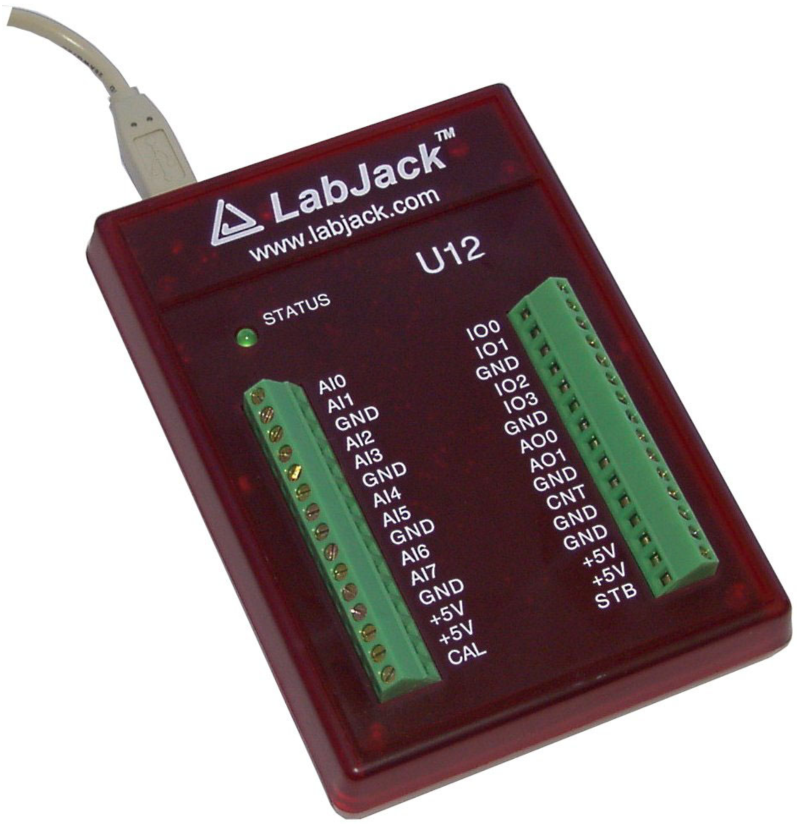
LabJack U12 User's Guide

Revision 1.04
11/5/2002

Meilhaus Electronic GmbH
www.meilhaus.com
support@meilhaus.com



For the latest version of the user's guide, the quickstart guide, or software, go to www.labjack.com.

Table Of Contents

Table Of Figures

1. Installation

The LabJack U12 requires a PC running Windows 98SE, ME, 2000, or XP. To determine your operating system version, go to

Start => Settings => Control Panel => System => General

and make sure the version number is 4.10.2222 or higher (Win98SE=4.10.2222, WinME=4.90.3000, Win2000=5.0.2195, WinXP=5.1.XXXX).

It does not matter if the hardware or software is installed first.

If you experience installation problems on Windows 98 Second Edition, before contacting us, please go to the downloads page at labjack.com and download "win98sehid.zip" (see the readme file for more information).

1.1 Hardware Installation

With the PC on and using the included cable, connect the LabJack U12 to the USB port on the PC or USB hub. The USB cable provides power and communication for the LabJack U12. The status LED should immediately blink 4 times (at about 4 Hz), and then stay off while the LabJack enumerates.

Enumeration is the process where the PC's operating system gathers information from a USB device that describes it and it's capabilities. The low-level drivers for the LabJack U12 come with Windows and enumeration will proceed automatically. The first time a device is enumerated on a particular PC, it can take a minute or two, and Windows might prompt you about installing drivers. Accept all the defaults at the Windows prompts, and reboot the PC if asked to do so. The Windows Installation CD might also be needed at this point. Make sure a CD with the correct version of Windows is provided. Enumeration occurs whenever the USB cable is connected, and only takes a few seconds after the first time.

When enumeration is complete, the LED will blink twice and remain on. This means Windows has enumerated the LabJack properly.

If the LabJack fails to enumerate:

- Make sure you are running Windows OS version 4.10.2222 or higher,
- Try connecting the LabJack to another PC,
- Try connecting a different USB device to the PC,
- Check our online forum and/or contact LabJack.

1.2 Software Installation

Although, the low-level USB drivers for the LabJack are included with Windows, high-level drivers are needed to send and receive data. The included LabJack CD installs the high-level drivers, example source code, and example applications.

Close all open applications, especially LabJack related software, and insert the LabJack CD. If autorun is enabled, the installation program should start automatically. If the installation does not start, you will have to manually double-click on LabJackVXXX.exe.

When the LabJack installation is finished, it will start the National Instruments LabVIEW Run-Time Engine (LVRTE) setup. The LVRTE is required for the example applications: LJconfig, LJlogger, LJscope, and LJtest. If prompted to reboot after this installation, go ahead and do so.

Virus scanners can often interfere with the installation of the LVRTE. If you have trouble running the example applications, repeat the LabJack software installation to make sure the LVRTE is installed.

To test the installation, start LJtest by selecting

Start => Programs => LabJack => LJtest.

Make sure “Test Fixture Installed” and “Continuous” are not selected, and press the “Run” button. LJtest will step through 8 separate tests and all should pass.

2. Hardware Description

The external features of the LabJack U12 are:

- USB connector,
- DB25 digital I/O connector,
- Status LED,
- 30 screw terminals.

The USB connection provides power and communication. No external power supply is needed. The +5 volt connections available at various locations are outputs, do not connect a power supply.

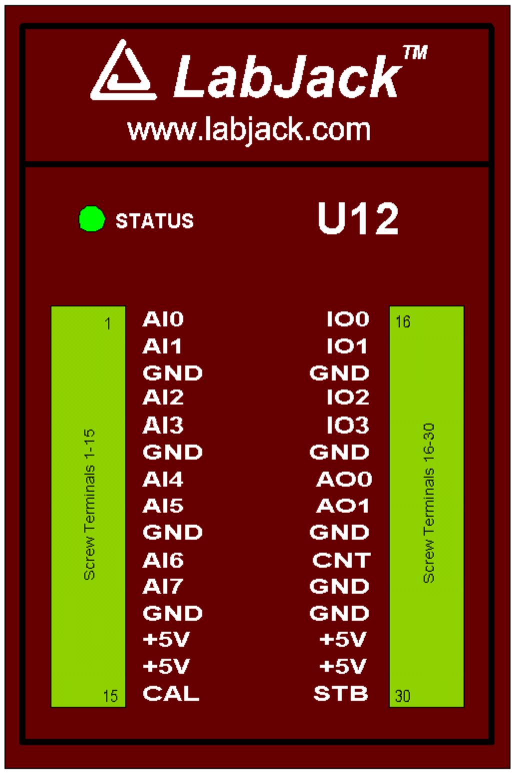


Figure 2-1. LabJack U12 top surface.

Figure 2-1 shows the top surface of the LabJack U12. Not shown is the USB and DB25 connector, which are both on the top edge. The DB25 connector provides connections for 16 digital I/O lines, called D0-D15. It also has connections for ground and +5 volts. All connections besides D0-D15, are provided by the 30 screw terminals shown in Figure 1. Each individual screw terminal has a label, AI0 through STB.

The status LED blinks 4 times at power-up, and then blinks once and stays on after enumeration (recognition of the LabJack U12 by the PC operating system). The LED also blinks during burst and stream operations, unless disabled. The LED can be enabled/disabled through software using the functions AISample, AIBurst, or AIStreamStart. Since the LED uses 4-5 mA of current, some users might wish to disable it for power-sensitive applications.

2.1 AI0 – AI7

Hardware

The LabJack U12 has 8 screw terminals for analog input signals. These can be configured individually and on-the-fly as 8 single-ended channels, 4 differential channels, or combinations in between. Each input has a 12-bit resolution and an input bias current of ±90 μA.

- Single-Ended: The input range for a single-ended measurement is ±10 volts.
- Differential channels can make use of the low noise precision PGA to provide gains up to 20, giving an effective resolution greater than 16-bits. In differential mode, the voltage of each AI with respect to ground must be between ±10 volts, but the range of voltage difference between the 2 AI is a function of gain (G) as follows:

G=1	±20 volts
G=2	±10 volts
G=4	±5 volts
G=5	±4 volts
G=8	±2.5 volts
G=10	±2 volts
G=16	±1.25 volts
G=20	±1 volt

The reason the range is ±20 volts at G=1 is that, for example, AI0 could be +10 volts and AI1 could be -10 volts giving a difference of +20 volts, or AI0 could be -10 volts and AI1 could be +10 volts giving a difference of -20 volts.

The PGA (programmable gain amplifier, available on differential channels only) amplifies the AI voltage before it is digitized by the A/D converter. The high level drivers then divide the reading by the gain and return the actual measured voltage.

Figure 2-2 shows a typical single-ended connection measuring the voltage of a battery. This same measurement could also be performed with a differential connection to allow the use of the PGA. In general, any single-ended measurement can be performed using a differential channel by connecting the voltage to an even-numbered analog input, and grounding the associated odd-numbered analog input (as shown by the dashed connection to AI1 in Figure 2-2).

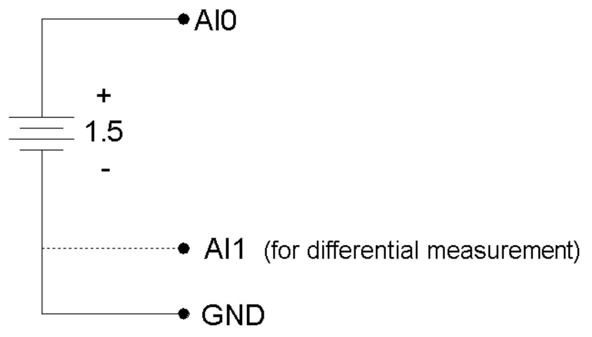


Figure 2-2. Single-ended measurement.

Figure 2-3 shows a typical differential connection measuring the voltage across a current shunt. A differential connection is required when neither leg of the shunt is at ground potential. Make sure that the voltage of both AI0 an AI1 with respect to ground is within ±10 volts. For instance, if the source (Vs) shown in Figure 2-3 is 120 VAC, the difference between AI0 and AI1 might be small, but the voltage from both AI0 and AI1 to ground will have a maximum value near 170 volts, and will seriously damage the LabJack.

Whether or not the ground (GND) connection is needed (Figure 2-3) will depend on the nature of Vs.

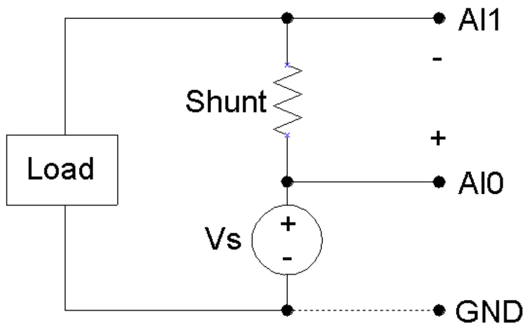


Figure 2-3. Differential measurement.

Figure 2-4 shows a single-ended connection used to measure the output voltage of a typical voltage-divider circuit. The voltage divider circuit is a simple way to convert a varying resistance (thermistor, photoresistor, potentiometer, etc.) to a varying voltage. With nothing connected to Va, the value of the unknown resistance, R2, can be calculated as:

$R2 = Va*R1 / (Vs-Va)$,

where Vs is the supply voltage (+5V in Figure 2-4).

When Va is connected to AI0, as shown in Figure 2-4, the input bias current of the LabJack affects the voltage divider circuit, and if the resistance of R1 and R2 is too large, this effect must be accounted for or eliminated. This is true for any signal with too high of a source impedance.

All measuring devices have maximum analog input bias currents that very from picoamps to milliamps. The input bias current of the LabJack U12's analog inputs varies from +70 to -94 microamps (μA). This is similar to an input impedance of about 100 kΩ, but because the current is nonzero at 0 volts, it is better to model the analog input as a current sink obeying the following rule:

$Iin = 8.181*Va – 11.67 \quad \mu A$

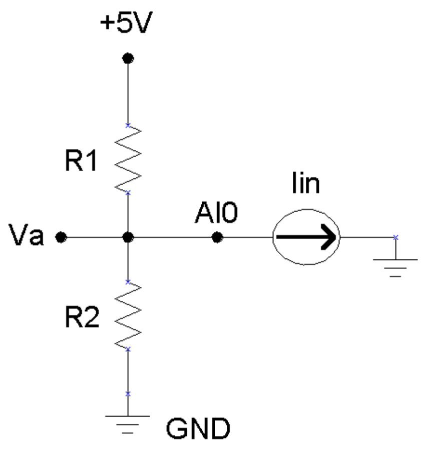


Figure 2-4. Single-ended measurement with voltage divider circuit.

Because the input bias current is known, as a function of input voltage, the simple voltage divider equation can be modified as follows to account for input bias current:

$R2 = Va / [((Vs-Va)/R1) – (8.181\mu * Va) + 11.67\mu]$

As an alternative to the equation above, Va can be buffered by a single-supply rail-to-rail operational amplifier, and the original simple voltage divider equation can be used. This solution works for any single-ended signal which stays between 0 and +5 volts. Some op-amp choices are:

- TLV2462
- LMC6482
- MAX4166

Software

Readings from the analog inputs are returned by the functions EAnalogIn, AISample, AIBurst, and AIStreamRead.

EAnalogIn is a simplified (E is for easy) function that returns a single reading from 1 analog input channel.

AISample returns a single reading of 1-4 channels, and takes up to 20 ms to execute, providing a maximum date rate of about 50 Hz per channel. This function also controls the status LED and sets the state of the IO pins.

AIBurst acquires multiple samples of 1-4 channels at a hardware-timed sample rate of up to 8192 Hz. The acquisition can be triggered based on the change of state of an IO pin. This function also controls the status LED and returns the states of the IO pins (which are read every 4 samples).

Internally, the actual number of samples collected and transferred by the LabJack during an AIBurst call is the smallest power of 2, from 64 to 4096, which is at least as big as numSamples. The execution time of this function, in milliseconds, can be estimated as (auto or turbo mode):

30+(1000*numSamplesActual/sampleRate)+(0.4*numSamplesActual)

numSamples = numScans * numChannels

sampleRate = scanRate * numChannels

AIStreamRead is called periodically during a stream acquisition started by AIStreamStart. Each call retrieves multiple samples of 1-4 channels from the LabJack stream buffer, along with the states of the IO pins (read every 4 samples). Hardware-timed sample rates of up 1200 Hz are available.

2.2 AO0 & AO1

The LabJack U12 has 2 screw terminals for analog output voltages. Each analog output can be set to a voltage between 0 and the supply voltage (+5 volts nominal) with 10-bits of resolution.

The output voltage is ratiometric with the +5 volt supply, which is generally accurate to ±5% (see Appendix A). If an output voltage of 5 volts is specified, the resulting output will be 100% of the supply voltage. Similarly, specifying 2.5 volts actually gives 50% of the supply voltage.

If improved accuracy is needed, measure the +5 volt supply with an analog input channel, and the actual output voltage can be calculated. For instance, if an analog output of 2.5 volts is specified and a measurement of +5V returns 5.10 volts, the actual output voltage is 2.55 volts. Alternatively, the analog output can itself be measured with an analog input.

There is a 1st order low-pass filter on each analog output with a 3dB frequency around 22 Hz.

Each analog output on the LabJack U12 is the output of an operational amplifier. Accidental application of an overvoltage or transient such as ESD (electro-static discharge) can damage these amplifiers. If your application warrants additional protection, a series resistor and zener diode to ground are recommended. A 5.6 volt zener such as the 1N4734 (Digi-Key part number 1N4734ADICT-ND) can be connected directly to the screw terminals. The thin stripe (cathode) goes to AOx and the other lead goes to a nearby GND. One end of the series resistor connects to AOx and the other end connects to your signal wire. The resistor should be as large as possible (10 ohms will help, 100 ohms is good, 1k ohms is very good), taking into account the voltage that will be dropped across the resistor.

Software

The analog outputs are set using the function EAnalogOut (easy function) or AOUpdate, which take up to 20 ms to execute, providing a maximum update rate of about 50 Hz per channel. AOUpdate also controls/reads all 20 digital I/O and the counter.

2.3 IO0 – IO3

Connections to 4 of the LabJack's 20 digital I/O are made at the screw terminals, and are referred to as IO0-IO3. Each pin can individually be set to input, output high, or output low. These 4 channels include a 1.5 k Ω series resistor that provides overvoltage/short-circuit protection. Each channel also has a 10 MΩ resistor connected to ground.

One common use of a digital input is for measuring the state of a switch as shown in Figure 2-5. If the switch is open, IO0 reads FALSE. If the switch is closed, IO0 reads TRUE.

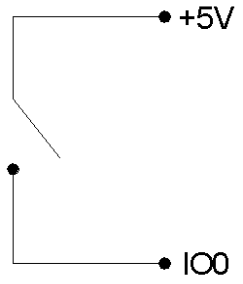


Figure 2-5. IO used to detect the state of a switch.

While providing overvoltage/short-circuit protection, the 1.5 k Ω series resistor on each IO pin also limits the output current capability. For instance, with an output current of 1 mA, the series resistor will drop 1.5 volts, resulting in an output voltage of about 3.5 volts.

Software

The easy functions EDigitalIn or EDigitalOut are used to read or set the state of one digital line, and both take up to 20 ms to execute.

The functions AOUpdate and DigitalIO are used to set the direction, set the state, and/or read the state, of each IO pin. Both of these functions take up to 20 ms to execute, providing a maximum update rate of about 50 Hz per pin.

The function AISample can set/read the state of each IO, and the function Counter reads the state of each IO.

The functions AIBurst and AIStreamRead, take a reading of the IO states and return it with the analog data. The states of the 4 IO are read simultaneously every 4 samples, providing a data rate of up to 2048 Hz per pin for burst mode, or 300 Hz per pin for stream mode.

2.4 D0 – D15

Connections to 16 of the LabJack's 20 digital I/O are made at the DB25 connector, and are referred to as D0-D15. These 16 lines have no overvoltage/short-circuit protection, and can sink or source up to 25 mA each (total sink or source current of 200 mA max for all 16). This allows the D pins to be used to directly control some relays. All digital I/O are CMOS output and TTL input except for D13-D15, which are Schmitt trigger input. Each D pin has a 10 MΩ resistor connected to ground.

DB25 Pinouts:

1: D0	6: D5	11: +5V	16: GND	21: D11
2: D1	7: D6	12: +5V	17: GND	22: D12
3: D2	8: D7	13: +5V	18: D8	23: D13
4: D3	9: NC	14: GND	19: D9	24: D14
5: D4	10: +5V	15: GND	20: D10	25: D15

These digital I/O can detect the state of a switch using the same circuit shown in Figure 2-5.

Because the D pins have no overvoltage/short-circuit protection, the user must be careful to avoid damage. A series resistor can provide substantial protection for these pins (see the CB25 datasheet). The following are examples of things that could damage a D pin and/or the entire LabJack:

- Shorting a high output to ground (or any potential other than +5V).
- Shorting a low output to a nonzero voltage (such as +5V).
- Exceeding the voltage limits specified in Appendix A.

Software
The easy functions EDigitalIn or EDigitalOut are used to read or set the state of one digital line, and both take up to 20 ms to execute.

The functions AOUpdate and DigitalIO are used to set the direction, set the state, and/or read the state, of each D pin. In addition, DigitalIO also returns the current state of the direction and output registers. Both of these functions take up to 20 ms to execute, providing a maximum update rate of about 50 Hz per pin.

2.5 CNT

The input connection to the 32-bit counter is made at screw-terminal CNT. The counter is incremented each time the voltage at CNT changes from less than 1 volt to greater than 4 volts, and is capable of counting frequencies up to at least1 MHz.

Software
The functions ECount (easy function), AOUpdate, and Counter are used to reset or read the counter. If a reset is specified, the counter is read first. All of these functions take up to 20 ms to execute, providing a maximum update rate of about 50 Hz.

Counter readings can also be returned in stream mode (AIStreamRead) at up to 300 Hz.

2.6 CAL – STB

These terminals are used during testing and calibration.

The CAL terminal is an operational amplifier output of the LabJack U12's 2.5 volt reference. Accidental application of an overvoltage or transient such as ESD (electro-static discharge) can damage this amplifier, and cause a failure of all analog inputs. In most cases, this terminal is unused and can be physically shielded. As an alternative, a series resistor and zener diode can be used to provide additional protection. A 5.6 volt zener such as the 1N4734 (Digi-Key part number 1N4734ADICT-ND) can be connected directly to the screw terminals. The thin stripe (cathode) goes to CAL and the other lead goes to a nearby GND. One end of the series resistor connects to AOx and the other end connects to your signal wire (the series resistor is not needed if nothing is connected to CAL). The resistor should be as large as possible (10 ohms will help, 100 ohms is good, 1k ohms is very good), taking into account the voltage that will be dropped across the resistor.

2.7 +5V

The LabJack has a nominal +5 volt internal power supply. Power can be drawn from this power supply by connecting to the +5V screw-terminals, or the +5V pins on the DB25 connector. The total amount of current that can be drawn from the +5V pins, analog outputs, and digital outputs, is 450 mA for most desktop computers and self-powered USB hubs. Some notebook computers and bus-powered hubs will limit this available current to about 50 mA.

2.8 GND

The GND connections available at the screw-terminals and DB25 connector provide a common ground for all LabJack functions. They are all the same.

3. Example Applications

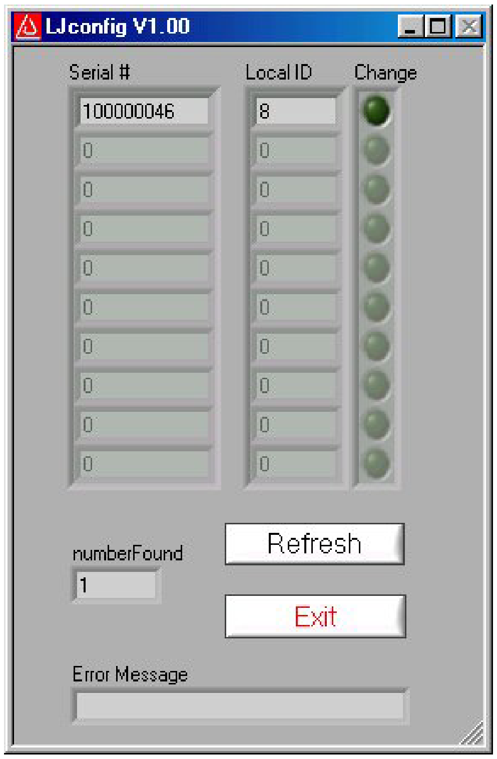
The LabJack U12 CD installs 7 example applications: LJconfig, LJcounter, LJfg, LJlogger, LJscope, LJstream, and LJtest.

- LJconfig: Lists all LabJacks connected to the USB and allows the local ID to be set on each.
- LJcounter: Reads the LabJack counter and provides the current frequency or count.
- LJfg (Function Generator): Outputs basic waveforms on AO0 (analog output zero).
- LJlogger: Saves data to disk, writes data to an HTML page on the Internet, and performs various actions (including email) on trigger events.
- LJscope: Simulates an oscilloscope by reading data from 2 AI channels in burst mode.
- LJstream: Uses stream mode to read, graph, and write to file, 4 AI channels.
- LJtest: Runs a sequence of tests on the LabJack itself.

The LabVIEW source code for most of these applications is installed in the examples directory.

3.1 LJconfig

Every LabJack has a local ID and serial number. The local ID is a value between 0 and 255 that can be changed by the user. The serial number is a value between 256 and 2,147,483,647 that is unique among all LabJacks and cannot be changed by the user. LJconfig is used to set the local ID of a particular LabJack.



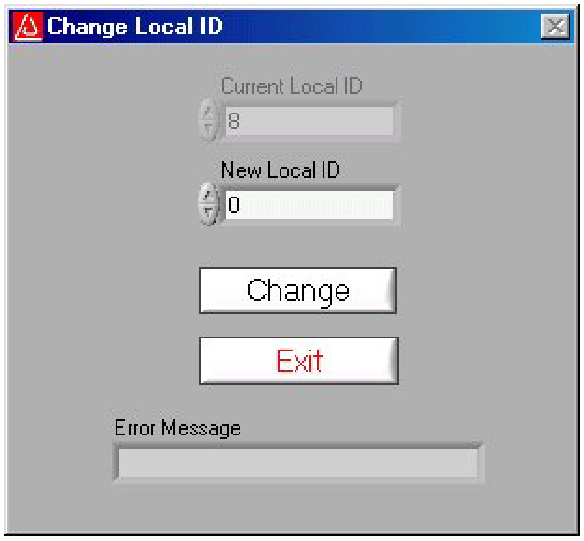
Serial #	Local ID	Change
100000046	8	●
0	0	●
0	0	●
0	0	●
0	0	●
0	0	●
0	0	●
0	0	●
0	0	●
0	0	●

numberFound
1

Error Message

Figure 3-1. LJconfig

Figure 3-1 shows the window that opens when LJconfig is run. Each time the “Refresh” button is pushed, LJconfig will scan the USB for all LabJacks. To change the local ID of a particular LabJack, push the “Change” button next to that LabJack, and the window shown in Figure 3-2 will appear.


Change Local ID

Current Local ID: 8

New Local ID: 0

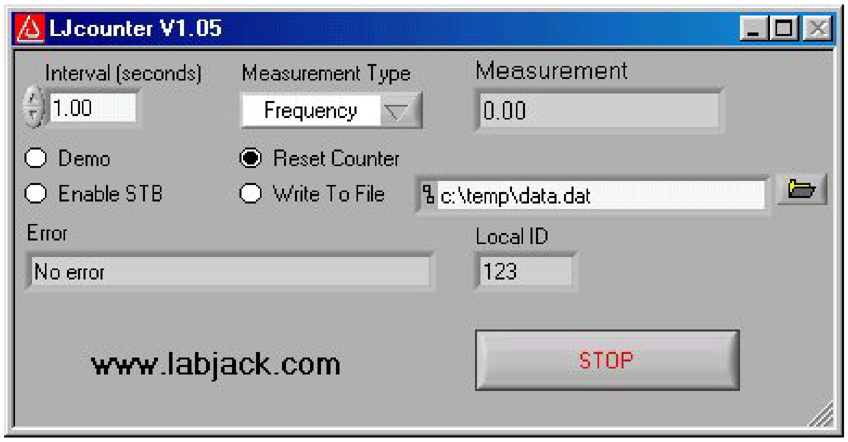
Error Message

Figure 3-2. LJconfig Change Local ID

Enter a new local ID between 0 and 255 and push the “Change” button. The new local ID will be written and the LabJack will be forced to re-enumerate.

3.2 LJcounter

Reads the LabJack counter and provides the current frequency or count.


LJcounter V1.05

Interval (seconds): 1.00

Measurement Type: Frequency

Measurement: 0.00

( ) Demo

(●) Reset Counter

( ) Enable STB

( ) Write To File: c:\temp\data.dat

Error: No error

Local ID: 123

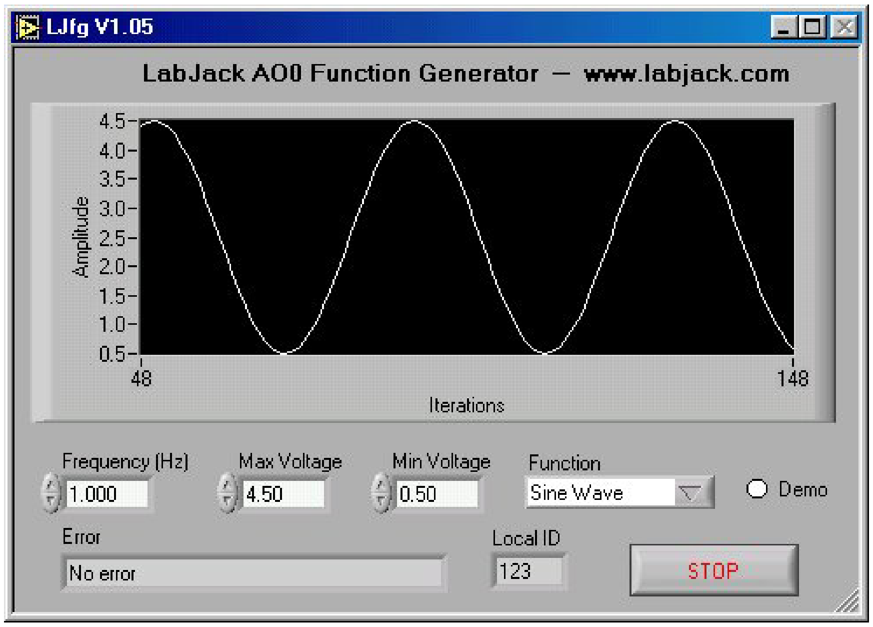
Figure 3-3. LJcounter

Figure 3-3 shows the LJcounter window:

- **Interval (seconds):** Specifies the interval, in seconds, between calls to the DLL function “Counter”.
- **Measurement Type:** If set to “Frequency”, this application divides the count by the interval to determine frequency in Hertz, and automatically resets the counter every read. If set to “Count”, the measurement is simply the current reading from the counter.
- **Measurement:** Displays frequency or count, depending on “Measurement Type”.

3.3 LJfg

This application allows the LabJack U12 to be used as a simple function generator. The DLL function “AOUpdate” is called every 25 milliseconds providing an update rate of 40 Hz, and thus a maximum reasonable signal frequency of a few Hertz.


LJfg V1.05

LabJack AO0 Function Generator — www.labjack.com

Frequency (Hz): 1.000

Max Voltage: 4.50

Min Voltage: 0.50

Function: Sine Wave

Demo

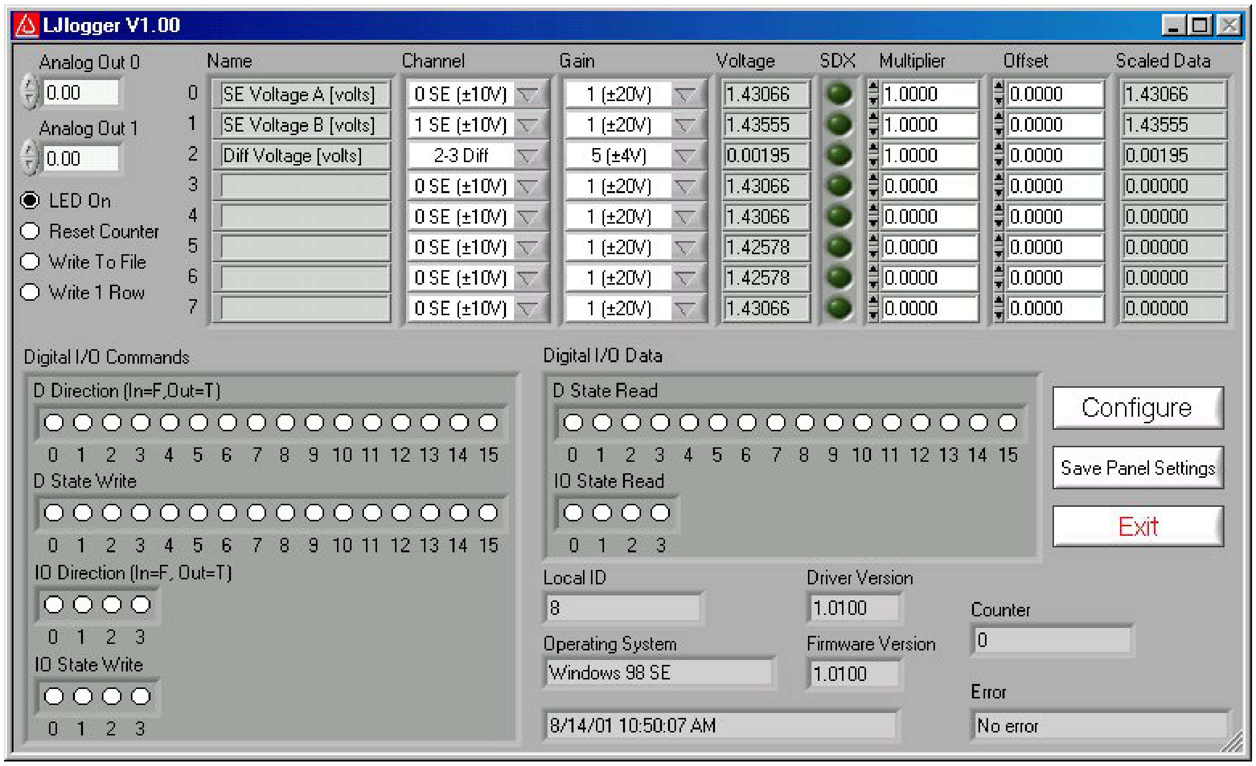
Error: No error

Local ID: 123

Figure 3-4. LJfg

3.4 LJlogger

LJlogger sends and receives data in command/response mode by making 2 DLL calls to "AISample" and 1 DLL call to "AOUpdate". It is capable of saving data to disk, writing data to an HTML page on the Internet, and performing various actions (including email) on trigger events.


LJlogger V1.00

Analog Out 0: 0.00

Analog Out 1: 0.00

[✓ LED On] Reset Counter Write To File Write 1 Row

	Name	Channel	Gain	Voltage	SDX	Multiplier	Offset	Scaled Data
0	SE Voltage A [volts]	0 SE (±10V)	1 (±20V)	1.43066	●	1.0000	0.0000	1.43066
1	SE Voltage B [volts]	1 SE (±10V)	1 (±20V)	1.43555	●	1.0000	0.0000	1.43555
2	Diff Voltage [volts]	2-3 Diff	5 (±4V)	0.00195	●	1.0000	0.0000	0.00195
3		0 SE (±10V)	1 (±20V)	1.43066	●	0.0000	0.0000	0.00000
4		0 SE (±10V)	1 (±20V)	1.43066	●	0.0000	0.0000	0.00000
5		0 SE (±10V)	1 (±20V)	1.42578	●	0.0000	0.0000	0.00000
6		0 SE (±10V)	1 (±20V)	1.42578	●	0.0000	0.0000	0.00000
7		0 SE (±10V)	1 (±20V)	1.43066	●	0.0000	0.0000	0.00000

Digital I/O Commands

D Direction (In=F,Out=T)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

D State Write
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

IO Direction (In=F, Out=T)
0 1 2 3

IO State Write
0 1 2 3

Digital I/O Data

D State Read
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

IO State Read
0 1 2 3

Local ID: 8

Driver Version: 1.0100

Operating System: Windows 98 SE

Firmware Version: 1.0100

Counter: 0

8/14/01 10:50:07 AM

Error: No error

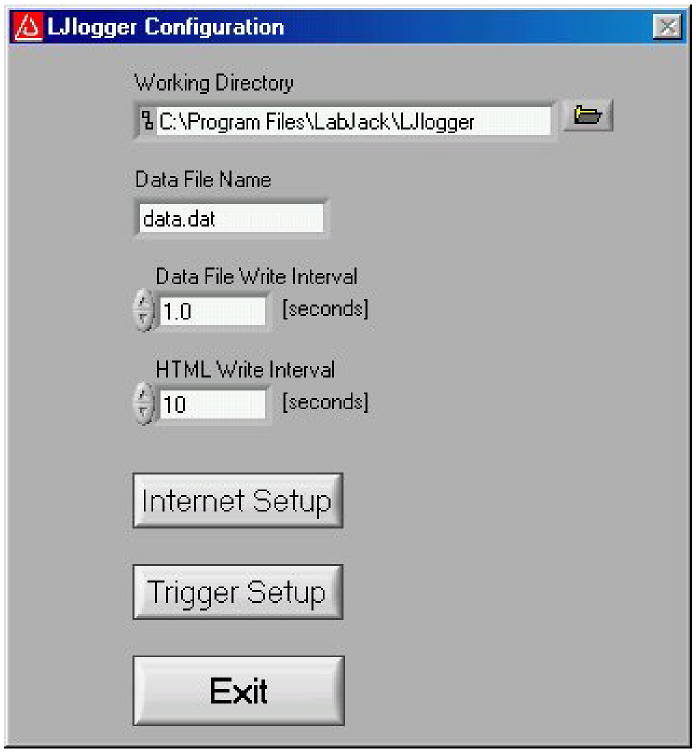
Figure 3-5. LJlogger

The main window for LJlogger is shown in Figure 3-5. The white colored items and the "SDX" buttons are controls to be edited/selected by the user. The grey colored items are indicators which display various information about the LabJack. Clicking the button labeled “Save Panel Settings” will save the current values of the controls as the default values.

If SDX is activated for a given analog input, the corresponding SDX DLL will be used to determine the scaled data. Users can make their own SDX DLLs (see the source code for more information), to provide more complex scaling or scaling that depends on other analog inputs.

Clicking the “Configure” button shown in Figure 3-5 brings up the window shown in Figure 3-6:

- **Working Directory:** This is the directory where data and configuration files will be written.
- **Data File Name:** Determines the name of the data file to which data will be written. New data is appended to the end of this file.
- **Data File Write Interval:** Determines the interval at which a new row of data will be written to the data file. Minimum of 0.5 seconds.
- **HTML Write Interval:** Determines the interval at which the HTML file is rewritten.


LJlogger Configuration

Working Directory: C:\Program Files\LabJack\LJlogger

Data File Name: data.dat

Data File Write Interval: 1.0 [seconds]

HTML Write Interval: 10 [seconds]

Figure 3-6. LJlogger Configuration

Clicking on the “Internet Setup” button in Figure 3-6 brings up the Internet configuration window shown in Figure 3-7. Basic customization of the HTML file can be done by clicking on “Advanced HTML Configuration” which brings up Figure 3-8.

Clicking on the “Trigger Setup” button in Figure 3-6 brings up the window shown in Figure 3-9.

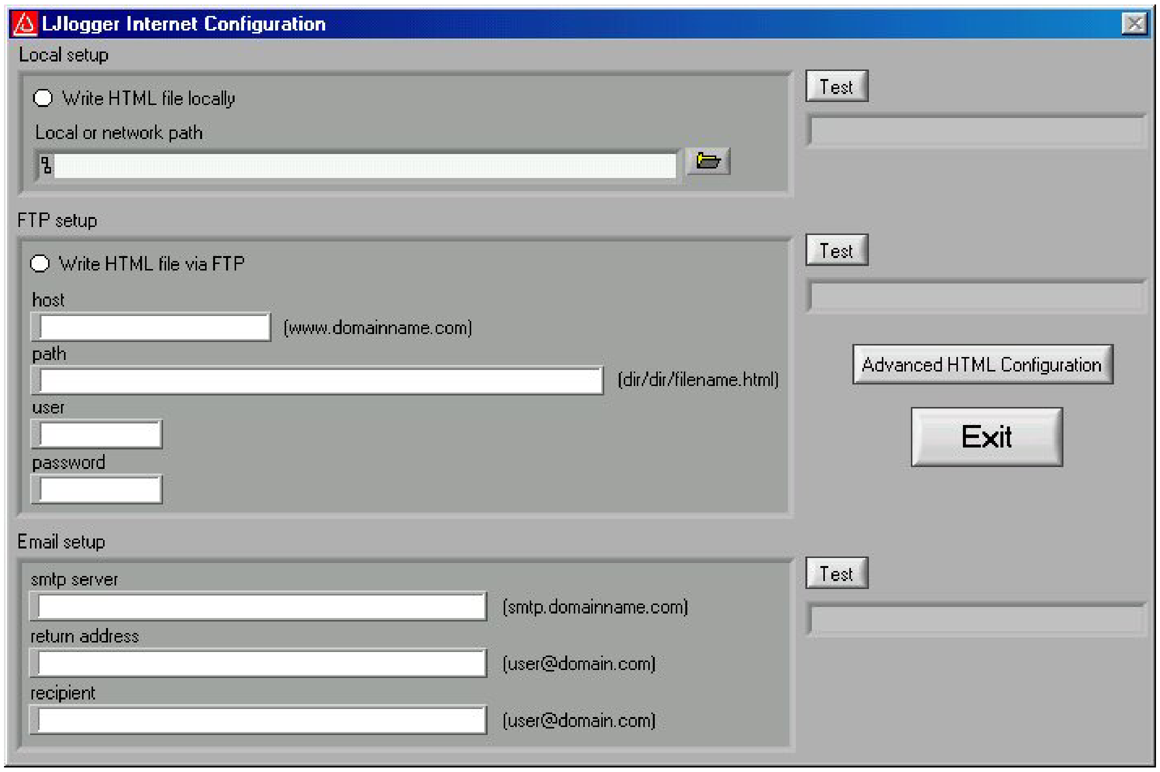


Figure 3-7. LJlogger Internet Configuration

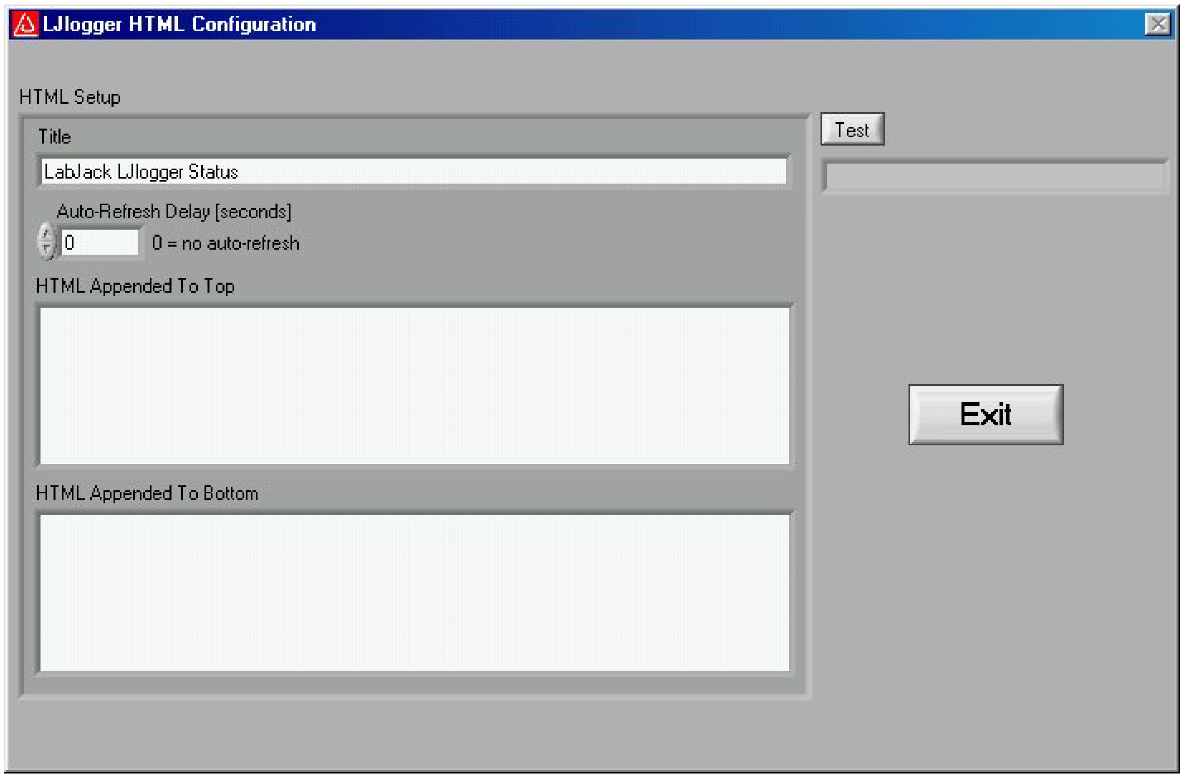
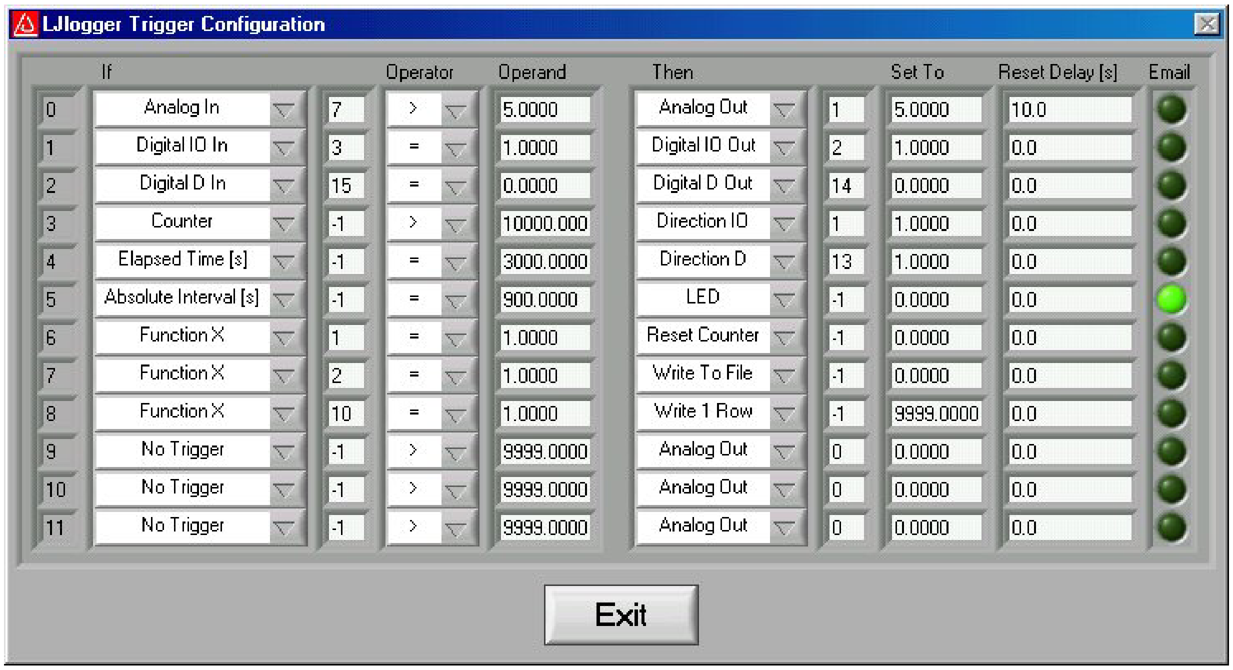


Figure 3-8. LJlogger HTML Configuration



	If		Operator	Operand	Then		Set To	Reset Delay [s]	Email
0	Analog In	7	>	5.0000	Analog Out	1	5.0000	10.0	●
1	Digital IO In	3	=	1.0000	Digital IO Out	2	1.0000	0.0	●
2	Digital D In	15	=	0.0000	Digital D Out	14	0.0000	0.0	●
3	Counter	-1	>	10000.000	Direction IO	1	1.0000	0.0	●
4	Elapsed Time [s]	-1	=	3000.0000	Direction D	13	1.0000	0.0	●
5	Absolute Interval [s]	-1	=	900.0000	LED	-1	0.0000	0.0	🟢
6	Function X	1	=	1.0000	Reset Counter	-1	0.0000	0.0	●
7	Function X	2	=	1.0000	Write To File	-1	0.0000	0.0	●
8	Function X	10	=	1.0000	Write 1 Row	-1	9999.0000	0.0	●
9	No Trigger	-1	>	9999.0000	Analog Out	0	0.0000	0.0	●
10	No Trigger	-1	>	9999.0000	Analog Out	0	0.0000	0.0	●
11	No Trigger	-1	>	9999.0000	Analog Out	0	0.0000	0.0	●

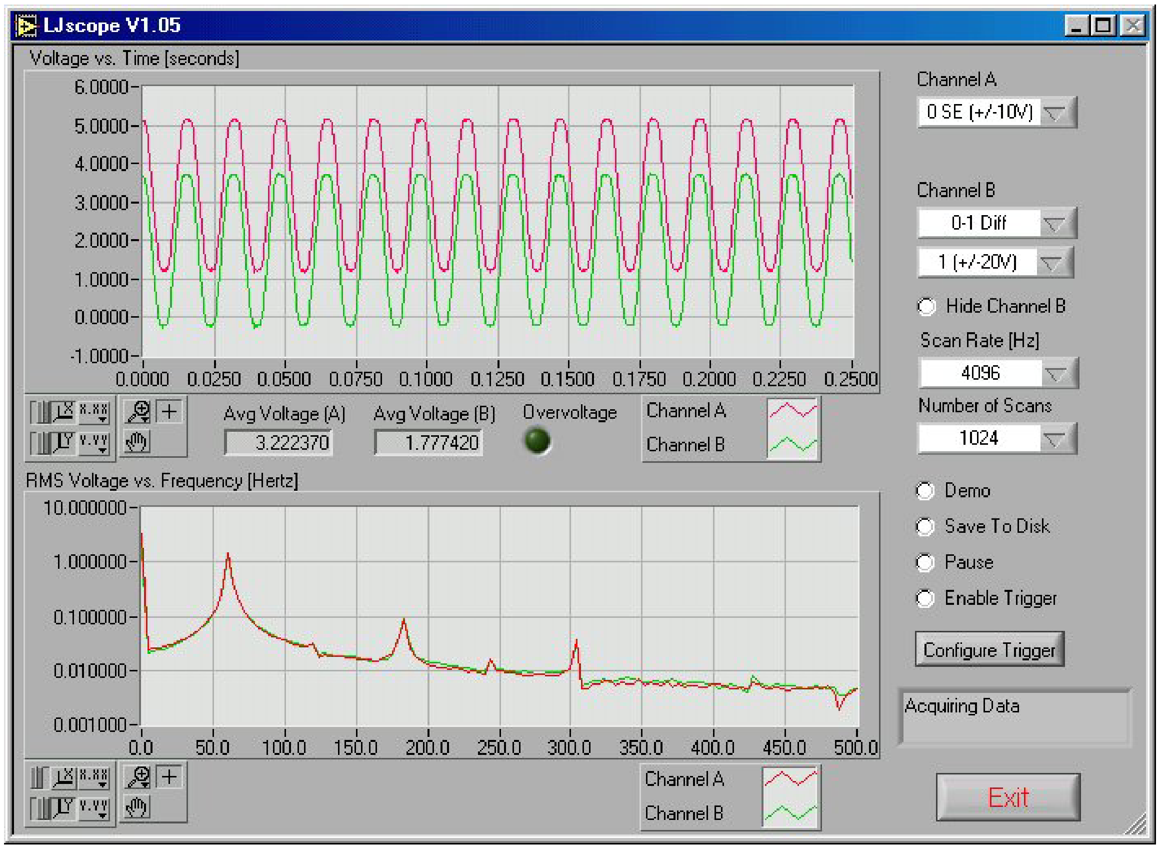
Figure 3-9. LJlogger Trigger Configuration

Figure 3-9 shows 9 example triggers:

- **Trigger #0:** If the scaled data from analog input row 7 (Figure 3-5) is greater than 5, then set AO1 to 5 volts. Once triggered, there is a 10 second delay before it can be triggered again.
- **Trigger #1:** If IO3 is high, set IO2 high. Reset delay is zero so this trigger can occur every iteration (every 0.1 seconds) if IO3 is high.
- **Trigger #2:** If D15 is low, set D14 low.
- **Trigger #3:** If the count is greater than 10,000, set IO1 to an output.
- **Trigger #4:** If it has been 3000 seconds since LJlogger started, set D13 to an output.
- **Trigger #5:** When the PC’s clock is at 15 minute intervals, the status LED will be turned off and an email will be sent.
- **Trigger #6:** Calls FunctionX from function1.dll. If the function returns True, reset the counter. Users can make their own FunctionX DLLs. See the source code for more information.
- **Trigger #7:** Calls FunctionX from function2.dll. If the function returns True, stop writing data to file.
- **Trigger #8:** Calls FunctionX from function10.dll. If the function returns True, write 1 row to the data file.

3.5 LJscope

LJscope simulates an oscilloscope by reading data from 2 analog input channels in burst mode.



LJscope V1.05

Channel A: 0 SE (+/-10V)

Channel B: 0-1 Diff, 1 (+/-20V)

Scan Rate [Hz]: 4096

Number of Scans: 1024

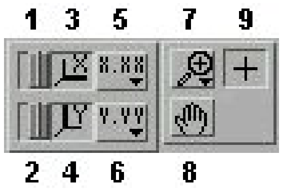
Avg Voltage (A): 3.222370

Avg Voltage (B): 1.777420

Acquiring Data

Figure 3-10. LJscope

There are two graphs on the LJscope main window (Figure 3-10), which show voltage versus time and voltage versus frequency. Both graphs have a palette to control various features such as autoscaling and zooming:



1. When you press this button it locks button 3 on (autoscale) position.
2. When you press this button it locks button 4 on (autoscale) position.
3. Pressing this button autoscales the x-axis.
4. Pressing this button autoscales the y-axis.
5. Miscellaneous x-axis formatting options.
6. Miscellaneous y-axis formatting options.
7. Zooming tool. Press this button to see different options for zooming. When collecting data, zooming will not work well unless autoscaling is off.
8. Panning tool. Allows you to drag and scroll around the graph.
9. Not applicable.

Other LJscope controls include:

- **Channel A/B:** Select the two AI channels that will be acquired. If a differential channel, is selected, the gain selection control will appear.
- **Hide Channel B:** When selected channel B will not be shown on the graph.
- **Scan Rate [Hz]:** (256 to 4096) Determines the scans/second for both channels.
- **Number of Scans:** (32 to 2048) Determines the number of scans that will be collected, and thus the total acquisition period. For example, if 1024 scans are collected at 4096 Hz, a quarter second of data will be collected (as shown in Figure 3-10).
- **Demo:** Calls “AIBurst” in demo mode so timing and data is simulated.
- **Save To Disk:** If selected a prompt will appear for a filename, and the current burst of data is saved to a tab-delimited file (time, channel A, channel B).
- **Pause:** Pauses data acquisition.
- **Enable Trigger:** Enables the IO trigger.
- **Configure Trigger:** Brings up the window shown in Figure 3-11. Choose the IO line to trigger on and whether to trigger when it is high or low. Also set the timeout period so the application will continue (with an error) if the trigger is not detected.

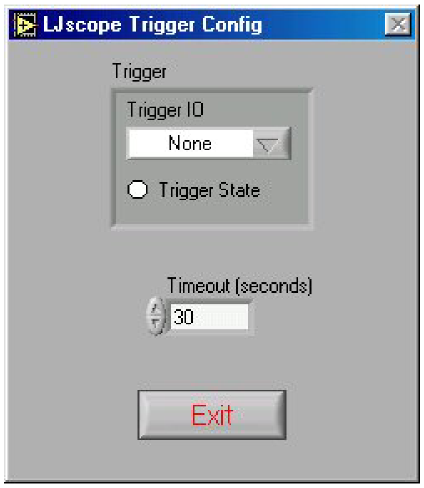
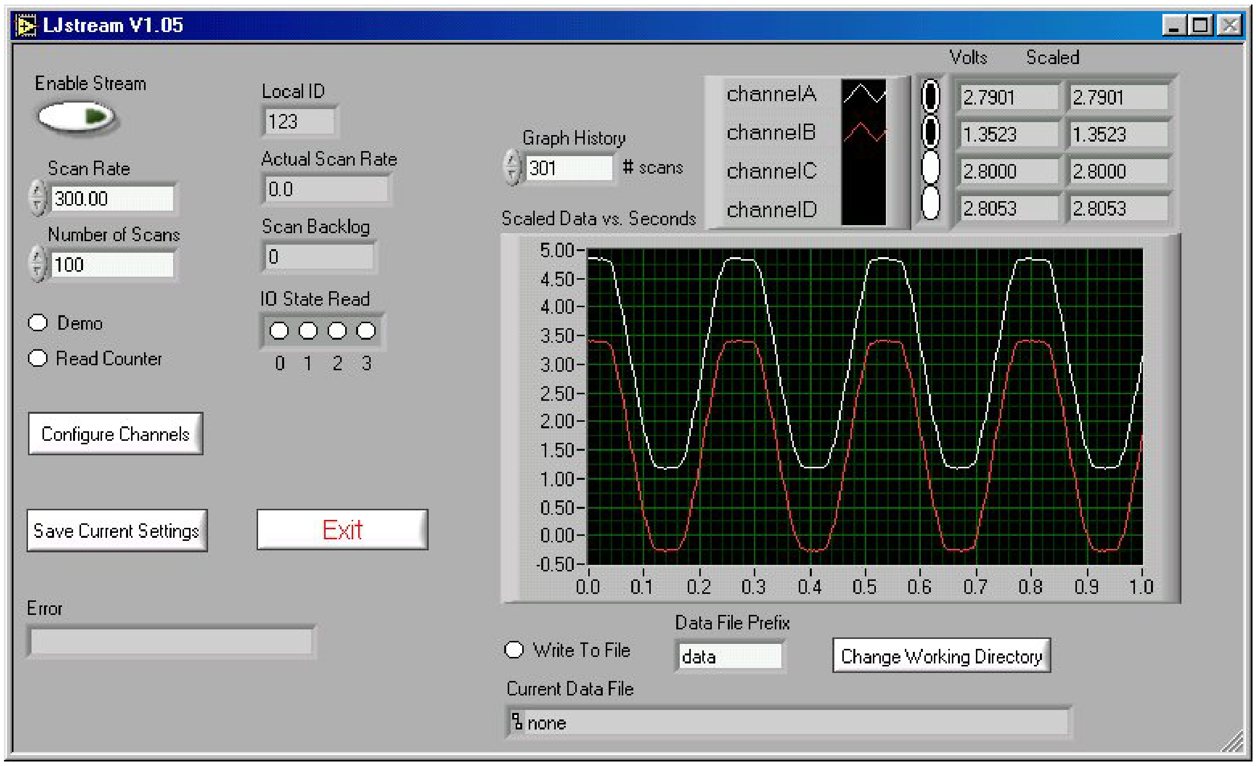


Figure 3-11. LJscope

3.6 LJstream

Uses stream mode to read, graph, and write to file, 4 AI channels. For more information, read about the stream functions (AIStreamStart, AIStreamRead, and AIStreamClear).

- **Enable Stream:** Starts and stops the stream acquisition.
- **Scan Rate:** Determines the scans/second (50 to 300).
- **Number of Scans:** Determines the number of scans that will be collected each iteration, and thus determines how fast this application iterates.
- **Demo:** Calls the “AIStream” functions in demo mode so timing and data is simulated.
- **Read Counter:** Collects 1 analog input and the counter if selected.
- **Configure Channels:** Click this button to bring up the channel configuration window shown in Figure 3-13.
- **Save Current Settings:** Saves all the current settings, including channel configuration.
- **Graph History:** Determines how much past history appears on the graph.


LJstream V1.05

Enable Stream

Scan Rate: 300.00

Number of Scans: 100

Local ID: 123

Actual Scan Rate: 0.0

Scan Backlog: 0

IO State Read: 0 1 2 3

Demo

Read Counter

Graph History: 301 # scans

	Volts	Scaled
channelA	2.7901	2.7901
channelB	1.3523	1.3523
channelC	2.8000	2.8000
channelD	2.8053	2.8053

Error

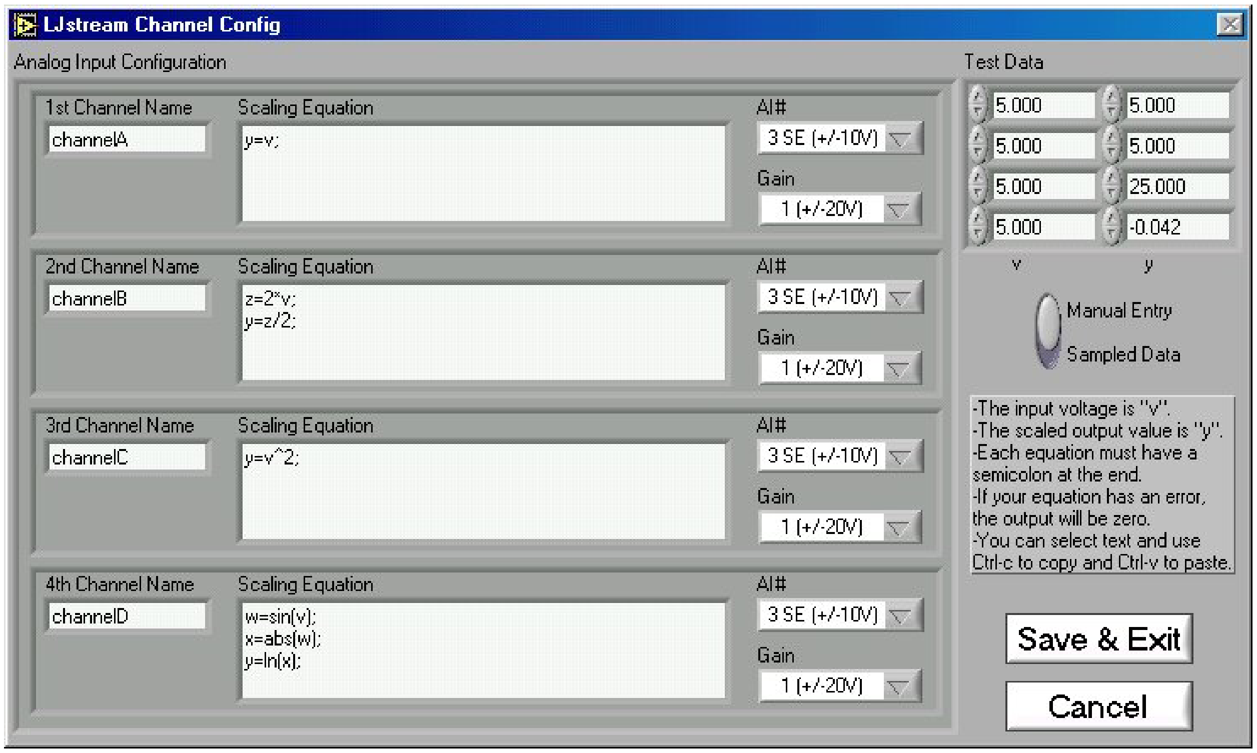
Write To File

Data File Prefix: data

Current Data File: none

Figure 3-12. LJstream

Figure 3-13 shows the LJstream channel configuration window. Here you can select analog inputs and gains and enter scaling equations. Use “Test Data” to see the effect of the scaling equations (“v” column is the measured voltage and the “y” column is the output of the scaling equations). “Manual/Sampled” determines where the “Test Data” in the “v” column originates.


LJstream Channel Config

Analog Input Configuration

Channel Name	Scaling Equation	AI#	Gain
1st Channel Name: channelA	y=v;	3 SE (+/-10V)	1 (+/-20V)
2nd Channel Name: channelB	z=2*v; y=z/2;	3 SE (+/-10V)	1 (+/-20V)
3rd Channel Name: channelC	y=v^2;	3 SE (+/-10V)	1 (+/-20V)
4th Channel Name: channelD	w=sin(v); x=abs(w); y=ln(x);	3 SE (+/-10V)	1 (+/-20V)

Test Data

v	y
5.000	5.000
5.000	5.000
5.000	25.000
5.000	-0.042

Manual Entry

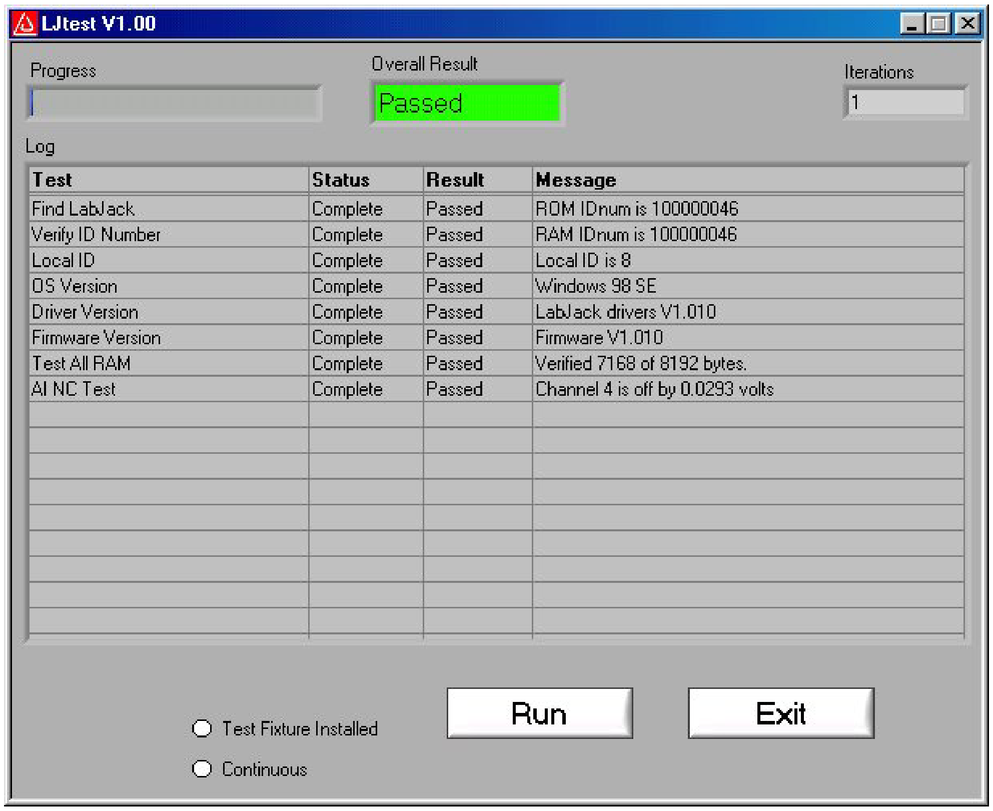
Sampled Data

-The input voltage is "v".
-The scaled output value is "y".
-Each equation must have a semicolon at the end.
-If your equation has an error, the output will be zero.
-You can select text and use Ctrl-c to copy and Ctrl-v to paste.

Figure 3-13. LJstream Channel Configuration

3.7 LJtest

LJtest runs a sequence of tests on the LabJack itself. Users will generally leave “Test Fixture Installed” unselected and execute the tests with nothing connected to the LabJack (except the USB of course).



Test	Status	Result	Message
Find LabJack	Complete	Passed	ROM IDnum is 100000046
Verify ID Number	Complete	Passed	RAM IDnum is 100000046
Local ID	Complete	Passed	Local ID is 8
OS Version	Complete	Passed	Windows 98 SE
Driver Version	Complete	Passed	LabJack drivers V1.010
Firmware Version	Complete	Passed	Firmware V1.010
Test All RAM	Complete	Passed	Verified 7168 of 8192 bytes.
AI NC Test	Complete	Passed	Channel 4 is off by 0.0293 volts

Figure 3-14. LJtest

4. Programming Reference

The LabJack U12 CD installs high-level drivers (ljackuw.dll), an ActiveX interface to the high-level drivers (ljackuwx.ocx), and LabVIEW6 VIs which call all the DLL functions. The DLL and OCX are installed in the Windows System directory. If the installation program can determine the LabVIEW6 directory, it copies the LabVIEW VIs into that directory (\vi.lib\addons\) so they show up on the function palette. Otherwise, the LabVIEW drivers are copied into the LabJack installation directory (c:\Program Files\LabJack)\drivers\labview, and can manually be transferred to the LabVIEW directory.

There are 27 functions exported by the LabJack DLL., and matching functions in the OCX and LabVIEW VIs. There are two additional support functions in the OCX, provided due to the limitations of ActiveX. All functions are command/response except for AIBurst and AIStreamStart/Read/Clear.

There are 2 parameters that are used by most functions:

- **errorcode –** A LabJack specific numeric error code. 0 means no error and 2 means no LabJacks were found. Use the function "GetErrorString" to get a description of the error, or see the list in Section 4.24 of this document.
- **idnum –** Functions with this input take either a local ID, serial number, or -1. A local ID or serial number will specify a specific LabJack, while –1 means the first found LabJack. Every LabJack has a local ID and serial number. The local ID is a value between 0 and 255 that can be changed by the user. The serial number is a value between 256 and 2,147,483,647 that is unique among all LabJacks and cannot be changed by the user.

4.1 EAnalogIn

Easy function. This is a simplified version of AISample. Reads the voltage from 1 analog input. Calling this function turns/leaves the status LED on. Execution time for this function is 20 milliseconds or less.

Declaration:

```
long EAnalogIn (      long *idnum,
                      long demo,
                      long channel,
                      long gain,
                      long *overVoltage,
                      float *voltage )
```

Parameter Description:

Returns: LabJack errorcodes or 0 for no error.

Inputs:

- ***idnum –** Local ID, serial number, or -1 for first found.
- **demo –** Send 0 for normal operation, >0 for demo mode. Demo mode allows this function to be called without a LabJack.
- **channel –** Channel command is 0-7 for single-ended, or 8-11 for differential.
- **gain –** Gain command is 0=1, 1=2, …, 7=20. This amplification is only available for differential channels.

Outputs:

- ***idnum –** Returns the local ID or –1 if no LabJack is found.
- ***overVoltage –** If >0, an overvoltage has been detected on one of the selected analog inputs.
- ***voltage –** Returns the voltage reading.

4.2 EAnalogOut

Easy function. This is a simplified version of AOUpdate. Sets the voltage of both analog outputs. Execution time for this function is 20 milliseconds or less.

Declaration:
```
long EAnalogOut (     long *idnum,
                      long demo,
                      float analogOut0,
                      float analogOut1 )
```

Parameter Description:
Returns: LabJack errorcodes or 0 for no error.
Inputs:
- ***idnum –** Local ID, serial number, or -1 for first found.
- **demo –** Send 0 for normal operation, >0 for demo mode. Demo mode allows this function to be called without a LabJack.
- **analogOut0 –** Voltage from 0.0 to 5.0 for AO0.
- **analogOut1 –** Voltage from 0.0 to 5.0 for AO1.

Outputs:
- ***idnum –** Returns the local ID or –1 if no LabJack is found.

4.3 ECount

Easy function. This is a simplified version of Counter. Reads & resets the counter (CNT). Calling this function disables STB (which is the default anyway). Execution time for this function is 20 milliseconds or less.

Declaration:
```
long ECount(  long *idnum,
              long demo,
              long resetCounter,
              double *count,
              double *ms )
```

Parameter Description:
Returns: LabJack errorcodes or 0 for no error.
Inputs:
- ***idnum –** Local ID, serial number, or -1 for first found.
- **demo –** Send 0 for normal operation, >0 for demo mode. Demo mode allows this function to be called without a LabJack.
- **resetCounter –** If >0, the counter is reset to zero after being read.

Outputs:
- ***idnum –** Returns the local ID or –1 if no LabJack is found.
- ***count –** Current count, before reset.
- ***ms –** Value of Window's millisecond timer at the time of the counter read (within a few ms). Note that the millisecond timer rolls over about every 50 days. In general, the millisecond timer starts counting from zero whenever the computer reboots.

4.4 EDigitalIn

Easy function. This is a simplified version of DigitalIO that reads the state of one digital input. Also configures the requested pin to input and leaves it that way. Execution time for this function is 20 milliseconds or less.

Note that this is a simplified version of the lower level function DigitalIO, which operates on all 20 digital lines. The DLL (ljackuw) attempts to keep track of the current direction and output state of all lines, so that this easy function can operate on a single line without changing the others. When the DLL is first loaded, though, it does not know the direction and state of the lines and assumes all directions are input and output states are low.

Declaration:

```
long EAnalogIn (     long *idnum,
                     long demo,
                     long channel,
                     long readD,
                     long *state )
```

Parameter Description:

Returns: LabJack errorcodes or 0 for no error.

Inputs:

- ***idnum –** Local ID, serial number, or -1 for first found.
- **demo –** Send 0 for normal operation, >0 for demo mode. Demo mode allows this function to be called without a LabJack.
- **channel –** Line to read. 0-3 for IO or 0-15 for D.
- **readD –** If >0, a D line is read as opposed to an IO line.

Outputs:

- ***idnum –** Returns the local ID or –1 if no LabJack is found.
- ***state –** The selected line is TRUE/Set if >0. FALSE/Clear if 0.

4.5 EDigitalOut

Easy function. This is a simplified version of DigitalIO that sets/clears the state of one digital output. Also configures the requested pin to output and leaves it that way. Execution time for this function is 20 milliseconds or less.

Note that this is a simplified version of the lower level function DigitalIO, which operates on all 20 digital lines. The DLL (ljackuw) attempts to keep track of the current direction and output state of all lines, so that this easy function can operate on a single line without changing the others. When the DLL is first loaded, though, it does not know the direction and state of the lines and assumes all directions are input and output states are low.

Declaration:

```
long EAnalogIn (     long *idnum,
                     long demo,
                     long channel,
                     long writeD,
                     long state )
```

Parameter Description:

Returns: LabJack errorcodes or 0 for no error.

Inputs:

- ***idnum –** Local ID, serial number, or -1 for first found.
- **demo –** Send 0 for normal operation, >0 for demo mode. Demo mode allows this function to be called without a LabJack.
- **channel –** Line to read. 0-3 for IO or 0-15 for D.
- **writeD –** If >0, a D line is written as opposed to an IO line.

- **state –** If >0, the line is set, otherwise the line is cleared.

Outputs:

- ***idnum –** Returns the local ID or –1 if no LabJack is found.

4.6 AISample

Reads the voltages from 1,2, or 4 analog inputs. Also controls/reads the 4 IO ports. Execution time for this function is 20 milliseconds or less.

Declaration:

```
long AISample (      long *idnum,
                     long demo,
                     long *stateIO,
                     long updateIO,
                     long ledOn,
                     long numChannels,
                     long *channels,
                     long *gains,
                     long disableCal,
                     long *overVoltage,
                     float *voltages )
```

Parameter Description:

Returns: LabJack errorcodes or 0 for no error.

Inputs:

- ***idnum –** Local ID, serial number, or -1 for first found.
- **demo –** Send 0 for normal operation, >0 for demo mode. Demo mode allows this function to be called without a LabJack.
- ***stateIO –** Output states for IO0-IO3.
- **updateIO –** If >0, state values will be written. Otherwise, just a read is performed.
- **ledOn –** If >0, the LabJack LED is turned on.
- **numChannels –** Number of analog input channels to read (1,2, or 4).
- ***channels –** Pointer to an array of channel commands with at least numChannels elements. Each channel command is 0-7 for single-ended, or 8-11 for differential.
- ***gains –** Pointer to an array of gain commands with at least numChannels elements. Gain commands are 0=1, 1=2, …, 7=20. This amplification is only available for differential channels.
- **disableCal –** If >0, voltages returned will be raw readings that are not corrected using calibration constants.
- ***voltages –** Pointer to an array where voltage readings are returned. Send a 4-element array of zeros.

Outputs:

- ***idnum –** Returns the local ID or –1 if no LabJack is found.
- ***stateIO –** Returns input states of IO0-IO3.
- ***overVoltage –** If >0, an overvoltage has been detected on one of the selected analog inputs.
- ***voltages –** Pointer to an array where numChannels voltage readings are returned.

ActiveX Function Differences:

The “channels” and “gains” arrays are replaced with “channelsPacked” and “gainsPacked”. The OCX has a function “FourPack” which will convert 4 elements to a packed value. The packed value is determined as: element[0] + (element[1] * 2^8) + (element[2] * 2^16) + (element[3] * 2^24).

The “voltages” array is replaced with 4 individual parameters.

Declaration (ActiveX):

```
long AISampleX (    long FAR* idnum,
                    long demo,
                    long FAR* stateIO,
                    long updateIO,
                    long ledOn,
                    long numChannels,
                    long channelsPacked,
                    long gainsPacked,
                    long disableCal,
                    long FAR* overVoltage,
                    float FAR* voltageA,
                    float FAR* voltageB,
                    float FAR* voltageC,
                    float FAR* voltageD )
```

4.7 AIBurst

Reads a specified number of scans (up to 4096) at a specified scan rate (up to 8192 Hz) from 1,2, or 4 analog inputs. First, data is acquired and stored in the LabJack’s 4096 sample RAM buffer. Then, the data is transferred to the PC.

If the LED is enabled (ledOn>0), it will blink at about 4 Hz while waiting for a trigger, turn off during acquisition, blink rapidly while transferring data to the PC, and turn on when done.

Declaration:

```
long AIBurst (  long *idnum,
                long demo,
                long stateIOin,
                long updateIO,
                long ledOn,
                long numChannels,
                long *channels,
                long *gains,
                float *scanRate,
                long disableCal,
                long triggerIO,
                long triggerState,
                long numScans,
                long timeout,
                float (*voltages)[4],
                long *stateIOout,
                long *overVoltage,
                long transferMode )
```

Parameter Description:

Returns:	LabJack errorcodes or 0 for no error.

Inputs:

- ***idnum –** Local ID, serial number, or -1 for first found.
- **demo –** Send 0 for normal operation, >0 for demo mode. Demo mode allows this function to be called without a LabJack.
- ***stateIOin –** Output states for IO0-IO3.
- **updateIO –** If >0, state values will be written. Otherwise, just a read is performed.
- **ledOn –** If >0, the LabJack LED is turned on.
- **numChannels –** Number of analog input channels to read (1,2, or 4).
- ***channels –** Pointer to an array of channel commands with at least numChannels elements. Each channel command is 0-7 for single-ended, or 8-11 for differential.
- ***gains –** Pointer to an array of gain commands with at least numChannels elements. Gain commands are 0=1, 1=2, …, 7=20. This amplification is only available for differential channels.
- ***scanRate –** Scans acquired per second. A scan is a reading from every channel (1,2, or 4). The sample rate (scanRate * numChannels) must be between 400 and 8192.
- **disableCal –** If >0, voltages returned will be raw readings that are not corrected using calibration constants.
- **triggerIO –** The IO port to trigger on (0=none, 1=IO0, …,4=IO3).
- **triggerState –** If >0, the acquisition will be triggered when the selected IO port reads high.
- **numScans –** Number of scans which will be returned. Minimum is 1. Maximum numSamples is 4096, where numSamples is numScans * numChannels.
- **timeout –** This function will return immediately with a timeout error if it does not receive a scan within this number of seconds.
- ***voltages –** Pointer to a 4096 by 4 array where voltage readings are returned. Send filled with zeros.
- ***stateIOout –** Pointer to a 4096 element array where IO states are returned. Send filled with zeros.
- **transferMode –**Always send 0.

Outputs:

- ***idnum –** Returns the local ID or –1 if no LabJack is found.
- ***scanRate –** Returns the actual scan rate, which due to clock resolution is not always exactly the same as the desired scan rate.
- ***voltages –** Pointer to a 4096 by 4 array where voltage readings are returned. Unused locations are filled with 9999.0.
- ***stateIOout –** Pointer to a 4096 element array where IO states are returned. Unused locations are filled with 9999.0.
- ***overVoltage –** If >0, an overvoltage has been detected on at least one sample of one of the selected analog inputs.

<u>ActiveX Function Differences:</u>
The “channels” and “gains” arrays are replaced with “channelsPacked” and “gainsPacked”. The OCX has a function “FourPack” (4.23) which will convert 4 elements to a packed value. The packed value is determined as: element[0] + (element[1] * 2^8) + (element[2] * 2^16) + (element[3] * 2^24).

The parameters “demo”, “ledOn”, “disableCal”, “transferMode”, “updateIO”, and “stateIOin”, are replaced by an “optionBits” parameter. Call the OCX function “BuildOptionBits” (4.22) to determine this parameter.

The “voltages” and “stateIOout” arrays are represented as strings. Floating point data is returned as 13 characters per number (XXXX.XXXXXXXX) and integers are returned as 10 characters per number (XXXXXXXXXX). Zeros are used for padding where necessary. The total number of bytes in the “voltages” string is 13*numSamples. The total number of bytes in the “stateIOout” string is 10*numScans.

Declaration (ActiveX):
```
long AIBurstX (          long FAR* idnum,
                         long numChannels,
                         long channelsPacked,
                         long gainsPacked,
                         float FAR* scanRate,
                         long triggerIO,
                         long triggerState,
                         long numScans,
                         long timeout,
                         BSTR FAR* voltages,
                         BSTR FAR* stateIOout,
                         long FAR* overVoltage,
                         long optionBits)
```

4.8 AIStreamStart

Starts a hardware timed continuous acquisition where data is sampled and stored in the LabJack RAM buffer, and can be simultaneously transferred out of the RAM buffer to the PC application. A call to this function should be followed by periodic calls to AIStreamRead, and eventually a call to AIStreamClear.

If the LED is enabled (ledOn>0), it will toggle every 40 samples during acquisition and turn on when the stream operation stops.

Declaration:
```
long AIStreamStart (  long *idnum,
                      long demo,
                      long stateIOin,
                      long updateIO,
                      long ledOn,
                      long numChannels,
                      long *channels,
                      long *gains,
                      float *scanRate,
                      long disableCal,
                      long reserved1,
                      long readCount )
```

Parameter Description:
Returns: LabJack errorcodes or 0 for no error.
Inputs:

- ***idnum –** Local ID, serial number, or -1 for first found.
- **demo –** Send 0 for normal operation, >0 for demo mode. Demo mode allows this function to be called without a LabJack.
- **stateIOin –** Output states for IO0-IO3.

- **updateIO –** If >0, state values will be written. Otherwise, just a read is performed.
- **ledOn –** If >0, the LabJack LED is turned on.
- **numChannels –** Number of analog input channels to read (1,2, or 4). If readCount is >0, numChannels should be 4.
- ***channels –** Pointer to an array of channel commands with at least numChannels elements. Each channel command is 0-7 for single-ended, or 8-11 for differential.
- ***gains –** Pointer to an array of gain commands with at least numChannels elements. Gain commands are 0=1, 1=2, …, 7=20. This amplification is only available for differential channels.
- ***scanRate –** Scans acquired per second. A scan is a reading from every channel (1,2, or 4). The sample rate (scanRate * numChannels) must be between 200 and 1200.
- **disableCal –** If >0, voltages returned will be raw readings that are not corrected using calibration constants.
- **reserved1 –** Reserved for future use. Send 0.
- **readCount –** If >0, the current count (CNT) is returned instead of the 2nd, 3rd, and 4th analog input channels. 2nd channel is bits 0-11. 3rd channel is bits 12-23. 4th channel is bits 24-31. This feature was added to the LabJack U12 starting with firmware version 1.03, and this input has no effect with earlier firmware versions.

Outputs:

- ***idnum –** Returns the local ID or –1 if no LabJack is found.
- ***scanRate –** Returns the actual scan rate, which due to clock resolution is not always exactly the same as the desired scan rate.

<u>ActiveX Function Differences:</u>
The “channels” and “gains” arrays are replaced with “channelsPacked” and “gainsPacked”. The OCX has a function “FourPack” (4.23) which will convert 4 elements to a packed value. The packed value is determined as: element[0] + (element[1] * 2^8) + (element[2] * 2^16) + (element[3] * 2^24).

The parameters “demo”, “ledOn”, “disableCal”, “updateIO”, and “stateIOin”, are replaced by an “optionBits” parameter. Call the OCX function “BuildOptionBits” (4.22) to determine this parameter.

<u>Declaration (ActiveX):</u>
```
long AIStreamStartX (        long FAR* idnum,
                             long numChannels,
                             long channelsPacked,
                             long gainsPacked,
                             float FAR* scanRate,
                             long optionBits,
                             long readCount)
```

4.9 AIStreamRead

Waits for a specified number of scans to be available and reads them. AIStreamStart should be called before this function and AIStreamClear should be called when finished with the stream.

<u>Declaration:</u>
```
long AIStreamRead (  long localID,
                     long numScans,
```

long timeout,
float (*voltages)[4],
long *stateIOout,
long *reserved,
long *ljScanBacklog,
long *overVoltage)

Parameter Description:

Returns: LabJack errorcodes or 0 for no error.

Inputs:

- **localID –** Send the local ID from AIStreamStart.
- **numScans –** Function will wait until this number of scans is available. Minimum is 1. Maximum numSamples is 4096, where numSamples is numScans * numChannels. Internally this function gets data from the LabJack in blocks of 64 samples, so it is recommended that numSamples be at least 64.
- **timeout –** Function timeout value in seconds.
- ***voltages –** Pointer to a 4096 by 4 array where voltage readings are returned. Send filled with zeros.
- ***stateIOout –** Pointer to a 4096 element array where IO states are returned. Send filled with zeros.

Outputs:

- ***voltages –** Pointer to a 4096 by 4 array where voltage readings are returned. Unused locations are filled with 9999.0.
- ***stateIOout –** Pointer to a 4096 element array where IO states are returned. Unused locations are filled with 9999.0.
- ***reserved –** Reserved for future use. Send 0.
- **ljScanBacklog –** Returns the scan backlog of the LabJack RAM buffer. The size of the buffer in terms of scans is 4096/numChannels.
- ***overVoltage –** If >0, an overvoltage has been detected on at least one sample of one of the selected analog inputs.

ActiveX Function Differences:

The “voltages” and “stateIOout” arrays are represented as strings. Floating point data is returned as 13 characters per number (XXXX.XXXXXXXX) and integers are returned as 10 characters per number (XXXXXXXXXX). Zeros are used for padding where necessary. . The total number of bytes in the “voltages” string is 13*numSamples. The total number of bytes in the “stateIOout” string is 10*numScans.

Declaration (ActiveX):

```
long AIStreamReadX (     long localID,
                         long numScans,
                         long timeout,
                         BSTR FAR* voltages,
                         BSTR FAR* stateIOout,
                         long FAR* ljScanBacklog,
                         long FAR* overVoltage)
```

4.10 AIStreamClear

This function stops the continuous acquisition. It should be called once when finished with the stream. The sequence of calls for a typical stream operation is: AIStreamStart, AIStreamRead, AIStreamRead, AIStreamRead, …, AIStreamClear.

Declaration:
long AIStreamClear (long localID)

Parameter Description:
Returns: LabJack errorcodes or 0 for no error.
Input:

- **localID –** Send the local ID from AIStreamStart/Read.

4.11 AOUpdate

Sets the voltages of the analog outputs. Also controls/reads all 20 digital I/O and the counter. Execution time for this function is 20 milliseconds or less.

Declaration:

```
long AOUpdate (      long *idnum,
                     long demo,
                     long trisD,
                     long trisIO,
                     long *stateD,
                     long *stateIO,
                     long updateDigital,
                     long resetCounter,
                     unsigned long *count,
                     float analogOut0,
                     float analogOut1)
```

Parameter Description:
Returns: LabJack errorcodes or 0 for no error.
Inputs:

- ***idnum –** Local ID, serial number, or -1 for first found.
- **demo –** Send 0 for normal operation, >0 for demo mode. Demo mode allows this function to be called without a LabJack.
- **trisD –** Directions for D0-D15. 0=Input, 1=Output.
- **trisIO –** Directions for IO0-IO3. 0=Input, 1=Output.
- ***stateD –** Output states for D0-D15.
- ***stateIO –** Output states for IO0-IO3.
- **updateDigital –** If >0, tris and state values will be written. Otherwise, just a read is performed.
- **resetCounter –** If >0, the counter is reset to zero after being read.
- **analogOut0 –** Voltage from 0.0 to 5.0 for AO0.
- **analogOut1 –** Voltage from 0.0 to 5.0 for AO1.

Outputs:

- ***idnum –** Returns the local ID or –1 if no LabJack is found.
- ***stateD –** States of D0-D15.
- ***stateIO –** States of IO0-IO3.
- ***count –** Current value of the 32-bit counter (CNT). This value is read before the counter is reset.

ActiveX Function Differences:
The counter read is returned as a double precision float, instead of an unsigned long.

Declaration (ActiveX):

```
long AOUpdateX (     long FAR* idnum,
                    long demo,
                    long trisD,
                    long trisIO,
                    long FAR* stateD,
                    long FAR* stateIO,
                    long updateDigital,
                    long resetCounter,
                    double FAR* count,
                    float analogOut0,
                    float analogOut1)
```

4.12 BitsToVolts

Converts a 12-bit (0-4095) binary value into a LabJack voltage.

Volts=((2*Bits*Vmax/4096)-Vmax)/Gain where Vmax=10 for SE, 20 for Diff.

Declaration:

```
long BitsToVolts (  long chnum,
                    long chgain,
                    long bits,
                    float *volts )
```

Parameter Description:

Returns: LabJack errorcodes or 0 for no error.

Inputs:

- **chnum –** Channel index. 0-7=SE, 8-11=Diff.
- **chgain –** Gain index. 0=1, 1=2, …, 7=20.
- **bits –** Binary value from 0-4095.

Outputs:

- ***volts –** Voltage.

4.13 VoltsToBits

Converts a voltage to it's 12-bit (0-4095) binary representation.

Bits=(4096*((Volts*Gain)+Vmax))/(2*Vmax) where Vmax=10 for SE, 20 for Diff.

Declaration:

```
long VoltsToBits (  long chnum,
                    long chgain,
                    float volts,
                    long *bits )
```

Parameter Description:

Returns: LabJack errorcodes or 0 for no error.

Inputs:

- **chnum –** Channel index. 0-7=SE, 8-11=Diff.
- **chgain –** Gain index. 0=1, 1=2, …, 7=20.
- **volts –** Voltage.

Outputs:

- ***bits –** Binary value from 0-4095.

4.14 Counter

Controls and reads the counter. The counter is disabled if the watchdog timer is enabled. Execution time for this function is 20 milliseconds or less.

Declaration:

```
long Counter ( long *idnum,
               long demo,
               long *stateD,
               long *stateIO,
               long resetCounter,
               long enableSTB,
               unsigned long *count )
```

Parameter Description:

Returns: LabJack errorcodes or 0 for no error.

Inputs:

- ***idnum –** Local ID, serial number, or -1 for first found.
- **demo –** Send 0 for normal operation, >0 for demo mode. Demo mode allows this function to be called without a LabJack.
- **resetCounter –** If >0, the counter is reset to zero after being read.
- **enableSTB –** If >0, STB is enabled. Used for testing and calibration. (This input has no effect with firmware V1.02 or earlier, in which case STB is always enabled)

Outputs:

- ***idnum –** Returns the local ID or –1 if no LabJack is found.
- ***stateD –** States of D0-D15.
- ***stateIO –** States of IO0-IO3.
- ***count –** Current value of the 32-bit counter (CNT). This value is read before the counter is reset.

4.15 DigitalIO

Reads and writes to all 20 digital I/O. Execution time for this function is 20 milliseconds or less.

Declaration:

```
long DigitalIO (long *idnum,
               long demo,
               long *trisD,
               long trisIO,
               long *stateD,
               long *stateIO,
               long updateDigital,
               long *outputD )
```

Parameter Description:

Returns: LabJack errorcodes or 0 for no error.

Inputs:

- ***idnum –** Local ID, serial number, or -1 for first found.
- **demo –** Send 0 for normal operation, >0 for demo mode. Demo mode allows this function to be called without a LabJack.
- ***trisD –** Directions for D0-D15. 0=Input, 1=Output.
- **trisIO –** Directions for IO0-IO3. 0=Input, 1=Output.
- ***stateD –** Output states for D0-D15.

- ***stateIO –** Output states for IO0-IO3.
- **updateDigital –** If >0, tris and state values will be written. Otherwise, just a read is performed.

Outputs:

- ***idnum –** Returns the local ID or –1 if no LabJack is found.
- ***trisD –** Returns a read of the direction registers for D0-D15.
- ***stateD –** States of D0-D15.
- ***stateIO –** States of IO0-IO3.
- ***outputD –** Returns a read of the output registers for D0-D15.

4.16 GetDriverVersion

Returns the version number of ljackuw.dll.

Declaration:
float GetDriverVersion (void)

Parameter Description:
Returns: Version number of ljackuw.dll.

ActiveX Function Differences:
Uses parameters to return DLL and OCX version.

Declaration (ActiveX):
```
void GetDriverVersionX (     float FAR* dllVersion,
                             float FAR* ocxVersion)
```

4.17 GetErrorString

Converts a LabJack errorcode, returned by another function, into a string describing the error.

Declaration:
```
void GetErrorString (  long errorcode,
                       char *errorString )
```

Parameter Description:
Returns: Nothing.
Inputs:

- **errorcode –** LabJack errorcode.
- ***errorString –** Pointer to a 50 element array of characters.

Outputs:

- ***errorString –** Pointer to a sequence of characters describing the error. Unused locations are filled with 0x00.

4.18 GetFirmwareVersion

Retrieves the firmware version from the LabJack's processor. Execution time for this function is 20 milliseconds or less.

Declaration:
float GetFirmwareVersion (long *idnum)

Parameter Description:
Returns: Version number of the LabJack firmware or 0 for error.

Inputs:

- ***idnum –** Local ID, serial number, or -1 for first found.

Outputs:

- ***idnum –** Returns the local ID or –1 if no LabJack is found.

4.19 GetWinVersion

Uses a Windows API function to get the OS version.

Declaration:

```
long GetWinVersion ( unsigned long *majorVersion,
                     unsigned long *minorVersion,
                     unsigned long *buildNumber,
                     unsigned long *platformID,
                     unsigned long *servicePackMajor,
                     unsigned long *servicePackMinor )
```

Parameter Description:

Returns: LabJack errorcodes or 0 for no error.

Outputs:

	Platform	Major	Minor	Build
Windows 3.1	0	-	-	-
Windows 95	1	4	0	950
Windows 95 OSR2	1	4	0	1111
Windows 98	1	4	10	1998
Windows 98SE	1	4	10	2222
Windows Me	1	4	90	3000
Windows NT 3.51	2	3	51	-
Windows NT 4.0	2	4	0	1381
Windows 2000	2	5	0	2195
Whistler	2	5	1	-

4.20 ListAll

Searches the USB for all LabJacks, and returns the serial number and local ID for each.

Declaration:

```
long ListAll (  long *productIDList,
                long *serialnumList,
                long *localIDList,
                long *powerList,
                long (*calMatrix)[20],
                long *numberFound,
                long *fcddMaxSize,
                long *hvcMaxSize )
```

Parameter Description:

Returns: LabJack errorcodes or 0 for no error.

Inputs:

- ***productIDList –** Pointer to a 127 element array. Send filled with zeros.
- ***serialnumList –** Pointer to a 127 element array. Send filled with zeros.
- ***localIDList –** Pointer to a 127 element array. Send filled with zeros.

- ***powerList –** Pointer to a 127 element array. Send filled with zeros.
- ***calMatrix –** Pointer to a 127 by 20 element array. Send filled with zeros.

Outputs:

- ***serialnumList –** Pointer to a 127 element array where serial numbers are returned. Unused locations are filled with 9999.0.
- ***localIDList –** Pointer to a 127 element array where local ID numbers are returned. Unused locations are filled with 9999.0.
- ***numberFound –** Number of LabJacks found on the USB.

ActiveX Function Differences:
The arrays are represented as strings with 10 characters per number (XXXXXXXXXX). Zeros are used for padding where necessary.

Declaration (ActiveX):
```
long ListAllX (          BSTR FAR* productIDList,
                         BSTR FAR* serialnumList,
                         BSTR FAR* localIDList,
                         BSTR FAR* powerList,
                         BSTR FAR* calMatrix,
                         long FAR* numberFound,
                         long FAR* fcddMaxSize,
                         long FAR* hvcMaxSize)
```

4.21 LocalID

Changes the local ID of a specified LabJack. Changes will not take effect until the LabJack is re-enumerated or reset, either manually by disconnecting and reconnecting the USB cable or by calling ReEnum or Reset.

Declaration:
```
long LocalID ( long *idnum,
               long localID )
```

Parameter Description:
Returns: LabJack errorcodes or 0 for no error.
Inputs:

- ***idnum –** Local ID, serial number, or -1 for first found.
- **localID –** New local ID.

Outputs:

- ***idnum –** Returns the local ID or –1 if no LabJack is found.

4.22 NoThread

This function is needed when interfacing TestPoint to the LabJack DLL on Windows 98/ME (see ljackuw.h for more information). Call this function to disable/enable thread creation for other functions. Normally, thread creation should be enabled, but it must be disabled for LabJack functions to work when called from TestPoint. One other situation where disabling thread creation might be useful, is when running a time-critical application in the Visual C debugger. Slow thread creation is a known problem with the Visual C debugger.

If the read thread is disabled, the "timeout" specified in AIBurst and AIStreamRead is also disabled.

Declaration:
long NoThread (long *idnum,
long noThread)

Parameter Description:
Returns: LabJack errorcodes or 0 for no error.
Inputs:

- ***idnum –** Local ID, serial number, or -1 for first found.
- **noThread –** If >0, the thread will not be used.

Outputs:

- ***idnum –** Returns the local ID or –1 if no LabJack is found.

4.23 ReEnum

Causes the LabJack to electrically detach from and re-attach to the USB so it will re-enumerate. The local ID and calibration constants are updated at this time.

Declaration:
long ReEnum (long *idnum)

Parameter Description:
Returns: LabJack errorcodes or 0 for no error.
Inputs:

- ***idnum –** Local ID, serial number, or -1 for first found.

Outputs:

- ***idnum –** Returns the local ID or –1 if no LabJack is found.

4.24 Reset (or ResetLJ)

Causes the LabJack to reset after about 2 seconds. After resetting the LabJack will re-enumerate. Reset and ResetLJ are identical.

Declaration:
long Reset (long *idnum)

Parameter Description:
Returns: LabJack errorcodes or 0 for no error.
Inputs:

- ***idnum –** Local ID, serial number, or -1 for first found.

Outputs:

- ***idnum –** Returns the local ID or –1 if no LabJack is found.

4.25 Watchdog

Controls the LabJack watchdog function. When activated, the watchdog can change the states of digital I/O if the LabJack does not successfully communicate with the PC within a specified timeout period. This function could be used to reboot the PC allowing for reliable unattended operation. The 32-bit counter (CNT) is disabled when the watchdog is enabled. Execution time for this function is 20 milliseconds or less.

If you set the watchdog to reset the LabJack, and choose too small of a timeout period, it might be difficult to make the device stop resetting. To disable the watchdog, reset the LabJack with IO0 shorted to STB, and then reset again without the short.

Declaration:

```
long Watchdog (     long *idnum,
                    long demo,
                    long active,
                    long timeout,
                    long reset,
                    long activeD0,
                    long activeD1,
                    long activeD8,
                    long stateD0,
                    long stateD1,
                    long stateD8 )
```

Parameter Description:

Returns: LabJack errorcodes or 0 for no error.

Inputs:

- ***idnum –** Local ID, serial number, or -1 for first found.
- **demo –** Send 0 for normal operation, >0 for demo mode. Demo mode allows this function to be called without a LabJack.
- **active –** Enables the LabJack watchdog function. If enabled, the 32-bit counter is disabled.
- **timeout –** Timer reset value in seconds.
- **reset –** If >0, the LabJack will reset on timeout.
- **activeDn –** If >0, Dn will be set to stateDn upon timeout.
- **stateDn –** Timeout state of Dn, 0=low, >0=high.

Outputs:

- ***idnum –** Returns the local ID or –1 if no LabJack is found.

4.26 ReadMem

Reads 4 bytes from a specified address in the LabJack's nonvolatile memory. Execution time for this function is 20 milliseconds or less.

Declaration:

```
long ReadMem (      long *idnum,
                    long address,
                    long *data3,
                    long *data2,
                    long *data1,
                    long *data0 )
```

Parameter Description:

Returns: LabJack errorcodes or 0 for no error.

Inputs:

- ***idnum –** Local ID, serial number, or -1 for first found.
- **address –** Starting address of data to read (0-8188).

Outputs:

- ***idnum –** Returns the local ID or –1 if no LabJack is found.
- ***data3 –** Byte at address.
- ***data2 –** Byte at address+1.
- ***data1 –** Byte at address+2.
- ***data0 –** Byte at address+3.

4.27 WriteMem

Writes 4 bytes to the LabJack's 8,192 byte nonvolatile memory at a specified address. The data is read back and verified after the write. Memory 0-511 is reserved for configuration and calibration data. Memory from 512-1023 is unused by the LabJack and available for the user (this corresponds to starting addresses from 512-1020). Memory 1024-8191 is used as a data buffer in hardware timed AI modes (burst and stream). Execution time for this function is 20 milliseconds or less.

Declaration:

```
long WriteMem (      long *idnum,
                     long unlocked,
                     long address,
                     long data3,
                     long data2,
                     long data1,
                     long data0 )
```

Parameter Description:

Returns: LabJack errorcodes or 0 for no error.

Inputs:

- ***idnum –** Local ID, serial number, or -1 for first found.
- **unlocked –** If >0, addresses 0-511 are unlocked for writing.
- **address –** Starting address for writing (0-8188).
- **data3 –** Byte for address.
- **data2 –** Byte for address+1.
- **data1 –** Byte for address+2.
- **data0 –** Byte for address+3.

Outputs:

- ***idnum –** Returns the local ID or –1 if no LabJack is found.

4.28 BuildOptionBits (ActiveX only)

This function is only in the OCX, and is used to build the optionBits parameter for AIBurst and AIStreamStart.

The parameter optionBits is made up of the following bits and can often just be set to 2 (normal operation with the LED on):

- bit 0 => demo
- bit 1 => ledOn
- bit 2 => disableCal
- bits 3,4 => transferMode
- bit 5 => updateIO
- bit 6 => stateIOin(0)
- bit 7 => stateIOin(1)
- bit 8 => stateIOin(2)
- bit 9 => stateIOin(3)

Declaration:

```
long BuildOptionBits (      long demo,
                            long ledOn,
                            long disableCal,
                            long transferMode,
                            long updateIO,
                            long stateIOin )
```

Parameter Description:
Returns: optionBits
Inputs:

- **demo –** Send 0 for normal operation, >0 for demo mode. Demo mode allows this function to be called without a LabJack.
- **ledOn –** If >0, the LabJack LED is turned on.
- **disableCal –** If >0, voltages returned will be raw readings that are not corrected using calibration constants.
- **transferMode –** Set to 0 (automatic).
- **updateIO –** If >0, state values will be written. Otherwise, just a read is performed.
- ***stateIOin –** Output states for IO0-IO3.

4.29 FourPack (ActiveX only)

This function is only in the OCX, and is used to convert a 4 element array into an integer. The packed value is determined as: valueA + (valueB * 2^8) + (valueC * 2^16) + (valueD * 2^24).

Declaration:

```
long FourPack (      long valueA,
                     long valueB,
                     long valueC,
                     long valueD )
```

Parameter Description:
Returns: Packed representation of a 4 element array.
Inputs:

- **valueA –** Element 0 of the array to be converted.
- **valueB –** Element 1 of the array to be converted.
- **valueC –** Element 2 of the array to be converted.
- **valueD –** Element 3 of the array to be converted.

4.30 Description of errorcodes.

It is recommended that the function GetErrorString be used to interpret errorcodes, but this list is provided as a convenience.

- **0 –** No error.
- **1 –** Unknown error.
- **2 –** No LabJacks found.
- **3 –** LabJack n not found.
- **4 –** Set USB buffer error.
- **5 –** Open handle error.
- **6 –** Close handle error.
- **7 –** Invalid ID.
- **8 –** Invalid array size or value.
- **9 –** Invalid power index.
- **10 –** FCDD size too big.
- **11 –** HVC size too big.

- **12 –** Read error.
- **13 –** Read timeout error.
- **14 –** Write error.
- **15 –** Turbo error.
- **16 –** Illegal channel index.
- **17 –** Illegal gain index.
- **18 –** Illegal AI command.
- **19 –** Illegal AO command.
- **20 –** Bits out of range.
- **21 –** Illegal number of channels.
- **22 –** Illegal scan rate.
- **23 –** Illegal number of samples.
- **24 –** AI response error.
- **25 –** LabJack RAM checksum error.
- **26 –** AI sequence error.
- **27 –** Maximum number of streams.
- **28 –** AI stream start error.
- **29 –** PC buffer overflow.
- **30 –** LabJack buffer overflow.
- **31 –** Stream read timeout.
- **32 –** Illegal number of scans.
- **33 –** No stream was found.
- **40 –** Illegal input.
- **41 –** Echo error.
- **42 –** Data echo error.
- **43 –** Response error.
- **48 –** Caps error.
- **49 –** Caps error.
- **50 –** Caps error.
- **51 –** HID number caps error.
- **52 –** HID get attributes warning.
- **57 –** Wrong firmware version error.
- **64 –** Could not claim all LabJacks.
- **65 –** Error releasing all LabJacks.
- **66 –** Could not claim LabJack.
- **67 –** Error releasing LabJack.
- **68 –** Claimed abandoned LabJack.
- **69 –** Local ID –1 thread stopped.
- **70 –** Stop thread timeout.
- **71 –** Thread termination failed.
- **72 –** Feature handle creation error.

If bit 8 is set, the error occurred in the stream thread. Bit 10 is set for Windows API errors.

A. Specifications

Parameter	Conditions	Min	Typical	Max	Units
General					
USB Cable Length				3	meters
User Connection(s) Length	CE compliance			3	meters
Supply Current (1)			20		mA
Operating Temperature		-40		85	°C
Clock Error	~ 25 °C			±30	ppm
	0 to 70 °C			±50	ppm
	-40 to 85 °C			±100	ppm
+5 Volt Power Supply (+5V)					
Voltage (Vs) (2)	Self-Powered	4.5		5.25	volts
	Bus-Powered	4.1		5.25	volts
Output Current (2) (3)	Self-Powered	450		500	mA
	Bus-Powered	50		100	mA
Analog Inputs (AI0 - AI7)					
Input Range For Linear Operation	AIx to GND, SE	-10		10	volts
	Aix to GND, Diff.	-10		20	volts
Maximum Input Range	AIx to GND	-40		40	volts
Input Current (4)	Vin = +10 volts		70.1		μA
	Vin = 0 volts		-11.7		μA
	Vin = -10 volts		-93.5		μA
Resolution (No Missing Codes)	C/R and Stream		12		bits
	Burst Diff.		12		bits
	Burst SE (5)		11		bits
Offset	G = 1 to 20		±1 * G		bits
Absolute Accuracy	SE		±1		% FS
	Diff.		±0.2		% FS
Integral Linearity Error			±1		bits
Differential Linearity Error			±0.5		bits
Repeatability			±1		bits
CAL Accuracy	CAL = 2.5 volts		±0.05	±0.25	%
CAL Current	Source			5	mA
	Sink	20	200		μA
Trigger Latency	Burst	25		50	μs
Trigger Pulse Width	Burst	40			μs
Analog Outputs (AO0 & AO1)					
Maximum Voltage (6)	No Load		Vs		volts
	At 1 mA		0.99 * Vs		volts
	At 5 mA		0.98 * Vs		volts
Output Current	Each AO			30	mA

Parameter	Conditions	Min	Typical	Max	Units
IO					
Low Level Input Voltage				0.8	volts
High Level Input Voltage		3		15	volts
Input Leakage Current			±1		µA
Output Short-Circuit Current (7)	Output High		3.3		mA
Output Voltage (7)	No Load	Vs - 0.4	Vs		volts
	At 1 mA		Vs - 1.5		volts
D					
Low Level Input Voltage (8)	D0 - D12			0.8	volts
	D13 - D15			1	volts
High Level Input Voltage (8)	D0 - D12	2		Vs + 0.3	volts
	D13 - D15	4		Vs + 0.3	volts
Input Leakage Current			±1		µA
Output Current (8)	Per Line			25	mA
	Total D0 - D15			200	mA
Output Low Voltage				0.6	volts
Output High Voltage		Vs - 0.7			volts
CNT					
Low Voltage (9)		GND		1	volts
High Voltage (9)		4		15	volts
Schmitt Trigger Hysteresis			20-100		mV
Input Leakage Current			±1		µA
Minimum High Time				500	ns
Minimum Low Time				500	ns
Maximum Input Frequency		1			MHz

(1) Current drawn by the LabJack through the USB. The status LED is responsible for 4-5 mA of this current.

(2) Self-powered would apply to USB hubs with a power supply, all known desktop computer USB hosts, and some notebook computer USB hosts. Bus-powered would apply to USB hubs without a power supply and some notebook computer USB hosts.

(3) This is the total current that can be sourced by +5V, analog outputs, and digital outputs.

(4) The input current at each analog input is a function of the voltage at that input (Vin) with respect to ground and can be calculated as: (8.181*Vin - 11.67) µA.

(5) Single-ended burst mode only returns even binary codes, and thus has a net resolution of 11 bits.

(6) Maximum analog output voltage is equal to the supply voltage at no load.

(7) The IO lines each have a 1500 ohm series resistor.

(8) These lines have no series resistor. It is up to the user to make sure the maximum voltages and currents are not exceeded.

(9) CNT is a Schmitt Trigger input.