

# Ocean Optics USB4000 with *usb4java*

Bernard Panneton  
December 2019

## Introduction

The `usb4java` library<sup>1</sup> is used to communicate with a USB peripheral. It used to interact with an Ocean Optics Inc. USB4000 spectrometer.

This library was used to build a collection of R functions stored in ***playWith\_usbjava.R*** file. The commands required to communicate with the spectrometer via the USB port are detailed in the device technical manual<sup>2</sup>.

## Functions in `playWith_usb4java.R`

### `init_usb()`

Initialize usb device:

1. load required library
2. init JVM
3. Set Java class paths
4. Define some objects
5. Define a C helper function for converting littleEndian byte vector to integer

RETURN: a list of 4 components:

1. context: a Java object of class `org.usb4java.Context`
2. dlist: a Java object of class `org.usb4java.DeviceList`
3. libusb: a Java object of class `org.usb4java.LibUsb`
4. bufutils: a Java object of class `org.usb4java.BufferUtils`

### `find_usb <- function(product,vendor,usbObjects, silent=TRUE)`

Given a vendor and a product ID number, find the corresponding USB device.

INPUTS:

- product: product ID number
- vendor: vendor ID number
- usbObjects: list returned by `init_usb`
- silent: when TRUE, no output at console.

OUTPUTS:

- a list with 2 components
  1. `usbDevice`: the device as obtained with `dlist$get(as.integer(dev_no))` where `dlist` was defined in `init_usb`
  2. `usbDescription`: obtained by `libusb$DeviceDescriptor(usbDevice, usbDescription)`

---

<sup>1</sup><http://usb4java.org/>

<sup>2</sup>USB4000-OEM-Data-Sheet.pdf stored in the **Doc** of the RStudio project **OceanOptics\_with\_usb4java\_in\_R**.

### **get\_OO\_name\_n\_serial(usbObjects, usbDevice)**

Function to get device name and version, device serial number and company names.

INPUTS:

- usbObjects: the list returned by `init_usb()`
- usbDevice: the list returned by `find_usb()`

OUTPUTS:

- list with 3 components:
  1. name: name of the USB device
  2. version: name of device with version number
  3. serialno: serial number

### **getWavelengths(usbObjects, usbDevice)**

To get the wavelength vector by reading the wavelength calibration coefficients

INPUTS:

- usbObjects: the list returned by `init_usb()`
- usbDevice: the list returned by `find_usb()`

OUTPUTS:

- a vector of wavelengths

### **jbyte\_2\_uint(x)**

Takes a vector of Java bytes and interprets and 0:255. Required as Java bytes are signed.

INPUTS:

- x: vector of Java bytes as seen in R

OUTPUTS:

- a vector of value in the range 0:255, same length as input.

### **queryStatus <- function(usbObjects, usbDevice)**

Query the USB device status.

INPUTS:

- usbObjects: the list returned by `init_usb()`
- usbDevice: the list returned by `find_usb()`

OUTPUTS:

- a list of 5 elements:
  1. nb\_pix: number of pixels in spectrum
  2. int\_time: current integration time
  3. pack\_in\_spectra: number of data packets per spectrum
  4. pack\_count:
  5. usb\_speed: speed of USB transfer (“full” or “high”)

### **setIntegrationTime(temps,usbObjects, usbDevice)**

Function to set spectrometer integration time.

INPUTS:

- temps: integration time in msec.
- usbObjects: the list returned by `init_usb()`
- usbDevice: the list returned by `find_usb()`

OUTPUTS:

- none

### **getSpectrum(pack\_in\_spectra=15, usbObjects, usbDevice)**

Function to retrieve a spectrum.

INPUTS:

- pack\_in\_spectra: number of data packets per spectrum
- usbObjects: the list returned by `init_usb()`
- usbDevice: the list returned by `find_usb()`

OUTPUTS:

- a spectrum as a numeric vector.

### **get\_N\_Spectrum(pack\_in\_spectra=15, nspectra=2, usbObjects, usbDevice)**

Function to retrieve a spectrum made as an average over a number of spectra.

INPUTS:

- pack\_in\_spectra: number of data packets per spectrum
- nspectra: number of spectrum to average over.
- usbObjects: the list returned by `init_usb()`
- usbDevice: the list returned by `find_usb()`

OUTPUTS:

- a spectrum as a numeric vector.

### **free\_Device(usbObjects)**

Function to free device. Required to cleanly end.

INPUTS:

- – usbObjects: the list returned by `init_usb()`

## **Example**

Connect a USB4000 to a USB port and copy the following lines to the R console.

```
#Testing functions
```

```
source("R/playWith_usb4java.R")
```

```

usbObjects <- init_usb()
product=0x1022
vendor=0x2457
usbDevice <- find_usb(product,vendor,usbObjects,TRUE)
name_serial <- get_OO_name_n_serial(usbObjects, usbDevice$usbDevice)
lapply(name_serial, print)
wv <- getWavelengths(usbObjects, usbDevice$usbDevice)
statut <- queryStatus(usbObjects, usbDevice$usbDevice)
setIntegrationTime(70,usbObjects,usbDevice$usbDevice)
(statut <- queryStatus(usbObjects, usbDevice$usbDevice))
dum <- getSpectrum(pack_in_spectra=15, usbObjects, usbDevice$usbDevice)
dum <- getSpectrum(pack_in_spectra=15, usbObjects, usbDevice$usbDevice)
plot(wv, dum[22:3669],type="l",col="red",lwd=2, ylim=c(0,7000))
sp <- get_N_Spectrum(pack_in_spectra=15, nspectra=20, usbObjects, usbDevice$usbDevice)
plot(wv,sp[22:3669],type="l",col="red",lwd=2, ylim=c(0,40000), main = paste0(name_serialname," –
Serialnumber : ",name_serialserialno), xlab = "Wavelength [nm]", ylab = "Intensity [A.U.]")
free_Device(usbObjects)

```

## Final words

The code in the R project works with a USB4000. Adaptation to other spectrometer should be fairly straightforward. One needs to check the product ID and the exact syntax of device specific commands and parameters (the number of pixels in a spectrum, the number of packets transmitted over the USB...).