sc -4 -4 -4 -4 3
sc -4 -4 -4 -4 3
sc 0 -4 -4 -4 3
sc -4 -4 -4 -4 3
sc -4 -4 -4 -4 3
```

float Output Format Command

The *float* command works in tandem with the *asc* command. Where the latter shows analog channel values in ADC counts, the *float* command converts ADC counts into a floating point number scaled in volts. You may toggle between *float* and *asc* to display volts and ADC counts respectively. The following is a typical output with all analog channels and the digital channel enabled while the *float* command is active:

```
sc  0.012  0.006  0.006  0.000 0
sc  0.012  0.012  0.006  0.000 0
sc  0.006  0.012 -0.006  0.000 0
sc -0.006  0.006  0.006  0.000 1
sc -0.006  0.006 -0.006  0.006 1
sc  0.000  0.006  0.000  0.000 2
sc  0.000  0.012  0.006  0.000 3
sc -0.006  0.012 -0.006  0.006 3
sc  0.006  0.006 -0.012 -0.006 3
sc  0.000  0.000  0.000 -0.012 1
sc  0.000  0.006  0.000  0.000 0
```

Go to the DI-145 Product Page