DI-155 Communication Protocol

DATAQ Instruments

Although DATAQ Instruments provides ready-to-run WinDaq software with its DI-155 Data Acquisition Starter Kits, programmers will want the flexibility to integrate the DI-155 in the context of their own application. To do so they want complete control over DI-155 hardware, which can be accomplished by using the device at the protocol level. This white paper describes how protocol-level programming of the DI-155 is implemented across the Windows and Linux operating systems. First, we'll describe the virtual COM operation of the DI-155's interface and how communicating with the DI-155 is accomplished via a COM port hooked by the operating system. Then we'll define the DI-155's command set and scan list architecture and finish with a description of the DI-155'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-155 driver package and connecting DI-155 hardware to the host computer's USB port results in a COM port being hooked by the operating system and assigned to the DI-155 device. Multiple DI-155 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-155 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-155'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-155s. The Vendor and Product ID combination for the DI-155 is: *Vid_0683&Pid_1550*. With this information and at least one connected DI-155, 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-155 connected to the PC. The Device Instance ID assumes the following typical string value: *USB\VID_0683&PID_1550\5&18B6E64F&0&8*. The first two sections of this string *(USB\VID_0683&PID_1550\)* are constant for all DI-155s. The second section *(5&18B6E64F&0&8)* will vary depending upon where in the USB port hierarchy the DI-155 is physically connected. Since more than one DI-155 cannot be connected to the same USB port this value will be unique for each concurrently connected DI-155. The entire string value is required for the second step.

2. Registry tree *HKEY_LOCAL_MACHINE \System\CurrentControlSet\Enum\ USB\VID_0683&PID_1550\5&18B6E64F&0&8* (continuing with the above example) shows a variable called *FriendlyName* set to string value *DATAQ DI-155 (COMXX)*, where *XX* is the COM port number assigned to the specified DI-155. This string may be parsed to extract the port number assigned to the DI-155. The process may be repeated using other Device Instance IDs determined from step (1) for other connected DI-155 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-155, 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_1550-** presents links to */dev/ttyACM** device files of currently plugged in DI-155 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-155:

modprobe usbserial vendor=0x0683 product=0x1550

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

DI-155 Command Set Overview

The DI-155 employs a simple ASCII character command set that allows complete control of the instrument. ASCII comes in handy when a terminal emulators such as HyperTerminal or PuTTY are 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) except the digital output command *Dhh* and the reset command *R1* (both D and R commands need NULL character to lead them).

DI-155 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-155 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
`srate arg0`	Defines scan rate
Output format commands	
`asc`	Delimited ASCII ouptut format, base 10 in ADC counts
`bin`	Packed binary output format
`float`	Floating point output scaled in volts
Digital out command*	
`dout arg0`	Outputs the specified data to the digital output port with a command echo
`Dhh`	Outputs the specified data to the digital output port without a command echo
Reset commands	
`reset arg0`	Performs various reset operations
`R1`	Same as *reset* command, but suppresses the echo

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

Basic Communication Commands

The DI-155 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-155 communication attempt. If these commands don't work with a functioning DI-155 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 1550(0x0D)
```

DI-155 Command Set	
ASCII Command*	**Action**
`info 0`	Returns "DATAQ"
`info 1`	Returns device name: "1550"
`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-155'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.

slist Command

The DI-155 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-155's scan list supports four types of inputs: Analog channels; Rate channel; Count channel; Digital inputs. These four type definitions may be placed in the DI-155's scan list in any order that satisfies the requirements of the application. The DI-155's scan list is a maximum of 11 elements long, which allows a hardware capacity measurement that's configured to sample all four analog channels, the counter and rate channels, and one 4-bit digital input port during one complete scan. Note that although the scan list is 11 positions in length, any analog, digital output, rate, or counter channel may appear in the scan list only once. Therefore, scan list lengths greater than 7 positions are undefined.

Each entry in the scan list is represented by a 16-bit number, which is defined in detail in the *DI-155 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 (even though only the first seven are valid). *config* is the 16-bit configuration parameter as defined in table *DI-155 Scan List Word Definitions*. For example, the command *slist 5 x000a* configures the sixth position of the scan list to specify data from the counter. Other examples follow:

asc this command is 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 x0009 configures the fourth scan list position to specify data from Rate input on its 10 kHz range.

slist 0 x0003 configures the first scan list position to specify data from analog channel 3 for its 50 V range

slist 5 xffff terminates the scan list.

Assuming that we wish to sample analog channels 2 (10 VFS), 3 (3.125 VFS), rate (100 Hz range), count, and finally the digital input port, the following scan list configuration would work:

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

slist 0 x0302

slist 1 x0603

slist 2 x0709

slist 3 x000a

slist 4 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 4. 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-155 Scan List Word Definitions†</th></tr>
<tr><th>Function</th><th colspan="16">Bit Position</th></tr>
<tr><th></th><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="5" rowspan="5">All Unused Bits = 0</td><td colspan="3" rowspan="4">Analog gain code
See Analog Gain Code Table</td><td colspan="4" rowspan="5">All 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="3">All 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 Range Tabel)</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 and counter channel numbers begin with 0 instead of 1 as indicated on the product label.

The protocol also supports a range setting for frequency measurements, which applies ONLY when the output coding format selection is binary. When the ASCII format is selected scaling is handled transparently by the firmware and presented as a floating point number in the output data stream. If the binary output mode is selected frequency counts are provided that range from 0 to 16,383 (14-bit.) Counts may be converted to a frequency in Hertz by applying the following formula:

$$\mathbf{Hz = (Range) \bullet (counts) \div 16384}$$

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

Rate Range Table (for DI2 connections)*				
Bit Position				Range (Hz)
11	10	9	8	
0	0	0	1	10,000
0	0	1	0	5,000
0	0	1	1	2,000
0	1	0	0	1,000
0	1	0	1	500
0	1	1	0	200
0	1	1	1	100
1	0	0	0	50
1	0	0	1	20
1	0	1	0	10
1	0	1	1	5

*A range selection DOES NOT apply when the output coding format is ASCII.

The DI-155 supports programmable gain on an analog channel-by-channel basis. The following Gain Code Table defines the selectable gains supported by the DI-155. Note that full scale measurement range at a gain of 1 is ±50 VFS.

DI-155 Analog Gain Code Table				
Bit Position			Gain Factor	± Volts Full Scale
10	9	8		
0	0	0	1	50.0
0	0	1	2	25.0
0	1	0	4	12.5
0	1	1	5	10.0
1	0	0	8	6.25
1	0	1	10	5.0
1	1	0	16	3.125
1	1	1	20	2.5

srate Scan rate Command

Command *srate* defines scanning speed, or the rate at which the DI-155 scans through items in the scan list that you defined above. *srate* is specified as an integer argument within the range of 75 to 65,535, and the resulting scan speed for scan list elements is defined by the following equation:

Total sample rate (Hz) = 750,000 ÷ srate, where
75 < srate < 65,535

This approach results in a sample throughput rate that ranges from 11.44 to 10,000 Hz so that the sample rate per channel is defined by the sample throughput rate divided by the number of enabled channels. The host program may achieve a further reduction in sample rate below 11.44 Hz by using selective sampling methods whereby every nth point is selected as the converted value. For example, a sample throughput rate of 5 Hz for a single enabled channel is achieved by applying an integer value of 30,000 to the *srate* command, and further selecting every 5th value from the reported data stream. Every 25th reading is effectively 1 Hz. Averaging every n values on each channel is more difficult but recommended since it reduces the noise by a factor of the square root of n.

Output Format Commands

The DI-155 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-155's output format commands. Command *asc* specifies the delimited ASCII output format in base 10 with analog channels displayed in ADC counts; command *float* specifies the delimited ASCII output format in base 10 with analog channels displayed as floating point voltages; *bin* specifies the binary output format. In all format instances, values for enabled channels (analog, digital, count, or rate) are output in precisely the same order that they were defined in the scan list through use of the slist command.

Unlike the DI-145 and the DI-149, where the state of the two least significant digital input bits DO (Remote event) and D1 (Remote stop/start) is embedded in the binary stream of each transmitted analog channel, the DI-155 uses these two bit positions on analog channels to accommodate its higher resolution ADC. Therefore, the state of all four digital inputs is determined by defining a digital input channel in the scan list (see the *DI-155 Scan List Word Definitions* table above.) Enabling the digital input channel will affect the per channel sample rate, since it occupies an active position in the scan list.

bin Output Format Command

The DI-155'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 stream sequence repeats until data acquisition is halted by the *stop* command.

DI-155 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	A6	A5	A4	A3	A2	A1	A0	0
		2	A13	A12	A11	A10	A9	A8	A7	1
1 (Analog in 1)	2	3	A6	A5	A4	A3	A2	A1	A0	1
		4	A13	A12	A11	A10	A9	A8	A7	1
2 (Analog in 2)	3	5	A6	A5	A4	A3	A2	A1	A0	1
		6	A13	A12	A11	A10	A9	A8	A7	1
3 (Analog in 3)	4	7	A6	A5	A4	A3	A2	A1	A0	1
		8	A13	A12	A11	A10	A9	A8	A7	1
8 (Digital in)	5	9	D0	0	0	0	0	0	0	1
		10	0	0	0	0	D3	D2	D1	1
9 (Rate in)	6	11	C6	C5	C4	C3	C2	C1	C0	1
		12	C13	C12	C11	C10	C9	C8	C7	1
10 (Counter in)	7	13	C6	C5	C4	C3	C2	C1	C0	1
		14	C13	C12	C11	C10	C9	C8	C7	1

Binary Analog Channel Coding

The DI-155 transmits a 14-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 the DI-155's resolution is slightly less than full 14-bit accuracy after calibration, and missing counts can and will exist between the full scale ranges for the following reason:

Although the DI-155's full scale measurement range is ±50 V at a gain factor of 1, the instrument's actual measurment span is nominally ±58.65 V. This "headroom" allows scaling and offset parameters to be applied per analog channel and for each measurment range to render more measurement accuracy. The consequence of this approach is that full 14-bit accuracy is sacrificed for a slightly lower value. A typical DI-155 operating between ±50V will consume approximately 85.25% of the ±58.65 V range, or approximately ±6983 codes for a calibrated device instead of the ±8192 codes available to it as a full 14-bit device. The resolution would then nominally be Log(6983*2)/Log(2), or 13.77 bits. The missing 1209 codes for each polarity of the typical DI-155 will be distributed evenly above and below 0 Volts, or approximately every 6 to 7 readings out of the 8192 possible output values for each polarity. When averaging is done by WinDaq or the scanning process, the noise in the input will allow all output values to occur.

Ideal DI-155 ADC Binary Coding*

AD13**	AD12	AD11	AD10	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	ADC Counts	Applied Voltage
0	1	1	1	1	1	1	1	1	1	1	1	1	1	8191	
0	1	1	1	1	1	1	1	1	1	1	1	1	0	8190	
0	0	0	0	0	0	0	0	0	0	0	0	1	0	2	
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	(50 ÷ analog gain) x (ADC Counts ÷ 8192)
1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1	
1	1	1	1	1	1	1	1	1	1	1	1	1	0	-2	
1	0	0	0	0	0	0	0	0	0	0	0	0	1	-8191	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	-8192	

* ADC counts versus applied voltage may vary depending upon calibration differences between individual DI-155 units.
**After inverting to yield a two's complement number

asc Output Format Command

Analog, digital and counter 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-155 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.

Since use of the ASCII output mode greatly multiplies the number of bytes transmitted by the DI-155 (e.g. as many as ten bytes per analog channel as opposed to two in the binary mode), the ASCII mode does not support the full sample rate capacity of the DI-155. Care should be taken to ensure that the value set for srate conforms to the following whenever the ASCII output mode is selected:

srate > 375 × (the number of active scan list elements)

DI-155 ADC channel data is output as a decimal number ranging from -8192 to 8191 as defined in the Ideal DI-155 ADC Binary Coding table above. The following is a typical output with four analog channels enabled:

Typical Output ASCII Decimal Output for Four Enabled Analog Channels

```
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
sc -8 -12 -12 -12
sc 104 96 96 96
sc 796 788 792 788
sc 320 316 316 316
sc 44 40 40 36
sc 796 792 792 792
sc 588 584 588 584
```

The single DI-155 digital input channel is represented as a 4-bit decimal number ranging from 0 to 15 according to the following table:

Digital Input Representation Using the ASCII Decimal Formatted Response				
ASCII Base 10 Output	Digital Inputs			
	D3	D2	D1	D0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
...				
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

The DI-155 counter channel is output as decimal values ranging from 0 to 16383:

Typical ASCII Decimal Output for the Counter Channel

```
sc 6003
sc 6004
sc 6005
sc 6006
sc 6007
sc 6008
sc 6009
sc 6010
sc 6011
sc 6012
```

The DI-155 rate channel is output as an auto-ranging floating point number:

Typical ASCII Decimal Output for the Rate Channel

```
sc 35.36
sc 35.69
sc 35.47
sc 34.58
sc 35.65
sc 35.47
sc 35.78
sc 35.77
sc 34.02
sc 38.98
```

If all channels are enabled on the DI-155 in the order of analog input channels 0-3, the digital input, and the counter and rate channels, the typical ASCII decimal output may look like this:

Typical ASCII Base 10 Output -- All Inputs Enabled

```
sc 592 588 588 588 15 5.99 599
sc -196 -200 -200 -204 15 6.00 600
sc -424 -424 -428 -428 15 6.01 601
sc 576 572 576 576 -12 15 6.02 602
sc -32 -36 -36 -36 -12 15 6.03 603
sc -568 -572 -572 -572 15 6.04 604
sc 388 384 384 388 -12 15 6.05 605
sc 140 132 132 132 -16 15 6.06 606
sc -640 -648 -648 -648 15 6.07 607
sc 188 188 188 188 -12 15 6.08 608
sc 300 296 296 292 -12 15 6.09 609
sc -564 -568 -568 -572 15 6.10 610
sc -8 -12 -12 -12 15 6.11 611
```

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.

Digital Output Commands

Two digital output commands are supported by the protocol, one that echoes the command and another that does not. The dout command will cause the DI-155 to echo the command and its arguments back to the issuing program. The D command is also supported by the protocol to perform the same function, but it does not echo. Since digital output states may be changed dynamically and asynchronously during data acquisition, echo suppression may be used to ensure that the echo does not corrupt the stream of acquired data flowing from the DI-155 to the controlling program.

dout command (echoed)

The *dout* command accepts one argument separated by a space (0x20) that defines the state of the digital output bits. The argument can be supplied as an ASCII decimal number. An ASCII hexadecimal value using the xh notation may also be used if the dout command has been preceded by the asc command. Note that all digital output bits are low true, a logic 1 written to a bit causes the port to sink current and assume a logic 0 state. A logic 0 written to a bit causes the port to source current and assume a logic 1 state.

Digital Output States					
dout Argument		Output Bit State (low true)			
Hexadecimal	Decimal	¬D3	¬D2	¬D1	¬D0
x0	0	1	1	1	1
x1	1	1	1	1	0
x2	2	1	1	0	1
...					
xd	13	0	0	1	0
xe	14	0	0	0	1
xf	15	0	0	0	0

D command (not echoed)

The *D* command (case sensitive) is used when an echo is not desired. The D command is evoked with a two-character argument without delimiters to define the state of the digital output bits. The argument is not case sensitive and is supplied as an ASCII hexadecimal number with a leading 0 (zero) character. Note that all digital output bits are low true, a logic 1 written to a bit causes the port to sink current and assume a logic 0 state. A logic 0 written to a bit causes the port to source current and assume a logic 1 state.

Digital Output States				
Dhh	Output Bit State (low true)			
where hh is	¬D3	¬D2	¬D1	¬D0
00	1	1	1	1
01	1	1	1	0
02	1	1	0	1
...				
0A or 0a	0	1	0	1
0B or 0b	0	1	0	0
0C or 0c	0	0	1	1
0D or 0d	0	0	1	0
0E or 0e	0	0	0	1
0F or 0f	0	0	0	0

Reset Commands

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

Reset Commands	
ASCII Command*	**Action**
`reset 1`	Resets the counter to a value of zero.
`R1`	Same as “reset 1” above, but without an echo.

DI-155 Product Page: http://www.dataq.com/products/startkit/di155.html.