User Guide for Agilent iOS IO

**Introduction:**

The Agilent iOS IO (AiOS\_IO) is a tool for iOS developers who want to create software or apps for the iPad / iPhone / iPod that communicate with LXI test and measurement instrumentation. The AiOS\_IO abstracts and handles the low level network sockets for LAN based communication, making sending data to and fetching data from LXI instruments much easier for the iOS programmer. To use an analogy, it creates a VISA like IO layer for the iOS.

The AiOS\_IO folder that is available for download consists of six objective C source code files. There are three classes in the six files, three .m files and three .h files. The three classes are AiOS\_IO, SocketIO, and Error their functionality will be discussed in more detail in the following sections. The source code that makes up the AiOS\_IO is being offered free of charge and “As is” so it is not supported by Agilent for more information see the disclosure statement at the end of this document. Feel free to use and modify the AiOS\_IO source code as you see fit.

**Technical Overview of AiOS\_IO:**

When using the AiOS\_IO tools you will only need to reference and access the AiOS\_IO class and its methods. The SocketIO and Error class are referenced by AiOS\_IO class and you do not need to reference them in your project. The Error and SocketIO class just need to be included in your iOS project. The following is a description of each of the three classes:

**SocketIO 🡪** A class to perform low level socket I/O communication between an iOS device and a LAN instrument. The socket communication is done using the BSD socket API. This class is abstracted by AiOS\_IO class and does not need to be referenced by the user. To learn more about the BSD socket API [click here](http://en.wikipedia.org/wiki/Berkeley_sockets).

**AiOS\_IO 🡪** A class for doing instrument I/O communication. It provides higher level functions applicable to instruments as compared to SocketsIO. It uses SocketIO as a lower communication layer. Additional layers could be added. You can open connection, print, scan, query, and close.

**Error 🡪** This class abstracts IO error handling. Error handling is done using the Objective C NSError class. [Click here](http://developer.apple.com/library/mac/#documentation/Cocoa/Reference/Foundation/Classes/NSError_Class/Reference/Reference.html) to learn more about NSError class.

The following is an overview of the methods found in the AiOS\_IO class:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Attempt to open connection to the specific address

address: ip or hostname

port: port number

error: object where error information will be placed

returns: YES on success. NO on error.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

- (BOOL)openWithAddress:(NSString \*)address port:(int)port error:(NSError \*\*)error;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Close the connection

error: object where error information will be placed

returns: YES on success. NO on error.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

- (BOOL)closeWithError:(NSError \*\*)error;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

When connecting using a sockets connect a separate connection on a different port needs to be made when sending a device clear. The instrument provides the port number to use for a device clear. Run this function after establishing a connection with open function

query: the query to send to the instrument to retrieve the port, pass NULL to use the default

error: object where error information will be placed

returns: YES on success. NO on error.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

- (BOOL)queryDeviceClearPort:(NSString \*)query error:(NSError \*\*)error;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Send a buffer (an array) to the instrument

buffer: an array to send

size: the size of the array (bytes)

error: object where error information will be placed

returns: YES on success. NO on error.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

- (BOOL)printBuffer:(char\*)buffer size:(int)size error:(NSError\*\*)error;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Send a string (array terminated by \0) to the instrument

message: text string to send

appendNewLine: should a new line be appended if not present in the message parameter

error: object where error information will be placed

returns: YES on success. NO on error.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

- (BOOL)print:(NSString\*)message appendNewLine:(BOOL)appendNewLine error:(NSError\*\*)error;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Send a string (array terminated by \0) to the instrument and a new line will be appended to message if not present

message: text string to send

error: object where error information will be placed

returns: YES on success. NO on error.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

- (BOOL)print:(NSString\*)message error:(NSError\*\*)error;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Receive a buffer from the instrument

buffer: a pointer to an allocated array

sizeToRead: size to read from socket (bytes)

sizeRead: size of that was actually read from the socket (bytes)

error: object where error information will be placed

returns: YES on success. NO on error.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

- (BOOL)scanBuffer:(char\*)buffer sizeToRead:(int)sizeToRead sizeRead:(int\*)sizeRead error:(NSError\*\*)error;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Receive a string terminated by a newline (line) from the instrument

response: pointer to a pointer where the string response will be placed. The pointer to a pointer should be an unallocated location, or there may be a memory leak.

trimNewLine: should the end line be removed from response after it's read

error: object where error information will be placed

returns: YES on success. NO on error.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

- (BOOL)scan:(NSString\*\*)response trimNewLine:(BOOL)trimNewLine error:(NSError\*\*)error;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Receive a string terminated by a newline (line) from the instrument and trim the new line from the response

response: pointer to a pointer where the string response will be placed.

the pointer to a pointer should be an unallocated location, or there

may be a memory leak.

error: object where error information will be placed

returns: YES on success. NO on error.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

- (BOOL)scan:(NSString\*\*)response error:(NSError\*\*)error;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Receive a binary definite size block(s) as defined in the 488.2 standard for transferring binary. Some commands transfer data back in binary format to reduce bandwidth as compared to text transfer. SCPI commands "format real" and "format:border swap" are often used to tell an instrument to use binary. You need to deallocate the arrays created in buffers after the call to this function.

blocksCount: number of definite size blocks to read

buffers: an array of arrays. The first dimension should be equal to blocksCount. The second dimension should be deallocated.

error: object where error information will be placed

returns: YES on success. NO on error.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

- (BOOL)scanBinaryDefiniteSizeBlocks:(int)blocksCount buffers:(char\*\*)buffers blocksRead:(int \*)blocksRead buffersReadSize:(int \*)buffersReadSize error:(NSError\*\*)error;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Send a string to the instrument, append a newline to the string if one is not present. Receive a string terminated by a newline (line) from the instrument, trim endline from response.

query: the string to send to the instrument

response: pointer to a pointer where the string response will be placed.

the pointer to a pointer should be an unallocated location, or there

may be a memory leak.

error: object where error information will be placed

returns: YES on success. NO on error.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

- (BOOL)query:(NSString\*)query response:(NSString\*\*)response error:(NSError\*\*)error;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Send device clear to the instrument. A device clear is needed when the instrument command parser is in a bad state and is not responding.

error: object where error information will be placed

returns: YES on success. NO on error.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

- (BOOL)deviceClearWithError:(NSError \*\*)error;

**AiOS\_IO Usage Instructions:**

The following brief instructions describe how to use the AiOS\_IO in your iOS project. You can download the source code for the “StarIDN” app, which demonstrates the use of AiOS\_IO. Instructions for using AiOS\_IO:

1. Add all 3 classes (6 files) to your iOS project in Xcode
2. Add a reference in your project to the AiOS\_IO class: #import "AiOS\_IO.h"
3. Create handle to the AiOS\_IO: AiOS \*io = [[AiOS\_IO alloc] init];
4. Open connection to LAN instrument:

bool connected = [io openWithAddress:@”1.1.1.1” port:5025 error:&hError]

1. Send \*IDN? SCPI and read response:

NSString \*response;

Bool success = [io query:@"\*idn?" response:&response error:&hError];

1. Close connection: bool closed = [io closeWithError:&hError];

**“As is” Disclosure:**

AGILENT PROVIDES THIS FREE SOFTWARE "AS IS". ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO ANY WARRANTY OF NON-INFRINGEMENT, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, REASONABLE CARE AND SKILL, AND FITNESS FOR A PARTICULAR PURPOSE ARE EXPRESSLY DISCLAIMED. Agilent shall not be liable in any way whatsoever for any damages of any nature arising from its use. In no event shall Agilent 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) except if caused by willful misconduct or gross negligence arising in any way out of the use of this software, even if advised of the possibility of such damage and if it has been ensured that such data can be reconstructed with reasonable expenditure from data material provided in machine-readable form.