Vaunix Technology Corporation

Lab Brick® LSW Switch Products

API User
Manual

Revision A

NOTICE

Vaunix has prepared this manual for use by Vaunix Company personnel and customers as a guide for the customized programming of Lab Brick products. The drawings, specifications, and information contained herein are the property of Vaunix Technology Corporation, and any unauthorized use or disclosure of these drawings, specifications, and information is prohibited; they shall not be reproduced, copied, or used in whole or in part as the basis for manufacture or sale of the equipment or software programs without the prior written consent of Vaunix Technology Corporation.

Table of Contents

1.0 OVERVIEW

The Lab Brick RF Switch Win32 SDK supports developers who want to control Lab Brick RF Switches from Windows programs, or who want to control the switches from LabVIEW[1] or other National Instruments programming environments. The SDK includes a dll which provides a Win32 API to find, initialize, and control the RF switches, along with header files and an example Win32 C program which demonstrates the use of the API. The standard version of the dll uses Microsoft's cdecl calling convention and dll linkage, so that the Lab Brick dll can be called by any software which can call a Windows Windows function. Other versions are available. If you need a dll with one of the other Microsoft supported naming conventions, or a similar library for use with the Linux family of operating systems please contact Vaunix technical support.

---

[1] LabVIEW is a trademark of National Instruments

[2] LabWindows is a trademark of National Instruments

2.0 USING THE SDK

The SDK consists of a dll, named VNX_switch.dll, along with this documentation, a C style header file, a library file for linking to the dll, and an example program written in C using Visual Studio 2008. Unzip the SDK into a convenient place on your hard disk, and then copy the dll and library file into the directory of the executable program you are creating. Add the header file (VNX_switch.h) to your project, being sure to include the surrounding extern C declaration, and include it with the other header files in your program. Make sure that the linker directives include the path of the library file.

3.0 PROGRAMMING

3.1 Overall Strategy and API Achitecture

The API provides functions for identifying how many and what type of Lab Brick RF switches are connected to the system, initializing the switches so that you can send them commands and read their state, functions to control the operation of the switches, and finally a function to close the software connection to each switch when you no longer need to communicate with it.

The API can be operated in a test mode, where the functions will simulate normal operation but will not actually communicate with the hardware devices. This feature is provided as a convenience to software developers who may not have a Lab Brick RF switch with them, but still want to be able to work on an applications program that uses the Lab Brick. Of course it is important to make sure that the API is in its normal mode in order to access the actual hardware!

Be sure to call fnLSW_SetTestMode(FALSE), unless of course you want the API to operate in its test mode. In test mode there will be 2 devices, an LSW-502PDT and an LSW-502P4T.

The first step is to identify the switches connected to the system. Call the function fnLSW_GetNumDevices() to get the number of switches attached to the system. Note that USB devices can be attached and detached by users at any time. If you are writing a program which needs to handle the situation where devices are attached or detached while the program is operating, you should periodically call fnLSW_GetNumDevices() to see if any new devices have been attached.[1]

Allocate an array big enough to hold the device ids for the number of devices present. While you should use the DEVID type declared in VNX_switch.h it's just an array of uints at this point. You may want to just allocate an array large enough to hold MAXDEVICES device ids, so that you do not have to handle the case where the number of attached devices increases.

Call fnLSW_GetDevInfo(DEVID *ActiveDevices), which will fill in the array with the device ids for each connected switch. The function returns an integer, which is the number of devices present on the machine.

---

[2] Usually it is a good idea to call fnLMS_GetNumDevices() at around 1 second intervals. While a short interval reduces the chances, it is still possible that the user will remove one device and replace it with another however, so to completely handle all the cases which can result from users hot plugging devices your application needs to check to see not only if the number of devices is different, but if the same number of devices are present, that they are not different devices.

The next step is to call fnLSW_GetModelName(DEVID deviceID, char *ModelName) with a null ModelName pointer to get the length of the model name, or just use a buffer that can hold MAX_MODELNAME chars. You can use the model name to identify the type of switch. Call fnLSW_GetSerialNumber(DEVID deviceID) to get the serial number of the switch. Based on that information, your program can determine which device to open.

Once you have identified the switch you want to send commands to, call fnLSW_InitDevice(DEVID deviceID) to actually open the device and get its various parameters like switch setting, pulse mode parameters, etc. After the fnLSW_InitDevice function has completed you can use any of the get functions to read the settings of the switch. It is best to use the InitDevice function once, at the beginning of operation, and then close the device when your program has completed its operations with the Lab Brick device.

To change one of the settings of the switch, use the corresponding set function. For example, to select a particular switch, call fnLSW_SetSwitch (DEVID deviceID, int switch_number). The first argument is the device id of the RF switch, the second is the desired switch selection. The RF outputs are numbered sequentially from 1 to 4 for the LSW-502P4T. For the LSW-502PDT the output 1 or 2 can be selected.

When you are done with the device, call fnLSW_CloseDevice(DEVID deviceID).

3.2 Status Codes

All of the set functions return a status code indicating whether an error occurred. The get functions normally return an integer value, but in the event of an error they will return an error code. The error codes can be distinguished from normal data by their numeric value, since all error codes have their high bit set, and they are outside of the range of normal data.

Functions that return a floating point result use specific, negative numeric values to indicate if an error occurred.

A separate function, fnLSW_GetDeviceStatus(DEVID deviceID) provides access to a set of status bits describing the operating state of the switch. This function can be used to check if a device is currently connected or open.

The values of the status codes are defined in the VNX_LSW_api.h header file.

3.3 Functions – Selecting the Device

VNX_SWITCH_API void fnLSW_SetTestMode(bool testmode)

Set testmode to FALSE for normal operation. If testmode is TRUE the dll does not communicate with the actual hardware, but simulates the basic operation of the dll functions. It does not simulate the operation of switch patterns or pulse mode operation generated by the actual hardware, but it does simulate the behavior of the functions used to get and set the parameters for the patterns and pulsed mode operation.

VNX_SWITCH_API int fnLSW_GetNumDevices()

This function returns a count of the number of connected switches.

VNX_SWITCH_API int fnLSW_GetDevInfo(DEVID *ActiveDevices)

This function fills in the ActiveDevices array with the device ids for the connected switches. Note that the array must be large enough to hold a device id for the number of devices returned by fnLSW_GetNumDevices. The function also returns the number of active devices, which can, under some circumstances, be less than the number of devices returned in the previous call to fnLSW_GetNumDevices.

The device ids are used to identify each device, and are used in the rest of the functions to select the device. Note that while the device ids may be small integers, and may, in some circumstances appear to be numerically related to the devices present, they should only be used as opaque handles.

VNX_SWITCH_API int fnLSW_GetModelName(DEVID deviceID, char *ModelName)

This function is used to get the model name of the switch. If the function is called with a null pointer, it returns just the length of the model name string. If the function is called with a non-null string pointer it copies the model name into the string and returns the length of the string. The string length will never be greater than the constant MAX_MODELNAME which is defined in VNX_switch.h This function can be used regardless of whether or not the switch has been initialized with the fnLSW_InitDevice function.

VNX_SWITCH_API int fnLSW_GetSerialNumber(DEVID deviceID)

This function is used to get the serial number of the switch. It can be called regardless of whether or not the switch has been initialized with the fnLSW_InitDevice function. If your system has multiple Lab Brick RF Switches, your software should use each device's serial number to keep track of each specific device. Do not rely upon the order in which the devices appear in the table of active devices. On a typical system the individual switches will typically be found in the same order, but there is no guarantee that this will occur.

VNX_SWITCH_API int fnLSW_GetDeviceStatus(DEVID deviceID)

This function can be used to obtain information about the status of a device, even before the device is initialized. (Note that information on the dynamic activity of the device, such as whether a pattern is active is not guaranteed to be available before the device is initialized.)

VNX_SWITCH_API int fnLSW_InitDevice(DEVID deviceID)

This function is used to open the device interface to the switch and initialize the dll's copy of the device's settings. If the fnLSW_InitDevice function succeeds, then you can use the various fnLSW_Get* functions to read the switch's settings. This function will fail, and return an error code if the switch has already been opened by another program.

VNX_SWITCH_API int fnLSW_CloseDevice(DEVID deviceID)

This function closes the device interface to the switch. It should be called when your program is done using the switch.

3.4 Functions – Setting parameters on the Switch

VNX_SWITCH_API LVSTATUS fnLSW_SetSwitch (DEVID deviceID, int select)

This function is used to set the position of the switch. The first argument is the device id of the RF switch, the second is the desired switch selection. The RF outputs are numbered sequentially from 1 to 4 for the LSW-502P4T. For the LSW-502PDT the output 1 or 2 can be selected.

VNX_SWITCH_API LVSTATUS fnLSW_SetUseExternalControl (DEVID deviceID, bool external);

This function is used to select internal or external control of the RF switches. If external is TRUE, then the Lab Brick RF switch will be controlled by the external control signal input or inputs.

VNX_SWITCH_API LVSTATUS fnLSW_SetPattern(DEVID deviceID, int num_entries, int sw_select[], int holdtime[])

This function sets the parameters for a switch pattern. A switch pattern consists of a set of pattern elements, where each element defines a switch setting and a hold time. When the pattern is activated the Lab Brick RF Switch steps through the pattern elements, waiting for the specified hold time at each step. Hold times are specified in milliseconds, with the minimum being 1 millisecond. Currently, a pattern can have at most four entries, so the maximum value for num_entries is 4. The array of switch selections, sw_select, has one element for each step in the pattern, and that element holds a switch number from 1 to 4. To start or stop a pattern use the fnLSW_StartPattern function.

VNX_SWITCH_API LVSTATUS fnLSW_StartPattern(DEVID deviceID, bool go)

Calling this function with go set to TRUE starts a switch pattern sequence at the beginning. To stop the pattern, call this function with go set to FALSE.

VNX_SWITCH_API LVSTATUS fnLSW_SetPatternType(DEVID deviceID, bool continuous)

Calling this function with continuous set to TRUE before starting the pattern in order to have the pattern repeat. If continuous is set to FALSE the pattern will only run once when it is started.

VNX_SWITCH_API LVSTATUS fnLSW_SetPatternEntry(DEVID deviceID, int sw_select, int holdtime, int index, bool last_entry)

This function can be used to set individual elements of the pattern. The argument sw_select is the switch setting, from 1 to 4. The argument holdtime is the length of time that the pattern will hold each switch setting, expressed as an integer number of milliseconds. The argument index is the zero based position in the pattern, ranging from 0 to 3. The last_entry argument should be set to TRUE only for the final element in the pattern. For example, the following set of calls define a pattern with three steps, where switch 1 is active for 1 second, switch 2 is active for .1 seconds, and switch 3 is active for 10 seconds on device 5:

```
result = fnLSW_SetPatternEntry(5, 1, 1000, 0, FALSE);
result = fnLSW_SetPatternEntry(5, 2, 100, 1, FALSE);
result = fnLSW_SetPatternEntry(5, 3, 10000, 2, TRUE);
```

VNX_SWITCH_API LVSTATUS fnLSW_SetFastPulsedOutput(DEVID deviceID, float pulseontime, float pulsereptime, bool on)

This function is the preferred way to control the internal pulse switching option. The pulseontime parameter is the length of the pulse on time (switch 1 active) in seconds. The pulsereptime parameter is the length of the repetition period in seconds. Both values can range from 100 nanoseconds (0.100e-6) to 1000 seconds (1.0e3). Set on = TRUE to start the pulsed output modulation.

VNX_SWITCH_API LVSTATUS fnLSW_SetPulseOnTime(DEVID deviceID, float pulseontime)

This function is used to set the length of the RF pulse on time of the device’s internal pulse switching. The pulseontime parameter is the length of the pulse on time (switch 1 active) in seconds, with a 100 nanosecond minimum. This function is not recommended for general use. Instead use the fnLSW_SetFastPulsedOutput function.

VNX_SWITCH_API LVSTATUS fnLSW_SetPulseOffTime(DEVID deviceID, float pulseofftime)

This function is used to set the length of the RF pulse off time of the device’s internal pulse switching. The pulseofftime parameter is the length of the pulse off time (switch 2 active) in seconds, with a 100 nanosecond minimum. The repetition period of the pulse modulation is equal to pulseontime + pulseofftime. This function is not recommended for general use. Instead use the fnLSW_SetFastPulsedOutput function.

VNX_SWITCH_API LVSTATUS fnLSW_EnableInternalPulseMod(DEVID deviceID, bool on)

This function is used to turn on and off the internal pulse switching. If on = TRUE the switch will switch the RF output between switch 1 and switch 2 according to the values set for the pulse on time and pulse off time using either the fnLSW_SetFastPulsedOutput function or the functions to set pulse on and off time directly . To stop the internal pulse switching, set on = FALSE. Always disable internal pulse switching before setting the pulse on and off time using the fnLSW_SetPulseOnTime and fnLSW_SetPulseOffTime functions.

VNX_SWITCH_API LVSTATUS fnLSW_SaveSettings(DEVID deviceID)

The Lab Brick RF Switches can save their settings, and then resume operating with the saved settings when they are powered up. Set the desired parameters, then use this function to save the settings.

802P4T/802PDT Switch API call

VNX_SWITCH_API LVSTATUS fnLSW_SetSwitchRFoutput(DEVID deviceID, int swindex, int swport)

This function is used to set the Switch RF output state for the corresponding switch index. The first argument is the switch index and second one is the RF outputs are numbered sequentially from 1 to 4(802P4T)/2(802PDT).

3.5 Functions – Reading parameters from the Switch

VNX_SWITCH_API int fnLSW_GetNumSwitches (DEVID deviceID)

This function returns the number of switches in the selected device. This is a read only value.

VNX_SWITCH_API int fnLSW_GetActiveSwitch (DEVID deviceID)

This function returns the current switch connection of the selected device. This value may differ from the current switch setting when an external signal is used to control the switch, or when a switch pattern is running, or during pulse mode operation. Note that for rapidly changing switch connections due to an external signal, switch patterns or pulse mode operation the value returned by the GetActiveSwitch function may not be a useful indicator of the actual switch connection since the value returned represents the switch connection at the last status report which is asynchronous with respect to the call to the GetActiveSwitch function.

VNX_SWITCH_API int fnLSW_GetSwitchSetting (DEVID deviceID)

This function returns the current switch setting of the selected device. In normal operation, this value is the same as the active switch, except in the conditions described above.

VNX_SWITCH_API int fnLSW_GetUseExternalControl (DEVID deviceID)

This function returns a non-zero value if the Lab Brick RF Switch has been set to use an external signal to control the switches.

VNX_SWITCH_API float fnLSW_GetPulseOnTime(DEVID deviceID)

This function returns the pulse on time, which is the length of time that RF input is connected to output switch 1 when internal pulse modulation is operating, in seconds.

VNX_SWITCH_API float fnLSW_GetPulseOffTime(DEVID deviceID)

This function returns the pulse off time, which is the length of time that RF output is connected to switch 2 when internal pulse modulation is operating, in seconds. The pulse repetition period is equal to the pulse on time added to the pulse off time.

VNX_SWITCH_API int fnLSW_GetPulseMode(DEVID deviceID)

This function returns an integer value which is 1 when the RF switch's internal pulse modulation is active, or 0 when the internal pulse modulation is off.

VNX_SWITCH_API int fnLSW_GetHasFastPulseMode(DEVID deviceID)

This function is included for compatibility with software developed for other Lab Brick products. All Lab Brick RF Switches have fast pulse mode switching.

VNX_SWITCH_API int fnLSW_GetPatternLength (DEVID deviceID);

This function returns an integer value which is the number of elements in the switch pattern. Currently, the maximum pattern length is 4. A pattern length of 0 indicates that no pattern has been loaded into the Lab Brick RF Switch.

VNX_SWITCH_API int fnLSW_GetPatternType (DEVID deviceID);

This function returns the current pattern type. A value of 1 indicates that a single shot pattern was selected, and a value of 2 indicates that a repeating pattern was selected.

VNX_SWITCH_API int fnLSW_GetPatternEntrySwitch (DEVID deviceID, int index);

This function returns the switch setting for a particular element in the array of switch settings that define the switch pattern. The index ranges from 0 to 3. A value of zero indicates the end of the pattern, while values of 1 to 4 indicate the switch setting for that step in the pattern.

VNX_SWITCH_API int fnLSW_GetPatternEntryTime (DEVID deviceID);

This function returns the hold time for a particular element in the array of switch settings that define the switch pattern. The index ranges from 0 to 3. The integer value returned is the length of time, in 1 millisecond increments that the switch will remain at that step in the pattern.

802P4T/802PDT Switch new API Calls

VNX_SWITCH_API int fnLDA_GetIPMode(DEVID deviceID)

This function is used to read the IP mode configuration of the device. Response data “0” represents the “Static” mode, “1” represents the “DHCP” mode.

VNX_SWITCH_API int fnLDA_GetIPAddress(DEVID deviceID, char *ip)

This function is used to read the IP address of the device.

VNX_SWITCH_API int fnLDA_GetNetmask(DEVID deviceID, char *netmask)

This function is used to read the netmask of the device.

VNX_SWITCH_API int fnLDA_GetGateway (DEVID deviceID, char *gateway)

This function is used to read the gateway address of the device.

VNX_SWITCH_API int fnLSW_GetMaxSwitchDevices(DEVID deviceID)

This function will return the Max switch devices.

VNX_SWITCH_API int fnLSW_GetSwitchRFoutput(DEVID deviceID, int swindex)

This function the switch port status of the corresponding switch index request.

4.0 PROGRAMMING SUPPORT

Lab Brick programming support is available from Vaunix Technology Corporation. Please contact our technical support group by email - LabBrickSupport@Vaunix.com.
Vaunix Technology also offers custom programming solutions. Send us your requirements to receive a fixed rate project quotation.
Thank you for using our Lab Brick products.