cmsplibExCC接口使用说明书

目录

一、常规通信接口

1. CMSPEX_Init

【函数原型】 int CMSPEX_Init(CMSPEX_HANDLE *hd, uint32_t uiCommMode)

【功能】 在调用接口前，对接口进行初始化

【参数说明】

hd：句柄；

uiCommMode：通信模式，0：USB 通信，1：串口通信；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
// 假设通信方式为 USB
CMSPEX_HANDLE hd;
int retValue = CMSPEX_Init(&hd, 0);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

【注意】

调用接口其他函数之前，必须先进行初始化，成功后才可调用其他函数。

2. CMSPEX_DeInit

【函数原型】 int CMSPEX_DeInit(CMSPEX_HANDLE hd)

【功能】 释放接口内存

【参数说明】

hd：句柄。

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd;//从 CMSPEX_Init()传出的有效句柄
int retValue = CMSPEX_DeInit(hd);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

【注意】

结束对通信接口的使用后，必须调用此函数释放接口内存。

3. CMSPEX_Open

【函数原型】 int CMSPEX_Open(CMSPEX_HANDLE hd, const char *dev)

【功能】 打开光谱仪设备

【参数说明】

hd：句柄；

dev：usb 连接则表示设备序列号，串口连接则表示串口号；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd;//从 CMSPEX_Init()传出的有效句柄
// “ABOOPXDF”为设备的序列号
int retValue = CMSPEX_Open(hd, "ABOOPXDF");
if (retValue == 0) {
    // success
}
else {
```

```
    // failure
}
```

【注意】

在对光谱仪进行操作之前，需要先打开设备，成功后才能进行其他操作。

4. CMSPEX_OpenByIndex

【函数原型】 int CMSPEX_OpenByIndex(CMSPEX_HANDLE hd, uint32_t devNum)

【功能】 打开光谱仪设备

【参数说明】

hd：句柄；

devNum：usb 连接则表示设备索引，串口连接则表示串口号；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd; //从 CMSPEX_Init()传出的有效句柄
// 0 为设备索引
int retValue = CMSPEX_OpenByIndex(hd, 0);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

【注意】

在对光谱仪进行操作之前，需要先打开设备，成功后才能进行其他操作。

5. CMSPEX_OpenByInfo

【函数原型】 int CMSPEX_OpenByInfo(CMSPEX_HANDLE hd, const char *dev, uint32_t baudrate, int32_t databit, uint32_t stopbit, uint32_t parity)

【功能】 打开光谱仪设备

【参数说明】

hd：句柄；

dev：usb 连接则表示设备序列号，串口连接则表示串口号；

baudrate：波特率；

databit：数据位；

stopbit：停止位，0：1bit，1：1.5bit，2：2bit；

parity：校验位，0：没有校验位，1：奇校验，2：偶校验；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd;//从 CMSPEX_Init()传出的有效句柄
// 假设串口号为“COM5”，波特率为 115200，数据位为 8，停止位 2bit,没有校验位
int retValue = CMSPEX_OpenByInfo(hd, "COM5", 115200, 8, 2, 0);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

【注意】

在对光谱仪进行操作之前，需要先打开设备，成功后才能进行其他操作。

6. CMSPEX_OpenByIndexByInfo

【函数原型】 int CMSPEX_OpenByIndexByInfo(CMSPEX_HANDLE hd, uint32_t devNum, uint32_t baudrate, int32_t databit, uint32_t stopbit, uint32_t parity)

【功能】 打开光谱仪设备

【参数说明】

hd：句柄；

devNum：usb 连接则表示设备索引，串口连接则表示串口号；

baudrate：波特率；

databit：数据位；

stopbit：停止位，0：1bit，1：1.5bit，2：2bit；

parity：校验位，0：没有校验位，1：奇校验，2：偶校验；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd; //从 CMSPEX_Init()传出的有效句柄
// 假设串口号为“COM5”，波特率为 115200，数据位为 8，停止位 2bit,没有校验位
int retValue = CMSPEX_OpenByIndexByInfo(hd, 5, 115200, 8, 2, 0);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

【注意】

在对光谱仪进行操作之前，需要先打开设备，成功后才能进行其他操作。

7. CMSPEX_Close

【函数原型】 int CMSPEX_Close(CMSPEX_HANDLE hd)

【功能】 关闭光谱仪设备

【参数说明】

hd：句柄；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd;// 从 CMSPEX_Init()传出的有效句柄
int retValue = CMSPEX_Close(hd);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

【注意】

结束对光谱仪的操作后，需要关闭设备。

8. CMSPEX_GetDevList

【函数原型】 int CMSPEX_GetDevList(CMSPEX_HANDLE hd, DEVLSTUSB* devlst, uint32_t &devCnts, uint32_t uiGetType)

【功能】 获取 USB 设备列表

【参数说明】

hd：句柄；

devlst：设备列表结构体；

devCnts：设备个数；

uiGetType：获取设备列表时的类型，0：获取设备列表信息，1：只获取设备个数，默认 0；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd;// 从 CMSPEX_Init()传出的有效句柄
uint32_t uiDevCnts;
int retValue = CMSPEX_GetDevList(hd, NULL, uiDevCnts, 1);
```

```
if (rslt != 0)
{
    // failure
    return ;
}
DEVLSTUSB *devlist = new DEVLSTUSB[uiDevCnts];
retValue = CMSPEX_GetDevList(hd, devlist, uiDevCnts, 0);
if (retValue ! = 0) {
    // failure
    return;
}
// success
// devlist 为获取到的设备列表信息，其中包括设备索引和设备序列号，可用于打开光谱仪
// devlist[0].devnums：第一个设备的索引
// devlist[0].serialnumber：第一个设备的序列号
```

【注意】

USB 连接时，可在打开光谱仪之前获取设备列表，根据设备列表的内容选择需要打开的设备。

9. CMSPEX_SetIntegrationTime

【函数原型】 int CMSPEX_SetIntegrationTime(CMSPEX_HANDLE hd, uint32_t uiInterTime)

【功能】 设置积分时间，单位 us，默认 60

【参数说明】

hd：句柄；

uiInterTime：积分时间，单位 us；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
// 假设设置积分时间为100us
CMSPEX_HANDLE hd;// 从CMSPEX_Init()传出的有效句柄
int retValue = CMSPEX_SetIntegrationTime(hd, 100);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

【注意】

积分时间范围：60~100 000 000（us）。

10. CMSPEX_GetIntegrationTime

【函数原型】 int CMSPEX_GetIntegrationTime(CMSPEX_HANDLE hd, uint32_t &uiInterTime)

【功能】 获取积分时间，单位us

【参数说明】

hd：句柄；

uiInterTime：积分时间，单位us；

【返回值说明】

成功返回0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd;// 从CMSPEX_Init()传出的有效句柄
uint32_t uiInterTime;
int retValue = CMSPEX_GetIntegrationTime(hd, uiInterTime);
if (retValue == 0) {
    // success -- uiInterTime为获取到的积分时间
}
```

```
else {
    // failure
}
```

【注意】

积分时间范围：60~100 000 000（us）。

11. CMSPEX_SetLampPulse

【函数原型】 int CMSPEX_SetLampPulse(CMSPEX_HANDLE hd, uint32_t uiPulseTime, uint32_t uiHighLevel)

【功能】 设置氙灯脉冲，单位 10ns，默认低电平 1500000，高电平 10000

【参数说明】

hd：句柄；

uiPulseTime：脉冲触发周期；

uiHighLevel：脉冲高电平；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
// 假设设置氙灯脉冲周期为 16ms，高电平 0.1ms
CMSPEX_HANDLE hd;// 从 CMSPEX_Init()传出的有效句柄
int retValue = CMSPEX_SetLampPulse(hd, 1600000, 10000);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

【注意】

建议氙灯脉冲周期>15ms，高电平>0.1ms。

12. CMSPEX_GetLampPulse

【函数原型】 int CMSPEX_GetLampPulse(CMSPEX_HANDLE hd, uint32_t &uiPulseTime, uint32_t &uiHighLevel)

【功能】 获取氙灯脉冲

【参数说明】

hd：句柄；

uiPulseTime：脉冲触发周期；

uiHighLevel：脉冲高电平；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd; // 从 CMSPEX_Init()传出的有效句柄
uint32_t uiPulseTime, uiHighLevel;
int retValue = CMSPEX_GetLampPulse(hd, uiPulseTime, uiHighLevel);
if (retValue == 0) {
    // success - uiPulseTime，uiHighLevel 为获取到的脉冲触发周期和高电平
}
else {
    // failure
}
```

【注意】

建议氙灯脉冲周期>15ms，高电平>0.1ms。

13. CMSPEX_LampEnable

【函数原型】 int CMSPEX_LampEnable(CMSPEX_HANDLE hd, uint32_t uiOn)

【功能】 设置氙灯开关，默认关闭

【参数说明】

hd：句柄；

uiOn：氙灯开关，0：关闭，1：连续脉冲，2：单次脉冲；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
// 假设设置氙灯开关为连续脉冲模式
CMSPEX_HANDLE hd; // 从 CMSPEX_Init()传出的有效句柄
int retValue = CMSPEX_LampEnable(hd, 1);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

【注意】

14. CMSPEX_LampStatus

【函数原型】 int CMSPEX_LampStatus(CMSPEX_HANDLE hd, uint32_t &uiOn)

【功能】 获取氙灯开关状态

【参数说明】

Hd：句柄；

uiOn：氙灯开关，0：关闭，1：连续脉冲，2：单次脉冲；

【返回值说明】

成功返回 0，失败返回负数。

【示例】

```
CMSPEX_HANDLE hd; // 从 CMSPEX_Init()传出的有效句柄
uint32_t uiOn;
int retValue = CMSPEX_LampStatus(hd, uiOn);
if (retValue == 0) {
    // success - uiOn 为获取到的氙灯开关状态
```

```
}
else {
    // failure
}
```

【注意】

15. CMSPEX_SetAccAvg

【函数原型】 int CMSPEX_SetAccAvg(CMSPEX_HANDLE hd, uint32_t uiAccTimes)

【功能】 设置底层平均次数，默认 20

【参数说明】

hd：句柄；

uiAccTimes：底层平均次数；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
// 假设设置平均次数为 1
CMSPEX_HANDLE hd;// 从 CMSPEX_Init()传出的有效句柄
int retValue = CMSPEX_SetAccAvg(hd, 1);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

【注意】

底层平均次数范围：1~255。

16. CMSPEX_GetAccAvg

【函数原型】 int CMSPEX_GetAccAvg(CMSPEX_HANDLE hd, uint32_t &uiAccTimes)

【功能】 获取底层平均次数

【参数说明】

hd：句柄；

uiAccTimes：底层平均次数；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd;// 从 CMSPEX_Init()传出的有效句柄
uint32_t uiAccTimes;
int retValue = CMSPEX_GetAccAvg(hd, uiAccTimes);
if (retValue == 0) {
    // success -- uiAccTimes 为获取到的底层平均次数
}
else {
    // failure
}
```

【注意】

底层平均次数范围：1~255。

17. CMSPEX_SetGPIOLevel

【函数原型】int CMSPEX_SetGPIOLevel(CMSPEX_HANDLE hd, unsigned char level)

【功能】 设置 GPIO 电平输出，默认 00h

【参数说明】

hd：句柄；

level： GPIO 电平输出；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
// 假设设置 GPIO 电平输出为第 1 个 GPIO 口高电平，其他 7 个低电平
```

```
CMSPEX_HANDLE hd;// 从CMSPEX_Init()传出的有效句柄
int retValue = CMSPEX_SetGPIOLevel(hd, 0x01);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

【注意】

取值范围为00(0x00)~255(0xFF),一共有8个GPIO口，从1~8分别对应二进制从右到左的8位，默认全部为0，为1则对应GPIO口为高电平。

18. CMSPEX_GetGPIOLevel

【函数原型】 int CMSPEX_GetGPIOLevel(CMSPEX_HANDLE hd, unsigned char &level)

【功能】 获取GPIO电平输出

【参数说明】

hd：句柄；

level：GPIO电平输出；

【返回值说明】

成功返回0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd;// 从CMSPEX_Init()传出的有效句柄
unsigned char level;
int retValue = CMSPEX_GetGPIOLevel(hd, level);
if (retValue == 0) {
    // success -- level为GPIO电平输出
}
else {
```

```
    // failure
}
```

【注意】

取值范围为00(0x00)~255(0xFF),一共有8个GPIO口，从1~8分别对应二进制从右到左的8位，默认全部为0，为1则对应GPIO口为高电平。

19. CMSPEX_Reset

【函数原型】 int CMSPEX_Reset(CMSPEX_HANDLE hd, uint32_t uiMode)

【功能】 复位

【参数说明】

hd：句柄；

uiMode：复位模式，1：打开复位；

【返回值说明】

成功返回0，失败返回负数；

【示例】

```
// 假设打开复位
CMSPEX_HANDLE hd;// 从CMSPEX_Init()传出的有效句柄
int retValue = CMSPEX_Reset(hd, 1);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

【注意】

复位后，光谱仪各参数不会恢复为默认值，断电后各参数会恢复默认值。

20. CMSPEX_GetResetStatus

【函数原型】 int CMSPEX_GetResetStatus(CMSPEX_HANDLE hd, uint32_t &uiMode)

【功能】 获取复位状态，默认为 0

【参数说明】

hd：句柄；

uiMode：复位状态，1：正在复位，0：空闲状态；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd;// 从 CMSPEX_Init()传出的有效句柄
uint32_t uiMode;
int retValue = CMSPEX_GetResetStatus(hd, uiMode);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

【注意】

复位后，光谱仪各参数不会恢复为默认值，断电后各参数会恢复默认值。

21. CMSPEX_GetWaveCalParams

【函数原型】 int CMSPEX_GetWaveCalParams(CMSPEX_HANDLE hd, double dWaveParams[4])

【功能】 获取波长标定参数

【参数说明】

hd：句柄；

dWaveParams：存放 4 个波长标定参数；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd;// 从CMSPEX_Init()传出的有效句柄
double dWaveParams[4];
memset(dWaveParams, 0, sizeof(double)*4);
int retValue = CMSPEX_GetWaveCalParams(hd, dWaveParams);
if (retValue == 0) {
    // success -- dWaveParams为波长标定参数
}
else {
    // failure
}
```

【注意】

22. CMSPEX_GetTemperature

【函数原型】 int CMSPEX_GetTemperature(CMSPEX_HANDLE hd, float &fTemper)

【功能】 获取温度

【参数说明】

hd：句柄；

fTemper：温度值；

【返回值说明】

成功返回0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd;// 从CMSPEX_Init()传出的有效句柄
float fTemper;
int retValue = CMSPEX_GetTemperature(hd, fTemper);
if (retValue == 0) {
    // success -- fTemper为获取的温度值
}
else {
    // failure
```

```
}
```

【注意】

23. CMSPEX_GetFwVer

【函数原型】 int CMSPEX_GetFwVer(CMSPEX_HANDLE hd, unsigned char *ver)

【功能】 获取固件版本号

【参数说明】

hd：句柄；

ver：光谱仪的固件版本（20 字节）；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd;// 从 CMSPEX_Init()传出的有效句柄
unsigned char ver[20] = { '0' };
int retValue = CMSPEX_GetFwVer(hd, ver);
if (retValue == 0) {
    // success -- ver 为获取到的固件版本号
}
else {
    // failure
}
```

【注意】

24. CMSPEX_SetBaudrate

【函数原型】 int CMSPEX_SetBaudrate(CMSPEX_HANDLE hd, uint32_t uiBaudrate)

【功能】 设置 usb 的波特率，默认为 100，表示每个字节传输时间为 1us

【参数说明】

hd：句柄；

uiBaudrate：波特率；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
// 假设设置 usb 波特率为 200，表示每个字节传输时间为 2us
CMSPEX_HANDLE hd; // 从 CMSPEX_Init()传出的有效句柄
int retValue = CMSPEX_SetBaudrate(hd, 200);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

【注意】

用户可设置 usb 波特率的范围为 100~100000，设置的值越大则传输速度越慢。

25. CMSPEX_GetBaudrate

【函数原型】 int CMSPEX_GetBaudrate(CMSPEX_HANDLE hd, uint32_t &uiBaudrate)

【功能】 查询 usb 的波特率

【参数说明】

hd：句柄；

uiBaudrate：波特率；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd; // 从 CMSPEX_Init()传出的有效句柄
uint32_t uiBaudrate;
int retValue = CMSPEX_GetBaudrate(hd, uiBaudrate);
if (retValue == 0) {
```

```
    // success - uiBaudrate 为查询到的波特率
}
else {
    // failure
}
```

【注意】

用户可设置 usb 波特率的范围为 100~100000，设置的值越大则传输速度越慢。

26. CMSPEX_GetSpecSerialNumber

【函数原型】 int CMSPEX_GetSpecSerialNumber(CMSPEX_HANDLE hd, unsigned char *serialNum)

【功能】 获取光谱仪序列号

【参数说明】

hd：句柄；

serialNum：光谱仪序列号（8 字节）；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd; //从 CMSPEX_Init()传出的有效句柄
unsigned char szSerialNum[8] = { '0' };
int retValue = CMSPEX_GetSpecSerialNumber(hd, szSerialNum);
if (retValue == 0) {
    // success -- szSerialNum 为获取到的光谱仪序列号
}
else {
    // failure
}
```

【注意】

27. CMSPEX_GetSpecType

【函数原型】 int CMSPEX_GetSpecType(CMSPEX_HANDLE hd, unsigned char *type)

【功能】 获取光谱仪型号

【参数说明】

hd：句柄；

type：光谱仪型号（16 字节）；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd; //从 SPLIBEX_Init()传出的有效句柄
unsigned char szType[16] = { '0' };
int retValue = CMSPEX_GetSpecType(hd, szType);
if (retValue == 0) {
    // success -- szType 为获取到的光谱仪型号
}
else {
    // failure
}
```

【注意】

28. CMSPEX_SetTrigGetSpecCtrl

【函数原型】 int CMSPEX_SetTrigGetSpecCtrl(CMSPEX_HANDLE hd, uint32_t uiAddr)

【功能】 设置外触发光谱获取控制，只有打开外触发并且已经触发后才可设置

【参数说明】

hd：句柄；

uiAddr：读光谱数据的地址，0~62；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
// 假设要读取第 1 帧的数据，首先需要设置外触发光谱获取控制
CMSPEX_HANDLE hd; //从 CMSPEX_Init()传出的有效句柄
int retValue = CMSPEX_SetTrigGetSpecCtrl(hd, 0);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

【注意】

29. CMSPEX_GetTrigGetSpecCtrl

【函数原型】 int CMSPEX_GetTrigGetSpecCtrl(CMSPEX_HANDLE hd, uint32_t &uiAddr, uint32_t &uiResult)

【功能】 查询外触发光谱获取控制结果

【参数说明】

hd：句柄；

uiAddr：读光谱数据的地址，0~62；

uiResult：执行结果，0：表示正在执行，1：表示执行成功；2：表示执行失败；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
// 假设要读取第 1 帧的数据，首先需要设置外触发光谱获取控制
CMSPEX_HANDLE hd; //从 CMSPEX_Init()传出的有效句柄
uint32_t uiAddr, uiResult;
int retValue = CMSPEX_SetTrigGetSpecCtrl(hd, 0);
if (retValue == 0) {
```

```
    // success
}
else {
    // failure
}
// 查询外触发光谱获取控制结果
retValue = CMSPEX_SetTrigGetSpecCtrl(hd, uiAddr, uiResult);
if (retValue == 0 && uiAddr == 0 && uiResult == 1) {
    // success -- 表示数据已经存放到光谱数据寄存器，可以获取了
}
else {
    // failure
}
```

【注意】

30. CMSPEX_GetUsbStatus

【函数原型】 int CMSPEX_GetUsbStatus(CMSPEX_HANDLE hd, uint32_t &uiMode)

【功能】 获取 USB 的连接状态

【参数说明】

hd：句柄；

uiMode：USB 连接状态，0：未连接，1：已连接；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd; // 从 CMSPEX_Init()传出的有效句柄
uint32_t uiMode;
int retValue = CMSPEX_GetUsbStatus(hd, uiMode);
if (retValue == 0) {
    // success -- uiMode 为 USB 的连接状态
```

```
}
else {
    // failure
}
```

【注意】

31. CMSPEX_SetSpTrigConfig

【函数原型】 int CMSPEX_SetSpTrigConfig(CMSPEX_HANDLE hd, uint32_t uiSwitch, uint32_t uiType, uint32_t uiNum)

【功能】 设置光谱仪外触发配置

【参数说明】

hd：句柄；

uiSwitch：外触发开关，0：关闭，1：开启；

uiType：开启外触发类型，1：上升沿触发；2：电平触发；

uiNum：光谱采集个数，表示一次触发获取的光谱数据数量，取值范围为 1～63，默认值为 1；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
// 假设打开外触发，类型为电平触发，每次触发采集数量为 10
CMSPEX_HANDLE hd;// 从 CMSPEX_Init()传出的有效句柄
int retValue = CMSPEX_SetSpTrigConfig(hd, 1,  2,  10);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

【注意】

32. CMSPEX_GetSpTrigConfig

【函数原型】 int CMSPEX_GetSpTrigConfig(CMSPEX_HANDLE hd, uint32_t &uiSwitch, uint32_t &uiType, uint32_t &uiNum)

【功能】 查询光谱仪外触发配置

【参数说明】

hd：句柄；

uiSwitch：外触发开关，0：关闭，1：开启；

uiType：开启外触发类型，1：上升沿触发；2：电平触发；

uiNum：光谱采集个数，表示一次触发获取的光谱数据数量，取值范围为 1～63，默认值为 1；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd; // 从 CMSPEX_Init()传出的有效句柄
uint32_t uiSwitch, uiType, uiNum;
int retValue = CMSPEX_GetSpTrigConfig(hd, uiSwitch, uiType, uiNum);
if (retValue == 0) {
    // success -- uiSwitch 为获取到的外触发开关；uiType 为获取到的外触发类型；uiNum 为获取到的光谱采集个数
}
else {
    // failure
}
```

【注意】

33. CMSPEX_GetSpecCollectionStatus

【函数原型】int CMSPEX_GetSpecCollectionStatus(CMSPEX_HANDLE hd, uint32_t &uiCollStatus, unsigned char *szDataStatus, uint32_t uiNum)

【功能】 查询光谱数据采集状态

【参数说明】

hd：句柄；

uiCollStatus：外触发数据更新状态，0：没有数据更新或者全部数据被取走，1：有数据更新；

szDataStatus：外触发光谱数据标志，与外触发配置中光谱数据对应，从第 0 个字节开始，每个字节数据对应一个光谱数据标志，为 1 表示当前光谱数据已存在，0 表示不存在；

uiNum：光谱采集个数，与光谱仪外触发配置的光谱采集个数一致；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
// 假设外触发获取光谱个数为 10
CMSPEX_HANDLE hd; // 从 CMSPEX_Init()传出的有效句柄
uint32_t uiCollStatus;
unsigned char szDataStatus[64] = { '0' };
int retValue = CMSPEX_GetSpecCollectionStatus(hd, uiCollStatus,
szDataStatus, 10);
if (retValue == 0) {
    // success -- uiCollStatus 为获取到的外触发数据更新状态；
    szDataStatus 为获取到的外触发光谱数据标志，例 szDataStatus[0]=1，
    表示第 1 帧数据已经更新完成，szDataStatus[0]=0,表示第 1 帧数据还未
    更新完成，以此类推
}
else {
    // failure
}
```

【注意】

34. CMSPEX_GetWavelength

【函数原型】int CMSPEX_GetWavelength(CMSPEX_HANDLE hd, unsigned char *buf, uint32_t addr, uint32_t len)

【功能】 获取波长数据

【参数说明】

hd：句柄；

buf：波长数据存储 buf；

addr：波长起始地址；

len：读取数据的字节长度（波长个数*4）；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
// 假设波长个数为 2048
CMSPEX_HANDLE hd;// 从 CMSPEX_Init()传出的有效句柄
uint32_t len;
len = 8192;   // (2048*4)
unsigned char buf[8192] = { '0' };
int retValue = CMSPEX_GetWavelength(hd, buf, 0, len);
if (retValue == 0) {
    // success
    // 进行数据转化，以低位在前，高位在后的方式，每 4 个字节为一个波长数据，以第一个点为例
    float data;
    memcpy(&data, buf, sizeof(char)*4);
    // 此时 data 为第一个波长数据，以此类推可得出全部波长数据
}
else {
    // failure
```

}

【注意】

波长数据个数和像素点数一一对应，数据第 0 个字节开始，每 4 个字节为一个像素点对应的波长数据（低位在前，高位在后）。

35. CMSPEX_Collection

【函数原型】 int CMSPEX_Collection(CMSPEX_HANDLE hd, unsigned char *buf, uint32_t addr, uint32_t len, uint32_t timeouts)

【功能】 数据采集

【参数说明】

hd：句柄；

buf：光谱数据存储 buf；

addr：数据起始地址，只能为 0；

len：读取数据的字节长度（波长个数*2）；

timeouts：超时；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
// 假设波长个数为 2048,积分时间 1000us,平均次数 2，外触发未开启
CMSPEX_HANDLE hd;// 从 CMSPEX_Init()传出的有效句柄
uint32_t len, timeouts;
len = 4096;   // (2048*2)
timeouts = 602;  // ((50+1)*2+500)
unsigned char buf[4096] = { ‘0’ };
int retValue = CMSPEX_Collection(hd, buf, 0, len, timeouts);
if (retValue == 0) {
    // success
    // 进行数据转化，以高位在前，低位在后的方式，每 2 个字节为一个光谱
```

```
    数据，以第一个点为例
    uint8_t tmp[2] = { '0' };
    uint16_t data;
    tmp[0] = buf[1];     tmp[1] = buf[0]
    memcpy(&data, tmp, sizeof(char)*2);
    // 此时 data 为第 1 个点的光谱值，以此类推可得出全部光谱数据
}
else {
    // failure
}
```

【注意】

光谱数据个数和像素点数一一对应，数据第 0 个字节开始，每 2 个字节为一个像素点对应的光谱数据（高位在前，低位在后）。

36. CMSPEX_UserFlashWrites

【函数原型】 int CMSPEX_UserFlashWrites(CMSPEX_HANDLE hd, uint32_t addr, unsigned char *buf, uint32_t cnts)

【功能】 用户写 flash，按扇区写，每个扇区 4096 个字节，一次最少写一个扇区

【参数说明】

hd：句柄；

addr：写扇区的起始地址（640~1023）；

buf：写进 flash 的数据；

cnts：写扇区的个数；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
// 假设从 645 扇区开始，写 2 个扇区
CMSPEX_HANDLE hd; // 从 CMSPEX_Init()传出的有效句柄
unsigned char buf[8192] = { '0' };
```

```
// 给 buf 赋值…
int retValue = CMSPEX_UserFlashWrites(hd, 645, buf, 2);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

【注意】

用户可操作的 flash 扇区地址的范围为 640~1023。

37. CMSPEX_UserFlashReads

【函数原型】 int CMSPEX_UserFlashReads(CMSPEX_HANDLE hd, uint32_t addr, unsigned char *buf, uint32_t cnts)

【功能】 用户读 flash，按扇区读，每个扇区 4096 个字节，一次最少读一个扇区

【参数说明】

hd：句柄；

addr：读扇区的起始地址（640~1023）；

buf：从 flash 读出的数据；

cnts：读扇区的个数；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
// 假设从 645 扇区开始，读 2 个扇区
CMSPEX_HANDLE hd; // 从 CMSPEX_Init()传出的有效句柄
unsigned char buf[8192] = { '0' };
int retValue = CMSPEX_UserFlashReads(hd, 645, buf, 2);
if (retValue == 0) {
    // success - buf 的数据就是从 flash 读出来的数据
```

```
}
else {
    // failure
}
```

【注意】

用户可操作的 flash 扇区地址的范围为 640~1023。

38. CMSPEX_CollectionByShortCmd

【函数原型】 int CMSPEX_CollectionByShortCmd(CMSPEX_HANDLE hd, unsigned char *buf, uint32_t len, uint32_t &rlen, uint32_t timeouts)

【功能】 数据采集

【参数说明】

hd：句柄；

buf：光谱数据存储 buf；

len：读取数据的字节长度（波长个数*2+32）；

rlen：实际读取到的字节长度；

timeouts：超时；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
// 假设波长个数为 2048,积分时间 1000us,平均次数 2，外触发未开启
CMSPEX_HANDLE hd;// 从 CMSPEX_Init()传出的有效句柄
uint32_t len, rlen, timeouts;
len = 4128;   // (2048*2+32)
Timeouts = 602; // ((50+1)*2+500)
unsigned char buf[4128] = { '0' };
int retValue = CMSPEX_CollectionByShortCmd(hd, buf, len, rlen,
timeouts);
if (retValue == 0) {
```

```
    // success
    // 进行数据转化，以高位在前，低位在后的方式，每 2 个字节为 1 个光谱
    数据，以第一个点为例
    uint8_t tmp[2] = { '0' };
    uint16_t data;
    tmp[0] = buf[29];    tmp[1] = buf[28]
    memcpy(&data, tmp, sizeof(char)*2);
    // 此时 data 为从 0 开始，第 0 个像素点的光谱值，以此类推可得出全部
    光谱数据
}
else {
    // failure
}
```

【注意】

受波长范围的影响，数据个数与波长个数相等，每个光谱数据占 2 个字节（高位在前，低位在后），从 0 开始，第 0~27 位为包头，第 28，29 为第 1 个光谱数据，第 30，31 为第 2 个光谱数据，依次类推，最后 4 个为 crc32 校验码。

39. CMSPEX_GetWavelengthByShortCmd

【函数原型】int CMSPEX_GetWavelengthByShortCmd(CMSPEX_HANDLE hd, unsigned char *buf, uint32_t len, uint32_t &rlen)

【功能】 获取波长数据

【参数说明】

hd：句柄；

buf：波长数据存储 buf；

len：读取数据的字节长度（波长个数*4+32）；

rlen：实际读取到的字节长度；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
// 假设读取全部波长数据
CMSPEX_HANDLE hd;// 从 CMSPEX_Init()传出的有效句柄
uint32_t len, rlen;
len = 8224;   // (2048*4+32)
unsigned char buf[8224] = { '0' };
int retValue = CMSPEX_GetWavelengthByShortCmd(hd, buf, len, rlen);
if (retValue == 0) {
    // success
    // 进行数据转化，以低位在前，高位在后的方式，每 4 个字节为 1 个波长数据，以第 1 个点为例
    float data;
    memcpy(&data, buf+28, sizeof(char)*4);
    // 此时 data 为第 1 个波长数据，以此类推可得出全部波长数据
}
else {
    // failure
}
```

【注意】

不受像素范围的影响，波长个数最多 2048 个，每个波长数据占 4 个字节（低位在前，高位在后），从 0 开始，第 0~27 位为包头，第 28，29，30，31 为第 1 个波长数据， 第 32，33，34，35 为第 2 个光谱数据，依次类推，最后 4 个为 crc32 校验码。

二、固化参数通信接口

1. CMSPEX_SetIntegrationTimeDown

【函数原型】 int CMSPEX_SetIntegrationTimeDown(CMSPEX_HANDLE hd,uint32_t uiInterTime)

【功能】 设置默认积分时间，即时生效，且固化进光谱仪，断电后此值为默认值，单位 us

【参数说明】

hd：句柄；

uiInterTime：积分时间，单位 us；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
// 假设设置默认积分时间为 100us
CMSPEX_HANDLE hd;// 从 CMSPEX_Init()传出的有效句柄
int retValue = CMSPEX_SetIntegrationTimeDown(hd, 100);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

【注意】

积分时间范围：60~100 000 000（us）。

2. CMSPEX_GetIntegrationTimeDown

【函数原型】 int CMSPEX_GetIntegrationTimeDown(CMSPEX_HANDLE hd,uint32_t &uiInterTime)

【功能】 获取默认积分时间，单位 us

【参数说明】

hd：句柄；

uiInterTime：积分时间，单位 us；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd;// 从 CMSPEX_Init()传出的有效句柄
uint32_t uiInterTime;
int retValue = CMSPEX_GetIntegrationTimeDown(hd, uiInterTime);
if (retValue == 0) {
    // success -- uiInterTime 为获取到的默认积分时间
}
else {
    // failure
}
```

【注意】

积分时间范围：60~100 000 000（us）。

3. CMSPEX_SetAccAvgDown

【函数原型】 int CMSPEX_SetAccAvgDown(CMSPEX_HANDLE hd, uint32_t uiAccTimes)

【功能】 设置默认底层平均次数，即时生效，且固化进光谱仪，断电后此值为默认值

【参数说明】

hd：句柄；

uiAccTimes：底层平均次数；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
// 假设设置默认平均次数为 1
CMSPEX_HANDLE hd; // 从 CMSPEX_Init()传出的有效句柄
int retValue = CMSPEX_SetAccAvgDown(hd, 1);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

【注意】

底层平均次数范围：1~255。

4. CMSPEX_GetAccAvgDown

【函数原型】 int CMSPEX_GetAccAvgDown(CMSPEX_HANDLE hd, uint32_t &uiAccTimes)

【功能】 获取默认底层平均次数

【参数说明】

hd：句柄；

uiAccTimes：底层平均次数；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd; // 从 CMSPEX_Init()传出的有效句柄
uint32_t uiAccTimes;
int retValue = CMSPEX_GetAccAvgDown(hd, uiAccTimes);
if (retValue == 0) {
    // success -- uiAccTimes 为获取到的默认底层平均次数
}
else {
```

```
    // failure
}
```

【注意】

底层平均次数范围：1~255。

5. CMSPEX_SetPixSmoothingWidthDown

【函数原型】 int CMSPEX_SetPixSmoothingWidthDown(CMSPEX_HANDLE hd, uint32_t uiWidth)

【功能】 设置光谱平滑宽度，即时生效，且固化进光谱仪，断电后此值为默认值

【参数说明】

hd：句柄；

uiWidth：光谱平滑宽度；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
// 假设设置光谱平滑宽度为 2
CMSPEX_HANDLE hd;// 从 CMSPEX_Init()传出的有效句柄
int retValue = CMSPEX_SetPixSmoothingWidthDown(hd, 2);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

【注意】

光谱平滑宽度范围：0~100。

6. CMSPEX_GetPixSmoothingWidthDown

【函数原型】 int CMSPEX_GetPixSmoothingWidthDown(CMSPEX_HANDLE hd,

uint32_t uiWidth)

【功能】 查询光谱平滑宽度

【参数说明】

hd：句柄；

uiWidth：光谱平滑宽度；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd;// 从 CMSPEX_Init()传出的有效句柄
uint32_t uiWidth;
int retValue = CMSPEX_SetPixSmoothingWidthDown(hd, uiWidth);
if (retValue == 0) {
    // success -- uiWidth 为获取到的光谱平滑宽度
}
else {
    // failure
}
```

【注意】

光谱平滑宽度范围：0~100。

7. CMSPEX_GetSlitWidthDown

【函数原型】 int CMSPEX_GetSlitWidthDown(CMSPEX_HANDLE hd, uint32_t &uiWidth)

【功能】 获取光谱仪狭缝宽度，单位μm

【参数说明】

hd：句柄；

uiWidth：光谱仪狭缝宽度，单位μm；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd;// 从CMSPEX_Init()传出的有效句柄
uint32_t uiWidth;
int retValue = CMSPEX_GetSlitWidthDown(hd, uiWidth);
if (retValue == 0) {
    // success -- uiWidth为获取到的光谱仪狭缝宽度
}
else {
    // failure
}
```

【注意】

8. CMSPEX_SetOemSpecSerialNumberDown

【函数原型】 int CMSPEX_SetOemSpecSerialNumberDown(CMSPEX_HANDLE hd, unsigned char *serialNum)

【功能】 设置OEM用户光谱仪序列号，固化进光谱仪

【参数说明】

hd：句柄；

serialNum：OEM用户光谱仪序列号，12字节；

【返回值说明】

成功返回0，失败返回负数；

【示例】

```
// 假设设置OEM用户光谱仪序列号为“M_UV1_S11639”
CMSPEX_HANDLE hd;// 从CMSPEX_Init()传出的有效句柄
unsigned char serialNum[12] = { ‘0’ };
sprintf(serialNum, "M_UV1_S11639");
int retValue = CMSPEX_SetOemSpecSerialNumberDown(hd, serialNum);
if (retValue == 0) {
    // success
```

```
}
else {
    // failure
}
```

【注意】

9. CMSPEX_GetOemSpecSerialNumberDown

【函数原型】 int CMSPEX_GetOemSpecSerialNumberDown(CMSPEX_HANDLE hd, unsigned char *serialNum)

【功能】 查询 OEM 用户光谱仪序列号

【参数说明】

hd：句柄；

serialNum：OEM 用户光谱仪序列号，12 字节；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd;// 从 CMSPEX_Init()传出的有效句柄
unsigned char serialNum[12] = { '0' };
int retValue = CMSPEX_GetOemSpecSerialNumberDown(hd, serialNum);
if (retValue == 0) {
    // success -- serialNum 为获取到的 OEM 用户光谱仪序列号
}
else {
    // failure
}
```

【注意】

10. CMSPEX_SetWaveRangeDown

【函数原型】 int CMSPEX_SetWaveRangeDown(CMSPEX_HANDLE hd, double dStart,

double dStop)

【功能】 设置光谱有效波长范围，即时生效，且固化进光谱仪，断电后此值为默认值，单位 nm

【参数说明】

hd：句柄；

dStart：波长起始位置,单位 nm；

dStop：波长结束位置,单位 nm；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
// 假设设置波长范围为 350~800nm
CMSPEX_HANDLE hd;// 从 CMSPEX_Init()传出的有效句柄
int retValue = CMSPEX_SetWaveRangeDown(hd, 350, 800);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

【注意】

11. CMSPEX_GetWaveRangeDown

【函数原型】 int CMSPEX_GetWaveRangeDown(CMSPEX_HANDLE hd, double &dStart, double &dStop)

【功能】 获取光谱有效波长范围

【参数说明】

hd：句柄；

dStart：波长起始位置,单位 nm；

dStop：波长结束位置,单位 nm；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd; // 从 CMSPEX_Init()传出的有效句柄
double dStart, dStop;
int retValue = CMSPEX_GetWaveRangeDown(hd, dStart, dStop);
if (retValue == 0) {
    // success -- dStart,dStop 为获取到的起始波长和结束波长
}
else {
    // failure
}
```

【注意】

12. CMSPEX_GetWaveNumberDown

【函数原型】 int CMSPEX_GetWaveNumberDown(CMSPEX_HANDLE hd, uint32_t &uiNum)

【功能】 获取光谱有效波长个数

【参数说明】

hd：句柄；

uiNum：光谱波长个数；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd; // 从 CMSPEX_Init()传出的有效句柄
uint32_t uiNum;
int retValue = CMSPEX_GetWaveNumberDown(hd, uiNum);
if (retValue == 0) {
    // success -- uiNum 为获取到的光谱波长个数
```

```
}
else {
    // failure
}
```

【注意】

每次更改波长范围后，像素范围和波长个数都会更新。

13. CMSPEX_GetPixelRangeDown

【函数原型】 int CMSPEX_GetPixelRangeDown(CMSPEX_HANDLE hd, uint32_t &uiStart, uint32_t &uiStop)

【功能】 获取光谱有效像素范围，根据设置的波长范围变化而变化

【参数说明】

hd：句柄；

uiStart：像素起始位置；

uiStop：像素结束位置；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd; // 从 CMSPEX_Init()传出的有效句柄
uint32_t uiStart, uiStop;
int retValue = CMSPEX_GetPixelRangeDown(hd, uiStart, uiStop);
if (retValue == 0) {
    // success -- uiStart,uiStop 为获取到的光谱像素范围
}
else {
    // failure
}
```

【注意】

每次更改波长范围后，像素范围和波长个数都会更新。

14. CMSPEX_GetOriWaveRangeDown

【函数原型】 int CMSPEX_GetOriWaveRangeDown(CMSPEX_HANDLE hd, double &dStart, double &dStop)

【功能】 获取光谱原始波长范围

【参数说明】

hd：句柄；

dStart：原始波长起始位置；

dStop：原始波长结束位置；

【返回值说明】

成功返回 0，失败返回负数；

【示例】

```
CMSPEX_HANDLE hd;// 从 CMSPEX_Init()传出的有效句柄
double dStart, dStop;
int retValue = CMSPEX_GetOriWaveRangeDown(hd, dStart, dStop);
if (retValue == 0) {
    // success -- dStart, dStop 为获取到的原始波长范围
}
else {
    // failure
}
```

【注意】

当需要获取完整光谱数据时，将有效波长范围改为原始波长范围即可。

三、获取光谱数据流程参考

1. 初始化

```
CMSPEX_HANDLE hd;
int retValue = CMSPEX_Init(&hd, 0);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

初始化成功后，才可对接口进行其它操作,注意选择设备的通信方式。

2. 打开光谱仪

打开（连接）光谱仪有两种方式，选一种打开即可：

① 用设备序列号打开

```
//USB连接时，“ABOOPXDF”为设备的序列号,可通过CMSPEX_GetDevList()获取
retValue = CMSPEX_Open(hd, "ABOOPXDF");
//串口连接时，“COM3”为串口号
retValue = CMSPEX_Open(hd, "COM3");
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

② 用设备索引打开

```
// USB连接时，0为设备索引，可通过CMSPEX_GetDevList()获取
retValue = CMSPEX_OpenByIndex(pgHandle, 0);
```

```
    //串口连接时，“COM3”为串口号
    retValue = CMSPEX_Open(hd, 3);
    if (retValue == 0) {
        // success
    }
    else {
        // failure
    }
```

补充：获取 USB 设备列表方法

```
uint32_t uiDevCnts;
retValue = CMSPEX_GetDevList(hd, NULL, uiDevCnts, 1);
if (rslt != 0)
{
    // failure
    return ;
}
DEVLSTUSB *devlist = new DEVLSTUSB[uiDevCnts];
retValue = CMSPEX_GetDevList(hd, devlist, uiDevCnts, 0);
if (retValue ! = 0) {
    // failure
    return;
}
// success
```

// devlist 为获取到的设备列表信息，其中包括设备索引和设备序列号，可用于打开光谱仪

```
// devlist[0].devnums：第一个设备的索引
// devlist[0].serialnumber：第一个设备的序列号
```

3. 设置参数（非必须，根据实际需求进行选择）

① 设置积分时间

当需要使用氙灯时，建议积分时间大于一个氙灯脉冲周期。

```
// 假设设置积分时间为16000us
retValue = CMSPEX_SetIntegrationTime(hd, 16000);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

② 设置平均次数

```
// 假设设置平均次数为1
retValue = CMSPEX_SetAccAvg(hd, 1);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

③ 设置有效波长范围

```
// 假设设置波长范围为200~800nm
retValue = CMSPEX_SetWaveRangeDown(hd, 200, 800);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

④ 设置氙灯脉冲

```
// 假设设置氙灯脉冲低电平 16ms，高电平 0.1ms
retValue = CMSPEX_SetLampPulse(hd, 1600000, 10000);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

⑤ 设置氙灯开关

```
// 假设设置氙灯开关为单次脉冲模式
retValue = CMSPEX_LampEnable(hd, 2);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

4. 获取参数

① 获取波长对应像素范围

```
uint32_t uiStart, uiStop;
int retValue = CMSPEX_GetPixelRangeDown(hd, uiStart, uiStop);
if (retValue == 0) {
    // success -- uiStart,uiStop 为获取到的光谱像素范围
}
else {
    // failure
}
```

② 获取波长个数，用于计算光谱数据个数

```
uint32_t uiNum;
retValue = CMSPEX_GetWaveNumberDown(hd, uiNum);
if (retValue == 0) {
    // success -- uiNum 为获取到的光谱波长个数
}
else {
    // failure
}
```

③ 获取波长数据，有以下两种方式，选一种即可

波长数据的个数和像素点数相同，并且每个波长数据和像素点一一对应。

1) 根据像素范围直接获取

```
uint32_t len, rlen;
len = uiNum*4;
unsigned char buf[8192] = { '0' };
double dWave[2048];
memset(dWave, 0, sizeof(double)*2048);
retValue = CMSPEX_GetWavelength(hd, buf, uiStart, len);
if (retValue == 0) {
    // success
    // 进行数据转化，以低位在前，高位在后的方式，每 4 个字节为一个
    波长数据
    float fData;
    for (int i = 0; i < uiNum; i++)
    {
        memcpy(&fData, buf+(i*4), sizeof(char)*4);
        dWave[i] = fData;
    }
```

```
}
else {
    // failure
}
```

2) 通过获取波长标定参数，再经过计算得出

```
// 获取标定系数
double dWaveCal[4];
memset(dWaveCal, 0, sizeof(double)*4);
int retValue = CMSPEX_GetWaveCalParams(hd, dWaveCal);
if (retValue == 0) {
    // success -- dWaveCal 为波长标定参数
}
else {
    // failure
}
// 计算波长数据
int i, j;
for (i = uiStart; i <= uiStop; i++)
{
    dWave[i] = 0;
    for (j = 0; j < 4; j++)
    {
        dWave[i] += dWaveCal[j]*pow((double)i, j);
    }
}
```

④ 获取积分时间（非必须）

```
uint32_t uiInterTime;
retValue = CMSPEX_GetIntegrationTime(hd, uiInterTime);
if (retValue == 0) {
```

```
    // success -- uiInterTime 为获取到的积分时间
}
else {
    // failure
}
```

⑤ 获取平均次数（非必须）

```
uint32_t uiAccTimes;
retValue = CMSPEX_GetAccAvg(hd, uiAccTimes);
if (retValue == 0) {
    // success -- uiAccTimes 为获取到的底层平均次数
}
else {
    // failure
}
```

用户也可根据实际需求对其它参数进行获取。

5. 获取光谱数据

5.1. 未开启外触发时

```
// 直接进行数据采集
uint32_t len, rlen, timeouts, uiInterTime2;
len = uiNum*2;    // ((uiStop-uiStart+1)*2)
uiInterTime2 = uiInterTime/1000.0;
timeouts = (50+ceil(uiInterTime2))*uiAccTimes + 500;
char buf[4116] = { '0' };
retValue = CMSPEX_Collection(hd, buf, 0, len, timeouts);
if (retValue == 0) {
    // success
    // 进行数据转化，以高位在前，低位在后的方式，每 2 个字节为一个光谱数据，以第一个点为例
```

```
        uint8_t tmp[2] = { '0' };
        uint16_t data;
        tmp[0] = buf[1];      tmp[1] = buf[0]
        memcpy(&data, tmp, sizeof(char)*2);
        // 此时 data 为第 1 个点的光谱值，以此类推可得出全部光谱数据
    }
    else {
        // failure
    }
```

5.2. 开启外触发时

① 设置光谱仪外触发配置

```
    // 假设设置打开外触发，外触发模式为上升沿触发，光谱采集个数为 10；
    retValue = CMSPEX_SetSpTrigConfig(fd, 1, 1, 10);
    if (retValue == 0) {
        // success
    }
    else {
        // failure
    }
```

② 获取光谱数据采集状态

注意：数据采集需要一定时间，包括积分时间，平均次数等都会直接关系到采集时间。

```
    uint32_t uiCollStatus;
    //长度对应上面的光谱采集个数，最大为 63
    unsigned char szDataStatus[10] = { '0' };
    retValue = CMSPEX_GetSpecCollectionStatus(fd, uiCollStatus,
szDataStatus, 10);
    if (retValue == 0) {
```

```
    // success
    // uiCollStatus：0：表示全部数据都被取走，即没有数据， 1：有数据更新；
    //szDataStatus：例 szDataStatus[0] = 0,表示第 1 帧数据还未采集完成或者已经取走，szDataStatus[0] = 1,表示第 1 帧数据已经采集完成，依次类推，得到每一帧数据的采集状态
}
else {
    // failure
}
```

③ 设置外触发光谱获取控制

```
// 假设现在获取第 1 帧数据
```

注意：在获取光谱数据采集状态时，第 1 帧数据已经采集完成才能取数据

```
retValue = CMSPEX_SetTrigGetSpecCtrl(fd, 0);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```

④ 获取外触发光谱获取控制

注意：等待一段时间再获取，在合理时间范围内，没有提示执行失败可多次获取。

```
uint32_t uiAddr, uiResult;
retValue = CMSPEX_GetTrigGetSpecCtrl(fd, uiAddr, uiResult);
if (retValue == 0) {
    // success
    // uIAddr：取第几帧光谱，此时应该为 0；
    // uiResult：0：表示正在 FPGA 还在运行中（取数据），1：表示执行
```

完成并成功，可以取数据了，2：执行失败，不可以取数据；

```
}
else {
    // failure
}
```

⑤ 获取光谱数据

```
uint32_t len, rlen, timeouts, uiInterTime2;
len = uiNum*2;    // ((uiStop-uiStart+1)*2)
uiInterTime2 = uiInterTime/1000.0;
timeouts = (50+ceil(uiInterTime2))*uiAccTimes + 500;
unsigned char buf[4116] = { '0' };
rslt = CMSPEX_Collection(fd, 0, len, timeouts);
if (retValue == 0) {
    // success
    // 进行数据转化，以高位在前，低位在后的方式，每 2 个字节为一个光谱数据，以第一个点为例
    uint8_t tmp[2] = { '0' };
    uint16_t data;
    tmp[0] = buf[1];      tmp[1] = buf[0]
    memcpy(&data, tmp, sizeof(char)*2);
    // 此时 data 为第 1 个点的光谱值，以此类推可得出全部光谱数据
}
else {
    // failure
}
```

6. 关闭

```
retValue = CMSPEX_Close(hd);
if (retValue == 0) {
```

```
    // success
}
else {
    // failure
}
```

7. 释放接口内存

```
retValue = CMSPEX_DeInit(hd);
if (retValue == 0) {
    // success
}
else {
    // failure
}
```