APSIN x000 API Description

Contents

1. Introduction to the APSIN3000

AnaPico allows customer access to Instrument DLLs for use with alternate interface and control programs. The AnaPico apsinx000.dll allows direct access to the APSIN product family through any software that can call a standard C.

2. Application Programming Interface

The AnaPico application programming interface (API) consists of the following files:

Apsin3000_lib.h
Apsin3000_lib.lib
Apsin3000_lib.dll

The header file defines the following constants:

```
#define VI_SUCCESS                  0x00000000
                    // Operation completed successfully
#define VI_SUCCESS_MAX_CNT          0x3FFF0006
                    // Number of bytes read is equal to the input count
#define VI_ERROR_SYSTEM_ERROR       0xBFFF0000
                    // Unknown system error
#define VI_ERROR_TMO                0xBFFF0015
                    // Timeout expired before operation completed
#define VI_ERROR_CONN_LOST          0xBFFF00A6
                    // Connection for the given session has been lost
#define VI_ERROR_FILE_ACCESS   0xBFFF00A1
                    // Error occurred while trying to open the specified file.
                    // Possible reasons include an invalid path or lack of access rights
```

The library provides the following commands:

apsin_find

`int apsin_find( char* devices, unsigned int tmo, int count, int *retCount )`

Locates devices in the same LAN subnet (sends broadcast message) and writes information about the located devices into *devices.

Parameters:

`devices`:	pointer to a buffer for the found devices string.
`tmo`:	search timeout, in ms
`count`:	size of the supplied buffer, in bytes
`retCount`:	pointer, returning the number of bytes written into the devices buffer, excluding the NULL-termination. Can be NULL.

Return Values:

`VI_SUCCESS`:	successful
`VI_ERROR_SYSTEM_ERROR`:	socket allocation failed

The returned `devices` string will be formatted as follows:
`"Instr Name:Serial No:IPv4 Addr;Instr Name:Serial No:IPv4 Addr;..."`

Each valid device will be identified with the instrument name, the instrument's serial number, and the IPv4 adress. The individual devices are separated with a ";" (semicolon). If no devices were found, the buffer will be an empty string "" and `retCount` will be 0.

Example:

```
char buf[512];
int retCount;
apsin_find( buf, 2000, 512, &retCount );
printf( buf );
```

will wait 2 seconds for devices to react on a broadcast call. If 3 devices were found, the output could be:

```
"INSTR0:31132200000016:192.168.1.55;INSTR0:61132200000049:192.168.1.49;INSTR0:62232211000003:192.168.1.46;"
```

apsin_sock_open

```
int apsin_sock_open( char* ipAddr, int port, unsigned int tmo, int* sock_id )
```

Opens a TCP socket connection to the specified IPv4 address and port.

Parameters:

`ipAddr`:	IPv4 address string (e.g. "192.168.1.50").
`port`:	TCP port, default is 18.
`tmo`:	connection timeout in ms, usually >= 1000.
`sock_id`:	pointer, returning the unique socket ID.

Return Values:

`VI_SUCCESS`:	successful
`VI_ERROR_SYSTEM_ERROR`:	socket allocation failed, connection refused by device, or no device listening at specified address.

Description:

Opens a TCP socket connection to the specified IPv4 address and port. The connection will be up until `apsin_sock_close()` is called. After opening, calling `apsin_sock_close()` is mandatory before exiting the program.

apsin_sock_close

```
int apsin_sock_close( int sock_id )
```

Close a TCP socket connection.

Parameters:

sock_id:	unique socket ID

Return Values:

`VI_SUCCESS`:	successful
`VI_ERROR_SYSTEM_ERROR`:	invalid socket ID (no socket connection)

apsin_sock_clear

`int apsin_sock_clear( int sock_id )`

Clears the socket receive buffer.

Parameters:

`sock_id`:	unique socket ID

Return Values:

`VI_SUCCESS`:	successful
`VI_ERROR_SYSTEM_ERROR`:	invalid socket ID (no socket connection)

apsin_sock_gets

`int apsin_sock_gets( int sock_id, char* buf, int count, int *retCount )`

Receive a string from a device.

Parameters:

`sock_id`:	unique socket ID
`buf`:	pointer to the buffer for the string received from the device
`count`:	buffer size, in bytes
`retCount`:	pointer, returning number of received bytes, can be NULL

Return values:

`VI_SUCCESS`:	successful
`VI_SUCCESS_MAX_CNT`:	response string is too long for the supplied buffer. Only `count` characters were stored in the buffer, and the buffer is not NULL-terminated. The remaining characters can be read by calling the function again.
`VI_ERROR_CONN_LOST`:	Connection lost, nothing was received

Description:

Use this function to receive requested data information from the APSIN.

apsin_sock_puts

`int apsin_sock_puts( int sock_id, char* buf, int count, int *retCount )`

Sends a string to a device.

Parameters:

`sock_id`:	unique socket ID
`buf`:	string being sent to the device
`count`:	string length, can be 0 if string is terminated with NULL or line feed '\n'
`retCount`:	pointer, returning number of sent bytes, can be 0

Return values:

`VI_SUCCESS`:	successful

VI_SUCCESS_MAX_CNT:	String is too long. Only the first 1024 characters were sent. The remaining characters must be sent again. No termination character was sent.
VI_ERROR_CONN_LOST:	Connection lost, nothing was sent

Description:

Send a string to a device. The string will be terminated according to the SCPI rules.

NULL or "\r\n" (carriage-return line-feed) will be replaced with "\n". Data located after those termination characters will be ignored. If the string contains no termination character, the function will terminate the string with '\n' after `count` characters.

Up to 1024 characters can be sent, including the string terminator.

3. Accessing the DLL with Matlab™

The commands used in Matlab™ to access DLLs are as follows:

Loadlibrary	Loads a dll
Unloadlibrary	Unloads a dll
Libisloaded	Checks to see if a library is loaded
Calllib	Calls a loaded dll function

To open the apsin3000_lib.dll, the following code sequence is used:

```
if libisloaded('apsin3000')==1,
   disp(' apsin3000 library is Loaded');
end
if libisloaded('apsin3000 ')==0,
      disp('Loading Library...');
      loadlibrary apsin3000_lib.dll apsin3000_lib.h alias apsin3000
      loadlibrary('apsin3000_lib.lib','apsin3000_lib.h');
      ipaddr = '192.168.1.50';
      socket_id = calllib( 'apsin3000', 'apsin_sock_open',
libpointer('cstring',ipaddr), size(ipaddr,2), 18, 0 );
    pause(1);
      if (sock_id < 0) || ( size(ipaddr,2) < 4 )
            fprintf(1,'\n  Unable to connect, quitting.\nPlease
verify LAN connection...\n');
            return
      end
      disp('Connection is established');
end
```

The "*libisloaded*" command checks to verify if the library has already been loaded. Therefore, this section of code can always be put in front of other commands rather than having to include separate scripts.
The "*loadlibrary*" command loads a DLL and the associated header. This command and header must be in the Matlab™ working directory or in a path that will Matlab™ recognize. Note that the name of the DLL can be aliased. In the above code, it has been aliased to "*apsin3000*". The "*calllib*" function calls out a function that is listed in the header and contained within the DLL.
The "`apsin_sock_open`" function is used here to open communications with the apsin3000. Note that the IP number of the device must be used to call the device and it is sent as a string. We use an IP 192.168.1.50 in the above example. While using the "`apsin_sock_open`" function, a 0 is returned if successful, a <0 if unsuccessful.

4. An C example program using the DLL

```
#include <stdio.h>
#include <stdlib.h>

#include "apsin3000_lib.h"  // APSIN communication library

#define DEV_STR_LEN 512     // device communication buffer size
#define TCP_PORT 18         // TCP port for socket connection

int main()
{
    char buf[DEV_STR_LEN];
    char ipAddr[32];
    int retCount, sock_id;

    /*************************************************************************
    * Open connection to device (to known IP address)
    *************************************************************************/
    printf( "Enter IP: " );
    scanf( "%s", ipAddr );
    printf( "\nOpening connection to %s on port %d\n", ipAddr, TCP_PORT );
    if ( apsin_sock_open( ipAddr, TCP_PORT, 1000, &sock_id ) != 0 )
    {
        printf( "  opening connection failed - enter to quit\n" );
        getchar();
        return -1;
    }

    /*************************************************************************
    * Communicate with device - commands and queries
    *************************************************************************/
    printf( "\nSending command: FREQ 1000000\n" );
    apsin_sock_puts( sock_id, "FREQ 1000000\n", 0, NULL );

    printf( "\nSending command: POW 0\n" );
    apsin_sock_puts( sock_id, "POW 0\n", 0, NULL );

    printf( "\nSending command: OUTP ON\n" );
    apsin_sock_puts( sock_id, "OUTP ON\n", 0, NULL );

    printf( "\nSending query: *IDN?\n" );
    apsin_sock_puts( sock_id, "*IDN?\n", 0, NULL );
    buf[0] = '\0';
    apsin_sock_gets( sock_id, buf, DEV_STR_LEN, NULL );
    printf( "  query response: %s", buf );

    /*************************************************************************
```

```
    * Close connection
    ***************************************************************************/
    printf( "\nClosing connection - enter to quit\n" );
    apsin_sock_close( sock_id );
    getchar();

    return 0;
}
```

Company Details

Address: AnaPico AG
Technoparkstr. 1
8005 Zurich
Switzerland

Phone: +41 (44) 440 00 51
Fax: +41 (44) 440 00 50
Email:
Technical Support: support@anapico.com
Sales: sales@anapico.com
Web site: www.anapico.com