Vaunix Technology Corporation
Lab Brick® Family of Digital Attenuators

API User Manual

Revision A1

Table of Contents

1.0 OVERVIEW 4
2.0 USING THE SDK 5
3.0 PROGRAMMING 6
3.1 Overall Strategy and API Achitecture 6
3.2 Status Codes 7
3.3 Functions – Selecting the Device 7
3.4 Functions – Setting parameters on the Attenuator 9
3.5 Functions – Reading parameters from the Attenuator 10
4.0 PROGRAMMING SUPPORT 12

1.0 OVERVIEW

The LabBrick Programmable Attenuator Win32 SDK supports developers who want to control LabBrick programmable attenuators from Windows programs, or who want to control the attenuators 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 attenuators, along with header files and an example Win32 C program which demonstrates the use of the API.

[1] LabView is a trademark of National Instruments

2.0 USING THE SDK

The SDK consists of a dll, named VNX_atten.dll, along with this documentation, a C style header file, a library file for linking to the dll, and a VC 6 example program. 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_atten.h) to your project, 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 LabBrick attenuators are connected to the system, initializing attenuators so that you can send them commands and read their state, functions to control the operation of the attenuators, and finally a function to close the software connection to the attenuator 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 LabBrick attenuator with them, but still want to be able to work on an applications program that uses the LabBrick. 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 fnLDA_SetTestMode(FALSE), unless of course you want the API to operate in its test mode. In test mode there will be 2 devices, an LDA-102 and an LDA-602.

The first step is to identify the attenuators connected to the system. Call the function fnLDA_GetNumDevices() to get the number of attenuators 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 fnLDA_GetNumDevices() to see if any new devices have been attached.[2]

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_atten.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 attenuators increases.

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

[2] Usually it is a good idea to call fnLDA_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, but they are different devices.

The next step is to call fnLDA_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 attenuator. Call fnLDA_GetSerialNumber(DEVID deviceID) to get the serial number of the attenuator. Based on that information, your program can determine which device to open.

Once you have identified the attenuator you want to send commands to, call fnLDA_InitDevice(DEVID deviceID) to actually open the device and get its various parameters like attenuation setting, attenuation ramp parameters, etc. After the fnLDA_InitDevice function has completed you can use any of the get functions to read the settings of the attenuator.

To change one of the settings of the attenuator, use the corresponding set function. For example, to set the attenuation level, call fnLDA_SetAttenuation(DEVID deviceID, int attenuation). The first argument is the device id of the attenuator, the second is the value of the attenuation you want to set the attenuator to. Attenuation is specified in .25 db units, so 10 db of attenuation is represented as 40, 6 db of attenuation is represented as 24, and .5 db, the minimum attenuation increment, is represented as 2.

When you are done with the device, call fnLDA_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.

A separate function, fnLDA_GetDeviceStatus(DEVID deviceID) provides access to a set of status bits describing the operating state of the attenuator. 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_atten header file.

3.3 Functions – Selecting the Device

VNX_ATTEN_API void fnLDA_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 attenuation ramps generated by the actual hardware, but it does simulate the behavior of the functions used to set the parameters for the ramps.

VNX_ATTEN_API int fnLDA_GetNumDevices()

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

VNX_ATTEN_API int fnLDA_GetDevInfo(DEVID *ActiveDevices)

This function fills in the ActiveDevices array with the device ids for the connected attenuators. Note that the array must be large enough to hold a device id for the number of devices returned by fnLDA_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 fnLDA_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_ATTEN_API int fnLDA_GetModelName(DEVID deviceID, char *ModelName)

This function is used to get the model name of the attenuator. 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_atten.h This function can be used regardless of whether or not the attenuator has been initialized with the fnLDA_InitDevice function.

VNX_ATTEN_API int fnLDA_GetSerialNumber(DEVID deviceID)

This function is used to get the serial number of the attenuator. It can be called regardless of whether or not the attenuator has been initialized with the fnLDA_InitDevice function. If your system has multiple attenuators, 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 attenuators will typically be found in the same order, but there is no guarantee that this will occur.

VNX_ATTEN_API int fnLDA_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 sweep or ramp activity of the device is not guaranteed to be available before the device is initialized.)

VNX_ATTEN_API int fnLDA_InitDevice(DEVID deviceID)

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

VNX_ATTEN_API int fnLDA_CloseDevice(DEVID deviceID)

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

3.4 Functions – Setting parameters on the Attenuator

VNX_ATTEN_API LVSTATUS fnLDA_SetAttenuation(DEVID deviceID, int attenuation)

This function is used to set the attenuation level of the programmable attenuator. The attenuation setting is encoded as an integer where each increment represents .25db of attenuation. The encoding is:

attenuation * .25db = Attenuation in db

For example, attenuation = 40 for 10db of attenuation, 2 for .5db of attenuation (the minimum resolution of the LDA-102 and LDA-602 series hardware), and 252 for 63db attenuation.

VNX_ATTEN_API LVSTATUS fnLDA_SetRampStart(DEVID deviceID, int rampstart)

This function sets the attenuation level at the beginning of an attenuation ramp or sweep. The encoding of rampstart, the attenuation level, is the same as the fnLDA_SetAttenuation function.

VNX_ATTEN_API LVSTATUS fnLDA_SetRampEnd(DEVID deviceID, int rampstop)

This function sets the attenuation level at the end of an attenuation ramp or sweep. The encoding of rampstop, the attenuation level, is the same as the fnLDA_SetAttenuation function.

VNX_ATTEN_API LVSTATUS fnLDA_SetAttenuationStep(DEVID deviceID, int attenuationstep)

This function sets the size of the attenuation step that will be used to generate the attenuation ramp or sweep. The encoding of attenuationstep, is the same as the fnLDA_SetAttenuation function. The smallest attenuation step size is 2 or .5 db.

VNX_ATTEN_API LVSTATUS fnLDA_SetDwellTime(DEVID deviceID, int dwelltime)

This function sets the length of time that the attenuator will dwell on each attenuation step while it is generating the attenuation ramp. The dwelltime variable is encoded as the number of milliseconds to dwell at each level. The minimum dwell time is 1 millisecond.

VNX_ATTEN_API LVSTATUS fnLDA_SetIdleTime(DEVID deviceID, int idletime)

This function sets the length of time that the attenuator will wait at the end of an attenuation ramp before beginning the ramp again when the ramp mode is set to SWP_REPEAT. The idletime variable is encoded as the number of milliseconds to dwell at each level. The minimum idle time is 0 milliseconds.

VNX_ATTEN_API LVSTATUS fnLDA_SetRFOn(DEVID deviceID, bool on)

This function allows rapid switching of the attenuator from its set value "on" (on = TRUE) to its maximum attenuation (on = FALSE).

VNX_ATTEN_API LVSTATUS fnLDA_SetRampDirection(DEVID deviceID, bool up)

This function is used to set the direction of the attenuation ramp. To create a ramp with increasing attenuation, set up = TRUE. Note that the ramp start attenuation value must be less than the ramp end attenuation value for a ramp with increasing attenuation. For a ramp with decreasing attenuation the ramp start value must be greater than the ramp end value.

VNX_ATTEN_API LVSTATUS fnLDA_SetRampMode(DEVID deviceID, bool mode)

This function is used to select either a single ramp or sweep of attenuation values, or a repeating series of ramps. If mode = TRUE then the ramp will be repeated, if mode = FALSE the ramp will only happen once.

VNX_ATTEN_API LVSTATUS fnLDA_StartRamp(DEVID deviceID, bool go)

This function is used to start and stop the attenuation ramps. If go = TRUE the attenuator will begin sweeping, FALSE stops the sweep.

VNX_ATTEN_API LVSTATUS fnLDA_SaveSettings(DEVID deviceID)

The LabBrick attenuators 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.

3.5 Functions – Reading parameters from the Attenuator

VNX_ATTEN_API int fnLDA_GetAttenuation(DEVID deviceID)

This function returns the current attenuation setting of the selected device. When an attenuation ramp is active this value will change dynamically to reflect the current setting of the device. The return value is in .25 db units.

VNX_ATTEN_API int fnLDA_GetRampStart(DEVID deviceID)

This function returns the current attenuation ramp start value setting of the selected device. The return value is in .25 db units.

VNX_ATTEN_API int fnLDA_GetRampEnd(DEVID deviceID)

This function returns the current attenuation ramp end setting of the selected device.

VNX_ATTEN_API int fnLDA_GetAttenuationStep(DEVID deviceID)

This function returns the current attenuation step size setting of the selected device. The return value is in .25 db units, so for example an attenuation step of 5db would be represented by a return value of 20.

VNX_ATTEN_API int fnLDA_GetDwellTime(DEVID deviceID)

This function returns the current dwell time for each step on the attenuation ramp in milliseconds. A one second dwell time, for example, would be returned as 1000.

VNX_ATTEN_API int fnLDA_GetIdleTime(DEVID deviceID)

This function returns the idle time, which is the delay between attenuation ramps when the device is in the repeating ramp mode, in milliseconds.

VNX_ATTEN_API int fnLDA_GetRF_On(DEVID deviceID)

This function returns an integer value which is 1 when the attenuator is “on”, or 0 when the attenuator has been set “off” by the fnLDA_SetRFOn function. Note that the function does not attempt to interpret attenuation settings as either “on” or “off”, so if you set the attenuation level to 63 db, (attenuation = 252) the output signal level would be the same as if you had used the fnLDA_SetRFOn function with the on = FALSE, but this function would not return 0.

VNX_ATTEN_API int fnLDA_GetMaxAttenuation(DEVID deviceID)

This function returns the maximum attenuation value that the device can provide. For the LDA-102 and LDA-602 programmable attenuators this value is 63 db, which is 252 .25 db units. Since future products may have different maximum attenuation capabilities your software should use this function to obtain the maximum attenuation possible.

VNX_ATTEN_API int fnLDA_GetMinAttenuation(DEVID deviceID)

This function returns the minimum attenuation value that the device can provide. For the LDA-102 and LDA-602 programmable attenuators this value is 0 db. Since future products may have different capabilities your software should use this function to obtain the minimum attenuation possible.

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.