

CVI Spectral Products - East

Spectrometers · Spectrophotometers · Color Instruments · Spectrographs · Monochromators

111 Highland Drive · Putnam, CT · 06260 · USA

PHONE (860) 928-5834 (860) 928-1928 · FAX (860) 928-1515 (860) 928-2676

<http://www.cvispectral.com>

SM32Pro SDK

Users Manual

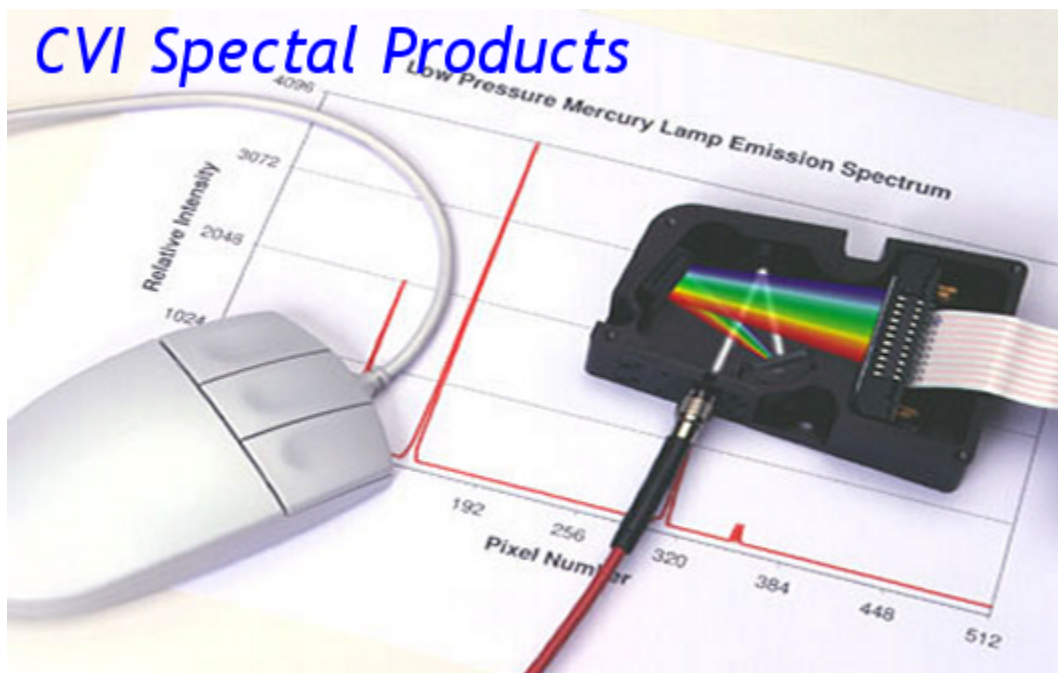


TABLE OF CONTENTS

GETTING STARTED	3
USING THE CVI LIBRARIES	5
GENERAL OVERVIEW	6
USING CURVE FITTING TO CALIBRATE SPECTRAM.....	7
SDK FUNCTIONS	9
ALPHABETICAL FUNCTION REFERENCE.....	10
<i>cviTestCard</i>	10
<i>cviSetIntEx</i>	11
<i>cviSetInt</i>	12
<i>cviTriggerReadEx</i>	13
<i>cviCloseEx</i>	14
<i>cviPolyFit</i>	15
<i>cviPolyCalc</i>	16
SAMPLE CODE.....	17

Getting Started

Welcome to the CVI Spectral Products **SM32Pro Software Development Kit**.

This kit documents a library of functions for accessing the **PCI/ISA/PCMCIA NI-DAQ Data Acquisition boards** used by the CCD detector units. Two fundamental CCD imaging concepts are the Collection of Data and Generating Meaningful Values from that data.

Collection of Data

In our CCD spectrometer, the light is dispersed across a charge-coupled device (CCD) 2086 pixel detector array. Data is collected by each pixel and converted to a relative value by the analog-to-digital (A/D) converter ranging from 0 to 4095 and representing the intensity of the light at each pixel. We are able to control the amount of time that the pixels collect light and thus read signals of varying strengths. Using this library of functions you will be able to adjust the signal capture time (Integration Time) and collect the converted signals from each pixel. The criteria to adjust the integration is to try different lengths of time with a reference signal representing the maximum possible signal level during the measurement until the peak value across the range of pixels is close to, but not at 4095. We recommend trying to get the peak between 3900 and 4000. This way you can be sure that you are getting an optimized measurement condition with no saturation of CCD elements.

Generating Meaningful Values

The information you collect from the system is only the relative signal size at each detector pixel until it has been calibrated to absolute (certified reference) values. That is in the raw data, the pixel location represents spatial distribution of the CCD elements and the corresponding intensity is represented by a digital value. Since we can adjust the strength of the light source and the integration time of the CCD, this A/D value only provides a relative representation of the real world parameters.

We address both of these problems. We connect pixels to wavelengths by use of known emission lines (e. g. CVI calibration light sources or narrow band filters) across the range of pixels together with a table relating pixels to wavelengths in nanometers. Curve fitting functions can then be applied to generate a polynomial function for the conversion of all other pixels to wavelengths.

We address the second problem in one of two ways. Both ways involve "normalization." Normalization involves measurements of signal strength based on its ratio with respect to reference signal intensity. If we are only interested in the spectral distribution of the sample signal, then we can normalize a sample signal scan to one reference value by, for example, intensities of all the array elements divided by the peak intensity. If we want to measure percent transmission, the light can be measured first with only air in the light path as 100% reference and then the sample can be inserted into the light path and a sample scan followed. Consequently, the divisions of sample scan readings by the 100% reference yields the relative transmission values, or percent transmission when multiplied by 100%. For reflectance measurement similar practice

can be applied and a high reflector can be used as a reference air in many cases.

Background Scan. It is strongly recommended that a measurement be taken prior to any sample scan, in which no external light but only the detector noise will be sensed and subsequently subtracted from all the following measurements. This will subtract thermally generated “background” DC offset level and statistically establish a baseline of zero for all subsequent measurements.

Using the CVI Libraries

NOTE:

All libraries mentioned below are installed when NI-Daq is installed.

For Windows C/C++, you must include the correct "include" file and "lib" file for the functions you are calling:

32-bit NI-DAQ applications also require the 32-bit NI-DAQ header, "nidaqex.h" and libraries, "nidaq32.lib" and "nidex32.lib". "Nidaqex.h" also references other NI-DAQ headers that you must include during compilation either by copying to the local build directory or by reference in your compiler's "directories" option.

NI-DAQ cards require the NIDAQ libraries, "nidaq32.lib" and "nidex32.lib" to be included in the program's link option because they reference "nidaq32.dll" and "nidex32.dll" dynamic link libraries which should have been installed into the "RootDrive:\Windows\System" directory when the NI-DAQ configuration software from the National Instruments CD was installed.

For Visual Basic, please include the correct "BAS" file. The function declarations listing in this file demonstrate the correct data types. To pass a "pointer" or array to a DLL from Visual basic, simply pass the first element of the array (IE SampleData[0]).

Please refer to the samples or give us a call if you need any further technical assistance.

Procedure for using the .dll with vc++

1. Copy cvidbni.dll to your **work** or **windows\system** folder.
2. Copy cvidbni.lib to your **work** or **release** folder.
3. Add **release\cvidbni.lib** to the **project/settings/link/object/library** module.

General Overview

Some basic fundamentals of utilizing the CVI SDK functions:

You must keep track of the base address of the acquisition board and the current integration time in your program.

Then you should first call ***cviTestCard*** to get the appropriate information of the board. Then you should call ***cviSetIntEx*** to set the integration time appropriately for your application. A **dummy read** of the data, described in the next paragraph should follow this function call. Optionally, integration time selection may be automated with code by reading a reference sample, scanning the data to find the peak value, scaling the integration time to move the peak to your desired range, and repeating this algorithm until your reference sample falls within a desired range. This process works best if you select an arbitrary high and low value for the integration time and make guesses in between the two. If a guess is too high (the peak is above your desired range), then your current guess becomes the previous guess averaged with the low value and the new high value becomes your previous guess. If a guess is too low (the peak value falls below your desired range), then your current guess becomes the average of the previous guess and the high value and the new low value becomes your previous guess.

Integration time is the time period the CCD pixels are exposed to light before the resulting charges are read out. A longer integration time can allow you to detect a lower light level signal. The longer your integration time is, the more background signal will accumulate.

Using Curve Fitting to Calibrate SM32Pro

Curve fitting is used in SM32Pro to correlate the physical locations of pixels on the

CCD with the known wavelength of the radiation falling on them.

This is done by identifying the pixel locations where the maximal of the known wavelength is at. These peak intensity wavelengths and pixels are used by ***cviTriggerRead*** to generate a correlating polynomial function which best represents all the data points. We have found that using a third order polynomial function produced the most desirable results for most cases. In cases where high dispersion elements are used, lower order polynomial functions may have to be utilized due to the limited known wavelengths available from the calibration lamps

Calibration Files

Each unit's calibration set is included in the SpectraM or SpectraM-XL software settings. For OEM customers who do not receive a data acquisition board, the calibration set is included on a floppy disk under the filename "wcal-xxxxxxx.txt", where the 7 "x" is the two letter, 5-digit unit serial number. This text file contains calibration data of the form "DataX=Wavelength;Pixel". The file also contains the regression coefficients "AZero value", "AOne value", ... , "BThree value" that satisfy the equations

$$\lambda_i = A_0 + A_1 P_i + A_2 P_i^2 + A_3 P_i^3$$

$$P_i = B_0 + B_1 \lambda_i + B_2 \lambda_i^2 + B_3 \lambda_i^3.$$

The A values are the coefficients for conversions from a pixel number to a wavelength in "nm" by use of the above first third order polynomial function. The B values allow the conversions from a desired wavelength in "nm" to a pixel number. For example, the following is a portation of file "wcal-HR90190.txt", which is for use with the example unit HR90190.

SDK Functions

Data Acquisition

cviTestCard

Used by the program to test the acquisition board.

cviSetIntEx

Used by the program to initialize the acquisition board.

cviSetInt

Used by the program to set integration time.

cviTriggerReadEx

Used by the program to collect spectral data.

cviCloseEx

Used by the program to reset all board options upon resetting the software.

Calibration***cviPolyFit***

Calculates the coefficients for a polynomial curve fitting function given an array of independent variables and a corresponding array of dependent variables. Used to generate a calibration function from pixels to wavelength.

cviPolyCalc

Calculates a polynomial function given the independent variable and a coefficient array. Used for determining the wavelength for a given pixel location.

Procedure of getting data***cviTestCard***

Tests the card to see if it is working properly.

cviSetIntEx

Initialize the board and sets integration time at the same time.

cviSetInt

Sets integration time.

cviTriggerReadEx

Reads acquired data.

cviCloseEx

Resets all the values to default values. (should be called at the exiting of program)

Alphabetical Function Reference

cviTestCard

int cviTestCard

(
short nDeviceNumber; // device number


```

short nBoardType;      // board type
short nGain;           // gain
)

```

This function is used to test the resources of the IO board

nDeviceNumber is the device number of the IO board.

nBoardType is the type of board. (If the board is PCI-1200, it should be 0, if the board is 6023, it should be 1)

nGain is the set intensity of acquired data (should be set to 2 always)

RETURN

If the board is found successfully the function will return a positive return value.

If the board is not found successfully the function will return a negative number

cviSetIntEx

```

long cviSetIntEx
(
int const nBase           //Base address of DPIO board
int const nPixelNumber    //a pixel number between 1 and 2086
long const lTime          //new integration time

```

```

int nDeviceNumber          //device number of installed card
int nChannel               //channel the device is found on
int nGain                  //set gain to amplify signal
)

```

This function is used to initialize the board.

wBase is the base address of the DPIO expansion board. (This should be 0x300 in Hex)
ITime is amount of time in milliseconds to set as the new integration time. This number should range between 35 and 65535 for the 1200 series or 11 and 65535 for the 6023.

nPixelNumber Informs the number of pixels (13-18 for optical blank and 33-2080 for the 2048 pixels)

nDeviceNumber informs the function of which IO device number to run. (You find this number in the NIMAX.exe utility installed with the device drivers.

nChannel informs the function of which Channel the device is located on

nGain informs the function of how much to amplify the signal. (This should be 2)

RETURN

Function returns the new integration time.

cviSetInt

```

long cviSetInt
(
int const nTriggerMode    //Sets trigger mode
long const ITime,         //new integration time
)

```

This function is used to change the integration time.

nTrigger is used to set the trigger mode. (Free Run = 10, Software Trigger = 11, Hardware Trigger = 12)

ITime is amount of time in milliseconds to set as the new integration time. This number should range between 35 and 65535 for the 1200 series or between 11 and 65535 for the 6023.

RETURN

Function returns the new integration time.

cviTriggerReadEx

```
int cviTriggerReadEx
(
int const nBase;    // Base address of acquisition board
long const Time;    // Current integration time
int const nUnitType; // nUnitType as defined above
USHORT* pArray;    // The array in which spectral data is stored
)
```

wBase is the base address of the data acquisition plug in board. (should be 0x300 in Hex)

Time is the current integration time. ***Time does not set the integration time, but is used for determining when the function should return an error if it gets no response.***

nUnitType informs the function of what type of spectrometer unit is connected to the IO board at wBase. (should be 0)

pArray points to a read memory address with 2086 unsigned short integers.

RETURN

If the board triggers successfully, then the function returns the number of data. If it does not trigger successfully, then the board returns -1.

cviCloseEx

int cviCloseEx

(
)

This function is called to reset all default valued for cviTestCard upon closing of the application

RETURN

1

cviPolyFit

```
int cviPolyFit
(
double far* x,      // Array of independent variables
double far* y,      // Array of dependent variables
int const numPts,   // Number of points in independent and dependent arrays
double far* coefs,  // Pointer to array to hold calculated coefficients [index from 0 to
order]
int const order     // Order of polynomial
)
```

This curve fitting function is used to find a polynomial function to calculate the wavelength

of a given pixel.

This function is used for calibration purposes. Either a calibration light source or a series of narrow band filters are scanned and the pixel location of all known peaks are identified along with the known wavelength at that peak. These peak locations and wavelengths are stored in the arrays *x* and *y*, respectively. The arrays indices should range from 0 to [Number_of_Points - 1]. They are passed to the function along with an requested order for the polynomial fitting function and an array large enough to hold the coefficients (). This array is then used with ***cviPolyCalc*** to calculate wavelength from pixels.

x is an array containing the independent variables. It should range from 0 to (numPts-1).

y is an array containing the dependent variables. It should range from 0 to (numPts-1).

numPts is the number of points in the variable arrays

coefs is a pointer to the array that will contain the polynomial coefficients. It should range from 0 to (order-1).

order is the desired order of the polynomial. We have determined third order to be the optimum for wavelength calibration for most cases.

RETURN

This function will return 1 if the function is successful. Otherwise it will return negative.

cviPolyCalc

```
void cviPolyCalc
(
double far* coefs, //
int const order,
int const x,
double far* y
)
```

This function calculates for the following formula:

$y = a_0 + a_1*x^1 + a_2*x^2 + \dots + a_N*x^N$, where * specified multiplication and ^ specifies "to the power of."

coefs is a pointer to an array containing the polynomial coefficients. These can be calculated using ***cviPolyFit***.

order specified the order of the polynomial equation and must be less than or to equal to the number of elements in coefs.

x is the independent variable, in this case, the pixel number.

y is the value to be calculated

RETURN

None

Example

```
#define DLLIMPORT extern "C" __declspec(dllimport)
#define FUNCTIONFLAGS __stdcall
DLLIMPORT int FUNCTIONFLAGS cviTestCard(short nDeviceNumber, short
nBoardType, short nGain);
DLLIMPORT int FUNCTIONFLAGS cviSetIntEx(int const nBase, int const nPixelNumber,
long const lTime, int nDeviceNumber, int nChannel, int nGain);
DLLIMPORT int FUNCTIONFLAGS cviSetInt(int const nTriggerMode, long const lTime);
DLLIMPORT int FUNCTIONFLAGS cviTriggerReadEx(int const nBase, long const lTime,
int const nUnitType, USHORT* pArray);
DLLIMPORT int FUNCTIONFLAGS cviCloseEx();
DLLIMPORT void FUNCTIONFLAGS cviPolyCalc(double *coefs, int order, int x, double
*y);
DLLIMPORT int FUNCTIONFLAGS cviPolyFit(double *x, double *y, int numPts, double
*coefs, int order);
```

```

main() {

USHORT *PixArray; //for pixel data
long lIntTime=35; //the least integration time is 11ms for 6023, and 35ms for 1200
int nPixelNo = 2086; //for Sony CCD


int nRwturnValue = cviTestCard(1, 1, 2); //DeviceNumber = 1; Board = 6023; Gain = 2;
if(nRwturnValue<0) //fail to test the board
{
    AfxMessageBox("The NI board does not work correctly.");
    return;
}

cviSetIntEx(0x300, 2086, lIntTime, 1,0, 2); // Initialize the board

cviSetInt(10, lIntTime); // Freerun trigger mode and set the integration time

PixArray = new USHORT[nPixelNo]; //Array for pixel data

nRwturnValue = cviTriggerReadEx(0x300, lIntTime, 0, PixArray);
if(nRwturnValue!=nPixelNo)
{
    AfxMessageBox("Data Aquisition does not work correctly.");
    delete[] PixArray;
    return;
}


//Calibration for the wavelength
double fWaveLength[14];
double fPixelNo[14];
int CalibrationNo=14;
double fCoefficient[5];
//Reandom calibration data
fWaveLength[0] = 296.7 ; fPixelNo[0] = 34;
fWaveLength[1] = 302.1; fPixelNo[1] = 81;
fWaveLength[2] = 313.2; fPixelNo[2] = 181;
fWaveLength[3] = 365.0; fPixelNo[3] = 226;
fWaveLength[4] = 435.8; fPixelNo[4] = 413;
fWaveLength[5] = 546.1; fPixelNo[5] = 697;
fWaveLength[6] = 577.0; fPixelNo[6] = 775;
fWaveLength[7] = 579.1; fPixelNo[7] = 781;
fWaveLength[8] = 696.5; fPixelNo[8] = 1073;
fWaveLength[9] = 763.5; fPixelNo[9] = 1237;
fWaveLength[10] = 772.4; fPixelNo[10] = 1259;
fWaveLength[11] = 811.5; fPixelNo[11] = 1354;
fWaveLength[12] = 912.3; fPixelNo[12] = 1595;
fWaveLength[13] = 1014.0; fPixelNo[13] = 1835;

```



```

cviPolyFit(fPixelNo, fWaveLength, CalibrationNo,fCoefficient, 3);

HFILE file = _lcreat("SMSDK_test.txt",0);
char strTemp[80];
sprintf(strTemp," Pixel      Wavelength(nm)      Intensity \r\n");
_lwrite(file,(LPSTR)strTemp,strlen(strTemp));
for(int i=32;i<2080;i++)
{
    double fWL;
    cviPolyCalc(fCoefficient,3,i-31 ,&fWL);
    sprintf(strTemp," %5d      %5.2f      %5d\r\n",i-31,fWL,(int)PixArray[i]);
    _lwrite(file,(LPSTR)strTemp,strlen(strTemp));
}
_lclose(file);

delete[] PixArray;

cviCloseEx(); //Close the board

}

```