IntelliDAQ

COM Automation API

DLL Version 1.0.0.4
Document Version 2.0.0.0
07 January 2018

Table of Contents

Summary

The IntelliDAQ automation interface allows you to configure and control IntelliDAQs as well as read out status and events for automated test control and processing. Python and C# examples are available upon request.

Interfaces

The functions described below describe the interfaces available for communicating with an IntelliDAQ device.

Any invalid parameters or function requests will result in a COM error which can be handled with the ErrorInfo class defined by Microsoft.

Connection

Before any other interfaces can be called, a connection must be made using one of the follow functions. Making a connection will disconnect any previous connection.

SetSocketConnection

Sets a socket connection to an Ethernet connected IntelliDAQ using the hostname and port specified. The default port on all IntelliDAQ devices is 8080.

Parameters		
hostName	BSTR (string)	Hostname or IP address of the IntelliDAQ device
port	UINT	IP port (range: 0 – 65535)
Returns		
Nothing		

SetSerialConnection

Sets a serial connection over a local RS232 connection to an IntelliDAQ device using the specified Comm port.

Parameters		
portName	BSTR (string)	Comm port to connect to (ex: “COM3”)
Returns		
Nothing		

SetUSBConnection

Sets a USB connection to a locally connected IntelliDAQ. If more than one IntelliDAQ is connected to the PC, the first one will be connected.

Parameters		
None		
Returns		
Nothing		

Control

These functions are used to start and stop continuous measurement tests.

StartTest

Starts a test.

WARNING: This will clear out all test data previously stored on the IntelliDAQ device.

Parameters		
None		
Returns		
Nothing		

StopTest

Stops a test.

Parameters		
None		
Returns		
Nothing		

Data Access

These functions are used to read the status and event log from the most recent test. The most recent test is the one currently running, or the last test ran if the test is already stopped.

ReadStatus

Reads the test status out of the currently connected IntelliDAQ device and returns is as a StatusDefn structure.

Parameters		
None		
Returns		
Status	StatusDefn	Status information. See StatusDefn.

ReadLog

Reads out the event log from the currently connected IntelliDAQ device and returns the number of entries read. This must be done before any calls to
GetLogEntry are used.

Parameters		
None		
Returns		
	ULONG	Number of entries in the log

GetLogEntry

Gets a log entry specified by the index that was previously read out by ReadLog.
The log is in ascending order (index 0 is the oldest entry).

NOTE: ReadLog must be called before this function can be used.

Parameters		
Index	ULONG	Index of requested log entry
Returns		
Nothing	LogEntryDefn	Log entry for the given index. See LogEntryDefn.

Data Storage and Retrieval

These functions are used for loading and storing test configurations and results.

LoadConfigFromFile

Loads a test config from a file, and writes it to the currently connected IntelliDAQ.
Test configuration files can be created using the SaveConfigToFile function or with the Waveform Capture and IntelliDAQ applications.

WARNING: This will overwrite the existing configuration in the device. Save existing configurations with SaveConfigToFile if the old configuration needs to be saved.

Parameters		
filename	BSTR (string)	Configuration filename. (file extension: *.cfg)
Returns		
Nothing		

SaveConfigToFile

Reads the test configuration from the currently connected IntelliDAQ device and saves it to a file.
This file can be loaded using the LoadConfigFromFile function or the Waveform Capture and IntelliDAQ applications.
If the file already exists, it will be overwritten.

Parameters		
filename	BSTR (string)	Configuration filename (include the extension *.cfg)
Returns		
Nothing		

SaveLogToFile

Reads the event log data from the currently connected IntelliDAQ device and saves it to a CSV file.
If the file already exists, it will be overwritten.

Parameters		
filename	BSTR (string)	Filename to save log to.
Returns		
Nothing		

SaveTestDataToFile

Saves all the test data from the currently connected IntelliDAQ in a binary format.
This file can be loaded using the
LoadTestDataFromFile function or viewed in the IntelliDAQ application.

Parameters		
filename	BSTR (string)	Filename to save test data to (file extension *.tst).
Returns		
Nothing		

LoadTestDataFromFile

Loads test data from a file.
Test data can be saved using the SaveTestDataToFile function or with the IntelliDAQ GUI application.

Parameters		
filename	BSTR (string)	Name of file to load (file extension *.tst)
Returns		
Nothing		

Identification

These functions are used to identify the currently connected IntelliDAQ device.

GetFixtureSerialNumber

Gets the serial number of the currently connected IntelliDAQ device as a string.

Parameters		
None		
Returns		
	BSTR (string)	Serial number (ex: “001002301”)

GetFixtureVersion

Gets the version of the currently connected IntelliDAQ device as a string.

Parameters		
None		
Returns		
	BSTR (string)	Version number as string (ex: "1.0")

GetFixtureName

Gets the fixture name of the currently connected IntelliDAQ device as a string.

Parameters		
None		
Returns		
	BSTR (string)	Fixture name (ex: "Lab Fixture 1")

Data Structures

StatusDefn

The StatusDefn structure is returned by the ReadStatus function.

```
typedef struct
{
   BSTR categoryName;                  // Name of the category
   DOUBLE consumption;                 // Category consumption (units: uA*sec)
   ULONG occurances;                   // Number of times an event met this category criteria
   DOUBLE percentage;                  // Percentage of current consumption in category vs. total current
                                       // consumption (0 – 100.00)
} CategoryInfo;

typedef struct
{
   DOUBLE total;                       // Total current consumption of test (units: uA*sec)
   ULONG time;                         // Number of seconds test has run
   ULONG startTime;                    // Local time (UTC with timezone and dst offset)
   DOUBLE average;                     // Average current of the test (units: uA, max fraction: 1/1000)
   DOUBLE instantaneousAverage;        // Average over last second (units: uA, max fraction: 1/1000)
   DOUBLE activePercentage;            // Percentage of overall run time in an event (0 – 100.0000)
   DOUBLE quiescentCurrent;            // Lowest 10 msec period of current in the last second (units: uA)
   VARIANT_BOOL isRunning;             // true if test is currently running
   VARIANT_BOOL inEvent;               // true if test is currently in an event
   VARIANT_BOOL isLogFull;             // true if the log is full
   CategoryInfo uncategorizedEvents;   // Category info for events not categorized in the 16 categories
   CategoryInfo categories[16];        // User defined categories
} StatusDefn;
```

LogEntryDefn

The LogEntryDefn structure is returned by the GetLogEntry function.

```
typedef struct
{
  BYTE entryType;                      // 0: current, 1-255: reserved
  BYTE categoryNumber;                 // Index of category event occurred in. 0 = uncategorized
  BSTR categoryName;                   // Name of category
  DOUBLE time;                         // Time since start of test (units: secs, resolution: 0.1 secs)
  DOUBLE duration;                     // Duration of the event (units: usecs)
  DOUBLE total;                        // Current Consumption of the event (units: uA*sec)
  DOUBLE maxValue;                     // Max Current during event (units: uA, max fraction: 1/1000)
  DOUBLE minVoltage;                   // Minimum voltage during the event (units: mV)
  DOUBLE maxVoltage;                   // Maximum voltage during the event (units: mV)
} LogEntryDefn;
```