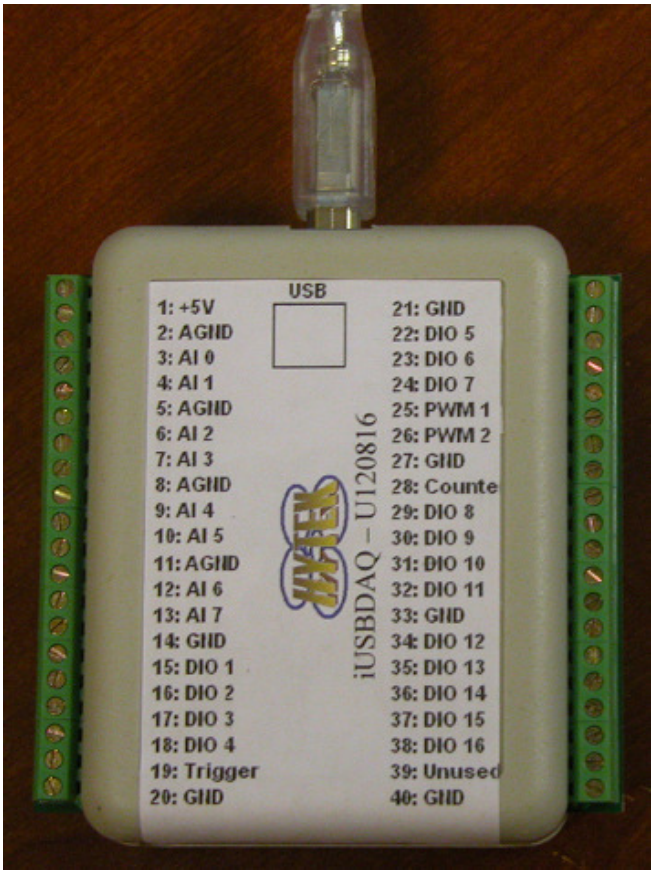
iUSBDAQ - U120816
iUSBDAQ - U1208LOG

USB 2.0 Full Speed Multi-Functions DAQ

User's Guide

Revision 2.2
Aug. 3, 2007

HYTEK Automation, Inc.
www.hytekautomation.com



Picture of iUSBDAQ – U120816

Picture of iUSBDAQ – U1208LOG

Important Information

Disclaimer:
The iUSBDAQ and associated products are not designed to be a critical component in life support or systems where malfunction can reasonably be expected to result in personal injury. Customers using these products in such applications do so at their own risk and agree to fully indemnify HYTEK Automation for any damages resulting from such applications.

All HYTEK Automation, Inc's hardware and software are provided "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall HYTEK Automation, Inc. be liable for any direct, indirect, incidental, special, exemplary, or consequential damages
(including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of HYTEK Automation's software and hardware, even if advised of the possibility of such damage.

Guarantee, Warranty and Trade In Policy:
Please check with your vendor or distributor for warranty period and trade in policy if applicable.

Trademark and Copyright Information:
iUSBDAQ and iDAQTest&Log are trademarks of HYTEK Automation, Inc.

LabVIEW is a trademark of National Instruments.

All other trademarks are the property of their respective owners.

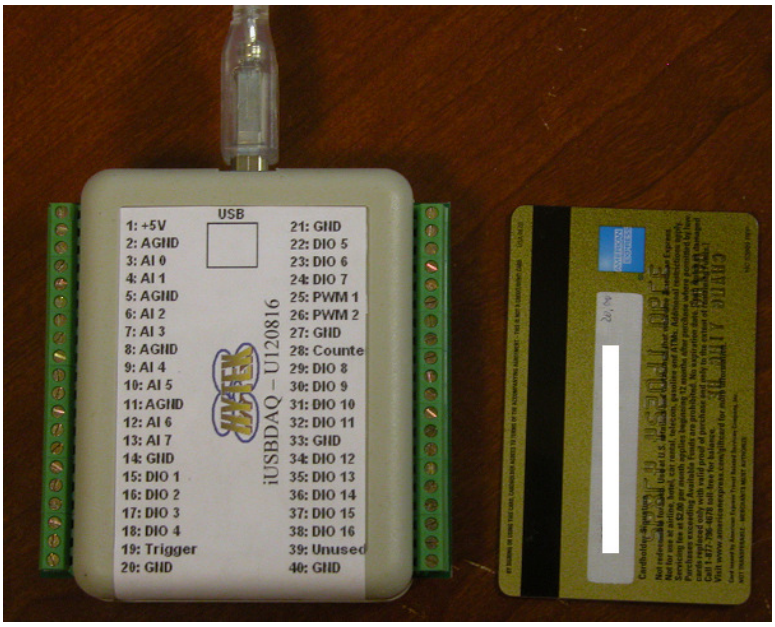
Reproduction of HYTEK Automation's hardware, software and document is prohibited without permission from HYTEK Automation, Inc.

Revision History:

July 2, 2007: refine the documentation for U1208LOG.
June, 2007: Add information for model iUSBDAQ – U1208LOG

Index

1. Overview of iUSBDAQ – U120816

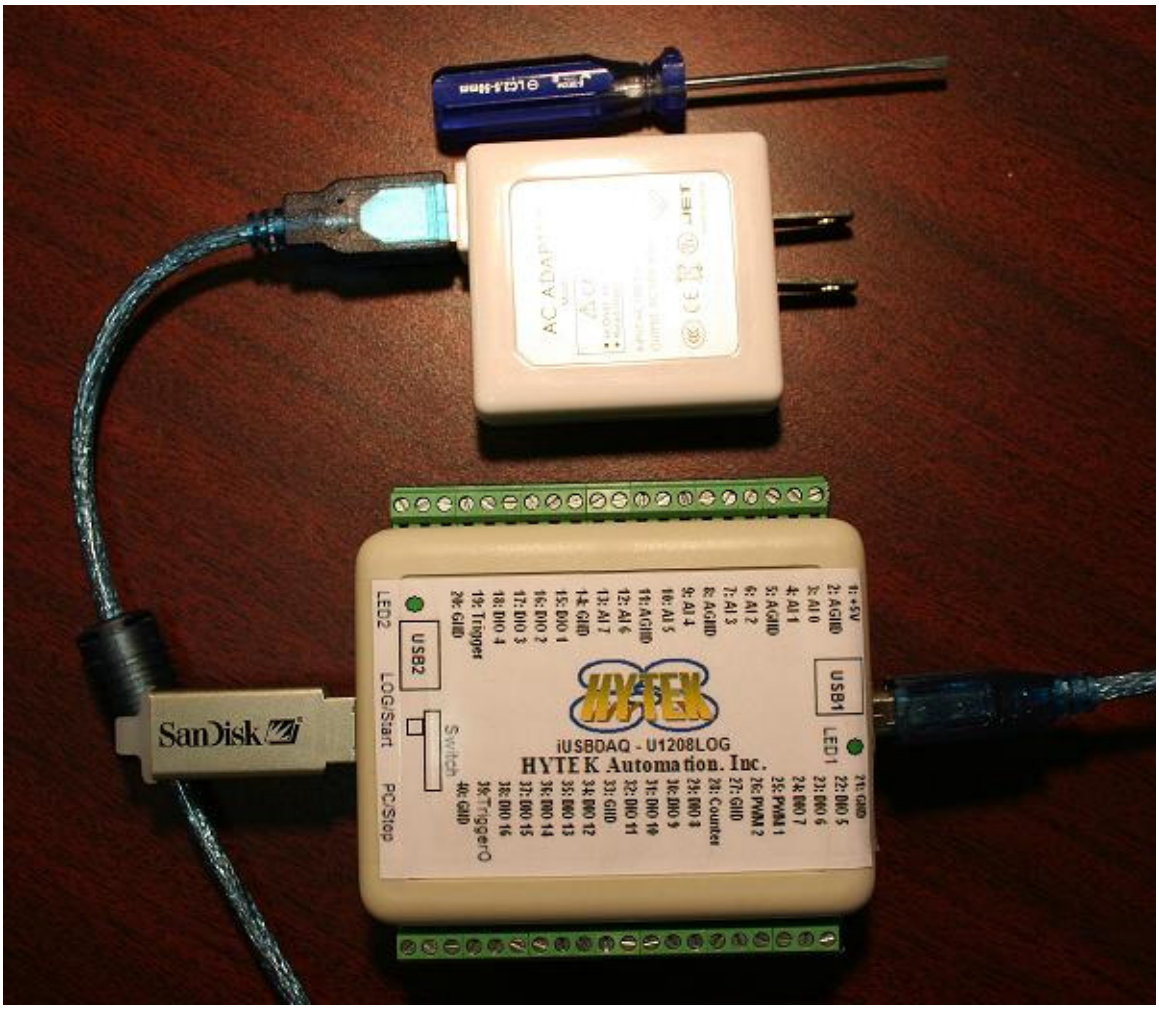


(The credit card is only used for compare with iUSBDAQ size)

- USB 2.0/1.1 Full Speed Interface
- USB Bus Powered
- 8 Single-Ended, 12-Bit Analog Inputs
- 0-4.096 V Analog Input Range
- Up to 32 kSamples/Sec Throughput with Single Channel Up to 13kSamples/Second for Streaming Mode
- Supports Both Scan Mode and Continuous Streaming Mode Data Acquisition
- One Dedicated Trigger Line for Streaming Mode Data Acquisition
- Two 10-bit PWM Outputs (3kHz- 333kHz)
- 16 Bi-Directional Digital I/O Lines (125HZ update rate)
- One 16-Bit Counter
- 240 bytes EEPROM Reserved for User Data
- Multiple iUSBDAQs Can Be Connected On Same Computer
- Works with Windows 98SE, ME, 2000, or XP
- *FREE* USB Cable
- *FREE* Screwdriver

- *FREE* Device Driver, Programming API (DLL), LabVIEW Drivers, Examples
- *FREE* Standalone Ready-to-Run *iDAQTest&Log* Software for Testing, Data Logging, Data Playback and Simple Analysis
- 30 Days Money Back Guarantee, 6 Month Warranty and Trade In Policy
- Approximately 3.5" x 3.375" x 1.125" (9cm x 8.5cm x 3cm)
- Industrial Temperature Range

2. Overview of iUSBDAQ – U1208LOG



Below highlights the key feature points of the iUSBDAQ - U1208LOG USB data acquisition module.

PC control mode:

- USB 2.0/1.1 full speed interface
- USB bus powered
- 8 Single-Ended, 12-Bit analog Inputs
- 0-4.096 V analog input range
- Up to **64 kSamples/Sec** throughput rate with single channel up to 53kSamples/Second for continuously streaming to PC

- One dedicated "trigger in" line for streaming mode data acquisition
- One dedicated video/Log "trigger out" line that sends out pulses for video frames or when log data in standalone data logging mode
- Two programmable 10-bit PWM outputs (3kHz- 333kHz)
- 16 channels of Bi-Directional Digital I/O lines (250HZ update rate)
- One 16-Bit counter
- 240 bytes EEPROM reserved for user data
- Multiple iUSBDAQs can be connected on same computer
- **Simultaneous Streaming from Multiple iUSBDAQs Possible, streaming data and streaming video at the same time possible.**
- Works with Windows 2000, or WinXP. Recommend 2.4GHZ or up CPU, at least 512M RAM.

USB flash drive standalone data logger mode:

- Virtually unlimited storage size for data logging, it's up to your USB flash disk's storage size.
- Can be powered by USB bus from computer or use included USB power adaptor without computer.
- Logging 8 channels of analog inputs, 16 channels of digital inputs and one 16 bit counter. All data are time stamped.
- Logging interval/frequency from 1 second to 12 hours, unlimited log period as long as USB disk has enough free space. Log frequency configurable with FREE iDAQTest&Log software.
- Automatic file name creation with time stamp.
- Raw data conversion utility and simple analysis functions provided in iDAQTest&Log software
- On board lithium battery to power the real time clock. Date/time setting of real time clock on board is configurable with iDAQTest&Log software
- Dedicated log "trigger out" line for synchronization of external devices.

Video Functions:

- Real time video display along with the data readings (8 ch. analog inputs, 16 ch. digital IOs and one counter)
- AVI recording along with data recording (8 ch. analog inputs, 16 ch. digital IOs and one counter)
- Playback AVI file along with data (8 ch. analog inputs, 16 ch. digital IOs and one counter)
- Dedicated video "trigger out" line
- Work with USB webcam or firewire camera that support WDM driver or are directshow compliant

General Information:

- 30 Days Money Back Guarantee, 6 Month Warranty and Trade In Policy
- Approximately 3.5" x 3.375" x 1.125" (9cm x 8.5cm x 3cm)
- 0°C to 70°C operating temperature range

3. Software Download and Upgrade

The iUSBDAQ device driver, LabVIEW interface, LabVIEW examples, C/C++ examples, dot net examples, iDAQTest&Log software and this document can be found and downloaded from http://www.hytekautomation.com/iDAQDownload.html

4. Software and Hardware Installation

For iUSBDAQ to work, windows 2000 or higher OS is required and computer should have usb 1.1 or usb 2.0 ports. Recommendation: WinXP or higher OS, 2.4GHZ with 512M RAM computer. **If you are using an usb hub, please make sure the hub is self-powered, not bus-powered because bus-powered hub will limit power to max 100mA.**

Following are the steps to install iUSBDAQ:

- Do not connect the iUSBDAQ to PC yet. If you have connected, please disconnect the device.
- Install the iUSBDAQ full installation program or at least iUSBDAQ device driver if you haven't done so. iUSBDAQ device driver is included in the full installation. It's the low level driver used for computer to recognize and enumerate the iUSBDAQ devices. This installation will also copy the iUSBDAQ.dll - the programming APIs that your software calls into computer system's directory.
- Now connect the iUSBDAQ to computer usb port with a USB cable **(if you have U1208LOG, please make sure you switch to PC/Stop mode first before connecting the cable, or else computer will not recognize this device**). The computer will automatically detect the new hardware if this is the first time iUSBDAQ is installed on a USB port, every time you change to a different USB port for the first time, the PC will prompt and saying new hardware detected, follow the instruction to install the driver. If windows popup saying that the driver for this hardware is not window's logo certified, please just ignore this message and hit "Install anyway". If it's looking for "mchpusb.sys" file, it's under the "..\system32\drivers". After successful installation, you should see "HYTEK iUSBDAQ" under your device manager in the category "HytekUSBDAQ". The LED beside the USB connector head should be green by now.
- Now you have the choice to run the standalone ready-to-run iDAQTest&Log software to see how device works or start right now programming with your favorite language with our programming API (DLL).

- This step is for U1208LOG model only. For checking the stand alone data logger mode of U1208LOG, you can leave the USB cable connected to PC or use the included USB power adaptor, plug into the 110V or 220V power plug, connecting the USB cable (**must be the one supplied by HYTEK Automation, or if you use your own USB cable, make sure it has noise reduction filter ring, or else logging mode may not work**) between USB power adaptor and U1208LOG **USB1** connector. At power up the U1208LOG module, you may notice that LED2 will flash 3 times at first, then turn off. Now switch the switch besides the USB2 connector to "Log/Start" mode, connecting an USB flash disk to USB2 connector. At this moment, you will notice that LED2 will turn on, LED1 will turn off and USB flash disk is flashing, after a while (time depends on different USB flash disk), LED2 will start flash in an interval of 1 second, this is an indication that the module is writing data into USB flash disk and by default it is set to 1 second logging interval. Now switch the switch to "PC/Stop" mode, this will stop the data logging and **it's strongly recommended that user should always put the switch to "PC/Stop" mode before unplug the USB flash disk to avoid data lose**. User can use iDAQTest&Log software to change the logging interval and convert logged raw data to HYTEK data format so that user can use the analysis screen in iDAQTest&Log software to do some analysis.

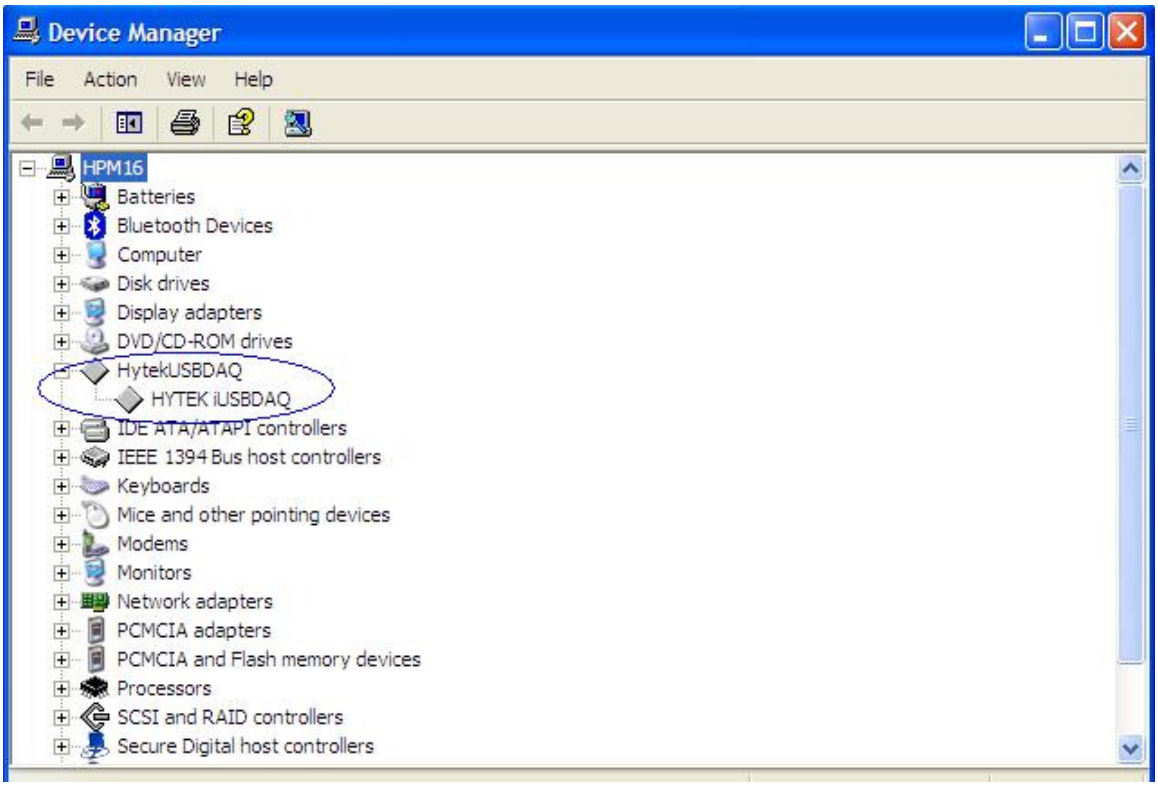


Figure 1. iUSBDAQ registered under device manager

5. Hardware Description

5.1 iUSBDAQ Block Diagram

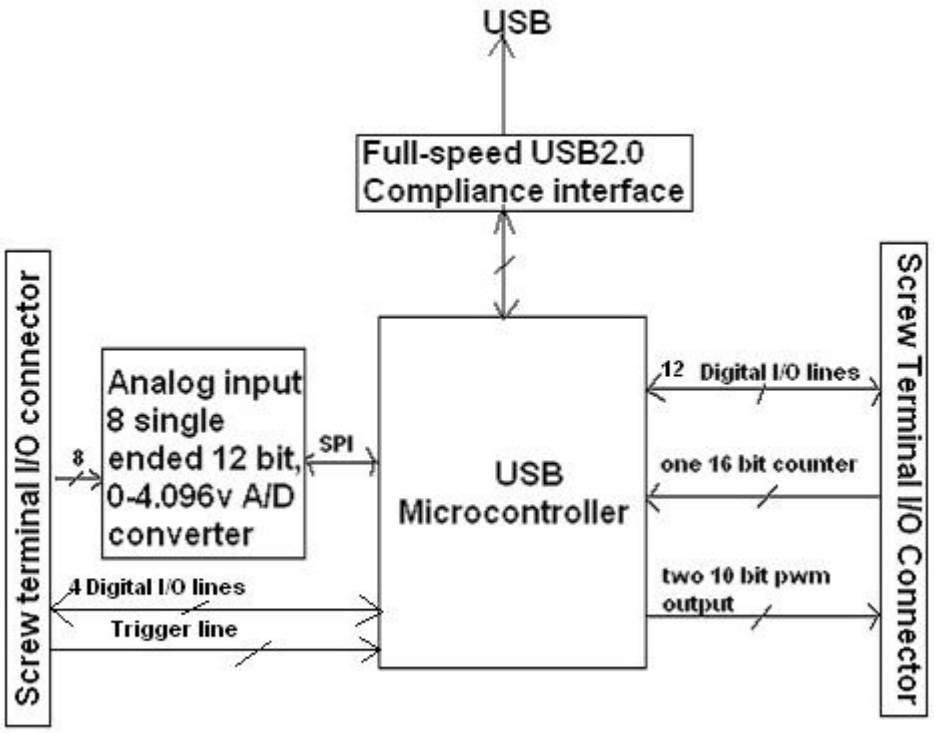
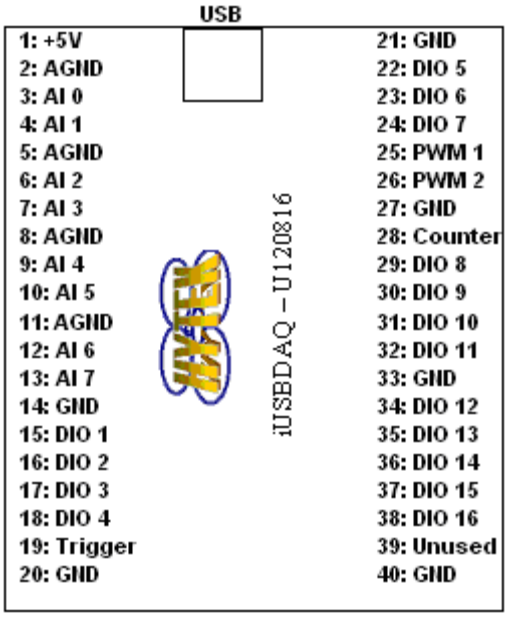
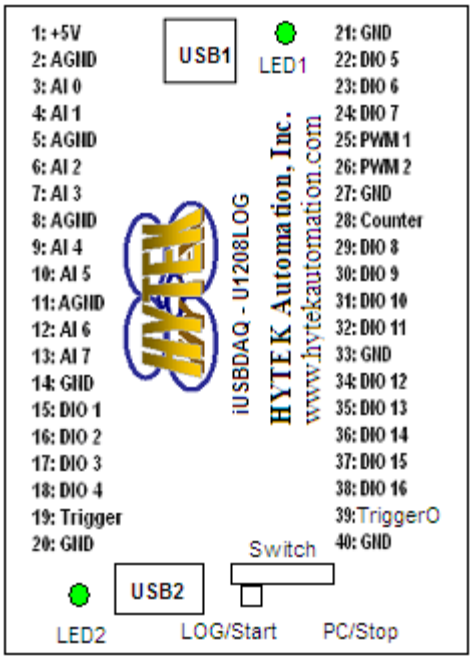


Figure 2. iUSBDAQ Block Diagram

5.2 Terminal Layout and Description

Figure 3 is the terminal layout of iUSBDAQ – U120816 and iUSBDAQ – U1208LOG





Terminal layout of U120816 **Terminal layout of U1208LOG**

U1208LOG Model: It has the same terminal layout as U120816 model above except terminal 39 is not "Unused", instead it becomes "TriggerO", used as video/LOG trigger out line.

Caution!the label on the cover may not align with the screw terminals, user should look at the back of the terminals for the pin numbers.

Figure 4 is the description of terminals

Terminal Number	Name of Terminal	Direction	Description
1	+5V	output	nominal +5 volt internal power supply
2	AGND	-	Analog ground
3	AI 0	input	Analog input channel 0
4	AI 1	input	Analog input channel 1
5	AGND	-	Analog ground
6	AI 2	input	Analog input channel 2
7	AI 3	input	Analog input channel 3
8	AGND	-	Analog ground
9	AI 4	input	Analog input channel 4
10	AI 5	input	Analog input channel 5
11	AGND	-	Analog ground

12	AI 6	input	Analog input 6
13	AI 7	input	Analog input 7
14	GND	-	Common ground
15	DIO 1	Bi-direction	Digital I/O 1
16	DIO 2	Bi-direction	Digital I/O 2
17	DIO 3	Bi-direction	Digital I/O 3
18	DIO 4	Bi-direction	Digital I/O 4
19	Trigger	input	External trigger line for analog streaming mode
20	GND	-	Common ground
21	GND	-	Common ground
22	DIO 5	Bi-direction	Digital I/O 5
23	DIO 6	Bi-direction	Digital I/O 6
24	DIO 7	Bi-direction	Digital I/O 7
25	PWM 1	output	PWM output 1
26	PWM 2	output	PWM output 2
27	GND	-	Common ground
28	Counter	input	Counter
29	DIO 8	Bi-direction	Digital I/O 8
30	DIO 9	Bi-direction	Digital I/O 9
31	DIO 10	Bi-direction	Digital I/O 10
32	DIO 11	Bi-direction	Digital I/O 11
33	GND	-	Common ground
34	DIO 12	Bi-direction	Digital I/O 12
35	DIO 13	Bi-direction	Digital I/O 13
36	DIO 14	Bi-direction	Digital I/O 14
37	DIO 15	Bi-direction	Digital I/O 15
38	DIO 16	Bi-direction	Digital I/O 16
39	Unused for U120816 “TriggerO” for U1208LOG	Output as “TriggerO”	For U1208LOG model, it’s video or LOG trigger out line
40	GND	-	Common ground

Figure 3. Terminal descriptions

5.3 +5V

This is a nominal +5 volt internal power supply (output). Power can be drawn from this power supply by connecting to the +5V terminal. The total output current drawn by the whole device connections (including this +5V power supply, the digital outputs, PWM outputs) should not exceed 450 mA for most desktop computers and self-powered USB hubs. We would recommend only draw around 250mA from this power terminal.

Some notebook computers and bus-powered hubs will limit total USB current for a device to about 100 mA. So the maximum current can be drawn from this power terminal

and all digital outputs, PWM outputs should not exceed 50mA because at power up, the device itself consumes 50mA.

Caution! The +5V terminal is an output. Do not connect an external power supply to this terminal or you may damage the iUSBDAQ – U120816 and possibly the computer.

5.4 AGND

This is the analog ground for analog input channels. In order to reduce noise, please use these ground only for analog inputs.

5.5 AI 0 - AI 7

These are 8 single ended analog input channels. It has a range of 0- 4.096V, 12 bit, so the resolution would be 1mv. iUSBDAQ supports scan and streaming mode for data acquisition. User can scan single channel or scan multi-channels at once. In scan mode, the execution time is 8ms for U120816, 4ms for U1208LOG. So this will make up125HZ per channel in scan mode (software timed data acquisition) for U120816 and 250HZ for U1208LOG. In streaming mode, the device can start streaming the data to PC right the way or waiting for external trigger. This is software configurable. In trigger mode, a high state on the trigger line will start the data acquisition. While in streaming, the LED will blink, but for higher scan rate, because LED is blinking so fast that user may not see it. The maximum throughput of streaming data varies with number of channels and they are also system dependant. For U1208LOG, beside the normal streaming mode as U120816 which can stream up to 32Ksamples/s, there is also a high speed streaming mode, which can stream up to 64Ksamples/s. Figure 5 shows the max throughputs.

Number Of Channels	Max Throughput (**samples/s**)
8	32000
7	30100
6	27000
5	25000
4	22000
3	19500
2	18400
1	13000
High speed streaming mode is for U1208LOG model only. Below is the throughput rate	
8	64000
7	57470
6	60000
5	60000

4	62520
3	61578
2	59368
1	53728

Figure 4. Table of analog streaming throughput rate

The relationship between sample rate, scan rate and number of channels is as below:

SampleRate=ScanRate*NumberOfChannels

For example, if you have 8 input channels with a scan rate of 4000, then the total sample rate would be 32ksamples/s.

For normal streaming mode, the minimum sample rate has to be 128. If lower than this required, user can use scan mode or over sampling. For example if you have two channels, the minimum scan rate has to be 64 in order to make up the minimum sample rate of 128.

For high speed streaming mode, the minimum sample rate can be 1sample/s due to different design method.

While streaming data in normal streaming mode, other functions can still be executed, such as read/write digital lines etc. but in high scan rate, the performance may be affected since the hardware CPU is shared doing other things besides hardware timed data acquisition. And overall performance is also system dependant.

In high speed streaming mode, no other functions should be executed in iUSBDAQ – U1208LOG except the stop streaming command.

Figure 6 shows a typical connection to the single ended analog input.

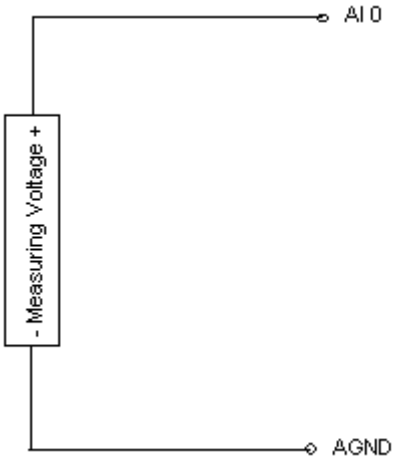


Figure 5. Typical connection to an analog input channel

5.6 Digital I/O

There are 16 bi-directional digital I/O lines in iUSBDAQ module. Each line is software configurable to be either input or output high or output low. At power-up or reset, all lines are set to be input by default. All digital lines are CMOS output and TTL level or Smith trigger input. All channels include a 470 Ω series resistor that provides overvoltage/short-circuit protection. Each channel also has a 1 MΩ resistor connected to ground.

Figure 7 shows a typical usage of a digital input for measuring the state of a switch. If switch is on +5V, the DIO 1 will read state TRUE, else if the switch is on GND, the DIO 1 will read the state of FALSE.

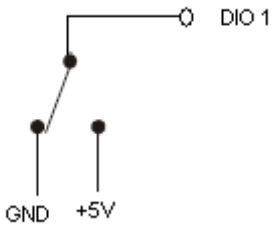


Figure 6. Typical usage of digital input for measuring the state

Caution! Do not connect a power source to DIO pins that are set to be output and do not connect a higher than specified voltage (see specification) to DIO input pins.

There is one DIO function used for read/write states and set pin directions for all DIO lines. It takes 8ms to execute for U120816 and 4ms for U1208LOG, so the update rate for DIO will be 125HZ for U120816 and 250HZ for U1208LOG. It's the high level application responsibilities to backup the existing states and directions for each pin if user only wants to update one pin at a time.

5.7 Trigger

This is the external trigger line for streaming mode data acquisition. It is software configurable if user wants to wait for this line. When in use, as soon as the line goes high, the device will start hardware timed data acquisition. This is a TTL level input.

5.8 PWM

There are two 10-bit PWM outputs in iUSBDAQ. Software can set period and duty cycle for each channel separately. The period is between 3 micro second to 333 micro second or the frequency is 333k to 3k. Duty cycle can be set between 0 – 100% or 0-1023 as

integer. Both PWM should be set to the same frequency, the last set frequency will be in effect. There are some protection circuits at the output of PWM, but user should still be careful not to connect power source to this PWM outputs!

The PWM outputs can be used to lit LEDs, drive motors, or convert to analog outputs by adding an extra circuit. See application note from microchip:

www.microchip.com
AN538 – Using PWM to generate analog output

Caution! Since PWM channels are outputs, please do not connect power source to these terminals, it may damage the device.

5.9 Counter

iUSBDAQ has a 16-bit counter. Software has the option to select if the counter should be reset after the read or not. The counter is incremented when it detects a falling edge followed by a rising edge. If you reset the counter while your signal is low, you will not get the first count until it goes high-low-high again.

5.10 GND

This is the common ground. All GNDs are the same. The grounds can be used for all terminals except analog input channels, which should use AGND.

5.11 EEPROM

iUSBDAQ has 240 bytes memory space reserved for user data. Read/write EEPROM execution time is 8ms for U120816 model and 4ms for U1208LOG model.

5.12 Video/Log TriggerO line (U1208LOG only)

IUSBDAQ – U1208LOG has a dedicated trigger out line at screw terminal pin39. It sends out pulses whenever there is video frame coming by in iDAQTest&Log software. Or when in standalone logger mode, it will also send out pulses whenever the data logging happens.

5.13 USB Flash Drive Logger (U1208LOG only)

IUSBDAQ – U1208LOG model can also be used as stand alone data logger. The connector USB2 is used for connecting USB flash disk, while USB1 connector is used for PC control or for USB power adaptor. The logging interval by default is 1 second, user can change this value in iDAQTest&Log software. It's between 1 second to 12 hours.

5.14 Switch (U1208LOG only)

IUSBDAQ – U1208LOG has a switch located beside the USB2 connector. On switching to PC/Stop side, this will allow the module to be controlled by PC. By switching to LOG/Start side, it will work as a standalone data logger and PC will not recognize this device when in LOG mode. **It is strongly recommended that user should always switch to PC/Stop mode before unplug the USB flash disk from USB2 connector, to avoid data lose.**

5.15 On Board Real Time Clock (U1208LOG only)

IUSBDAQ – U1208LOG has a real time clock on PCB board that is powered by a 3V lithium battery. This battery is exchangeable. The time of the clock can be set at iDAQTest&Log software. This clock is used to time stamp the log file and logging data in stand alone logger mode.

5.16 Automatic File Name Generation (U1208LOG only)

When in logger mode, iUSBDAQ – U1208LOG will automatically generate a new log file name whenever the USB flash disk is inserted into USB2 connector or when the log file has logged over 30,000 scans of data (each scan includes 8 analog inputs data, 16 digital lines data and one counter data), it will also create a new file name. The file name is starting with "DL" following by the time stamp (DL[Month][Day][Hour].[Minutes]). For example this is a file name "DL060413.45". This will prevent data lose in big files and make easy for raw data file conversion. The logged raw data files can be converted to HYTEK format data files with iDAQTest&Log software. **Always use the switch to stop data logging before unplug the USB disk from USB2 connector, to prevent data lose.**

4 iDAQTest&Log Software

5.17 Overview

iDAQTest&Log is a standalone ready-to-run software that can be used to test the iUSBDAQ device and log analog input's data and read back logged data for basic analysis. With this software, user can browser through all the features in iUSBDAQ, such as read single analog channel, multiple analog channels, streaming analog input data continuously for display and log into file with or without external trigger, set DIO directions and read/write DIO states, set PWM output, read counter etc. Version 2.0 also adds the video functions for U1208LOG module.

Figure 8 shows an overall view of the iDAQTest&Log screenshot. Details of how each buttons, controls work, will also be explained graphically.

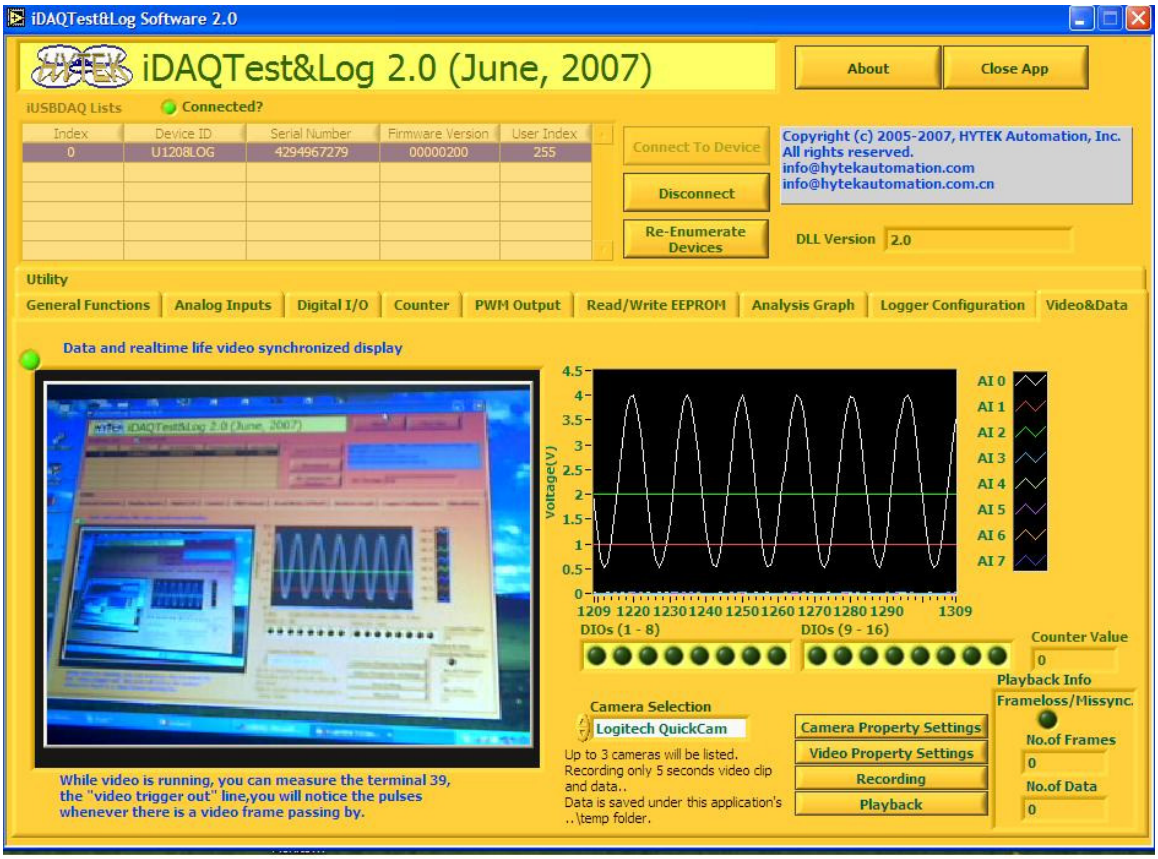


Figure 7. Overview of iDAQTest&Log software

There are 10 tabs in this software. Below will go over each corner of the software and explain the operations.

5.18 Top Part

(This is v1.0 top, will update to v2.0)

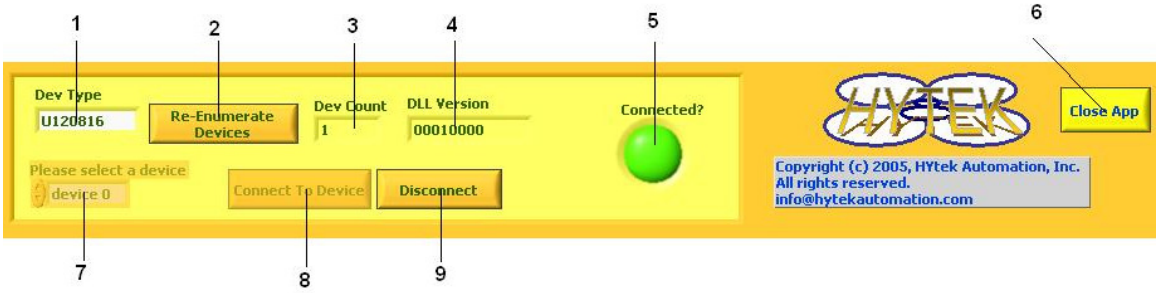


Figure 8. Top part of the iDAQTest&Log software

1: This is the device type selection field. For iDAQTest&Log version 1.0, there is only one device type available, which is iUSBDAQ.

2: This button will cause the software to close all opened sessions and re-enumerate all the iUSBDAQs connected to the computer. The total number of found devices will be displayed in field 3.
3: Display the total number of iUSBDAQs found in a computer.
4: Display the iUSBDAQ.dll version used for this application.
5: If light green, means the selected device's session is opened and connected successfully.
6: Press this button will exit the whole application.
7: This list box lists all the devices of the selected device type (field 1). User can select one device by drop down the list box.
8: Press this button will try to connect to the selected device in field 7. If successful, the LED indicator in field 5 will turn to light green and the tabs will be enabled.
9: Press this button will disconnect the device and the tabs will disabled, user will see all the tabs are grey out.

5.19 General Functions Tab

In the general functions' tab, the information such as firmware version, serial number and designer will be displayed.

Figure 9. General Functions Tab

5.20 Analog Inputs Tab

In this tab, user can scan one single analog channel, or scan multiple channels or start streaming mode continuous data acquisition. The max number of channels is 8 and it depends on which channel is starting channel. The following condition has to meet:

StartingChannel+Number of Channels<=8

The max scan rate that can be set varies with the number of channels. Here is the max throughput table again:

Number Of Channels	Max Throughput (**samples/s**)
8	32000
7	30100
6	27000
5	25000
4	22000
3	19500
2	18400
1	13000
High speed streaming mode is for U1208LOG model only. Below is the throughput rate	
8	64000
7	57470
6	60000
5	60000
4	62520
3	61578
2	59368
1	53728

Figure 10. Max throughput table

The relationship between sample rate, scan rate and number of channels is as below:

SampleRate=ScanRate*NumberOfChannels

For example, if you have 8 input channels with a scan rate of 4000, then the total sample rate would be 32ksamples/s.

The minimum sample rate128 is required in normal streaming mode. If lower than this required, user can use scan mode. For example if you have two channels, the minimum scan rate has to be 64.

For high speed streaming mode, the minimum sample rate can be 1sample/s due to different design method.

While streaming data in normal streaming mode, other functions can still be executed, such as read/write digital lines etc. but in high scan rate, the performance may be affected since the hardware CPU is shared doing other things besides hardware timed data acquisition. And overall performance is also system dependant.

In high speed streaming mode, no other functions should be executed in iUSBDAQ – U1208LOG except the stop streaming command.

If wrong number of channels or scan rate is given, the software will popup a dialog box, telling the error code and error source. Just click on OK and give a correct parameter, the software will continue.

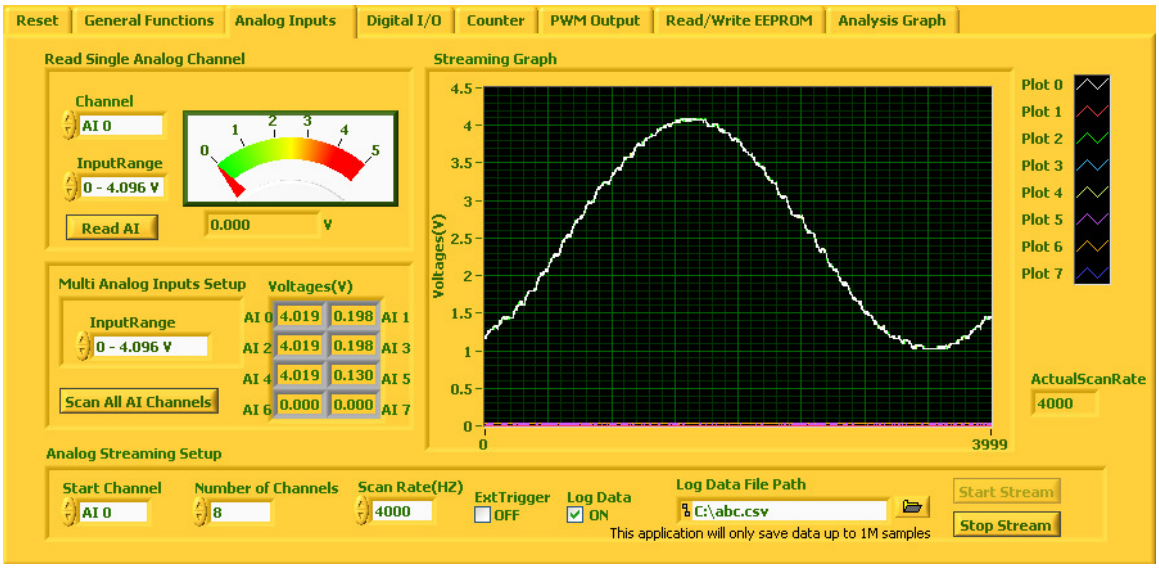


Figure 11. Analog Inputs Tab

By checking the "Log Data" box and specify a file path, the data will be logged into the file. The iDAQTest&Log software allows up to 10M samples to be logged. We recommend to use file ending *.csv so that the data file can be easily opened in Excel. Figure 14 shows a sample log file in Excel. The logged data file can be also opened in "Analysis Graph Tab" (see session 4.9) and perform some basic analysis.

HYTEK iUSBDAQ Data File
12:33:02 30/07/2005

Start Channel=0	Number of Channels=3	ScanRate=2000HZ
0.528	0	0.528
0.544	0	0.528
0.528	0.032	0.544
0.528	0.032	0.528
0.528	0.032	0.528
0.528	0.032	0.528
0.528	0.032	0.528
0.528	0.032	0.528
0.544	0.032	0.544
0.528	0.032	0.528
0.528	0.032	0.528
0.528	0.032	0.528
0.528	0.032	0.544
0.528	0.032	0.528
0.544	0.032	0.528

0.528	0.032	0.528
0.544	0.032	0.528
0.528	0.032	0.528
0.544	0.032	0.528
0.528	0.032	0.528
0.528	0.032	0.528

Figure 12. Sample Data Log File

If the "Ext Trigger" box is checked, after user press the "Start Stream" button, the device will wait for a high state on the trigger line to start data acquisition.

5.21 Digital I/O Tab

In this tab user can set digital I/O directions, write output lines and read the DIO states. And caution should be taken, that no power source is connected to output DIO lines.

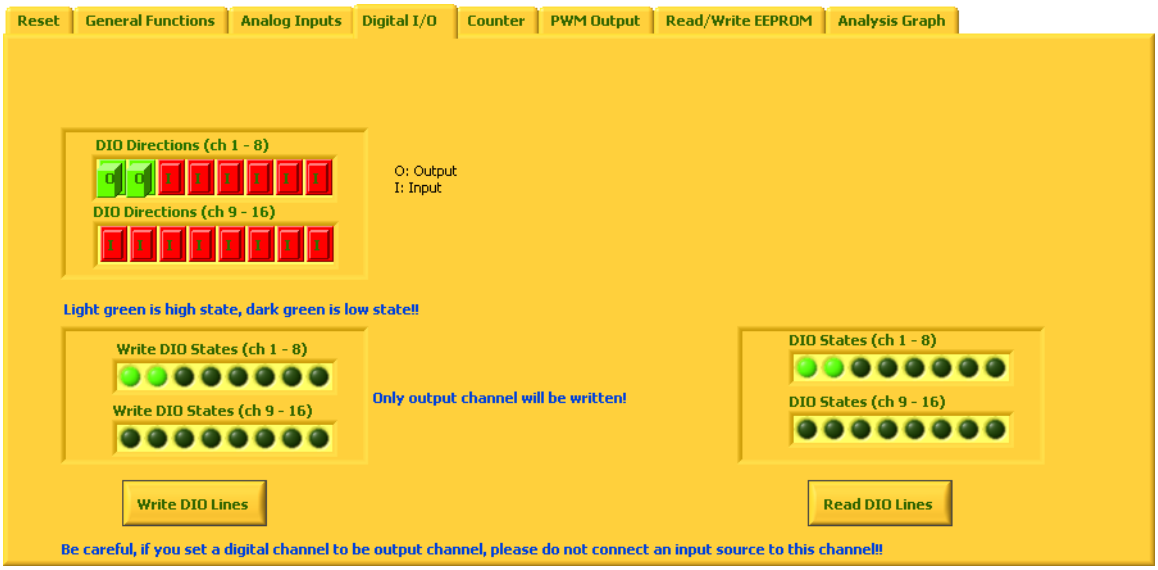


Figure 13. Digital I/O Tab

5.22 Counter Tab

In this counter tab, user can read counter, reset counter. A rising edge on the counter pin will increment counter. Counter will roll over after 65536 (16 bit).

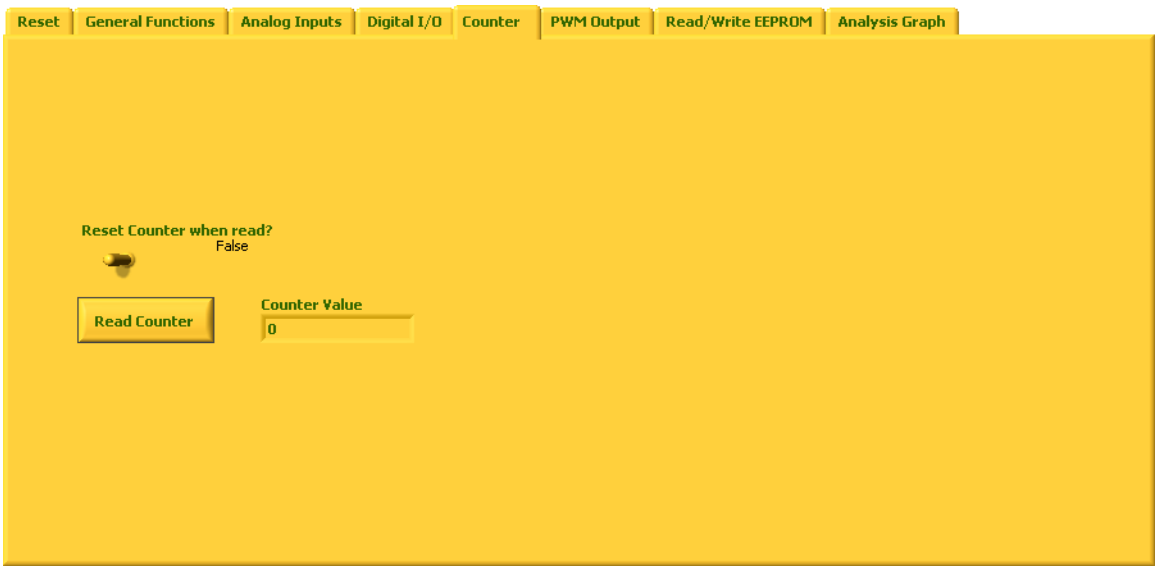


Figure 14. Counter Tab

5.23 PWM Output Tab

Set the PWM output. There are two 10-bit PWM channels, ch1 and ch2. The PWM period is between 3- 333micro second. Duty cycle is between 0 -100.

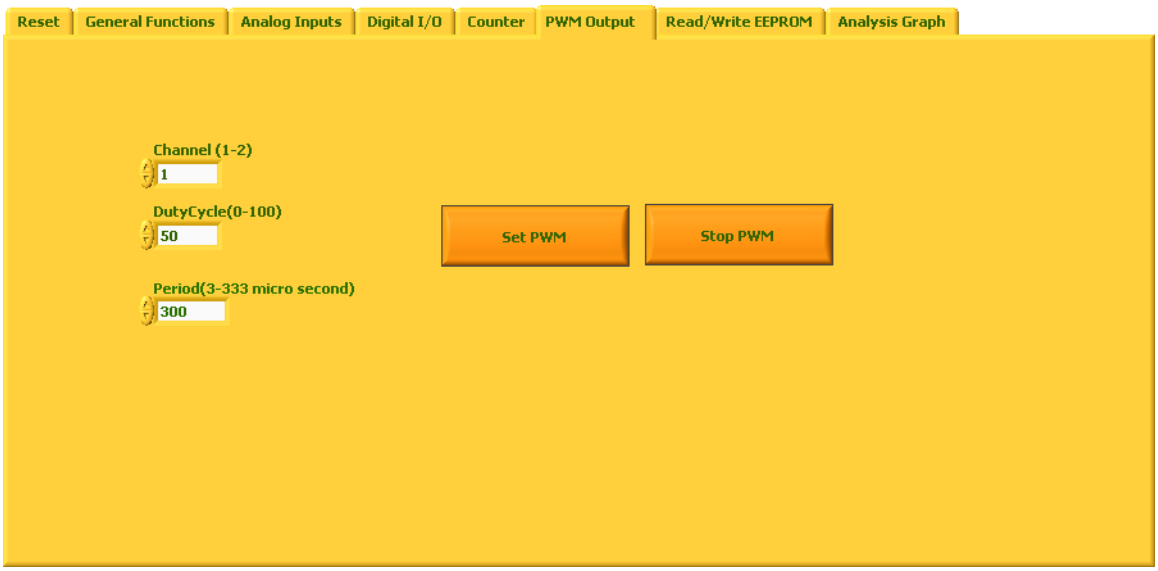


Figure 15. PWM Output Tab

5.24 Read/Write EEPROM Tab

iUSBDAQ has 240 bytes memory space reserved for user data. In this tab, user can write data into EEPROM at a specified address and read back from a specified address. The following condition has to meet:

Starting Address+Bytes To Read/Write<=240

If the above condition is not met, a popup dialog box will appear, telling the error code and error source. Just click on OK and continue using the software by giving the correct address or number of bytes to read/write.

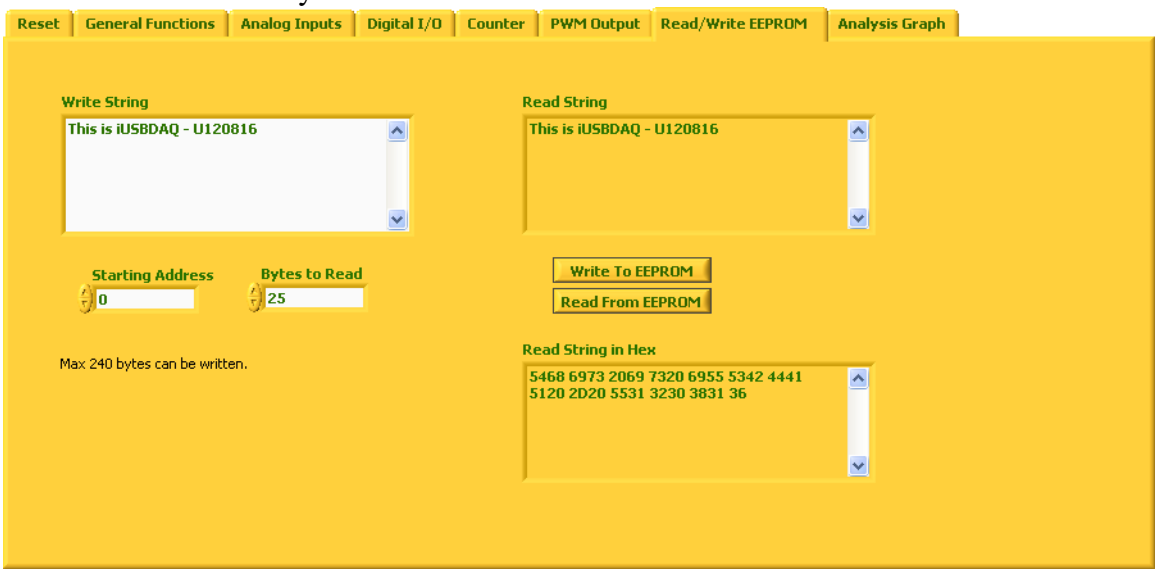


Figure 16. Read/Write EEPROM Tab

5.25 Analysis Graph Tab

Use can load a saved data file in this tab. iDAQTest&Log provides limited analysis function, user can view the multi-plot graph. Two cursors can be used for measuring the data points. The XY axis by default is in auto scale mode, right click on the graph, a popup menu will appear, user can turn auto scale on or off or just click on the axis' number to change the graph range. Future version will provide more analysis functions in this tab.

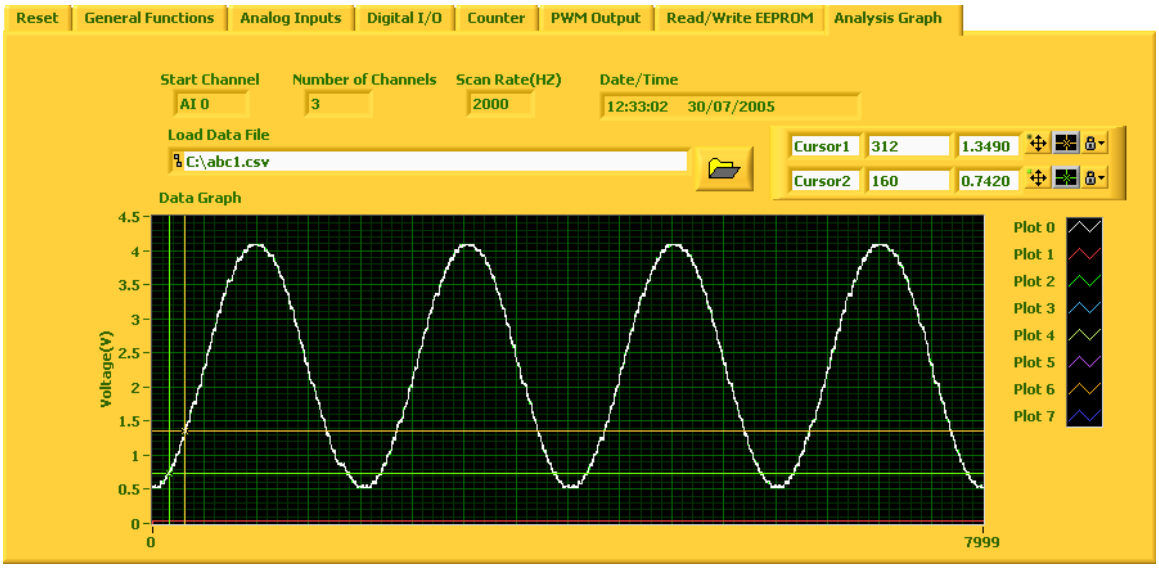


Figure 17. Analysis Graph Tab

5.26 Utility Tab (U1208LOG only)

In this tab, user can convert the raw logging file from U1208LOG to HYTEK formatted data file so that it can be viewed and analyzed.

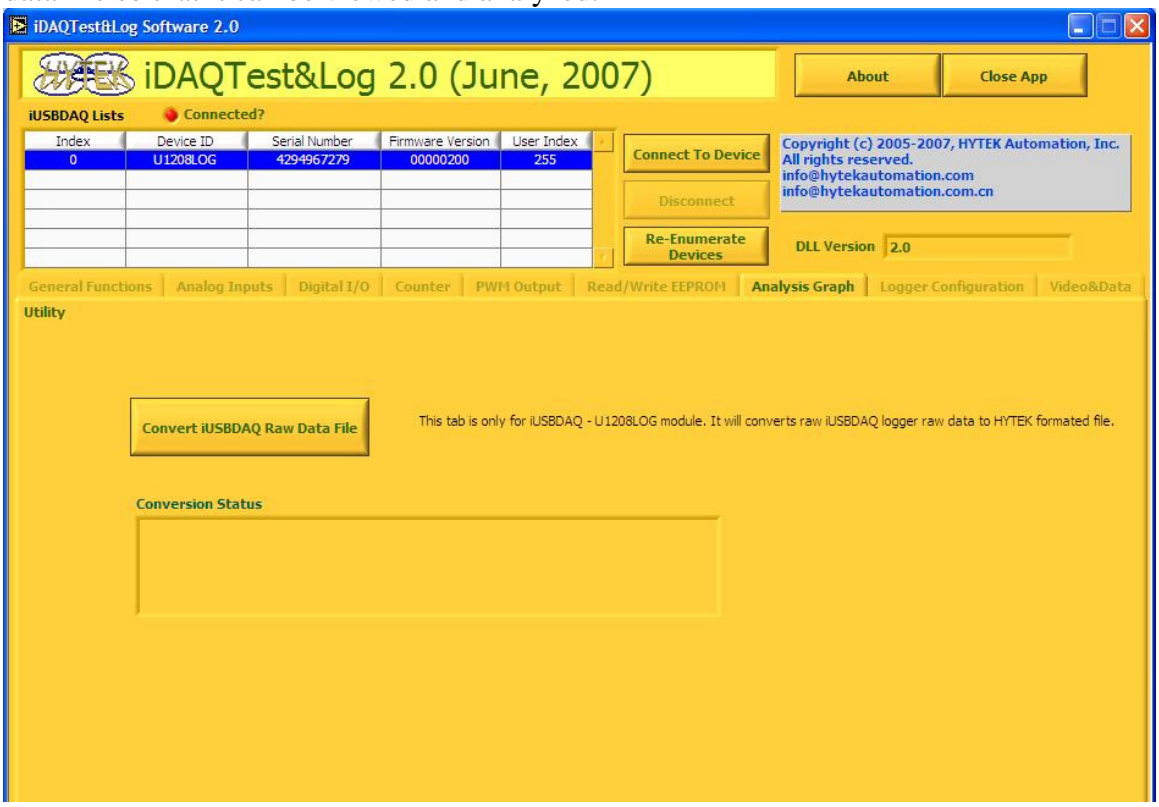


Figure 19: Utility Tab

The converted data is in the order of:
AI0,AI1,AI2,AI3,AI4,AI5,AI6,AI7,DIO1-8 (in Byte),DIO9-16(in Byte),Counter, Year, Month, Day, Hour, Minute, Second

5.27 Logger Configuration Tab (U1208LOG only)

In this tab, user can set the real time clock, read the current date/time from real time clock. User can also set the logging interval between 1 second to 12 hours.

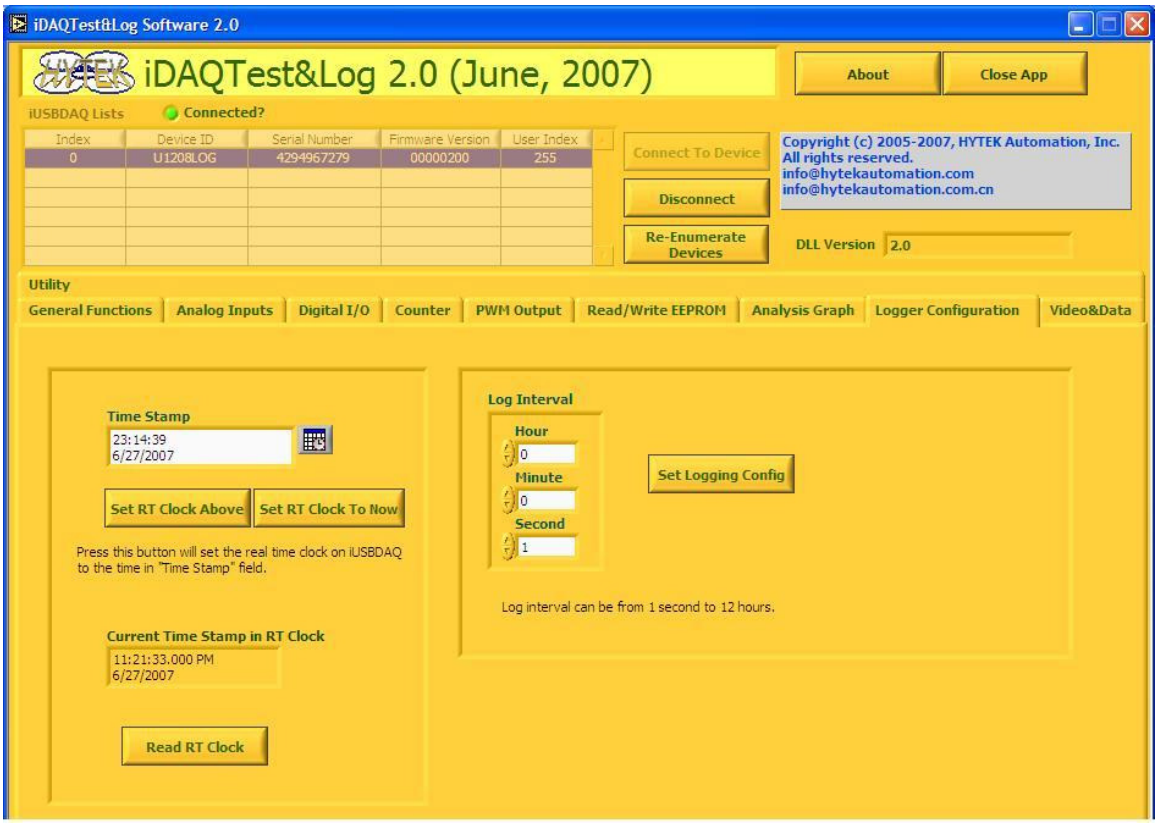


Figure 20: Logger Configuration Tab

5.28 Video&Data Tab (U1208LOG only)

In this tab, user can select a directshow supporting USB webcam or firewire camera. It will display live video synchronized with data scan (8 analog inputs, 16 DIOs, one counter), record AVI file and data, playback AVI file with data etc. Up to 3 cameras are listed and up to 5 seconds video clip and data can be recorded with this iDAQTest&Log software.

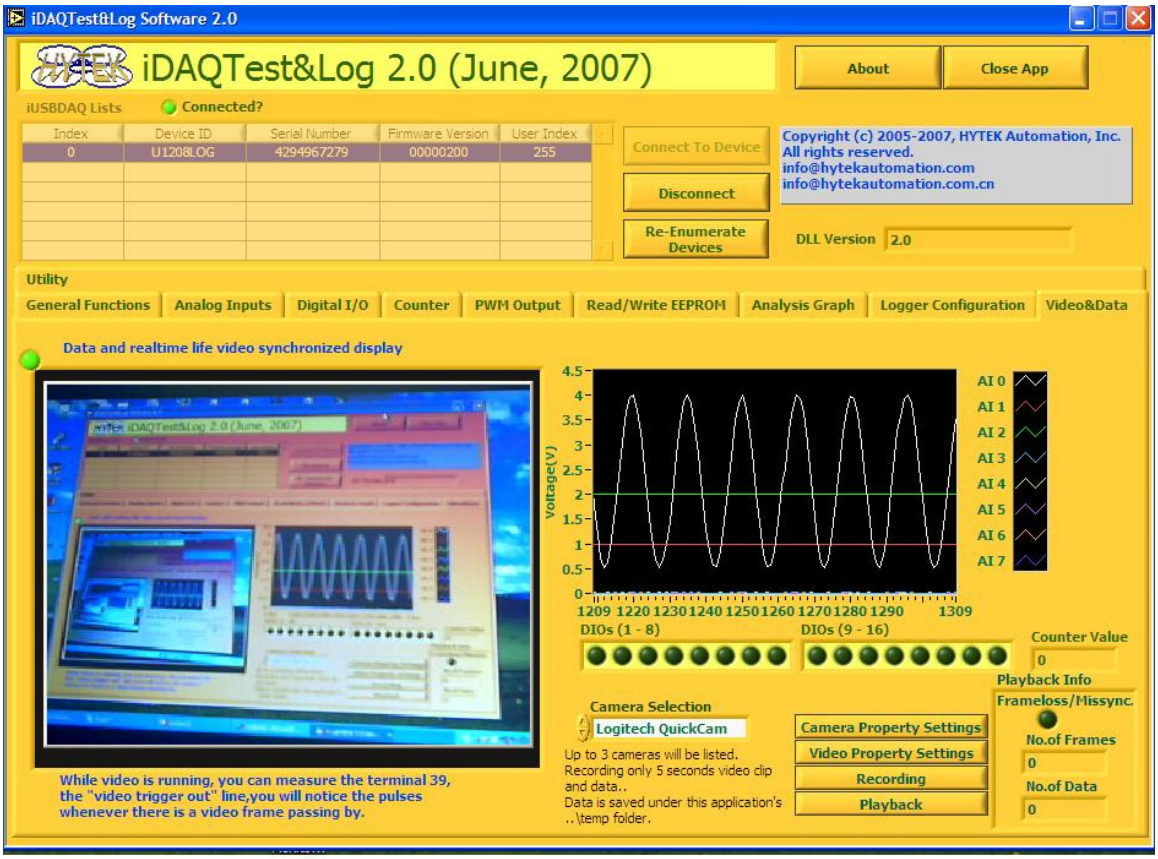


Figure 21: Video and Data Display tab

6 Programming Reference (iUSBDAQ.dll)

The iUSBDAQ device driver installs the programming API iUSBDAQ.dll into the computer system's directory. This DLL is called by user application.

6.1 DevSession Typedef Structure

This is the device session typedef structure definition.

```
typedef struct
{
        int                     DevIndex;
        int                     DevInstance;
        unsigned long           DevType;
        HANDLE                  iSession1;
        HANDLE                  iSession2;
} DevSession;
```

6.2 DevInfo Typedef Structure

```
typedef struct
{
```

```
    unsigned long serialNumber;
    BYTE DevID;
    BYTE UserIndex;
    DWORD Version;
}DevInfo;
```

6.3 iUSBDAQ_EnumerateDev

This is the first function that need to be called for a selected device type before using any iUSBDAQ functions. It enumerates the computer to create a mapping of the iUSBDAQs, allocates some internal memory. If any device's sessions were opened before, they will be closed by this function in order to create the internal map. It also returns the total number of iUSBDAQs of the selected type. Normally this function only need to be called once at very beginning of application, but user may want to re-enumerate if some devices are disconnected while in use to make sure the mapping is updated.

Syntax:

```
int iUSBDAQ_EnumerateDev (unsigned long DevType,
                          unsigned long* Count
                         );
```

Parameters:

DevType: input. Device type, currently we have only U120816, so it's value should be 0.
Count: output. It's the total number of all found iUSBDAQs of the selected type.

Return:

iUSBDAQ error code or 0 for no error

6.4 iUSBDAQ_OpenDevice

This function opens an iUSBDAQ session. This session will be used to call other device related functions. The order of using a device function is enumeration, open session, calling device related functions... at very end close device session. There is no need to enumerate, open session and close session every time.

Syntax:

```
int iUSBDAQ_OpenDevice ( unsigned long DevType,
                         unsigned long DevIndex,
                         DevSession* Session
                         );
```

Parameters:

DevType: input. Device type, currently we have only U120816, so it's value should be 0.
DevIndex: input. It's the index of iUSBDAQs of the selected type that you want to use.
Session: Output. This is the session that being opened and will be used by other device related function calls.

Return:
iUSBDAQ error code or 0 for no error

6.5 iUSBDAQ_Reset

Reset the device.

Syntax:
int iUSBDAQ_Reset (DevSession* Session);

Parameters:
Session: input, device session that being opened.

Return:
iUSBDAQ error code or 0 for no error

6.6 iUSBDAQ_ReleaseDevice

This function closes an opened device session. Normally user only needs to close a device session when they don't use that device anymore. The order of using a device function is enumeration, open session, calling device related functions... at very end close device session. There is no need to enumerate, open session and close session every time.

Syntax:
int iUSBDAQ_ReleaseDevice (DevSession* Session);

Parameters:
Session: input, device session that being opened.

Return:
iUSBDAQ error code or 0 for no error

6.7 iUSBDAQ_GetDeviceSerialNo

This function returns an iUSBDAQ device serial number. Each iUSBDAQ has an unique serial number. User can use this unique ID to distinguish the devices if there are more than one iUSBDAQ connected to the same computer.

Syntax:
```
int iUSBDAQ_GetDeviceSerialNo   (DevSession* Session,
                                 unsigned long* SerialNumber
                                 );
```

Parameters:
Session: input, device session that being opened.

SerialNumber: output, device serial number.

Return:
iUSBDAQ error code or 0 for no error

6.8 iUSBDAQ_GetCredits

This function returns the designer's information.

Syntax:
```
int iUSBDAQ_GetCredits (   DevSession* Session,
                           BYTE* data
                           );
```

Parameters:
Session: input, device session that being opened.
data: output, an array of size 64, type of BYTE. User should allocate this array. It returns the designer's info.

Return:
iUSBDAQ error code or 0 for no error

6.9 iUSBDAQ_GetDLLVersion

This function returns the iUSBDAQ.dll version number.

Syntax:
```
DWORD iUSBDAQ_GetDLLVersion ();
```

Parameters:
none

Return:
iUSBDAQ.dll version number

6.10 iUSBDAQ_GetFirmwareVersion

This function returns the iUSBDAQ firmware version number.

Syntax:
```
int iUSBDAQ_GetFirmwareVersion (DevSession* Session,
                                DWORD* Version
                                );
```

Parameters:
Session: input, device session that being opened.
Version: output, iUSBDAQ.dll version number

Return:
iUSBDAQ error code or 0 for no error

6.11 iUSBDAQ_GetErrorDes

This function takes an iUSBDAQ error code and outputs the error string for detailed error source.

Syntax:
```
void iUSBDAQ_GetErrorDes (int ErrorCode,
                          char* ErrorString
                         );
```

Parameters:
ErrorCode: input, iUSBDAQ error code.
ErrorString: output, error description string. User should allocate a string length of 256 byte.

Return:
none

6.12 iUSBDAQ_ReadSingleAnalogIn

This function reads voltage from one analog channel.

Syntax:
```
int iUSBDAQ_ReadSingleAnalogIn ( DevSession* Session,
                                 int      Channel,
                                int InputRange,
                                 float * Voltage,
                                 int * Reserved
                                );
```

Parameters:
Session: input, device session that being opened.
Channel: input, the channel to read data from, range 0 -7, 0 is AI 0 and so on.
InputRange: input, since currently iUSBDAQ has only one voltage range, so this parameter should be 0 for now.
Voltage: output, the read data from the analog channel.
Reserved: output, reserved parameter, it always returns 0 for now.

Return:
iUSBDAQ error code or 0 for no error

6.13 iUSBDAQ_ReadMultiAnalogIn

This function reads voltage from multiple analog channels at once.

Syntax:
int iUSBDAQ_ReadMultiAnalogIn (DevSession* Session,
int StartCh,
int NofChs,
int InputRange,
float *Voltages,
int * Reserved
);

Parameters:
Session: input, device session that being opened.
StartCh: input, the starting channel, range 0 -7, 0 is AI 0 and so on.
NofChs: input, number of channels to read data from starting from the channel StartCh. The maximum number of channels is 8, and it depends on which channel is the starting channel. The StartCh+NofChs should not exceed 8.
InputRange: input, since currently iUSBDAQ has only one voltage range, so this parameter should be 0 for now.
Voltages: output, the read data from the analog channels. User should allocate this array, it should has a size of NofChs with floating data type.
Reserved: output, reserved parameter, it always returns 0 for now.

Return:
iUSBDAQ error code or 0 for no error

6.14 iUSBDAQ_AIStartStream

This function starts the streaming mode analog inputs acquisition. This function returns right the way after the call and the streaming is in background with an internal pc side data buffer. Please read more about this data buffer in " iUSBDAQ_AIGetScans" description.

The maximum number of channels is 8, and it depends on which channel is the starting channel. The StartChannel+NumberOfChs should not exceed 8.

The maximum sampling throughput varies with number of Channels. Please refer to the hardware part of iUSBDAQ User's Guide for details. The relationship between sampling rate and scan rate is:

SampleRate=ScanRate*NumberOfChannels

For example, if scan rate is 3000, number of channels is 8, the total sample rate will be 24,000samples/s.

The minimum sample rate has to be 128. If lower than this required, user can use scan mode or over sampling. For example if you have two channels, the minimum scan rate

has to be 64 which will result a sample rate of 128samples/s.

If Ext Trigger is true, after this vi returns, the device will wait for a high state from the trigger line to start the data acquisition. False will start the data acquisition right after this call into pc side background data buffer.

This function will return actual scan rate, it may not always the same as the specified input scan rate.

This function will return an error if the channel's settings are not correct.

Syntax:
```
int iUSBDAQ_AIStartStream (DevSession* Session,
                           int StartCh,
                           int NofChs,
                           int InputRange,
                           int ScanRate,
                          int * ActualRate,
                          int withExternalTrigger
                          );
```

Parameters:
Session: input, device session that being opened.
StartCh: input, the starting channel, range 0 -7, 0 is AI 0 and so on.
NofChs: input, number of channels to read data from starting from the channel StartCh. The maximum number of channels is 8, and it depends on which channel is the starting channel. The StartCh+NofChs should not exceed 8.
InputRange: input, since currently iUSBDAQ has only one voltage range, so this parameter should be 0 for now.
ScanRate: input, the number of scans per second. A scan is the reading from every selected channels.
ActualRate: output, the returned actual scan rate. It may not always be the same as specified ScanRate because of the device internal timer.
withExternalTrigger: input, if 1, will wait for trigger line to go high, if 0, will start background streaming right after this call.
Return:
iUSBDAQ error code or 0 for no error

6.15 iUSBDAQ_AIGetScans

This function gets the data back from the internal PC side data buffer.The number of channels has to be the same as the number of channels in "iUSBDAQ_AIStartStream". User can specify how many scans need to be read back. This function only returns when all the asked scans are read. To stop in between, user needs to call the "iUSBDAQ_AIStopStream" and this function will error out and return. The timeout is for read of 128 samples, not for the total number of scans asked. 1000ms usually is a good number for a timeout. The returned data is interleaved, for example, if you start streaming

with starting channel of AI 0, number of channels is 2, then the output data is in the order of AI 0, AI 1, AI 0, AI 1..... This function also returns the actual number of scans, it may not always be the same as input number of scans. The relation between data array size and actual number of scans is:
Data array size=actual number of scans x number of channels

The minimum samples requested must not be less than 128, for example if you have two channels, then the minimum number of scans must not be less than 64. There is no limit for maximum number of scans requested.

The internal data buffer has a size of 12.8k samples, if this buffer is full, the later coming data will be ignored. Usually user should start calling this function right after start streaming.

There are two possible ways user can use this function, one way is passing in a big total number of scans to read at once (can be bigger than 12.8k because the iUSBDAQ.dll will take care of this), this function will wait until all asked scans returned. The other way is asking for a small amount number of scans, and repeat calling this function again and again. The only risk of second method is if there is too many delays between two consecutive calls of this function, the internal data buffer may be full, some data will be ignored which will result none continuous data between two calls.

In iDAQTest&Log software, it uses the second method, each time it asks for one second long data, which means the asked number of scans equals to scan rate and repeat calling this function again and again, concatenate the data to get a continuous graph. User may also want to consider using a separate data engine to handle streaming, getting data so that this action will not block other user interface actions if any.

Syntax:

```
int iUSBDAQ_AIGetScans (  DevSession* StreamSession,
                          int NumbofScans,
                          int Timeout,
                          float* buffer,
                          int* actualScans
                         );
```

Parameters:
Session: input, device session that being opened.
NumbofScans: input, number of scans requested.
Timeout: timeout for reading 128 samples in unit of ms. Usually pass in 1000ms would work for most cases.
Buffer: output, an array allocated by the user to hold the returned data. It must has a size of (Number of Scans requested x Number of channels). The actual returned data size maybe equal or smaller than that, because the actual number of scans maybe smaller than the requested number of scans. And the data is interleaved in the order of channels, for example AI0, AI1, AI0, AI1…if you have two channels starting from channel AI0.

actualScans: output, returns the actual number of scans read, it's equal or smaller than the number of scans requested.

Return:
iUSBDAQ error code or 0 for no error

6.16 iUSBDAQ_AIStopStream

This function stops the data streaming.

Syntax:
int iUSBDAQ_AIStopStream (DevSession* Session);

Parameters:
Session: input, device session that being opened.

Return:
iUSBDAQ error code or 0 for no error

6.17 iUSBDAQ_ReadCounter

This function reads the counter.

Syntax:
```
int iUSBDAQ_ReadCounter (DevSession* Session,
                         int Reset,
                          unsigned int *Count
                         );
```

Parameters:
Session: input, device session that being opened.
Reset: input, 1 will reset the counter to start over after the read. Value 0 will continue the counting after the read.
Count: output, returned counted number. For how counter works, please refer to the hardware part of this user's guide.

Return:
iUSBDAQ error code or 0 for no error

6.18 iUSBDAQ_DIO

This function reads the counter. The directions and states are represented in a byte, which is 8 bits. The lowest bit represents the lowest DIO pin number and highest bit represents the highest DIO pin number.
Syntax:

```
int iUSBDAQ_DIO (  DevSession* Session,
                   BYTE Port1to8dir,
                   BYTE Port9to16dir,
                   BYTE *Port1to8state,
                   BYTE *Port9to16state,
                    int updateDio
                   );
```

Parameters:
Session: input, device session that being opened.
Port1to8dir: input, one byte represents DIO pin 1-8 directions. If BYTE data type expressed in binary, 1 for input, 0 for output. For example, 11011110 would result a value of 222, it represents that all pins from 1 -8 are inputs pins except pin1 and pin6 are output pins.
Port9to16dir: input, one byte represents DIO pin 9 – 16 directions.
Port1to8state: input and output, DIO 1 – DIO 8 states. For pins that directions being set to output, if "updateDio" is 1, which means write/update output pins, the states will be written and the actual states will be read back and returned to the same parameter. Or if "udateDio" is 0, only read action is performed and returned back to this parameter.
Port9to16state: input and output, DIO 9 – DIO 16 states. For pins that directions being set to output, if "updateDio" is 1, which means write/update output pins, the states will be written and the actual states will be read back and returned to the same parameter. Or if "udateDio" is 0, only read action is performed and returned back to this parameter.
updateDio: if 1, a write/update action will be performed for all pin directions and output pins' states and read action will be performed for all DIO pin's states. If 0, only read action will be performed for all DIO pin's states.

Return:
iUSBDAQ error code or 0 for no error

6.19 iUSBDAQ_PWMOut

This function sets the PWM output channels and parameters.

Syntax:

```
int iUSBDAQ_PWMOut (    DevSession* Session,
                        BYTE Channel,
                        int DutyCycle,
                        int Period
                        );
```

Parameters:
Session: input, device session that being opened.
Channel: input, PWM channel. Either 1 or 2.
DutyCycle: input, value between 0 – 100.

Period: input, PWM period, value between 3us to 333 us which results a frequency of 333kHz to 3kHz pulse. Both channels should have the same frequency or period, the last set frequency will be used for both channels.

Return:
iUSBDAQ error code or 0 for no error

6.20 iUSBDAQ_STOPPWM

This function stops the PWM output of the selected channel.

Syntax:
```
int iUSBDAQ_STOPPWM ( DevSession* Session,
                      int Channel
                      );
```

Parameters:
Session: input, device session that being opened.
Channel: input, PWM channel. Either 1 or 2.

Return:
iUSBDAQ error code or 0 for no error

6.21 iUSBDAQ_ReadMemory

This function reads data from EEPROM.

Syntax:
```
int iUSBDAQ_ReadMemory (DevSession* Session,
                        BYTE ByteCount,
                        BYTE Address,
                        BYTE* Data
                        );
```
Parameters:
Session: input, device session that being opened.
ByteCount: input, number of bytes to read. The following condition must meet: Address+ByteCount<=240 because there are only 240 bytes space in EEPROM reserved for user data.
Address: input, the starting address to read data from.
Data: output, an array that holds the data read back from EEPROM. An array of type BYTE, the size of the array should not be smaller than the “ByteCount”.

Return:
iUSBDAQ error code or 0 for no error

6.22 iUSBDAQ_WriteMemory

This function writes data to EEPROM.

Syntax:
int iUSBDAQ_WriteMemory (DevSession* Session,
BYTE ByteCount,
BYTE Address,
BYTE* Data
);

Parameters:
Session: input, device session that being opened.
ByteCount: input, number of bytes to write. The following condition must meet: Address+ByteCount<=240 because there are only 240 bytes space in EEPROM reserved for user data.
Address: input, the starting address to write data to.
Data: input, an array that holds the data to write to the EEPROM.

Return:
iUSBDAQ error code or 0 for no error

7 iUSBDAQ.DLL v2.0 New added Functions

Below are the new added functions in iUSBDAQ.dll version2.0. Unless otherwise specified, these functions will apply to both U120816 and U1208LOG models.

7.1 iUSBDAQ_10bitPWMOut

This function sets duty cycle of 10bit PWM output in integer value from 0-1023.

Syntax:
int iUSBDAQ_10bitPWMOut (
DevSession* Session,
BYTE Channel,
int DutyCycle,
int period);

Parameters:
Session: input, device session that being opened.
DutyCycle: input, value from 0-1023.
Period: input, PWM period, value between 3us to 333 us which results a frequency of 333kHz to 3kHz pulse. Both channels should have the same frequency or period, the last set frequency will be used for both channels.

Return:
iUSBDAQ error code or 0 for no error

7.2 API iUSBDAQ_OpenDeviceBySN

This function opens device session by unique serial number. This is very useful when there are multiple iUSBDAQs in the computer, this prevents the miss ordering of USB DAQs because Window OS will not remember the order of USB DAQs if they are plugged into the different USB ports.

Syntax:
iUSBDAQ_OpenDeviceBySN(
unsigned long SerialNumber,
DevSession* Session);

Parameters:
Session: output, device session that being opened.
SerialNumber: input, the unique device serial number.

Return:
iUSBDAQ error code or 0 for no error

7.3 iUSBDAQ_OpenDeviceByUserIndex

This function opens device session by user defined device index. The user device index can be set in iDAQTest&Log software's general Tab. They should be set to an unique value to avoid mixing up. This helps to manage the multiple iUSBDAQs in the same computer because Windows OS will not remember the ordering of USB DAQs if they are plugged into different USB ports.

Syntax:
int iUSBDAQ_OpenDeviceByUserIndex(unsigned long UserIndex,
DevSession* Session);
Parameters:
Session: output, device session that being opened.
UserIndex: input, the user defined index. They should be unique for each device.

Return:
iUSBDAQ error code or 0 for no error

7.4 iUSBDAQ_GetDeviceInfo

This function gets device information such as serial number, user index, device ID and firmware version etc.

Syntax:
Int iUSBDAQ_GetDeviceInfo(unsigned long DevIndex,
DevInfo *devInfo);

Parameters:
DevIndex: input, device index.
devInfo: output, please refer to DevInfo typedef.

Return:
iUSBDAQ error code or 0 for no error

7.5 iUSBDAQ_AIStartStream_HS (U1208LOG only)

This function starts the high speed streaming mode of analog inputs acquisition. It has identical parameters as function “iUSBDAQ_AIStartStream”, but allows higher speed sampling. For high speed sampling rate please refer to chapter 5.5 of this document. This function returns right the way after the call and the streaming is in background with an internal pc side data buffer. Please read more about this data buffer in " iUSBDAQ_AIGetScans" description.

The maximum number of channels is 8, and it depends on which channel is the starting channel. The StartChannel+NumberOfChs should not exceed 8.

The maximum sampling throughput varies with number of Channels. Please refer to the hardware part, chapter 5.5 of this document for details. The relationship between sampling rate and scan rate is:

SampleRate=ScanRate*NumberOfChannels

For example, if scan rate is 3000, number of channels is 8, the total sample rate will be 24,000samples/s.

If Ext Trigger is true, after this function returns, the device will wait for a high state from the trigger line to start the data acquisition. False will start the data acquisition right after this call into pc side background data buffer.

This function will return actual scan rate, it may not always the same as the specified input scan rate.

This function will return an error if the channel's settings are not correct.

Note: in high speed streaming mode, the hardware will not response to any other commands except the “iUSBDAQ_AIStopStream_HS” below. Please do not call any other functions that will cause the communication with the hardware while in high speed streaming, otherwise it will cause error.

Syntax:
int iUSBDAQ_AIStartStream_HS (DevSession* Session,

```
 int StartCh,
 int NofChs,
 int InputRange,
 int ScanRate,
int * ActualRate,
int withExternalTrigger
);
```

Parameters:
Session: input, device session that being opened.
StartCh: input, the starting channel, range 0 -7, 0 is AI 0 and so on.
NofChs: input, number of channels to read data from starting from the channel StartCh. The maximum number of channels is 8, and it depends on which channel is the starting channel. The StartCh+NofChs should not exceed 8.
InputRange: input, since currently iUSBDAQ has only one voltage range, so this parameter should be 0 for now.
ScanRate: input, the number of scans per second. A scan is the reading from every selected channels.
ActualRate: output, the returned actual scan rate. It may not always be the same as specified ScanRate because of the device internal timer.
withExternalTrigger: input, if 1, will wait for trigger line to go high, if 0, will start background streaming right after this call.

Return:
iUSBDAQ error code or 0 for no error.

7.6 iUSBDAQ_AIStopStream_HS (U1208LOG only)

This function stops the data streaming in high speed mode.

Syntax:
int iUSBDAQ_AIStopStream_HS (DevSession* Session);

Parameters:
Session: input, device session that being opened.

Return:
iUSBDAQ error code or 0 for no error

7.7 EnumerateCameras (U1208LOG only)

This function will enumerate cameras in the system and return the total number of cameras found. IUSBDAQ.dll limits number of cameras to 3. This function should be called once and at first before using any camera/video related functions.

Syntax:

int EnumerateCameras(DevSession* Session);

Parameters:
Session: input, device session that being opened.

Return:
Number of cameras in system.

7.8 iUSBDAQ_OpenCam (U1208LOG only)

This function opens a camera session associated with the camera index. Before using other camera/video related functions for the selected camera index, this function should be called first.

Syntax:
```
int iUSBDAQ_OpenCam(DevSession* Session,
                    int CamIndex,
                    int VideoProperty,
                    HWND owner);
```

Parameters:
Session: input, iUSBDAQ device session that being opened.
CamIndex: input, selected camera index. Starting from 0, should not exceed the "number of cameras in system – 1". The maximum number of cameras allowed with iUSBDAQ.dll is limited to 3.
VideoProperty: input, either 0 or 1. If 1, the software will popup a video property window where user can set frame rate, resolution and color space for the camera.
Owner: input, the parent window's handle. If "VideoProperty" parameter is 0, this parameter is ignored. If "VideoProperty" parameter is 1 and this parameter is not NULL, then the popup video property window will be the child window of this parent window. Otherwise, the video property window will be a floating window.

Return:
iUSBDAQ error code or 0 for no error.

7.9 iUSBDAQ_VideoGetData (U1208LOG only)

If there is new video frame passing by, this function will send out pulse at TriggerO line (screw terminal pin 39) and get 8 analog channels' data, 16 digital channels' status and the 16 bit counter's value. If there is no new video frame passed, then it will return right the way with error. This function should be pulled frequently with an interval less than the camera frame rate. Say if your camera has 30 frames/s, ideally this function should be called at interval less than 33ms, this way it will make sure it will not miss any video frames passing by. The synchronization accuracy of data and video will be within one video frame time. Note, while putting this function in a loop for pulling, you can still call

other functions that will communicate with the hardware, such as start streaming in normal mode (not high speed streaming mode for U1208LOG, that is the only exception), read analog data or digital lines etc, but you need to make sure they happen in a sequence, not parallel, maybe by using a semaphore, otherwise error may occur due to both functions trying to communicate with hardware at the same time.

Syntax:

```
int iUSBDAQ_VideoGetData(       DevSession* Session,
                                int CamIndex,
                                float *Voltages,
                                BYTE *Port1to8state,
                                BYTE*Port9to16state,
                                unsigned int *Counter);
```

----------not finished yet

Parameters:

Session: input, iUSBDAQ device session that being opened.

CamIndex: input, selected camera index. Starting from 0, should not exceed the "number of cameras in system – 1". The maximum number of cameras allowed with iUSBDAQ.dll is limited to 3.

Voltages: output, either 0 or 1. If 1, the software will popup a video property window where user can set frame rate, resolution and color space for the camera.

Owner: input, the parent window's handle. If "VideoProperty" parameter is 0, this parameter is ignored. If "VideoProperty" parameter is 1 and this parameter is not NULL, then the popup video property window will be the child window of this parent window. Otherwise, the video property window will be a floating window.

Return:

iUSBDAQ error code or 0 for no error.

7.10 iUSBDAQ_CloseCam (U1208LOG only)

IUSBDAQ_API int IUSBDAQ_STD_API iUSBDAQ_CloseCam(DevSession* Session,int CamIndex);

7.11 iUSBDAQ_Video_Run (U1208LOG only)

IUSBDAQ_API int IUSBDAQ_STD_API iUSBDAQ_Video_Run(DevSession* Session,int CamIndex);

7.12 iUSBDAQ_Video_Stop (U1208LOG only)

IUSBDAQ_API int IUSBDAQ_STD_API iUSBDAQ_Video_Stop(DevSession* Session,int CamIndex);

7.13 iUSBDAQ_Get_CamName (U1208LOG only)

IUSBDAQ_API int IUSBDAQ_STD_API iUSBDAQ_Get_CamName(DevSession* Session,int CamIndex, char* value);//value 100 byte

7.14 iUSBDAQ_SetVWindow_Cam (U1208LOG only)

IUSBDAQ_API int IUSBDAQ_STD_API iUSBDAQ_SetVWindow_Cam(DevSession* Session,int CamIndex, const char* winname, HWND owner, int left, int top, int width, int height,int visibility);//winname 100 byte

7.15 iUSBDAQ_Camera_Property (U1208LOG only)

IUSBDAQ_API int IUSBDAQ_STD_API iUSBDAQ_Camera_Property(DevSession* Session,int CamIndex, HWND owner);

7.16 iUSBDAQ_Start_Vrecord (U1208LOG only)

IUSBDAQ_API int IUSBDAQ_STD_API iUSBDAQ_Start_VRecord(DevSession* Session,int CamIndex, char* VFilePath);

7.17 iUSBDAQ_Stop_Vrecord (U1208LOG only)

IUSBDAQ_API int IUSBDAQ_STD_API iUSBDAQ_Stop_VRecord(DevSession* Session,int CamIndex);

//All AVI functions are for U1208LOG only

7.18 iUSBDAQ_AVI_Open (U1208LOG only)

IUSBDAQ_API unsigned int IUSBDAQ_STD_API iUSBDAQ_AVI_Open(DevSession* Session,char* fileName);//return file session

7.19 iUSBDAQ_AVI_Run (U1208LOG only)

IUSBDAQ_API int IUSBDAQ_STD_API iUSBDAQ_AVI_Run(DevSession* Session,unsigned int FileSession);

7.20 iUSBDAQ_AVI_Stop (U1208LOG only)

IUSBDAQ_API int IUSBDAQ_STD_API iUSBDAQ_AVI_Stop(DevSession* Session,unsigned int FileSession);

7.21 iUSBDAQ_AVI_Pause (U1208LOG only)

IUSBDAQ_API int IUSBDAQ_STD_API iUSBDAQ_AVI_Pause(DevSession* Session,unsigned int FileSession);

7.22 iUSBDAQ_AVI_Resume (U1208LOG only)

IUSBDAQ_API int IUSBDAQ_STD_API iUSBDAQ_AVI_Resume(DevSession* Session,unsigned int FileSession);

7.23 iUSBDAQ_AVI_Seek (U1208LOG only)

IUSBDAQ_API int IUSBDAQ_STD_API iUSBDAQ_AVI_Seek(DevSession* Session,unsigned int FileSession, unsigned int Position, unsigned int stopPos);

7.24 iUSBDAQ_AVI_GetPosition (U1208LOG only)

IUSBDAQ_API int IUSBDAQ_STD_API iUSBDAQ_AVI_GetPosition(DevSession* Session,unsigned int FileSession, unsigned int* Position);

7.25 iUSBDAQ_AVI_GetDuration (U1208LOG only)

IUSBDAQ_API int IUSBDAQ_STD_API iUSBDAQ_AVI_GetDuration(DevSession* Session,unsigned int FileSession, unsigned int * Duration);

7.26 iUSBDAQ_AVI_Close (U1208LOG only)

IUSBDAQ_API int IUSBDAQ_STD_API iUSBDAQ_AVI_Close(DevSession* Session,unsigned int FileSession);

7.27 iUSBDAQ_SetVWindow_AVI (U1208LOG only)

IUSBDAQ_API int IUSBDAQ_STD_API iUSBDAQ_SetVWindow_AVI(DevSession* Session, unsigned int FileSession, const char* winname, HWND owner, int left, int top, int width, int height,int visibility);

8 Error Code Description

This is the mapping of error code and error descriptions. The function “iUSBDAQ_GetErrorDes” in API also outputs error description when input an error code.

0-"No Error",
1-"Unknown Error",
2-"device index exceed the max device number of that type",
3-"No such iUSBDAQ module type or wrong device type",

4-"Open device session error",
5-"device maybe used by other application or session already been opened",
6-"Write error, it could be the device disconnected",
7-"Read error, it could be the device disconnected",
8-"NULL session",
9-"Partially failed, incorrect receive length",
10-"the byte count number is invalid or exceed the eeprom size",
11-"channel number is incorrect",
12-"number of channels invalid",
13-"Wrong input voltage range",
14-"Duty cycle value should be between 0- 1023",
15-"The IO channel is not an output channel",
16-"wrong PWM period",
17-"Too many data sets requested, max. allowed number of samples is 3200",
18-"Timeout when waiting for required number of samples",
19-"At least 128 samples required",
20-"Exceed maximum data throughput",
21-"Try to write/read data beyond allowed EEPROM boundary",
22-"PWM period parameter too big, max. allowed 333 us",
23-"PWM period parameter too small, min. allowed 3 us",
24-"Null Video Session, the camera maybe in use or session already opened",
25-"Data not ready",
26-"Exceed max allowed camera numbers",
27-"Camera is in use",
28-"General video related error",
29-"This video device is not supported",
30-"Video recording error",
31-"AVI file playback error",

9 LabVIEW Interface Description

The LabVIEW interface vis are wrapper around iUSBDAQ.dll which located in computer system's directory after the iUSBDAQ device driver being installed. Below are vis's simple descriptions, **more details about functions and parameter's meaning, please also refer to chapter 7 of this document.**

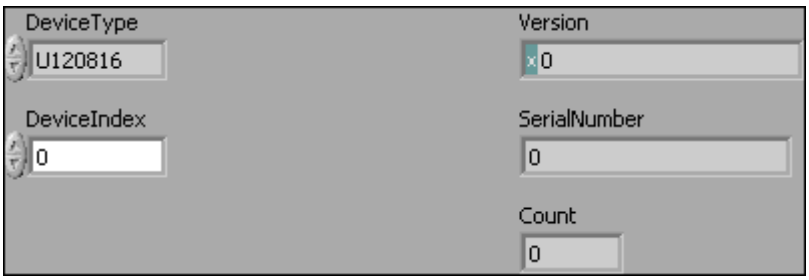
9.1 LabVIEW Examples

9.1.1 iUSBDAQ_GetInfo_Example.vi

This vi enumerates (init) the devices and gets the firmware version, device serial number and total number of iUSBDAQs in the computer of the selected type.

Connector Pane

Front Panel

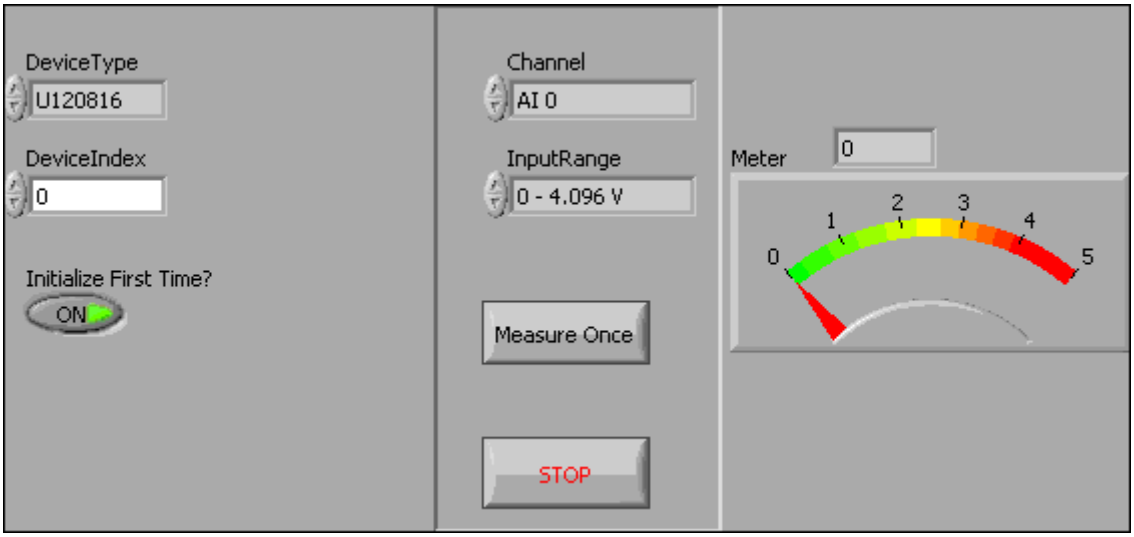


9.1.2 iUSBDAQ_SingleAI_Example.vi

This examples hows how to read data from a single analog channel.

Connector Pane

Front Panel

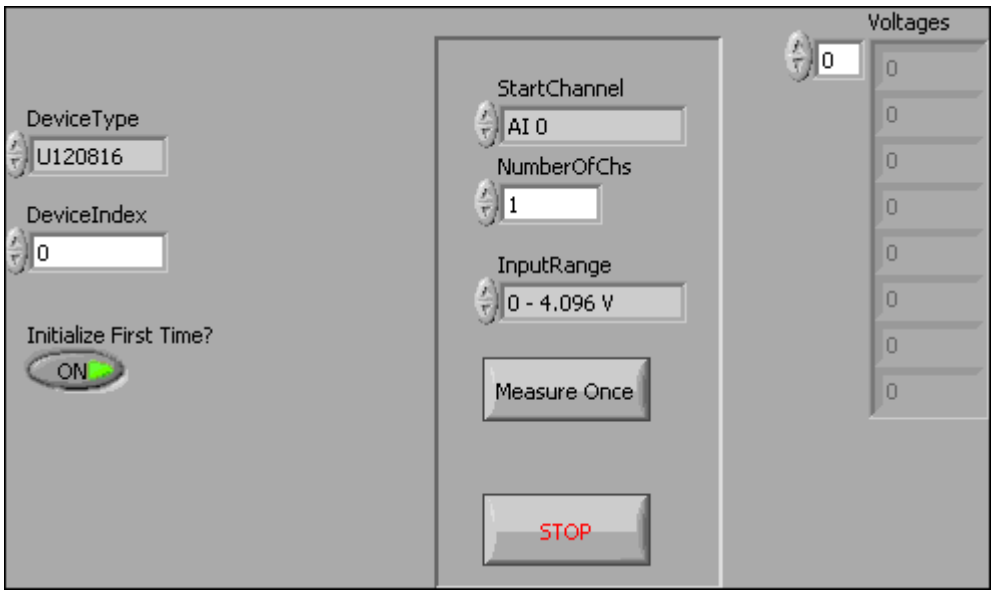


9.1.3 iUSBDAQ_MultipleAIs_Example.vi

This example shows how to read data from multiple analog channels.

Connector Pane

Front Panel

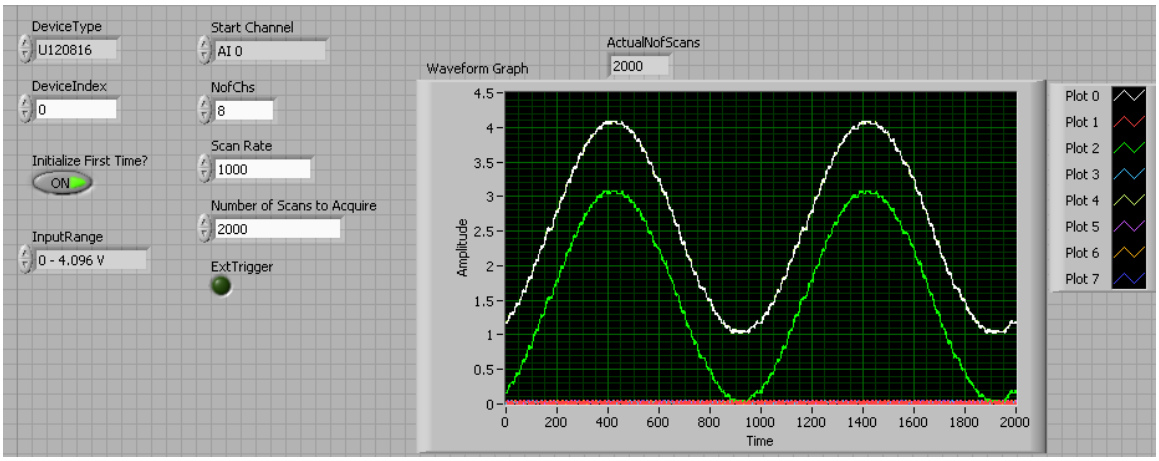


9.1.4 iUSBDAQ_StreamingAIs_Example.vi

This example shows how to acquire analog data using streaming functions.

Connector Pane

Front Panel

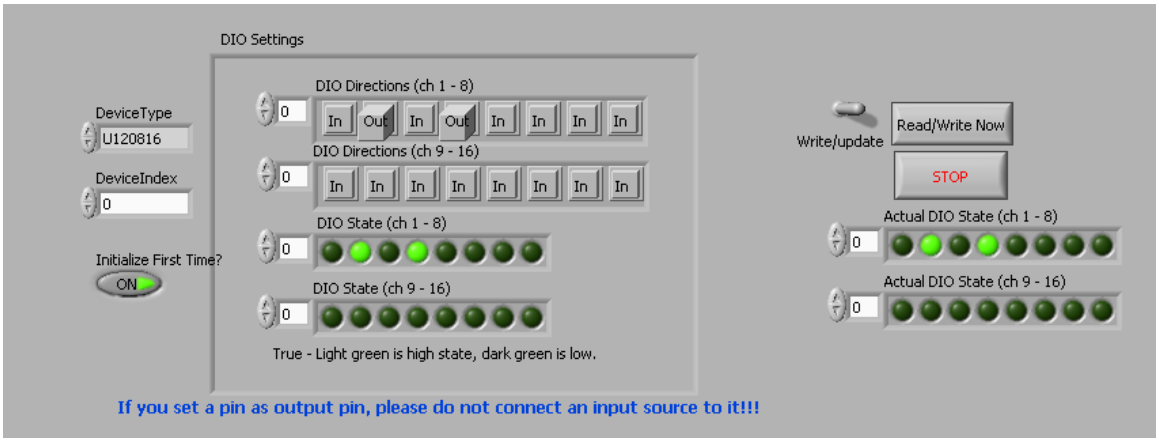


9.1.5 iUSBDAQ_DIO_Example.vi

This example shows how to set the digital I/O directions, read/write DIO states.

Connector Pane

Front Panel

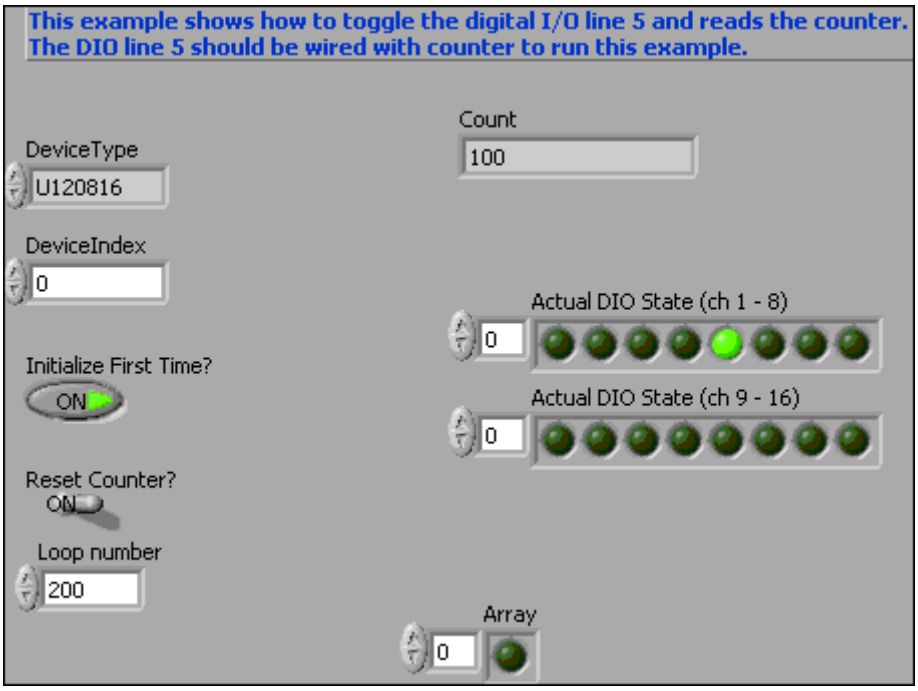


9.1.6 iUSBDAQ_DIO_Counter.vi

This example shows how to toggle the digital I/O line 5 and reads the counter. The DIO line 5 should be wired with counter to run this example.

Connector Pane

Front Panel

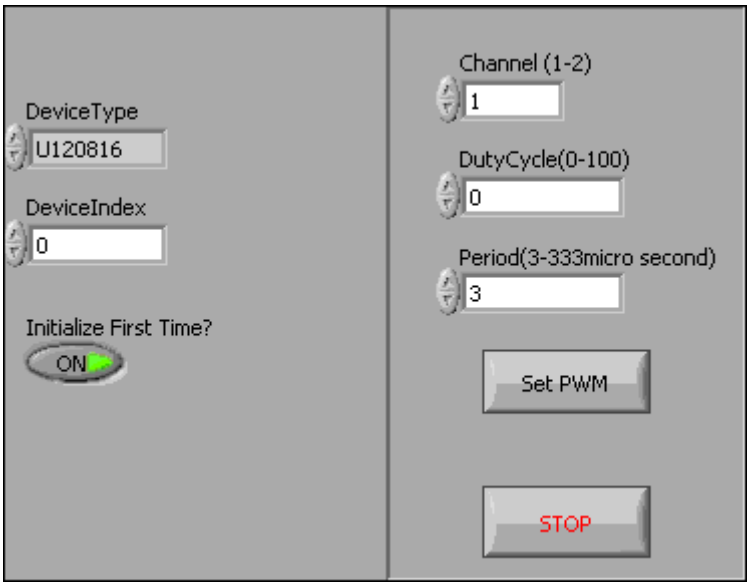


9.1.7 iUSBDAQ_PWM_Example.vi

This example shows how to set PWM outputs.

Connector Pane

Front Panel

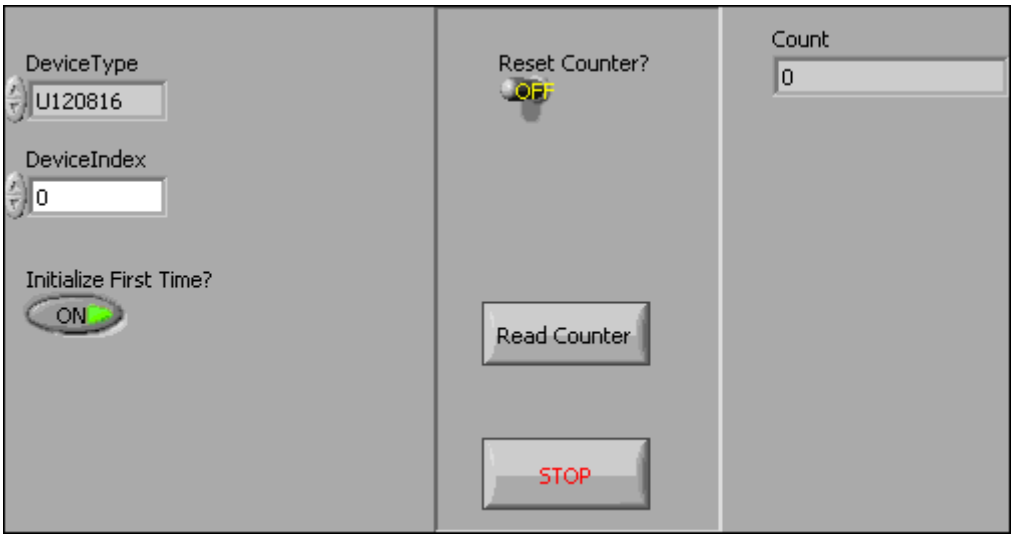


9.1.8 iUSBDAQ_Counter_Example.vi

This example shows how to read counter.

Connector Pane

Front Panel

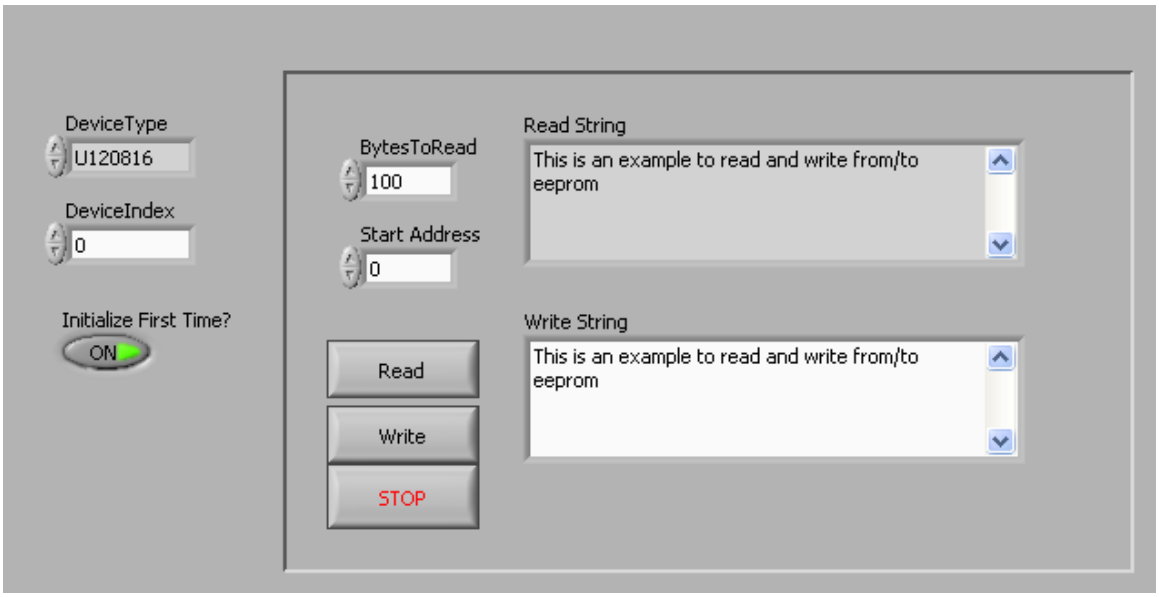


9.1.9 iUSBDAQ_ReadWriteEEPROM_Example.vi

This example shows how to write to eeprom at specified address and read back eeprom.

Connector Pane

Front Panel

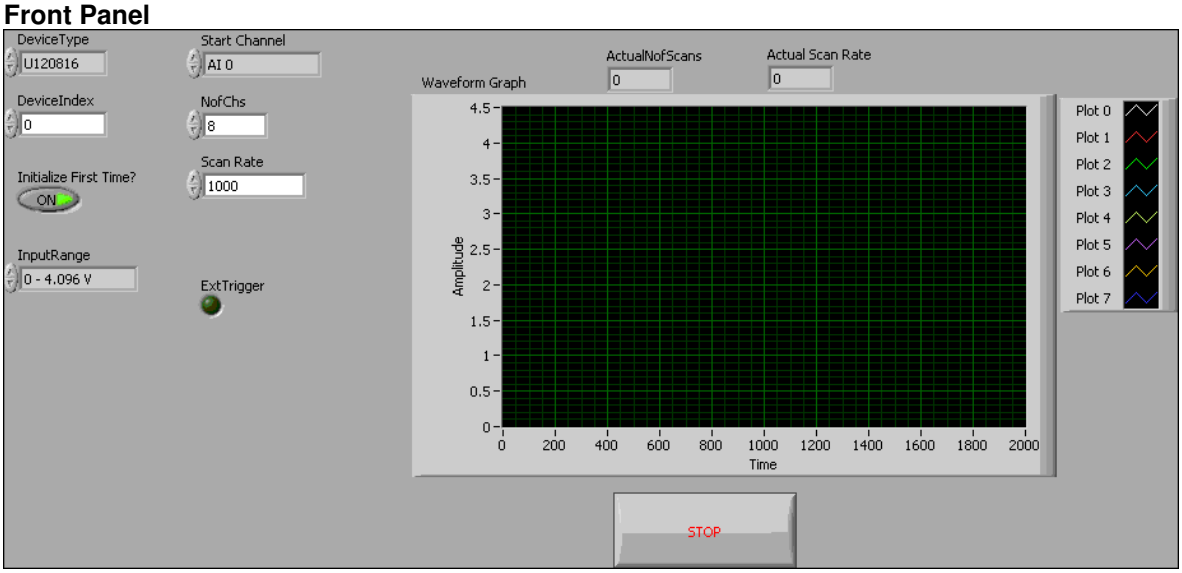


9.1.10 iUSBDAQ_StreamingAIs_Cont_Example.vi

This example shows how to stream analog data continuously.

Connector Pane

Front Panel

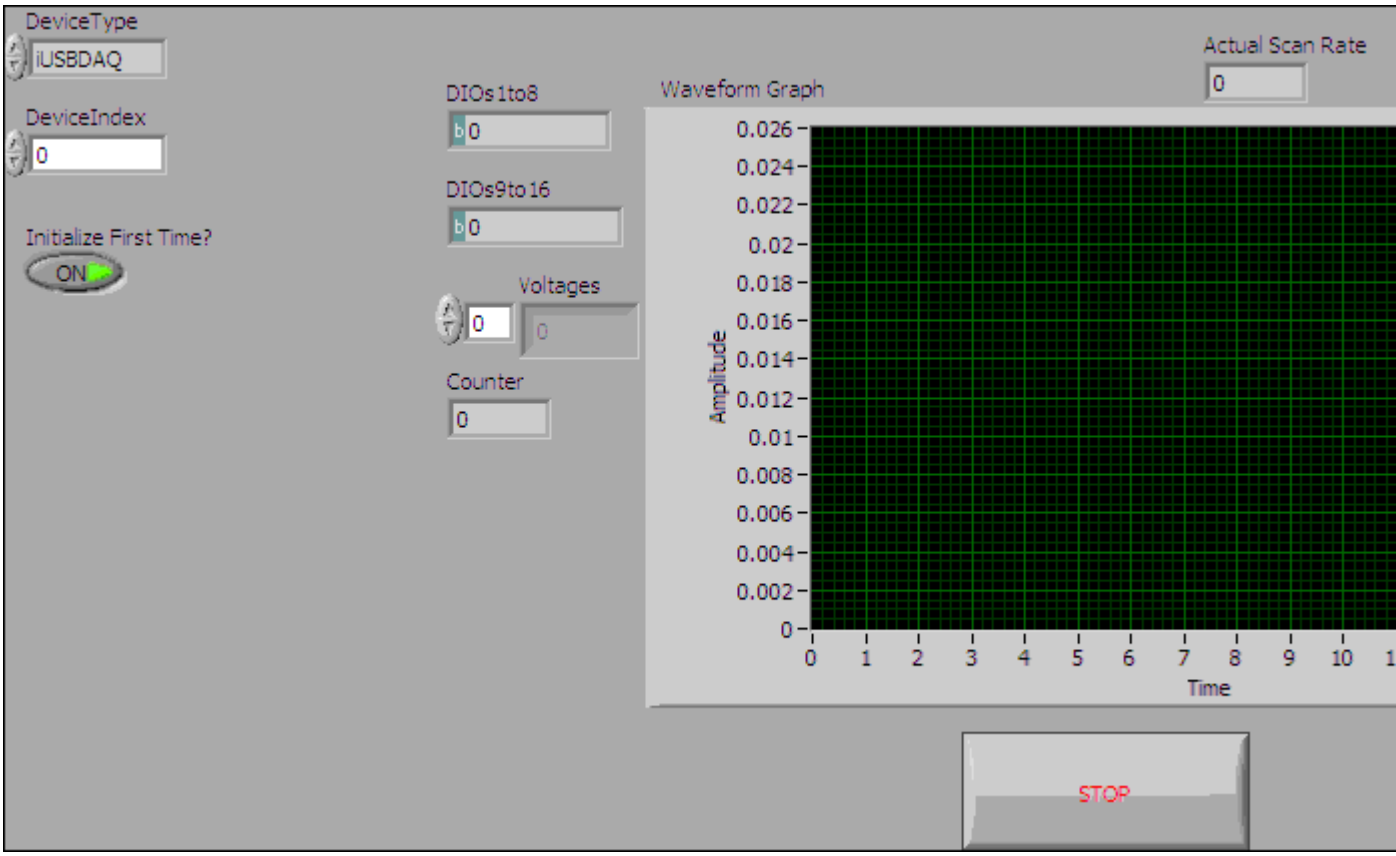


9.1.11 iUSBDAQ_StreamingAIs_Cam_Example.vi (U1208LOG only)

This example shows how to stream analog data continuously while video is streaming, at the same time getting 8 analog input's data, 16 digital line status and counter value.

Connector Pane

Front Panel

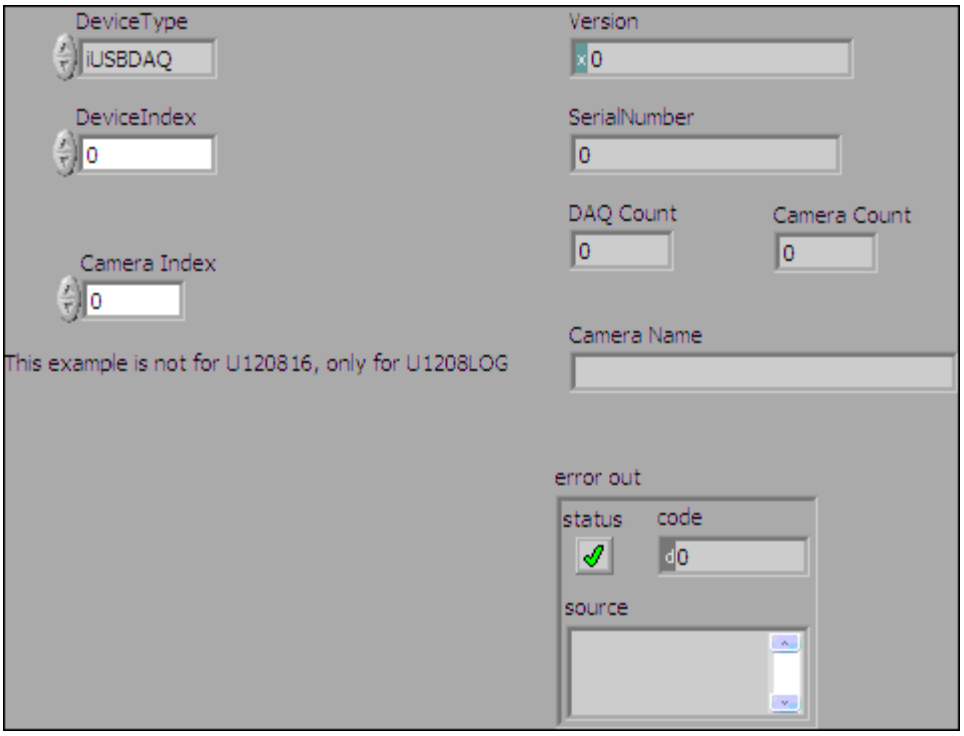


9.1.12 iUSBDAQ_GetDAQCAM_Info_Example.vi (U1208LOG only)

This vi enumerates (init) the iUSBDAQ devices as well as cameras and gets the DAQ firmware version, DAQ device serial number and total number of iUSBDAQs in the computer, also gets the total number of cameras and camera name for the selected camera in "Camera Index" control.

Connector Pane

Front Panel

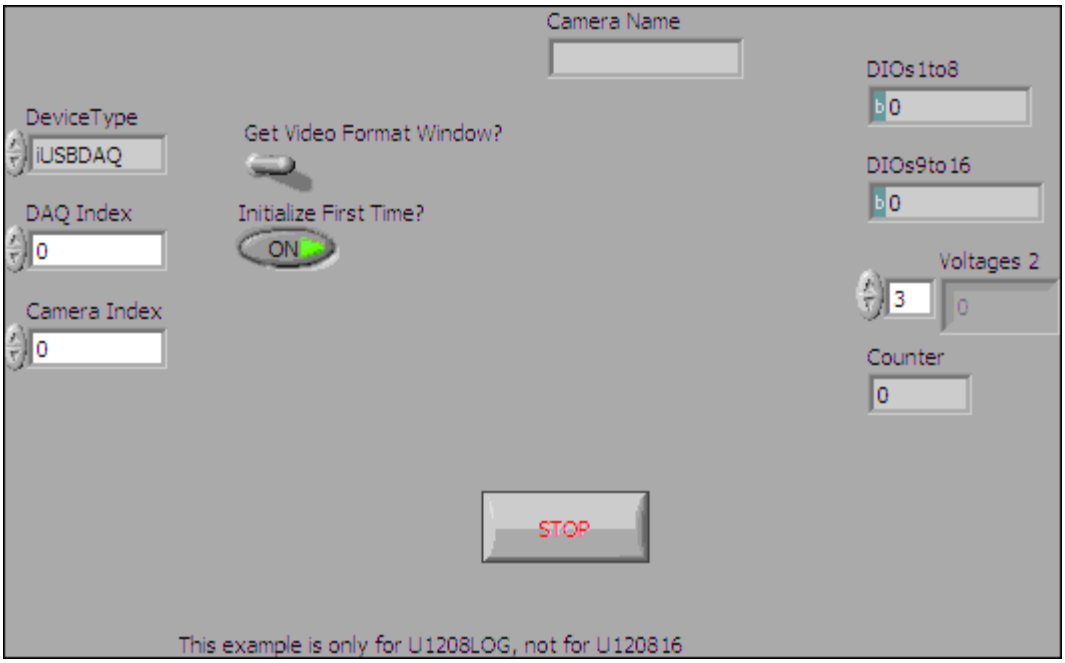


9.1.13 iUSBDAQ_DAQCAM_Example1.vi (U1208LOG only)

This example shows how to read data from multiple analog channels while the video is streaming.

Connector Pane

Front Panel

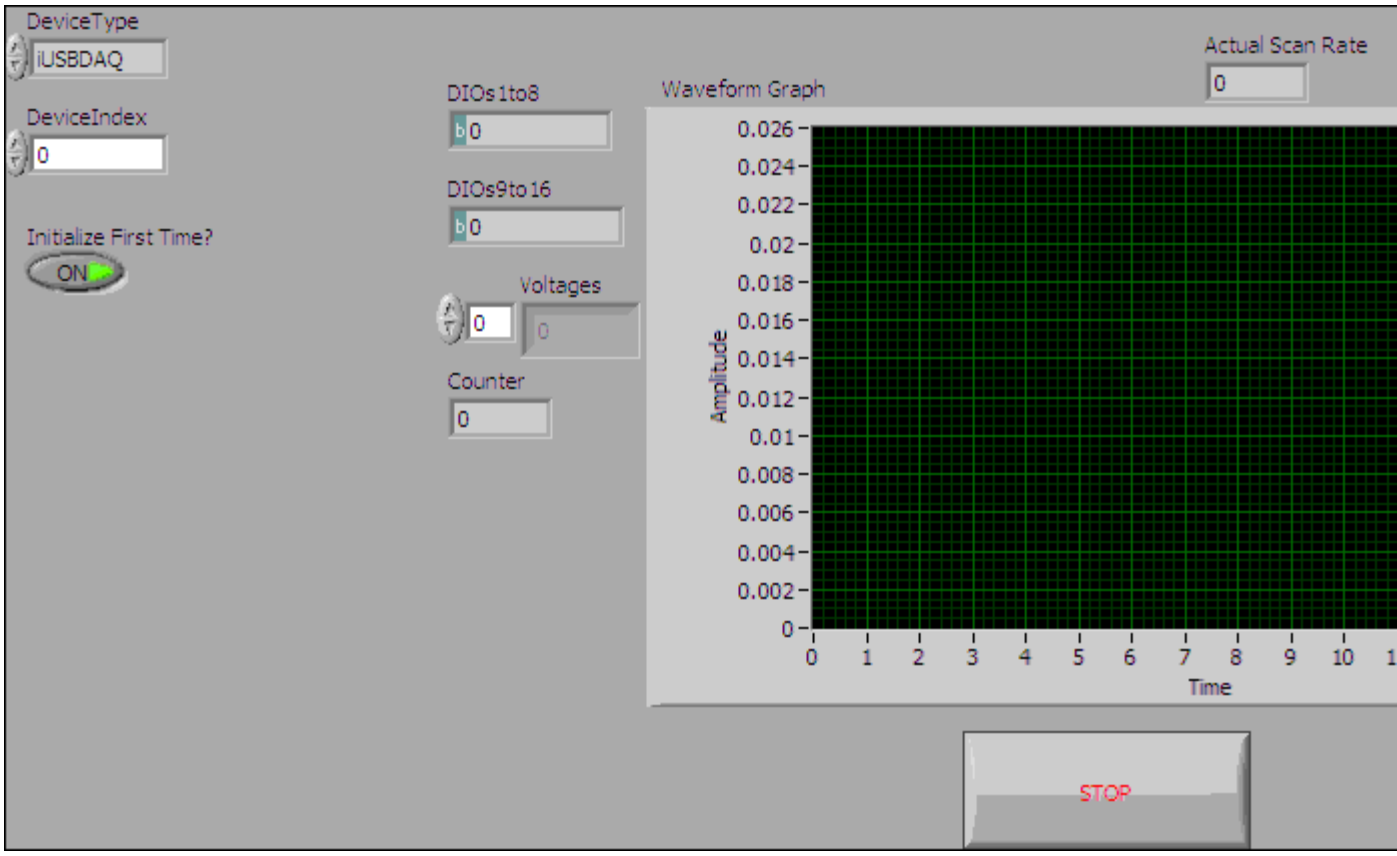


9.1.14 iUSBDAQ_StreamingAIs_Cam_Example.vi (U1208LOG only)

This example shows how to stream analog data continuously while video is streaming, at the same time getting 8 analog input's data, 16 digital line status and counter value.

Connector Pane

Front Panel



9.1.15 iUSBDAQ_AVI_Example1.vi (U1208LOG only)

This example shows how to playback AVI file.

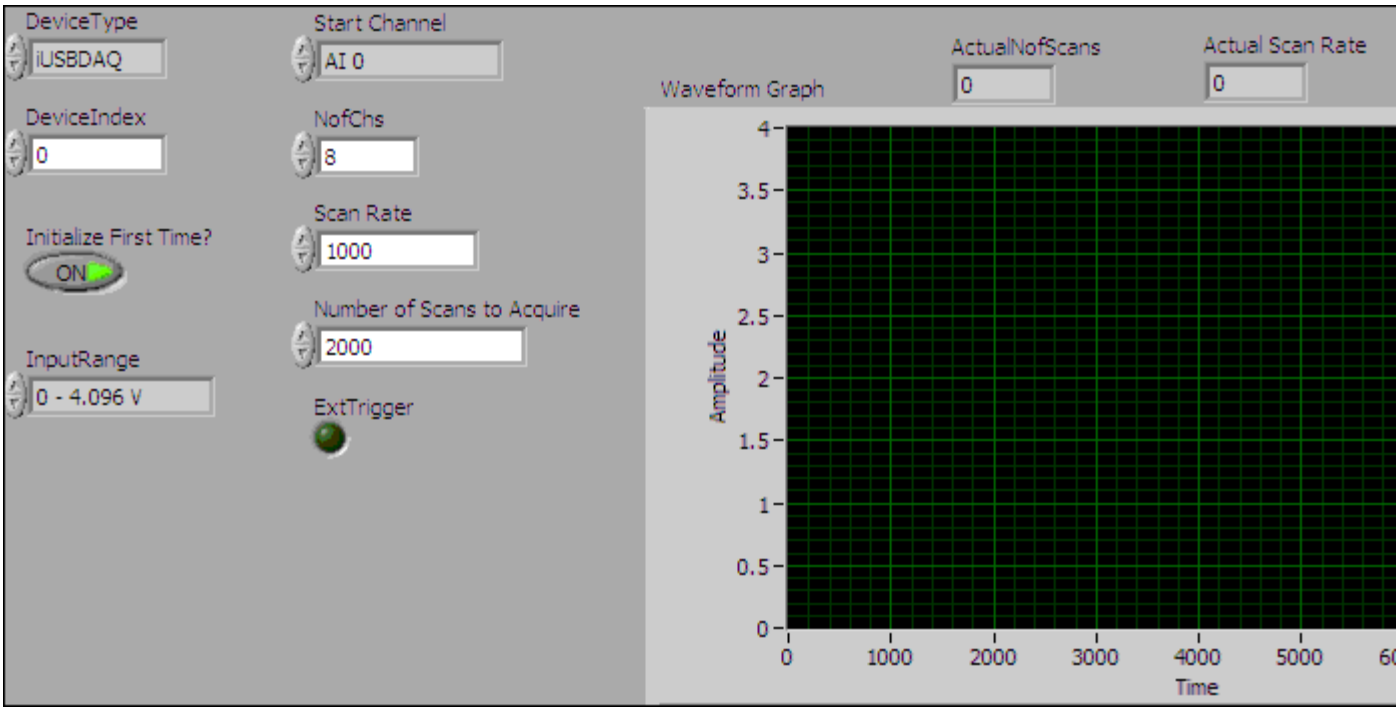
Connector Pane

Front Panel

iUSBDAQ_Streaming_HS_Example.vi
This example shows how to acquire analog data using high speed streaming functions. This is only for U1208LOG model.

Connector Pane

Front Panel



9.2 General Device Functions

9.2.1 iUSBDAQ_All_Vis.vi

Open this vi's diagram will see all iUSBDAQ functions.

Connector Pane

9.2.2 iUSBDAQ_GetErrorString.vi

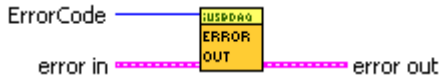
This vi takes a iUSBDAQ error code and returns the error description string.

Connector Pane

9.2.3 iUSBDAQ_ErrorOut.vi

This vi generates an error out cluster with the iUSBDAQ error code and description string as error source.

Connector Pane



9.2.4 iUSBDAQ_GetDLLVersion.vi

This vi gets the current iUSBDAQ.dll version.

Connector Pane

9.2.5 iUSBDAQ_Get_DesignerInfo.vi

This vi returns the iUSBDAQ designer's info. And it's HYTEK Automation, Inc.

Connector Pane

9.2.6 iUSBDAQ_GetFWVersion.vi

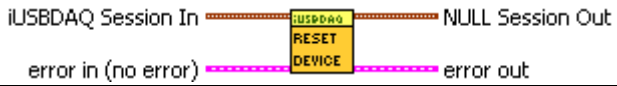
This vi gets the firmware version inside the hardware.

Connector Pane

9.2.7 iUSBDAQ_Reset.vi

This vi resets the iUSBDAQ.

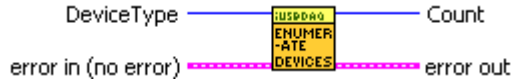
Connector Pane



9.2.8 iUSBDAQ_EnumerateDevices.vi

This is the first vi that need to be called for a selected device type before using any iUSBDAQ functions. It enumerates the computer to create a mapping of the iUSBDAQ, allocates some internal memory. If any device's sessions were opened before, they will be closed by this vi in order to creat the internal map. It also returns the total number of iUSBDAQs of the selected type. Normally this vi only need to be called once at very beginning of application, but user may want to re-enumerate if some devices are disconnected while in use to make sure the mapping is updated.

Connector Pane

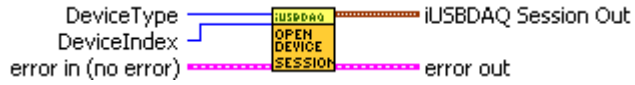


9.2.9 iUSBDAQ_OpenDeviceSession.vi

This vi opens an iUSBDAQ session. This session will be used to call other device related

functions. The order of using a device function is enumerate, open session, calling device related functions... at very end close device session. There is no need to enumerate, open session and close session everytime.

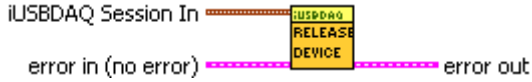
Connector Pane



9.2.10 iUSBDAQ_ReleaseDevice.vi

This vi closes a opened device session. Normally user only need to close a device session when they don't use that device anymore. The order of using a device function is enumerate, open session, calling device related functions... at very end close device session. There is no need to enumerate, open session and close session everytime.

Connector Pane



9.2.11 iUSBDAQ_Get_SerialNumber.vi

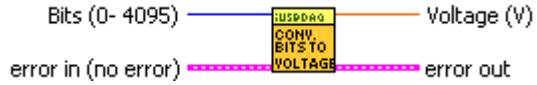
This vi returns an iUSBDAQ device serial number. Each iUSBDAQ has an unique serial number. User can use this unique ID to distinguish the devices if there are more than one iUSBDAQs connected to the same computer.

Connector Pane

9.2.12 iUSBDAQ_Bits_To_Voltage.vi

converts bits to voltage.

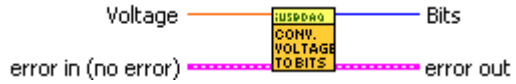
Connector Pane



9.2.13 iUSBDAQ_Voltage_To_Bits.vi

Converts voltage to bits.

Connector Pane

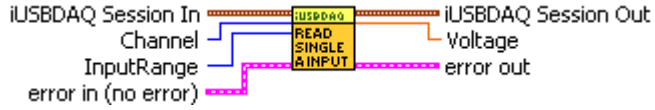


9.3 Analog Input Functions

9.3.1 iUSBDAQ_Read_SingleAI.vi

This vi reads voltage from one analog channel.

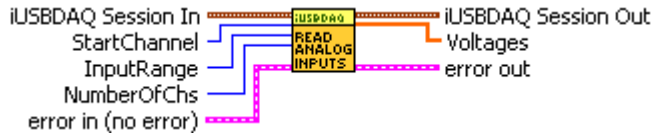
Connector Pane



9.3.1 iUSBDAQ_Read_Multi_AI.vi

This vi scans multiple analog channels once and return the voltages. The maximum Number of Chs is 8, and it depends on which channel is the starting channel. The StartChannel+NumberOfChs should not exceed 8.

Connector Pane



9.3.2 iUSBDAQ_StartStream.vi

This vi starts the streaming mode analog inputs acquisition. This vi returns right the way after the call and the streaming is in background with an internal pc side data buffer. Please read more about this data buffer in "iUSBDAQ_GetStreamSamples.vi" description.

The maximum Number of Chs is 8, and it depends on which channel is the starting channel. The StartChannel+NumberOfChs should not exceed 8.

The maximum sampling throughput varies with number of Channels. Please refer to iUSBDAQ User's Guide for details. The relationship between sampling rate and scan rate is: SampleRate=ScanRate*NumberOfChannels

For example, if scan rate is 3000, number of channels is 8, the total sample rate will be 24,000samples/s.

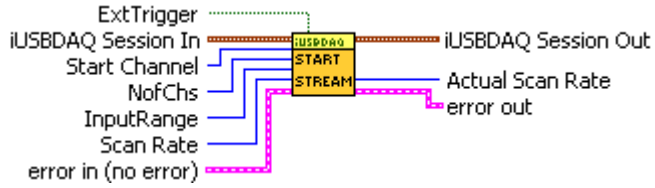
The minimum sample rate has to be 128. If lower than this required, user can use scan mode. So for example if you have two channels, the minimum scan rate has to be 64.

If Ext Trigger is true, after this vi returns, the device will wait for a high state from the trigger to start the data acquisition. False will start the data acquisition right after this call into pc side background data buffer.

This vi will return actual scan rate, it may not always the same as the specified input scan rate.

This vi will return an error if the channel's settings are not correct.

Connector Pane



9.3.3 iUSBDAQ_StopStream.vi

This vi stops the streaming.

Connector Pane

9.3.4 iUSBDAQ_StartStream_HS.vi (U1208LOG only)

This vi starts the streaming mode analog inputs acquisition in high speed mode, this function is only available for U1208LOG model. This vi returns right the way after the call and the streaming is in background with an internal pc side data buffer. Please read more about this data buffer in "iUSBDAQ_GetStreamSamples.vi" description.

The maximum Number of Chs is 8, and it depends on which channel is the starting channel. The StartChannel+NumberOfChs should not exceed 8.

The maximum throughput varies with number of Channels. Please refer to iUSBDAQ User's Guide for details. The relationship between sampling rate and scan rate is:
Sample Throughput Rate=ScanRate*NumberOfChannels

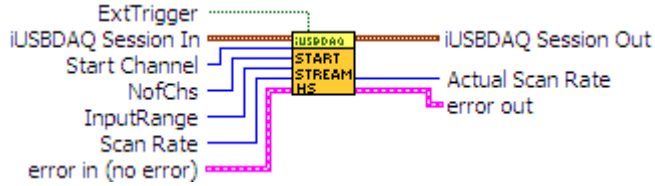
For example, if scan rate is 3000, number of channels is 8, the total sample throughput rate will be 24,000samples/s.

If Ext Trigger is true, after this vi returns, the device will wait for a high state from the trigger to start the data acquisition. False will start the data acquistion right after this call into pc side background data buffer.

This vi will return actual scan rate, it may not always the same as the specified input scan rate.

This vi will return an error if the channel's settings are not correct.

Connector Pane



9.3.5 iUSBDAQ_StopStream_HS.vi (U1208LOG only)

This vi stops the high speed data streaming of iUSBDAQ - U1208LOG.

Connector Pane

9.3.6 iUSBDAQ_GetStreamSamples.vi

This vi gets the data back from the internal PC side data buffer.The number of channels has to be the same as the number of channels in "iUSBDAQ_StartStream.vi". User can specify how many scans need to be read back. This vi only returns when all the asked scans are read. To stop inbetween, user need to call the "iUSBDAQ_StopStream.vi" and this vi will error out and return. The timeout is for read of 128 samples, not for the total number of scans asked. The returned data is interleaved, for example, if you start streaming with starting channel of AI 0, number of channels is 2, then the output data is in the order of AI 0, AI 1, AI 0, AI 1..... This vi also returns the actual number of scans, it may not always be the same as input number of scans. The relation between data array size and actual number of scans is:
Data array size=actual number of scans x number of channels

The minimum samples requested must not be less than 128, for example if you have two channels, then the minimum NofScans must not be less than 64. There is no limit for maximum NofScans requested.

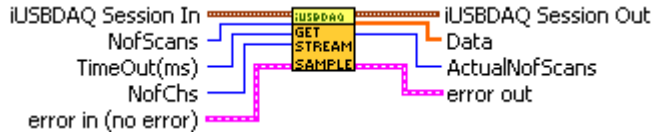
The internal data buffer has a size of 12.8k samples, if this buffer is full, the later coming data will be ignored. Usually user should start calling this vi right after start streaming.

There are two possible ways user can use this vi, pass in a big total number of scans to read at once (can be bigger than 12.8k because the iUSBDAQ.dll will take care of this), this vi will wait until all asked scans returned. The other way is asking for a small amount number of scans, and repeat calling this vi again and again. The only risk of second method is if there is too many delays between two consecutive calls of this vi, the internal data buffer may be full, some data will be ignored which will result none continuous data between two calls.

In iDAQTest&Log software, it uses the second method, each time it asks for one second long data, which means the asked number of scans equals to scan rate and repeat calling this vi again and again, concatenate the data to get a continuous graph. User may also want to consider using a separate data engine vi to handle streaming, getting data so that this action will not block other user interface actions if any.

Please check out the example vis related to this function, "iUSBDAQ_StreamingAIs_Example.vi" and "iUSBDAQ_StreamingAIs_Cont_Example.vi".

Connector Pane



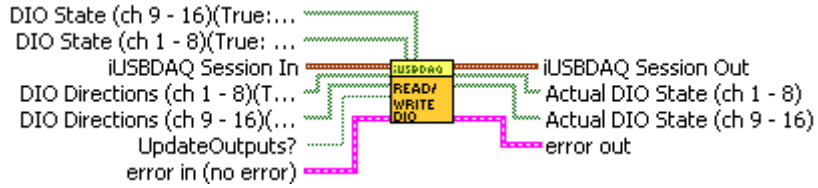
9.4 Digital I/O Functions

9.4.1 iUSBDAQ_DIO.vi

This vi sets DIO directions, read/write DIO states. If "Update Outputs?" control is true, the output DIO pins states will be written. A read of all DIO states is always performed at the end no matter

what. For DIO directions, true is input, false is output and for DIO states, true is high state, false is low state.

Connector Pane

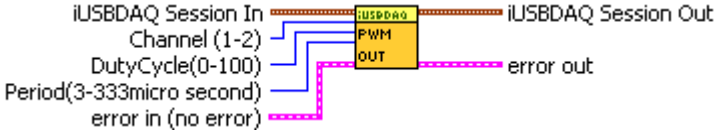


9.5 PWM Functions

9.5.1 iUSBDAQ_PWM_Out.vi

This vi sets PWM outputs. Both channels should have the same frequency or period. The last set period will be in effect. Duty cycle is between 0-100%.

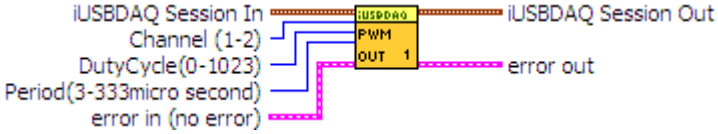
Connector Pane



9.5.2 iUSBDAQ_PWM_Out1.vi

This vi sets PWM outputs. Both PWM channels use the same period or frequency. The last set period will be in effect. The duty cycle is between 0 – 1023.

Connector Pane



9.5.3 iUSBDAQ_StopPWM.vi

This vi stops PWM.

Connector Pane

9.6 Counter Functions

9.6.1 iUSBDAQ_Read_Counter.vi

This vi reads counter. If "Reset Counter" switch is true, the counter get reset to 0 after the read.

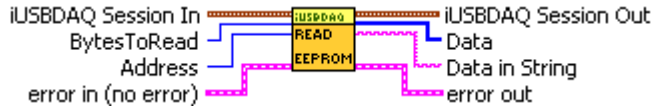
Connector Pane

9.7 Read/Write EEPROM Functions

9.7.1 iUSBDAQ_ReadEEPROM.vi

This vi reads from EEPROM. The following condition must meet:
Starting Address+Bytes To Read<=240

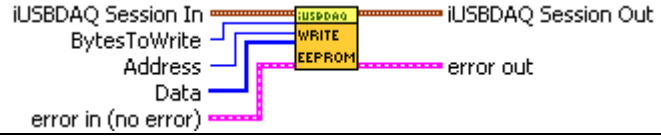
Connector Pane



9.7.2 iUSBDAQ_WriteEEPROM.vi

This vi writes bytes to EEPROM. The following condition must meet:
Starting Address+Bytes To Write<=240

Connector Pane



9.8 Video Functions (U1208LOG only)

9.8.1 iUSBDAQ_EnumerateCams.vi

This vi enumerates the camreas in the system and return the total cameras available. The library may have limit the max number of cameras that can be used. Please refer to user manual.

This vi has to be called first before any video related functions can be used.

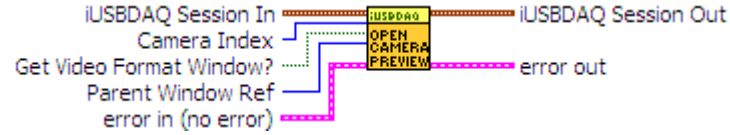
Connector Pane

9.8.2 iUSBDAQ_OpenCam_Preview.vi

This vi opens camera session in preview mode. Before you can run or stop camera video, you

should always first open camera session.

Connector Pane



9.8.3 iUSBDAQ_Close_Camera.vi

This vi closes camera session. After you finishes using the camera session, please close it.

Connector Pane

9.8.4 iUSBDAQ_Video_Run.vi

This vi runs the life camera video.

Connector Pane

9.8.5 iUSBDAQ_Video_Stop.vi

This vi stops the life camera video. It does not close the camera session. You can run the video again with video run function.

Connector Pane

9.8.6 iUSBDAQ_Get_CamName.vi

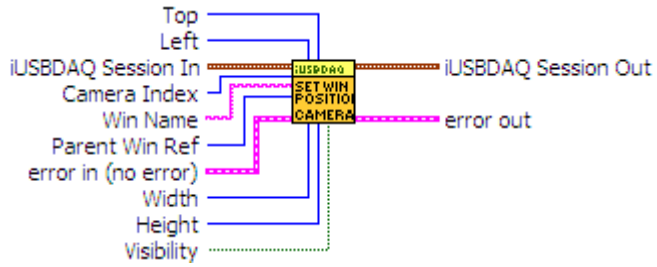
This vi gets the camera name.

Connector Pane

9.8.7 iUSBDAQ_SetVWindow_Cam.vi

This vi sets the camera video window's name, position, parent window, visibility etc.

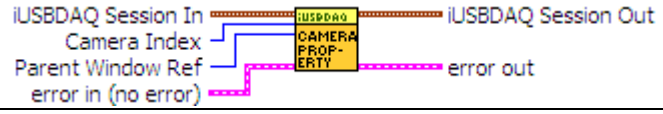
Connector Pane



9.8.8 iUSBDAQ_Camera_Property.vi

This vi will bring out the camera property window, where you can set brightness, contrast, white balance, hue, saturation etc. camera properties.

Connector Pane

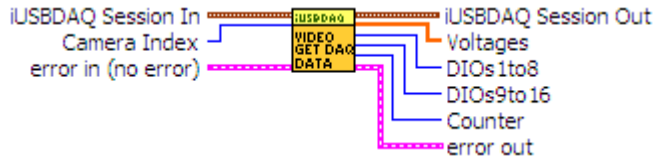


9.8.9 iUSBDAQ_Video_GetData.vi

This vi will pull the data that are synchronized with video frames. It can only output data when video is running. It returns error when there is no new data ready.

If there is new video frame passing by, this function will send out pulse at TriggerO line (screw terminal pin 39) and get 8 analog channels' data, 16 digital channels' status and the 16 bit counter's value. If there is no new video frame passed, then it will return right the way with error. This function should be called frequently with an interval less than the camera frame rate. Say if your camera has 30 frames/s, ideally this function should be called at interval less than 33ms, this way it will make sure it will not miss any video frames passing by. The synchronization accuracy of data and video will be within one video frame time. Note, while putting this function in a loop for pulling, you can still call other functions that will communicate with the hardware, such as start streaming in normal mode (not high speed streaming mode for U1208LOG, that is the only exception), read analog data or digital lines etc, but you need to make sure they happen in a sequence, not parallel, maybe by using a semaphore, otherwise error may occur due to both functions trying to communicate with hardware at the same time.

Connector Pane



9.8.10 iUSBDAQ_Video_StartRecord.vi

This vi starts the video recording into avi file. The camera session should be pre opened by IVision_OpenCam_Preview.vi.

Connector Pane

9.8.11 iUSBDAQ_Video_StopRecord.vi

This vi stops the video recording into avi file. This vi will not close the camera session, so the existing camera session will still be usable. User to call the IVision_Video_Run.vi to run the video after this vi.

Connector Pane

9.9 AVI Functions (U1208LOG only)

9.9.1 iUSBDAQ_AVI_Open.vi

This vi will open an AVI file session. It returns the file session back. This file session is needed to call other AVI related functions.

Connector Pane

9.9.2 iUSBDAQ_AVI_Run.vi

This vi will run the avi file to playback. The file session should be preopened by AVI open function and pass into this function.

Connector Pane

9.9.3 iUSBDAQ_AVI_Stop.vi

This vi will stop the avi file to playback. The file session should be preopened by AVI open function and pass into this function.

Connector Pane

9.9.4 iUSBDAQ_AVI_Pause.vi

This vi will pause the avi file to playback. The file session should be preopened by AVI open function and pass into this function.

Connector Pane

9.9.5 iUSBDAQ_AVI_Resume.vi

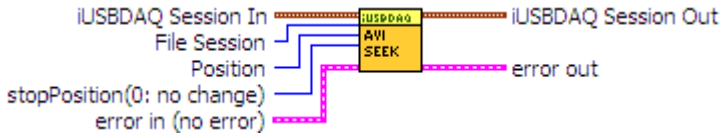
This vi will resume the avi file to playback if it was paused previously. The file session should be preopened by AVI open function and pass into this function.

Connector Pane

9.9.6 iUSBDAQ_AVI_Seek.vi

This vi will set the AVI to a specified position. It can also set the stop position of avi file.

Connector Pane



9.9.7 iUSBDAQ_AVI_GetPosition.vi

This vi will get the AVI current frame position.

Connector Pane

9.9.8 iUSBDAQ_AVI_Close.vi

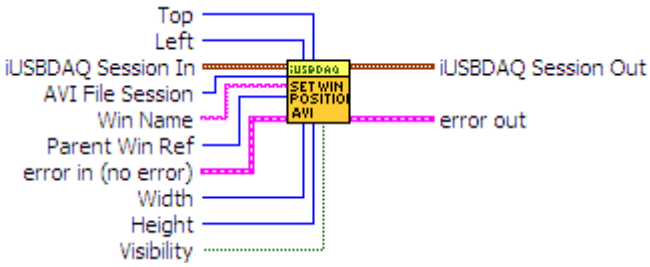
This vi will close the opened AVI file session. After closing the avi file session, no other functions related to avi file are possible to be called for this avi file.

Connector Pane

9.9.9 iUSBDAQ_SetVWindow_AVI.vi

This vi sets the AVI file video window's name, position, parent window, visibility etc.

Connector Pane



9.9.10 iUSBDAQ_AVI_GetDuration.vi

This vi will get the AVI total frame numbers.

Connector Pane

10 Runtime Distribution of iUSBDAQ Driver

When you want to distribute the system or software that integrates iUSBDAQ, there is only one file that you need to distribute for iUSBDAQ to work, the iUSBDAQ device driver. It's the low level driver used for computer to recognize and enumerate the iUSBDAQ devices. This installation will also copy the iUSBDAQ.dll - the programming APIs that your software calls into computer system's directory.

The method how you distribute, either includes iUSBDAQ device driver in your full installation program with your own software or distribute it separately, is up to the customers.

11 Troubleshooting

All iUSBDAQs are tested on good working condition prior shipping, but in case the device still does not work after proper installation and the green light is not lit, there are few things to check

- If the device was installed properly, check the device manager, see if you see HYTEK iUSBDAQ listed under "HytekUSBDAQ" category.
- Change another USB cable, prefer using a cable that has noise protection filter built in.
- Change to another USB port on the same computer.
- Make sure, if it's laptop, it's not on battery, because the USB port power supply maybe low.
- Change to another computer, so that you know if device or computer is the problem.
- For U1208LOG, must use a USB cable that has noise protection filter built in when connecting to the USB power adaptor.
- For U1208LOG, if LED2 does not flash in the preset logging interval, unplug USB flash disk and plug in again or re-power up the device. Also make sure the switch is on the LOG/Start side.

- Please check out the iUSBDAQ FAQ or post questions at: forum.hytekautomation.com, if you can not solve the problem yourself.

12 Specifications

The spec is for 25 °C typically.

General

Parameter	**Specification**
Device type	USB 2.0 full speed
Device compatibility	USB 1.1, USB 2.0
Operating Temperature range	-40 to 85℃ for U120816, 0 to 70℃ for U1208LOG
Dimension	3.5" x 3.375" x 1.125" (9cm x 8.5cm x 3cm)
Connector Type	Screw terminal
Power Supply	USB bus powered, min 4.5V max 5.5V

Analog Input

Parameter	Min	Typical	Max	Unit	Specification
A/D converter type					Successive approximation type
Input voltage range for linear operation	0		Vref	V	
Mode					Single ended
Number of channels					8
Resolution					12 bit
Vref Voltage	4.055	4.096	4.137	V	
Accuracy	-41	1	+41	mV	Depends on the accuracy of Vref
Throughput		32k 30.1k 27k 25k 22k 19.5k 18.4k 13k 125		Samples /s	Number of channels=8 Numberof channels=7 Numberof channels=6 Numberof channels=5 Numberof channels=4 Numberof channels=3 Numberof channels=2 Numberof channels=1 Software timed scan
Maximum input voltage range	-0.6		7.6	V	

Integral Nonlinearity		+-1.0	+-2.0	LSB	
Differential Nonlinearity		+-0.5	+-1.0	LSB	No missing codes over-temperature
Offset Error		+-1.25	+-3	LSB	
Leakage Current		0.001	+-1	uA	
Trigger source					Software or Trigger line
Gain Error		+-1.25	+-5	LSB	

Digital input/output

Number of IOs	16 bi-directional
Digital type	CMOS output, TTL or Schmitt trigger input
Pull up	All pins pull up to Vs via 470 ohm, 1M to ground
Input high voltage	2.0V min, 5.5V absolute max
Input low voltage	-0.3V absolute min, 0.8V max
Output voltage	(Vs-0.47) V at 1mA
Output short circuit current	10.6mA at Vs=5V
Maximum output current sunk	25mA
Maximum output current sourced	25mA
Power on states	All are inputs
Input leakage current	+-1uA

Counter

Resolution	16 bit
Maximum input frequency	1M HZ
High pulse width	0.5us min
Low pulse width	0.5us min
Input leakage current	+-1uA
Input high voltage	4.0V min, 5.5V absolute max
Input low voltage	0.8Vmax, -0.3V absolute min
Pull up	470ohm in series, 1M ohm to ground

PWM

Resolution	10 bit
Number of channels	2
Pull up	470ohm pull up to Vs, 1M ohm to ground
Period	3us – 333us
Frequency	333kHz – 3kHz

Trigger Line

Pull up/pull down	470ohm in series, 1M ohm to ground
Input leakage current	+-1uA
Input high voltage	2.0V min, 5.5V absolute max
Input low voltage	-0.3V absolute min, 0.8V max
Trigger mode	High state will start data acquisition if used

+5V Power

Parameter	Condition	Specification
Output voltage	Connected to self-powered hub Connected to externally powered root hub	4.5V min, 5.25V max
	Connected to bus powered hub	4.1V min, 5.25Vmax
Output current	Connected to self-powered hub Connected to externally powered root hub	450mA max
	Connected to bus powered hub	50mA max

EEPROM

Size	240 bytes
Address range	0 - 239