1 MATLAB Function Calls for the LabJack U12

Contained within Matlab_funcs.zip are all of the necessary files to communicate with the LabJack U12 within the MATLAB environment. Due to the nature in which MATLAB communicates with the LabJack drivers, some extra time is required to make function calls. For single function calls the execution time is increased form 20 milliseconds to approximately 50 milliseconds. However, if the function calls are made repeatedly within a loop, the execution time decreases to values of 20 milliseconds or less. Multiple function calls in succession allows MATLAB to keep the necessary files in memory, decreasing total execution time. If knowing function execution time is critical, it is suggested that the user make use of MATLAB functions to calculate the execution time. See MATLAB documentation for more information on this topic.

The LabJack functions were all compiled using MATLAB version 6.1. Use of these functions with older versions of MATLAB may not be possible. If difficulties arise when using these functions, please contact LabJack (mailto:support@labjack.com).

1.1 Getting Started

To use the LabJack functions for MATLAB, start by extracting the files in Matlab_funcs.zip to a folder located somewhere on your hard drive. Open up MATLAB, choose *File* from the menu bar, and select *Set Path* from the list of options. A window will open that allows the user to set the path of the file containing the LabJack function calls. In the *Set Path* window press the *Add Folder* button and locate the folder containing the LabJack files. Once the file has been selected press *OK*. The directory path of the LabJack function folder should now appear at the top of the list in the MATLAB search path. If it is present press the *Save* button. The MATLAB search path has now been modified to include the file containing all the LabJack functions. Close the Set Path window by pressing the *Close* button.

Note: Any time a function is called, MATLAB must locate its source code. In the case of the LabJack functions, MATLAB must locate the compiled *.dll files used for communication with the LabJack drivers. At the time of a function call, MATLAB will always start looking for the required files in the current working directory. From there it moves on to all of the files listed in the MATLAB search path. If the LabJack folder is moved, the search path has to be modified to reflect its new location. Otherwise MATLAB will be unable to locate the required files.

1.2 Using the LabJack Functions

Calling the LabJack functions within MATLAB is done just as any standard MATLAB function. Each function has a set of input arguments and return values. The general calling syntax for the MATLAB functions is

[Output Values] = Function_name(Input Arguments)

The first term, *[Output Values],* represents an array of variables that MATLAB will set equal to the returned values of the function. Each function has a different number of return values. The number of elements in the return variable array can be any number up to the total number of return values of the function. If the return variable array has fewer elements than returned values,

MATLAB will start assigning return values to the each return variable until it has run out of return variables. Assignment of return values always starts with the first return value and the first return variable. If there are more return variables than returned values an error message will be returned by the function.

Function_name is the name of the particular function to be called. The function name is not case sensitive, but must be spelled exactly as it appears in all function documentation. The function name is always followed by a list of input values contained within parenthesis. However, if the function does not require input arguments, the parenthesis must be omitted. Below are some examples of calling LabJack functions from within the MATLAB environment.

1.2.1 Example 1 (EAnalogIn called from workspace)

EanalogIn is an "easy" function for reading in an analog voltage from one of the eight analog input channels. The MATLAB syntax and variable definition are listed below.

MATLAB Syntax:

[voltage overVoltage errorcode idnum] = EEnlaogIn(idnum, demo, channel, gain)

Inputs:

idnum - Local Id, serial number, or -1 for first LJ found.
demo - 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.

Outputs:

voltage - Returns the voltage reading.
overVoltage - If >0 overvoltage has been detected on one of the selected analog inputs
errorcode - LabJack errorcodes or 0 for no error.
idnum - Local ID number of Labjack, or -1 if no LabJack is found

If EAnalogIn is called from the MATLAB workspace, the function call and returned values will look something like this:

```
>> [a b c d] = eanalogin(-1,0,1,0)

a =
   1.4307

b =
    0

c =
    0
```

```
d =
   123

>>
```

In the above example, EAnalogIn was called to read in an analog voltage from the first LabJack found (-1), in standard operation mode (0), from AI1 (1), with a gain of zero (0). The values listed below the function call are the returned values assigned to each of the variables in the return variable array.

- a = 1.4037V = AI0 voltage
- b = 0 = No overvoltage occurred
- c = 0 = No errors occurred
- d = 123 = LabJack ID number

If the return array has fewer elements than the return values for the function, only return values that have corresponding return variables will be assigned. For example, if EAnalogIn is called with only a two-element return array, the first two return values are assigned (Voltage and Overvoltage).

```
>> [a b] = eanalogin(-1,0,1,0)

a =
    1.4307

b =
     0

>>
```

If no return variables are defined MATLAB will return the first return value and assign it to the variable “ans”, by default.

```
>> eanalogin(-1,0,1,0)

ans =

    1.4307

>>
```

1.2.2 Example 2 (EAnalogIn used in a m-file)

Labjack functions can be called from the MATLAB workspace, or they can be called from an m-file to make use of other MATLAB code. Below is a simple example that uses EAnalogIn to

collect data from AI1 for one second, and plots the data to screen. Points are sampled about every 50 ms.

```
function analog_sample()

for (i = 1:20);
      start = cputime;
      time = (i-1)*0.050;
      volts = eanalogin(-1,0,1,0);       % Sample analog channel 1
      data(i,:) = [time volts];          % Data array for plotting
      elapsed = start-cputime;           % Calculate the time elapsed during function call
      pause(0.050-elapsed);              % Pause for remainder of 50 ms sample period
end

% Plot data
plot(data(:,1),data(:,2));
title('Analog Input Channel 1');
xlabel('Time (s)');
ylabel('AI1 Input (V)');
grid on;
```

This file can be run if the above code is cut and pasted into a new m-file, and saved to the hard drive.

2 Function Documentation

The following section documents all of the LabJack MATLAB function calls. A brief description of each function is included as well as definitions for all input/output arguments.

2.1 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 50 ms or less.

MATLAB Syntax:

[voltage overVoltage errorcode idnum] = EAnlaogIn(idnum, demo, channel, gain)

Inputs:

idnum - Local Id, serial number, or -1 for first LJ found
demo - 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.

Outputs:

voltage - Returns the voltage reading.
overVoltage - If >0 over voltage has been detected on one of the selected analog inputs
errorcode - LabJack error codes or 0 for no error.

idnum - Local ID number of Labjack, or -1 if no LabJack is found.

2.2 EAnalogOut

Easy function. This is a simplified version of AOUpdate. Sets the voltage of both analog outputs. Execution time for this function is 50 milliseconds or less.

MATLAB Syntax:

[errorcode idnum] = EAnlaogOut(idnum, demo, analogOut0, analogOut1)

Inputs:

idnum - Local Id, serial number, or -1 for first LJ found
demo - 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 - Local ID number of Labjack, or -1 if no LabJack is found.
errorcode - LabJack error codes or 0 for no error.

2.3 Ecount

Easy function. This is a simplified version of Counter. Reads and resets the counter (CNT) Calling this function disables STB (which is the default anyway). Execution time for this function is 50 milliseconds or less.

MATLAB Syntax:

[count ms errorcode idnum] = ECount (idnum, demo, resetCounter)

Inputs:

idnum - Local Id, serial number, or -1 for first LJ found
demo - 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:

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.
errorcode - LabJack error code or 0 for no error.
idnum - Returns the local ID or -1 if no LabJack is found. ECount

2.4 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 leave it that way. Execution time for this function is 50 ms 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 for the lines and assumes all directions are inputs and output stares are low.

MATLAB Syntax:

[state errorcode idnum] = EDigitalIn(idnum, demo, channel, readD)

Inputs:

idnum - Local Id, serial number, or -1 for first LabJack found
demo - 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:

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

2.5 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 leave it that way. Execution time for this function is 50 ms 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 for the lines and assumes all directions are input and output stares are low.

Matlab Syntax:

[errorcode idnum] = EDigitalOut(idnum, demo, channel, writeD, state)

Inputs:

idnum - Local Id, serial number, or -1 for first LJ found

demo - 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:
errorCode - LabJack errorcode or 0 for no error.
idnum - Returns the local ID or -1 if no LabJack is found.

2.6 AISample

Reads the voltage from 1,2, or 4 analog inputs. Also controls/reads the 4 IO Ports. Execution time for this function is 50 ms or less

MATLAB Syntax:
[voltages stateIO overVoltage errorcode idnum] = AISample(idnum, demo, stateIO, updateIO, ledOn, numChannels, channels, gains, diableCal)

Input:
idnum - Local Id, serial number, or -1 for first LJ found
demo - 0 for normal operation, >0 for demo mode. Demo mode allows this function to be called without a LabJack.
stateIO - Output states for IO3 - IO4 (Decimal value).
updateeIO - If > 0, sate 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 - Array of channel commands with at least numChannels elements. Each channel command is 0-7.for single-ended, or 8-11 for differential.
gains - Array of gain commands with at least numChannel elements. Gain commands are 0=1, 1=2, ..., 7=20. This amplification is available only for differential channels.
disableCal - If >0, voltages returned will be raw readings that are note corrected using calibration constants.

Output:
voltages - Array where voltage readings are returned. Same length as numChannels.
stateIO = Returns input states for IO0 - IO3. Decimal value ranging from 0 - 15.
overVoltage = if > 0 over voltage has been detected on one of the selected analog inputs
errorcode = LabJack error codes or 0 for no error.
idnum = Local ID number of Labjack, or -1 if no LabJack is found.

2.7 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.

Matlab Syntax:

[voltages stateIOout scanRate overVoltage errorcode idnum] = AIBurst(idnum, demo, stateIOin, updateIO, ledOn, numChannels, channels, gains, scanRate, diableCal, trigerIO, triggerState, numScans, timeout, transfermode)

Inputs:

idnum - Local Id, serial number, or -1 for first LJ found
demo - 0 for normal operation, >0 for demo mode. Demo mode allows this function to be called without a LabJack.
stateIOin - Output states for IO3 - IO4.
updateIO - If > 0, sate 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 - Array of channel commands with at least numChannels elements. Each channel command is 0-7 for single-ended, or 8-11 for differential.
gains - Array of gain commands with at least numChannel elements. Gain commands are 0=1, 1=2, ..., 7=20. This amplification is available only 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 note 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
transferMode - Always send 0

Outputs:

voltages - Array where voltage readings are returned. Size of array is numScans x numChannels
stateIOout = Array where IO states are returned.
scanRate - Returns the actual scan rate, which due to clock resolution is not always exactly the same as the desired scan rate.
overVoltage = if > 0 over voltage has been detected on one of the selected analog inputs
errorcode = LabJack error codes or 0 for no error.
idnum = Local ID number of LabJack, or -1 if no LabJack is found.

2.8 AIStreamStart

Starts 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 be toggled every 40 samples during acquisition and turn on when the stream operation stops.

MATLAB Syntax:

[scanRate errorcode idnum] = AIStreamStart(idnum, demo, stateIOin, updateIO, ledOn, numChannels, channels, gains, scanRate, disableCal, readCount)

Inputs:

idnum – Local ID, serial number, or –1 for first LabJack found.
demo – Send 0 for normal operation, >0 for demo mode. Demo mode allows this function to be called without a LabJack.
stateIOin – Output state 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 – Array of channel commands with at least numChannels elements. Each channel command is 0-7 for single-0ended, or 8-11 for differential.
gains – 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.
readCount - If>0, the current count (CNT) is returned instead of the 2^{nd}, 3^{rd}, and 4^{th} analog input channels. 2^{nd} channel is bits 0-11, 3^{rd} channel is bists 12-23. 4^{th} 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:

scanRate – Returns the actual scan rate, which due to clock resolution is not always exactly the same as the desired scan rate.
errorcode = LabJack error codes or 0 for no error.
idnum = Local ID number of LabJack, or -1 if no LabJack is found.

2.9 AIStreamRead

Waits for 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.

MATLAB Syntax:

[voltages stateIOout LjScanBacklog overVoltage errorcode idnum] =
AIStreamRead(localID, numScans, timeout)

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.

Outputs:

voltages – Array where voltage readings are returned. Array size is numScans x numChannels.
stateIOout - Array where IO readings are returned. Array size is numScans x numChannels.
ljScanBacklog – Returns the number of scans backlogged in the LabJack Ram buffer. This indicates the number of scans that are left in the buffer after AIStreamRead has completed. The total 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.
errorcode = LabJack error codes or 0 for no error.

2.10 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, AIStremRead, AIStreamRead, …, AIStreamClear.

MATLAB Syntax:

[errorcode] = AIStreamClear(localID)

Inputs:

localID – Send the local ID from AIStreamStart/Read

Ouputs:

errorcode = LabJack error codes or 0 for no error.

2.11 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 50 ms or less.

MATLAB syntax:

[stateD stateIO count errorcode idnum] = aoupdate(idnum, demo, trisD, trisIO, stateD, stateIO, updateDigital, resetCounter, analogOut0, analogOut1)

Inputs:

idnum – Local ID, serial number, or –1 for first LabJack 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:

stateD – States for D0 – D15.
stateIO – States for IO0 – IO3.
count – Current value of the 32-bit counter (CNT). This value is read before the counter is reset.
errorcode = LabJack errorcodes or 0 for no error.
idnum = Local ID number of Labjack, or -1 if no LabJack is found.

2.12 AsynchConfig

Requires firmware V1.08 or higher. This function writes to the asynch registers and sets the direction of the D lines (input/output) as needed. The actual 1-bit time is about 1.833 plus a "full" delay (us). The actual 1/2-bit time is about 1.0 plus a "half" delay (us).

MATLAB Syntax:

[errorcode idnum] = AsynchConfig(idnum, demo, timeoutMult, configA, configB, configTE, fullA, fullB, fullC, halfA, halfB, halfC)

Inputs:

idnum – Local ID, Serial Number, or -1 for first found (I32).

demo – Send 0 for normal operation, >0 for demo mode (I32). Demo mode allows this function to be called without a LabJack, and does little but simulate execution time.
timeoutMult – If enabled, read timeout is about 100 milliseconds times this value (I32, 0-255).
configA – If >0, D8 is set to output-high and D9 is set to input (I32).
configB – If >0, D10 is set to output-high and D11 is set to input (I32).
configTE – If >0, D12 is set to output-low (I32).
fullA – A time value for a full bit (I32, 1-255).
fullB – B time value for a full bit (I32, 1-255).
fullC – C time value for a full bit (I32, 1-255).
halfA – A time value for a half bit (I32, 1-255).
halfB – B time value for a half bit (I32, 1-255).
halfC – C time value for a half bit (I32, 1-255).

Outputs:
idnum - Returns the local ID or -1 if no LabJack is found.
errorcode - LabJack errorcodes or 0 for no error.

2.13 Asynch

Requires firmware V1.05 or higher. This function writes and then reads half-duplex asynchronous data on 1 of two pairs of D lines (8,n,1). Call AsynchConfig to set the baud rate. Similar to RS232, except that logic is normal CMOS/TTL (0=low=GND, 1=high=+5V, idle state of transmit line is high). Connection to a normal RS232 device will probably require a converter chip such as the MAX233.

PortA => TX is D8 and RX is D9
PortB => TX is D10 and RX is D11
Transmit Enable is D12

Up to 18 bytes can be written and read. If more than 4 bytes are written or read, this function uses calls to WriteMem/ReadMem to load/read the LabJack's data buffer.

MATLAB Syntax
[data errorcode idnum] = Asynch(idnum, demo, portB, enableTE, enableTO, enableDel, baudrate, numWrite, numRead, data)

Inputs:
idnum – Local ID, Serial Number, or -1 for first found (I32).
demo – Send 0 for normal operation, >0 for demo mode (I32). Demo mode allows this function to be called without a LabJack, and does little but simulate execution time.
portB – If >0, asynch PortB is used instead of PortA.
enableTE – If >0, D12 (Transmit Enable) is set high during transmit and low during receive (I32).
enableTO – If >0, timeout is enabled for the receive phase (per byte).
enableDel – If >0, a 1 bit delay is inserted between each transmit byte.

Baudrate – -This is the bps as set by AsynchConfig. Asynch needs this so it has an idea how long the transfer should take.
numWrite – Number of bytes to write (I32, 0-18).
numRead – Number of bytes to read (I32, 0-18).
data – Serial data buffer. Send an 18 element array. Fill unused locations with zeros (I32).

Outputs:
idnum – Returns the Local ID or -1 if no LabJack is found (I32).
data – Serial data buffer. Returns any serial read data. Unused locations are filled with 9999s. (I32).
errorcode - LabJack errorcodes or 0 for no error.

Time: 20 ms to read & write up to 4 bytes, plus 40 ms for each additional 4 bytes to read or write. Possibly extra time for slow baud rates.

2.14 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.

MATLAB Syntax:
[volts errorcode] = bitstovolts(chnum, chgain, bits)

Input:
chnum - Channel index. 0-7 SE, 8-11 Diff.
chgain - Gain index. 0=1, 1=2, .., 7=20.
bits - Binary value from 0-4096.
Output:
volts - Voltage.

2.15 VoltToBits

Converts a voltage to its 12-bit (0-4095) binary representation.
Bits=(4096*((Volts*Gain)+Vmax))/(2*Vmax) where Vmax=10 for SE, 20 for Diff.

MATLAB Syntax:
[bits errorcode] = VoltsToBits(chnum, chgain, volts)

Input:
chnum - Channel idex. 0-7 SE, 8-11 Diff.
chgain - Gain index. 0=1, 1=2, .., 7=20.
volts – Voltage.
Output:
bits - Binary value 0-4095.

2.16 Counter

Controls and reads the counter. The counter is disabled if the watchdog timer is enabled. Executing time for this function is 50 ms or less.

MATLAB Syntax:

[stated stateIO count errorcode idnum] = Counter(idnum, demo, resetCounter, enableSTB)

Inputs:

idnum – Local ID, serial number, or –1 for first LabJack 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:

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.
errorcode = LabJack error codes or 0 for no error.
idnum = Local ID number of Labjack, or -1 if no LabJack is found.

2.17 DigitalIO

Reads and writes to all 20 digital I/O. Execution time for this function is 50 ms or less.

MATLAB Syntax:

[stateD stateIO trisD outputD errorcode idnum] = DigitalIO(idnum, demo, trisD, trisIO, stateD, stateIO, updateDigital)

Inputs:

idnum - Local Id, serial number, or -1 for first LJ found
demo - 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:

stateD - States of D0-D15.

stateIO - States of IO0-IO3.
trisD - Returns a read of the direction registers for D0-D15.
outputD – Returns a read from the output registers for DO-D15.
errorcode - LabJack error codes or 0 for no error.
idnum - Local ID number of Labjack, or -1 if no LabJack is found.

2.18 GetDriverVersion

Returns the version number of ljackuw.dll.

MATLAB Syntax:
[version] = GetDriverVersion

Inputs:
None.

Outputs:
version - Version number of ljackuw.dll.

2.19 GetErrorString

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

Matlab Syntax:
[errorstring] - GetErrorString(errorcode)

Inputs:
errorcode - LabJack error code.

Outputs:
errorstring - Character string describing error.

2.20 GetFirmwareVersion

Retrieves the firmware version from the LabJack's processor.

MATLAB Syntax:
[version idnum] = GetFirmwareVersion(idnum)

Inputs:
idnum - Local Id, serial number, or -1 for first LJ found

Outputs:

version - Version number of the LabJack firmware or 0 for error.
idnum - Local ID, serial number, or -1 for first found.

2.21 GetWinVersion

Uses a Windows API function to get the OS version

MATLAB Syntax:
[majorVersion minorVersion buildNumber platformID servicePackMajor servicePackMinor errorcode] = GetWinVersion

Inputs:
None

Outputs:

	Platform	Major	Minor	Build
Windows 3.1	0	-	-	-
Windows 95	1	4	0	950
Windows 95 OSRS	1	4	0	1111
Windows 98	1	4	10	1998
Windows 98 SE	1	4	10	2222
Windows ME	1	4	90	3000
Windows NT4 3.51	2	3	51	-
Windows NT 4.0	2	4	0	1381
Windows 2000	2	5	0	2195
Whistler	2	5	1	-

errorcode - LabJack error codes or 0 for no error.

2.22 ListAll

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

MATLAB Syntax:
[serialnumList localIDList numberFound errorcode] = ListAll

Input:
none

Output:
serialnumList - Array where serial numbers are returned.

localIDList - Array were local ID numbers are returned.
numberFound - Number of LabJacks found on the USB.
fcddMaxSize - Max size of fcdd.
hvcMinSize - Max size of hvc.
errorcode - Labjack error codes or 0 for no error.

2.23 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 ReEnumn or ResetLJ.

MATLAB Syntax:
[localID errorcode] = LocalID(idnum, localID)

Inputs:
idnum - Local ID, serial number, or -1 for first found
localID - New local ID number

Outputs:
idnum - Returns the local ID or -1 if no LabJack is found.
errorcode - LabJack errorcodes or 0 for no error.

2.24 PulseOut

Requires firmware V1.1 or higher. The timeout of this function, in milliseconds, is set to: 5000+numPulses*((B1*C1*0.02)+(B2*C2*0.02)). This command creates pulses on any/all of D0-D7. The desired D lines must be set to output using another function (DigitalIO or AOUpdate). All selected lines are pulsed at the same time, at the same rate, for the same number of pulses.

This function commands the time for the first half cycle of each pulse, and the second half cycle of each pulse. Each time is commanded by sending a value B & C, where the time is,

1st half-cycle microseconds = ~17 + 0.83*C + 20.17*B*C
2nd half-cycle microseconds = ~12 + 0.83*C + 20.17*B*C

which can be approximated as,

microseconds = 20*B*C

For best accuracy when using the approximation, minimize C. B and C must be between 1 and 255, so each half cycle can vary from about 38/33 microseconds to just over 1.3 seconds. If you

have enabled the LabJack Watchdog function, make sure it's timeout is longer than the time it takes to output all pulses.

MATLAB Syntax:

[errorcode idnum] = PulseOut(idnum, demo, lowFirst, bitSelect, numPulses, timeB1, timeC1, timeB2, timeC2)

Inputs:

idnum - Local ID, Serial Number, or -1 for first found (I32).
demo - Send 0 for normal operation, >0 for demo mode (I32). Demo mode allows this function to be called without a LabJack, and does little but simulate execution time.
lowFirst - If >0, each line is set low then high, otherwise the lines are set high then low (I32).
bitSelect - Set bits 0 to 7 to enable pulsing on each of D0-D7 (I32, 0-255).
numPulses - Number of pulses for all lines (I32, 1-32767).
timeB1 - B value for first half cycle (I32, 1-255).
timeC1 - C value for first half cycle (I32, 1-255).
timeB2 - B value for second half cycle (I32, 1-255).
timeC2 - C value for second half cycle (I32, 1-255).

Outputs:

idnum - Returns the Local ID or -1 if no LabJack is found (I32).
errorcode - LabJack errorcodes or 0 for no error.

Time: 20 ms plus pulse time (make sure watchdog is longer if active)

2.25 PulseOutStart

Requires firmware V1.07 or higher. PulseOutStart and PulseOutFinish are used as an alternative to PulseOut. PulseOutStart starts the pulse output and returns without waiting for the finish. PulseOutFinish waits for the LabJack's response which signifies the end of the pulse output. If anything besides PulseOutFinish is called after PulseOutStart, the pulse output will be terminated and the LabJack will execute the new command.

Note that due to boot-up tests on the LabJack U12, if PulseOutStart is the first command sent to the LabJack after reset or power-up, there would be no response for PulseOutFinish. In practice, even if no precautions were taken, this would probably never happen, since before calling PulseOutStart a call is needed to set the desired D lines to output.

Also note that PulseOutFinish must be called before the LabJack completes the pulse output to read the response. If PulseOutFinish is not called until after the LabJack sends it's response, the function will never receive the response and will timeout.

This command creates pulses on any/all of D0-D7. The desired D lines must be set to output using another function (DigitalIO or AOUpdate). All selected lines are pulsed at the same time, at the same rate, for the same number of pulses.

This function commands the time for the first half cycle of each pulse, and the second half cycle of each pulse. Each time is commanded by sending a value B & C, where the time is,

1st half-cycle microseconds = ~17 + 0.83*C + 20.17*B*C
2nd half-cycle microseconds = ~12 + 0.83*C + 20.17*B*C

which can be approximated as,

microseconds = 20*B*C

For best accuracy when using the approximation, minimize C. B and C must be between 1 and 255, so each half cycle can vary from about 38/33 microseconds to just over 1.3 seconds.

If you have enabled the LabJack Watchdog function, make sure it's timeout is longer than the time it takes to output all pulses.

MATLAB Syntax:

[errorcode idnum] = PulseOutStart(idnum, demo, lowFirst, bitSelect, numPulses, timeB1, timeC1, timeB2, timeC2)

Inputs:

idnum - Local ID, Serial Number, or -1 for first found (I32).
demo - Send 0 for normal operation, >0 for demo mode (I32). Demo mode allows this function to be called without a LabJack, and does little but simulate execution time.
lowFirst - If >0, each line is set low then high, otherwise the lines are set high then low (I32).
bitSelect - Set bits 0 to 7 to enable pulsing on each of D0-D7 (I32, 0-255).
numPulses - Number of pulses for all lines (I32, 1-32767).
timeB1 - B value for first half cycle (I32, 1-255).
timeC1 - C value for first half cycle (I32, 1-255).
timeB2 - B value for second half cycle (I32, 1-255).
timeC2 - C value for second half cycle (I32, 1-255).

Outputs:

idnum - Returns the Local ID or -1 if no LabJack is found (I32).
idnum - Returns the local ID or -1 if no LabJack is found.
errorcode - LabJack errorcodes or 0 for no error.

2.26 PulseOutFinish

Requires firmware V1.1 or higher. See PulseOutStart for more information.

MATLAB Syntax

[errorcode idnum] = PulseOutFinish(idnum, demo, timeoutMS)

Inputs:

idnum - Local ID, Serial Number, or -1 for first found (I32).
demo - Send 0 for normal operation, >0 for demo mode (I32). Demo mode allows this function to be called without a LabJack, and does little but simulate execution time.
timeoutMS - Amount of time, in milliseconds, that this function will wait for the Pulseout response (I32).

Outputs:

idnum - Returns the Local ID or -1 if no LabJack is found (I32).
idnum - Returns the local ID or -1 if no LabJack is found.
errorcode - LabJack errorcodes or 0 for no error.

2.27 PulseOutCalc

This function can be used to calculate the cycle times for PulseOut or PulseOutStart.

MATLAB Syntax

[errorcode frequency timeB timeC] = PulseOutCalc(frequency, timeB, timeC)

Inputs:

frequency - Desired frequency in Hz (SGL).

Outputs:

frequency - Actual best frequency found in Hz (SGL).
timeB - B value for first and second half cycle (I32).
timeC - C value for first and second half cycle (I32).
idnum - Returns the local ID or -1 if no LabJack is found.
errorcode - LabJack errorcodes or 0 for no error.

2.28 ReEnum

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

MATLAB Syntax:

[idnum errorcode] = ReEnum(idnum)

Inputs:

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

Outputs:

idnum - Returns the local ID or -1 if no LabJack is found.
errorcode - LabJack errorcodes or 0 for no error.

2.29 ResetLJ

Causes the LabJack to reset after about 2 seconds. After resetting the LabJack will re-enumerate.

MATLAB Syntax:
[idnum errorcode] = ResetLJ(idnum)

Input:
idnum - Local ID, serial number, or -1 for first found

Output:
idnum - Returns the local ID or -1 if no LabJack is found.
errorcode - LabJack error codes or 0 for no error.

2.30 SHT1X

This function retrieves temperature and/or humidity readings from a SHT1X sensor. Data rate is about 2 kbps with firmware V1.1 or higher (hardware communication). If firmware is less than V1.1, or TRUE is passed for softComm, data rate is about 20 bps.

DATA = IO0
SCK = IO1

The EI-1050 has an extra enable line that allows multiple probes to be connected at the same time using only the one line for DATA and one line for SCK. This function does not control the enable line.

This function automatically configures IO0 has an input and IO1 as an output.

Note that internally this function operates on the state and direction of IO0 and IO1, and to operate on any of the IO lines the LabJack must operate on all 4. The DLL keeps track of the current direction and output state of all lines, so that this function can operate on IO0 and IO1 without changing IO2 and IO3. 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.

MATLAB Syntax
[errorcode idnum tempC tempF rh] = SHT1X(idnum, demo, softComm, mode,
statusReg, tempC,)

Inputs:
idnum - Local ID, Serial Number, or -1 for first found (I32).
demo - Send 0 for normal operation, >0 for demo mode (I32). Demo mode allows this function to be called without a LabJack, and does little but simulate execution time.
softComm - If >0, forces software based communication. Otherwise software communication is only used if the LabJack U12 firmware version is less than V1.1.

mode - 0=temp and RH,1=temp only,2=RH only. If mode is 2, the current temperature must be passed in for the RH corrections using *tempC.
statusReg - Current value of the SHT1X status register. The value of the status register is 0 unless you have used advanced functions to write to the status register (enabled heater, low resolution, or no reload from OTP).
tempC - If mode is 2, the current temperature must be passed in for the RH corrections.

Outputs:
idnum - Returns the Local ID or -1 if no LabJack is found (I32).
tempC - Returns temperature in degrees C.
tempF - Returns temperature in degrees F.
rh - Returns RH in percent.
errorcode - LabJack error codes or 0 for no error.

Time: About 20 ms plus SHT1X measurement time for hardware comm. Default measurement time is 210 ms for temp and 55 ms for RH. About 2 s per measurement for software comm.

2.31 SHTComm

Low-level public function to send and receive up to 4 bytes to from an SHT1X sensor. Data rate is about 2 kbps with firmware V1.09 or higher (hardware communication). If firmware is less than V1.1, or TRUE is passed for softComm, data rate is about 20 bps.
DATA = IO0
SCK = IO1

The EI-1050 has an extra enable line that allows multiple probes to be connected at the same time using only the one line for DATA and one line for SCK. This function does not control the enable line.

This function automatically configures IO0 has an input and IO1 as an output.

Note that internally this function operates on the state and direction of IO0 and IO1, and to operate on any of the IO lines the LabJack must operate on all 4. The DLL keeps track of the current direction and output state of all lines, so that this function can operate on IO0 and IO1 without changing IO2 and IO3. 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.

MATLAB Syntax
[errorcode idnum datarx] = SHT1X(idnum, softComm, waitMeas, serialReset,
dataRate, numWrite, numRead, datatx, datarx)

Inputs:
idnum - Local ID, Serial Number, or -1 for first found (I32).
softComm - If >0, forces software based communication. Otherwise software communication is only used if the LabJack U12 firmware version is less than V1.1.

waitMeas - If >0, this is a T or RH measurement request.
serialReset - If >0, a serial reset is issued before sending and receiving bytes.
dataRate - 0=no extra delay (default),1=medium delay,2=max delay.
numWrite - Number of bytes to write (0-4,I32).
numRead - Number of bytes to read (0-4,I32).
datatx - Array of 0-4 bytes to send. Make sure you pass at least
numWrite - number of bytes (U8).

Outputs:
idnum - Returns the Local ID or -1 if no LabJack is found (I32).
datarx - Returns 0-4 read bytes as determined by numRead (U8).
errorcode - LabJack error codes or 0 for no error.

Time: About 20 ms plus SHT1X measurement time for hardware comm. Default measurement time is 210 ms for temp and 55 ms for RH. About 2 s per measurement for software comm.

2.32 SHTCRC

Checks the CRC on a SHT1X communication. Last byte of datarx is the CRC. Returns 0 if CRC is good, or SHT1X_CRC_ERROR_LJ if CRC is bad.

MATLAB Syntax
[errorcode] = SHTCRC(statusReg, numWrite, numRead, datatx, datarx)

Inputs:
statusReg - Current value of the SHT1X status register.
numWrite - Number of bytes that were written (0-4).
numRead - Number of bytes that were read (1-4).
datatx - Array of 0-4 bytes that were sent.
datarx - Array of 1-4 bytes that were read.

Outputs:
errorcode - LabJack error codes or 0 for no error.

2.33 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 periods. 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 50 ms or less.

If you set the watchdog to reset the LabJack, and choose too small of 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.

MATLAB Syntax:

[idnum errorcode] = Reset(idnum, demo, active, timeout, reset, activeD0, activeD1, activeD8, stateD0, stateD1, stateD8)

Input:

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 (activeD0 - D8).
stateDn - Timeout state of Dn, 0 = Low, >0 = High (stateD0 - D8)

Output:

idnum - Returns the local ID or -1 if no LabJack is found.
errorcode - LabJack error codes or 0 for no error.

2.34 ReadMem

Reads 4 bytes from a specified address in the LabJack's nonvolatile memory. Execution time for this function is 50 ms or less.

MATLAB Syntax:

[data3 data2 data1 data0 idnum errorcode] = GetErrorString(idnum, address)

Input:

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

Output:

data3 - Byte at address.
data2 - Byte at address+1.
data1 - Byte at address+2.
data0 - Byte at address+3.
idnum - Returns the local ID or -1 if no LabJack is found.
errorcode - LabJack error codes or 0 for no error.

2.35 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 the 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 50 milliseconds or less.

MATLAB Syntax:

[errorcode idnum] = WriteMem(idnum, unlocked, address, data3, data2, data1, data0)

Inputs:

idnum - Local ID, serial number, or -1 for first LabJack found.
unlocked - If>0, addresses 0-511 ae unlocked fro writing.
address - Starting address of data to read (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.
errorcode - LabJack error codes or 0 for no error.