EasyNirLib二次开发用户手册

编写人员：沈玉杰

编写时间：2023年9月24日

手册版本：UG-ENL-V1.2

技术支持：深圳谱研互联科技有限公司

导读：

本文适用于谱研互联近红外光谱仪和模组适配的easynirlib二次开发包（sdk）的1.2版本通用版。sdk库的函数相对比较多，但在实际使用时，大多数都用不到。本文介绍几个常用api接口的逻辑调用流程，用户依此即可获取完整的光谱数据。

本文示例代码均使用C语言描述，主要用于展示调用流程和方法，其他编程语言可根据所使用的集成开发环境（IDE）适当修改。

文中为了便于较为独立完整地描述函数接口，大多数采用的局部变量定义，如数据长度、波长指针、光谱强度指针等，用户在实际开发软件时，应当用全局变量做定义。

在描述函数接口的参数时，[in]表示该参数是外部输入，[out]表示向外输出，out主要用于指针参数。

一、SDK概述

Easynirlib是基于C语言封装的api接口库，可以较为方便地被其他语言调用，如C++、C#、Labview等，其他语言只要能直接调用C语言动态库的均可适用。支持Windows、MAC和Linux等主流操作系统。

Easynirlib虽然有很多API接口，但有些接口参数很多，理解起来也比较困难，而且在实际使用中也不是必需的，所以本文仅列举了操作光谱仪必需的和使用率较高的函数接口，已经可以满足用户实际使用。

用户如需要查询或设置配置信息，可以在电脑的ISC软件进行设置，实际使用时不需要再修改。不建议在二次开发时频繁对光谱仪进行配置信息设置。

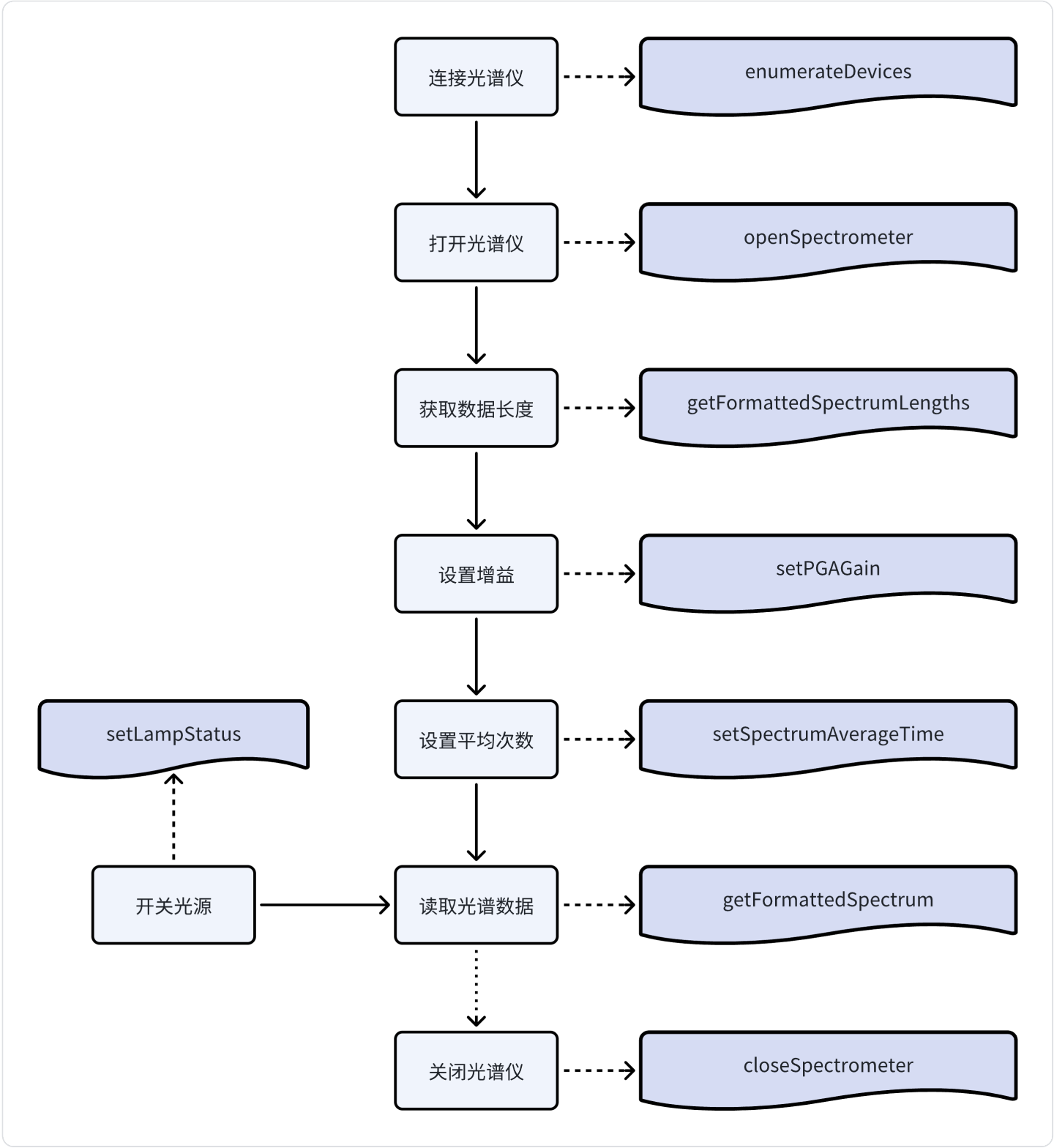
本文介绍的函数接口，适用于适用的型号模组和光谱仪如下：

型号	描述
NIR-M-R2	反射型，波长范围900-1700nm，内置两只卤素灯，支持USB、UART通信

NIR-R210	反射型，波长范围900-1700nm，内置两只卤素灯、锂电池，带壳体，支持USB、UART、蓝牙通信
NIR-M-R11	反射型，波长范围1350-2150nm，内置四只卤素灯，支持USB、UART通信，裸机
NIR-R310L	反射型，波长范围1350-2150nm，内置四只卤素灯，内置锂电池、支持USB、蓝牙、UART通信，带壳体
NIR-M-F1	光纤型，波长范围900-1700nm，SMA905接口，支持USB、UART通信
NIR-F210	光纤型，波长范围900-1700nm，SMA905接口，支持USB、UART通信，带壳体
NIR-M-F11	光纤型，波长范围1350-2150nm，SMA905接口，SMA905接口，支持USB、UART通信
NIR-F310	光纤型，波长范围1350-2150nm，SMA905接口，SMA905接口，支持USB、UART通信，带壳体
NIR-M-T1	透射型，波长范围900-1700nm，光源、比色皿支架、光谱仪一体集成，支持USB、UART通信
NIR-T210	透射型，波长范围900-1700nm，光源、比色皿支架、光谱仪一体集成，支持USB、UART通信，带壳体
NIR-M-T11	透射型，波长范围1350-2150nm，光源、比色皿支架、光谱仪一体集成，支持USB、UART通信
NIR-T310	透射型，波长范围1350-2150nm，光源、比色皿支架、光谱仪一体集成，支持USB、UART通信，带壳体

二、函数调用流程图

二次开发时按照图中流程操作，列出的8个函数接口即可满足实际需求。



以上函数接口并不是严格的顺序关系。其中连接光谱仪、打开光谱仪、获取数据长度、设置增益、设置平均次数都属于读取光谱数据之前的准备工作，可以在初始化时完成。读取光谱数据实际上包括了读取波长和对应的强度值。开关光源可以读取光谱数据前处理，对于光纤型光谱仪，可以先设置为打开光源状态，以缩短采集时间。关闭光谱仪可以在软件关闭时调用。设置增益、设置平均次数、开关光源均可在过程中设置，下次读光谱数据时生效。

三、重点接口函数说明

3.1 连接光谱仪

3.1.1 函数说明

enumerateDevices是连接设备的函数接口，函数内部完成了光谱仪的必要初始化配置，是必需函数，不可以跳过该步骤而直接打开光谱仪。

函数说明如下：

```
int enumerateDevices(int* num);
```

函数描述：获取连接的光谱仪设备数量

参数说明：

Param[out]：int* num，设备数量指针

返回值：

Return ⩾ 0: 连接成功

Return < 0：连接失败

3.1.2 代码示例

```
void connectDevice()
{
    int num = 0;    //获取设备连接数
    int res = enumerateDevices(&num);    //接口返回值

    if(res<0)
        printf("Connect device failed: %d\n", res);
    else
        printf("Connect device succeeded: %d\n", num);
}
```

连接一台光谱仪成功时的控制台打印值如下：

Connect device succeeded: 1

本示例代码中，定义一个整型变量num并初始化赋值分配内存后，将变量地址传参给enumerateDevices函数，若连接成功，函数内部把设备数量赋值到num变量所在的内存地址，再打印出的num值，即为连接设备数量。

如果没有连接光谱仪，连接失败，控制台打印值为负值：

Connect device failed: -1

3.2 打开和关闭光谱仪

3.2.1 函数说明

openSpectrometer是打开设备的函数接口，是必需函数，打开设备后才可以执行其他设置或读取光谱仪的操作。

closeSpectrometer是关闭设备的函数接口，通常在软件关闭时执行，也可以在打开设备之前先执行一次关闭设备函数，再打开设备，以避免两次打开出现冲突。

```
1 int openSpectrometer(int index);
```

函数描述：打开设备

参数说明：

Param[in]：int index，设备索引，0表示第1台设备, 1表示第2台设备，以此类推

返回值：

Return ⩾ 0: 打开成功

Return < 0：打开失败

```
1 void closeSpectrometer();
```

函数描述：关闭设备

参数说明：无

返回值： 无

3.1.2 代码示例

```
1 void openDevice()
2 {
3     int res;    //接口返回值
4 
5     closeSpectrometer();    //先关闭光谱仪
6 
7     res = openSpectrometer(DEVICE_INDEX);     //打开光谱仪
```

```
    if(res<0)
        printf("Open device failed: %d\n", res);
    else
        printf("Open device succeeded: %d\n", res);
}
```

仅连接一台光谱仪时，控制台打印值如下：

Open device succeeded: 0

本示例代码中，先关闭一次光谱仪设备，以避免重复打开。DEVICE_INDEX是已连接光谱仪的索引号，常用宏定义。若仅连接一台光谱仪，DEVICE_INDEX即为0，此时也可以直接用0作为openSpectrometer的参数。

```
res = openSpectrometer(0);    //仅连接一台光谱仪时，设备索引为0
```

3.3 获取数据长度

3.3.1 函数说明

getFormattedSpectrumLengths是获取数据长度的函数接口，是必需函数，在申请用于存储波长和光谱强度所用的内存时，需要用到数据长度。该步骤非常重要，用户需要理解示例代码，特别是存储波长和光谱数据的指针和内存申请部分。

```
int getFormattedSpectrumLengths(int* fileSize);
```

函数描述：获取数据长度

参数说明：

Param[out]：int* fileSize，数据长度指针

返回值：

Return ⩾ 0: 获取成功

Return < 0：获取失败

3.3.2 代码示例

```
void getSpectrumLength()
{
    int     fileSize = 0;       //数据长度
```

```
    double *pWavelengths;       //用于存储波长
    int    *pIntensities;       //用于存储光谱强度

    int res = getFormattedSpectrumLengths(&fileSize); //获取光谱数据长度
    if(res<0)
        printf("Get fileSize failed: %d\n", res);
    else
        printf("Get fileSize succeeded: %d\n", fileSize);

    pWavelengths = (double*)malloc(fileSize*sizeof(double));    //申请波长内存空间
    pIntensities = (int*)malloc(fileSize*sizeof(int));          //申请光谱强度内存
}
```

仅连接一台光谱仪时，控制台打印值如下：

Get fileSize succeeded: 228

本示例代码中，定义了数据长度fileSize ，同时又定义了分别用于指向波长和光谱数据的指针pWavelengths和pIntensities。在获取了数据长度后，采用malloc函数分别给波长和光谱数据分配了对应长度的内存空间，并分别用pWavelengths和pIntensities指向了申请的空间。

示例代码为了方便理解，把fileSize、pWavelengths、pIntensities都放在函数中定义，但在实际使用时，这几个变量被其他函数频繁引用，因此都是放在全局变量中定义。

对于刚入坑的学生党，虽然可能学过c语言，但对动态申请内存可能并不熟悉，这里就对malloc稍作讲解。

malloc的参数是以字节为单位，我们已知波长是double型，光谱强度是int型，也知道了数据长度，因此先用sizeof(double)和sizeof(int)来获取当前系统中的double和int对应的字节数，再乘以fileSize，即fileSize*sizeof(double)，即可得到我们需要的总字节数，作为参数传给malloc，就可以申请到对应的内存空间。

由于我们使用了指针，数据会使用指针索引来表示，所以把申请到的总内存空间按照double和int型，分别分成小单元，即(double*)和(int*)，这样就可以直接使用指针索引了，如pWavelengths[5]就表示第6个double型的波长值。

3.4 设置增益

3.4.1 函数说明

setPGAGain是设置增益倍率的函数接口，是可选函数，推荐在初始化的时候调用。光谱仪默认的增益倍率是1，读取的光谱强度值偏小，在实际使用时常用的是16、32或64，可根据需求进行设置。

由于光谱仪内部ADC是24位，满量程在1600多万counts值，设备内部电路通过增益对ADC值做了管理，用户只需要用电脑端ISC做评估，选择合适的增益即可，最大值推荐不要超过600万counts，否

则容易引起adc error错误，导致数据失真。

增益分固定和自适应两种模式，对于二次开发用户，推荐使用固定模式，否则容易在弱光时设备自适应高倍增益导致数据出现管理混乱的现象。在用ISC软件做参比板评估时，可以用自适应来选取合适增益，选取好后在做二次开发时推荐直接设成固定模式写入。

```
int setPGAGain(int isFixed, unsigned char gainVal);
```

函数描述：设置平均次数

参数说明：

Param[in]：int isFixed，1:固定 0:自适应

Param[in]：unsigned char gainVal，增益倍率，可设置1、2、4、8、16、32或64

返回值：

Return ⩾ 0: 获取成功

Return < 0：获取失败

3.4.2 代码示例

```
void setPgaGain()
{
    unsigned char pga = 32;        //增益倍率为32

    int res = setPGAGain(1, pga); //设置成固定模式
    if(res<0)
        printf("Set PGA failed: %d\n", res);
    else
        printf("Set PGA succeeded: %d\n", res);
}
```

仅连接一台光谱仪时，控制台打印值如下：

Set PGA succeeded: 0

本示例代码中，定义了增益倍率pga并赋值32，设置成固定模式后调用setPGAGain函数，再打印设置增益函数接口的返回值。

3.5 设置平均次数

3.5.1 函数说明

setSpectrumAverageTime是设置平均次数的函数接口，是可选函数，默认平均次数是6。平均次数是正整数，最小为1。顾名思义，平均次数是光谱仪内部多次测量求平均值时所设置的次数。平均次数越高，获取的光谱数据重复性越好，但用时也越长。推荐平均次数不要超过10，否则测量时间会比较长。

平均次数也可以在电脑端ISC软件的配置信息中设置，且会被写到光谱仪内部存储器中。若实际使用时平均次数为固定值，则二次开发可以不再设置。

```
int setSpectrumAverageTime(unsigned short time);
```

函数描述：设置测量平均次数

参数说明：

Param[in]：unsigned short time，平均次数，最小值为1，新光谱仪的默认值为6

返回值：

Return ⩾ 0: 获取成功

Return < 0：获取失败

3.5.2 代码示例

```
void setAvgTimes()
{
    int avg = 6;    //定义平均次数并赋值6
    int res = setSpectrumAverageTime(avg);  //设置平均次数
    if(res<0)
        printf("Set avg failed: %d\n", res);
    else
        printf("Set avg succeeded: %d\n", res);
}
```

仅连接一台光谱仪时，控制台打印值如下：

Set avg succeeded: 0

本示例代码中，定义了平均次数avg并赋值6，调用setSpectrumAverageTime，再打印设置平均次数函数接口的返回值。

3.6 获取光谱数据

3.6.1 函数说明

getFormattedSpectrum是获取光谱数据的函数接口，光谱数据包括波长值和相对光谱强度值，是必需函数。在调用该函数前，需要执行**2.3节**的获取数据长度函数示例代码，确保波长和光谱强度指针已经分配了内存空间。

```
int getFormattedSpectrum(double *pWavelength, int *pData, int *dataSize);
```

函数描述：获取光谱数据

参数说明：

Param[out]：double *pWavelength，波长值输出到该指针指向的内存空间予以存储

Param[out]：int *pData，相对光谱强度值输出到该指针指向的内存空间予以存储

Param[out]：int *dataSize，数据长度输出到该指针指向的变量

返回值：

Return ⩾ 0: 获取成功

Return < 0：获取失败

💡 注意，这里的dataSize与3.2节的fileSize有些区别，dataSize是当前读取光谱数据时的输出数据长度，fileSize是申请内存空间的数据长度，fileSize⩾dataSize，可以确保内存空间足够用。正常来说二者是相同的，但封装api接口时需要考虑到光谱仪配置参数在二次开发时被动态修改的状况，比如900-1700nm波长范围的光谱仪默认fileSize是228，但在过程中被修改为900-1600nm，此时dataSize<fileSize，即不再是228个数据。

3.6.2 代码示例

```
void getSpectrumData()
{
    int dataSize = 0;    //获取输出数据数量

    int res = getFormattedSpectrum(pWavelengths,pIntensities,&dataSize);    //获

    if(res<0)
        printf("Get spectrum data failed: %d\n", res);
    else
        printf("Get spectrum data succeeded: %d\n", res);

    //打印波长值和对应的光谱强度值
    for (int i=0;i<dataSize;i++)
        printf("Wavelength: %f, Intensity: %d\n", pWavelengths[i], pIntensities[
```

```
15 }
```

仅连接一台光谱仪时，控制台打印值如下：

Get spectrum data succeeded: 0

Wavelength: 900.993140, Intensity: 4435

Wavelength: 904.952494, Intensity: 5309

Wavelength: 908.907419, Intensity: 6267

……

Wavelength: 1695.003875, Intensity: 13358

Wavelength: 1697.944533, Intensity: 11394

Wavelength: 1700.880762, Intensity: 10001

本示例代码中，定义了输出数据数量dataSize并赋值为0，调用getFormattedSpectrum函数接口后，获取的波长和光谱强度数据分别存在pWavelengths和pIntensities指向的内存空间，打印获取光谱数据函数接口的返回值，再通过pWavelengths和pIntensities指针打印出波长值和对应的光谱强度值。

💡 该示例代码中的pWavelengths和pIntensities应该是在全局变量中定义，且完成了2.3节描述的内存空间申请。2.3节示例中局部定义pWavelengths和pIntensities，只是为了展示功能，实际上也应该使用全局变量，保证不同函数操作的是相同pWavelengths和pIntensities。

3.7 设置内置光源开关状态

3.7.1 函数说明

setLampStatus是设置内置光源开关状态的接口函数，是可选函数。对于内置光源的光谱仪或模组，一般设置为关闭状态；对于光纤型光谱仪或模组，一般设置为开启状态。默认为关闭状态。

当处于关闭状态时，光谱仪工作时会有开灯时间延时，扫描速度较慢；当处于开启状态时，光谱仪工作时无开灯时间延时，扫描速度较快。光纤型光谱仪没有内置光源，因此常设置为开启状态以提升扫描速度。而内置光源型光谱仪较少用于在线测量，对速度要求较低，其测量时会短暂自动开灯，因此常设置为默认关闭状态。

```
1 int setLampStatus(int status);
```

函数描述：设置内置光源开关状态

参数说明：

Param[in]：int status，开关状态， 0:关闭 ，1:开启

返回值：

Return ⩾ 0: 获取成功

Return < 0：获取失败

3.7.2 代码示例

```
void setLampStatus()
{
    int res = setLampStatus(0); //设置内置光源为关闭状态

    if(res<0)
        printf("Set lamp failed: %d\n", res);
    else
        printf("Set lamp succeeded: %d\n", res);
}
```

仅连接一台光谱仪时，控制台打印值如下：

Set lamp succeeded: 0

本示例代码中，调用setLampStatus，打印函数接口返回值。

四、完成代码示例

4.1 完整C语言代码

完整示例代码用C语言编写，仅用于展示调用流程，其他编程语言和IDE可按在界面上配合按钮调用。各调用函数除2.3节把局部变量改为全局变量外，其他都使用了第3章示例代码。

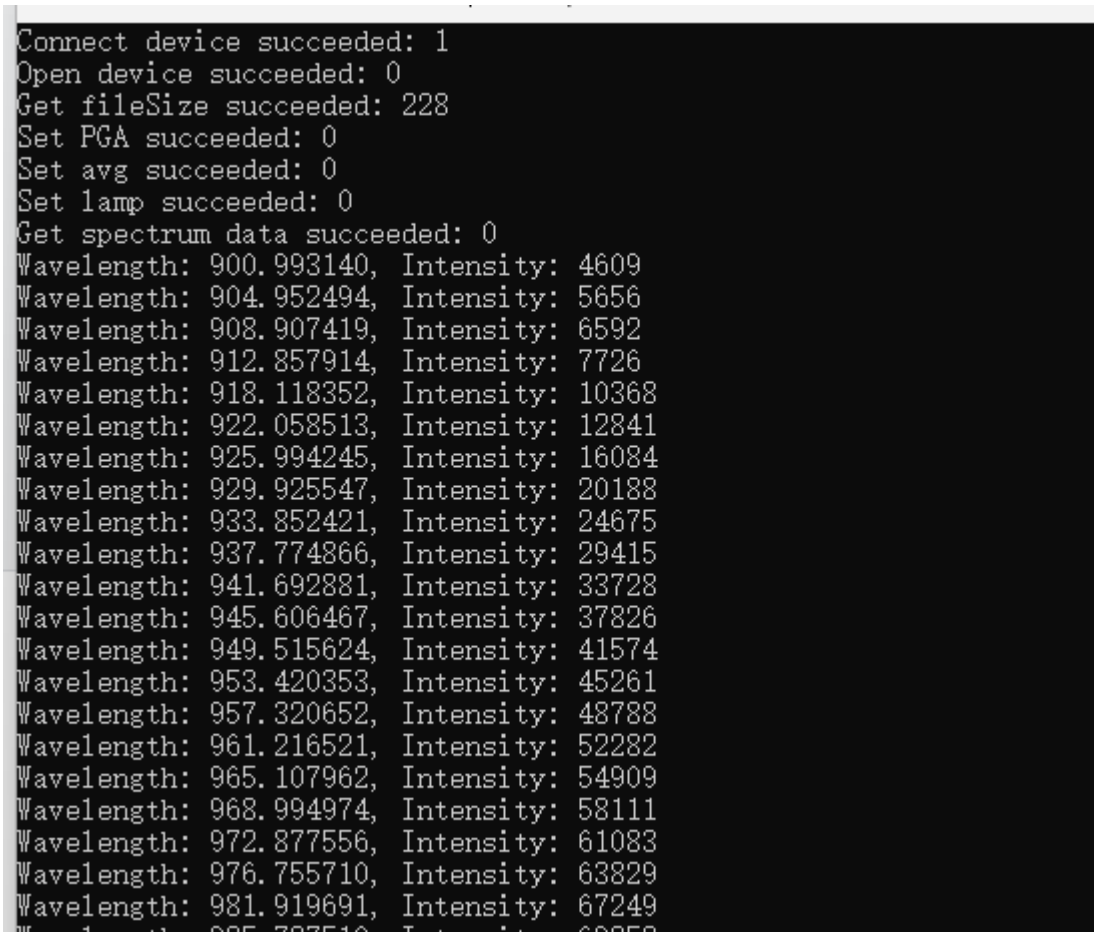
```
#include "easynirwrapper.h"

#define DEVICE_INDEX 0

//定义全局变量
int     fileSize = 0;       //数据长度
double *pWavelengths;       //用于存储波长
int    *pIntensities;       //用于存储光谱强度

//连接光谱仪设备
void connectDevice()
{
```

```
    int num = 0;    //获取设备连接数
    int res = enumerateDevices(&num);   //接口返回值

    if(res<0)
        printf("Connect device failed: %d\n", res);
    else
        printf("Connect device succeeded: %d\n", num);
}

//打开光谱仪设备
void openDevice()
{
    int res;    //接口返回值

    closeSpectrometer();    //先关闭光谱仪

    res = openSpectrometer(DEVICE_INDEX);     //打开光谱仪
    if(res<0)
        printf("Open device failed: %d\n", res);
    else
        printf("Open device succeeded: %d\n", res);
}

//获取数据长度
void getSpectrumLength()
{
    int res = getFormattedSpectrumLengths(&fileSize); //获取光谱数据长度
    if(res<0)
        printf("Get fileSize failed: %d\n", res);
    else
        printf("Get fileSize succeeded: %d\n", fileSize);

    pWavelengths = (double*)malloc(fileSize*sizeof(double));    //申请波长内存空间
    pIntensities = (int*)malloc(fileSize*sizeof(int));          //申请光谱强度内存
}

//设置增益
void setPgaGain()
{
    unsigned char pga = 32;       //增益倍率为32

    int res = setPGAGain(1, pga); //设置成固定模式
    if(res<0)
        printf("Set PGA failed: %d\n", res);
    else
        printf("Set PGA succeeded: %d\n", res);
}
```

```
60
61 //设置平均次数
62 void setAvgTimes()
63 {
64     int avg = 6;    //定义平均次数并赋值6
65     int res = setSpectrumAverageTime(avg);  //设置平均次数
66     if(res<0)
67         printf("Set avg failed: %d\n", res);
68     else
69         printf("Set avg succeeded: %d\n", res);
70 }
71
72 //获取光谱数据
73 void getSpectrumData()
74 {
75     int dataSize = 0;    //获取输出数据数量
76
77     int res = getFormattedSpectrum(pWavelengths,pIntensities,&dataSize);    //获
78
79     if(res<0)
80         printf("Get spectrum data failed: %d\n", res);
81     else
82         printf("Get spectrum data succeeded: %d\n", res);
83
84     //打印波长值和对应的光谱强度值
85     for (int i=0;i<dataSize;i++)
86         printf("Wavelength: %f, Intensity: %d\n", pWavelengths[i], pIntensities[
87 }
88
89 //设置内置光源开关状态
90 void setLampStatus()
91 {
92     int res = setLampStatus(0); //设置内置光源为关闭状态
93
94     if(res<0)
95         printf("Set lamp failed: %d\n", res);
96     else
97         printf("Set lamp succeeded: %d\n", res);
98 }
99
100 int main(int argc, char *argv[])
101 {
102     QCoreApplication a(argc, argv);
103
104     connectDevice();
105     openDevice();
106     getSpectrumLength();
```

```
    setPgaGain();
    setAvgTimes();
    setLampStatus();
    getSpectrumData();

    return a.exec();
}
```

4.2 控制台结果



控制台结果总结如下：

Connect device succeeded: 1，打印值是已连接的光谱数量，即1台

Open device succeeded: 0，返回值为0，打开设备成功

Set PGA succeeded: 0，返回值为0，设置增益倍率成功

Set avg succeeded: 0，返回值为0，设置平均次数成功

Set lamp succeeded: 0，返回值为0，设置内置光源开关状态成功

Get spectrum data succeeded: 0，获取光谱数据成功

Wavelength: 900.993140, Intensity: 4609

Wavelength: 904.952494, Intensity: 5656

Wavelength: 908.907419, Intensity: 6592

……

打印波长和对应相对光谱强度