Package: rOmniDriver (via r-universe)

June 28, 2024

Type Package

Title Omni Driver R wrapper

Version 0.1.20

Date 2024-03-29

Maintainer Pedro J. Aphalo <pedro.aphalo@helsinki.fi>

Description This package is a wrapper of the OmniDriver java driver for Ocean Optics spectrometers.

License GPL (>= 2)

Depends R (>= 4.2.0), rJava (>= 1.0-6)

SystemRequirements Java JDK (>= 1.8), OmniDriver (>= 2.43)

Suggests knitr (>= 1.40), rmarkdown (>= 2.20), roxygen2 (>= 7.2.1)

VignetteBuilder knitr

LazyLoad yes

Encoding UTF-8

URL http://www.r4photobiology.info, https://github.com/aphalo/romnidriver

BugReports https://github.com/aphalo/romnidriver/issues

RoxygenNote 7.3.1

Repository https://aphalo.r-universe.dev

RemoteUrl https://github.com/aphalo/rOmniDriver

RemoteRef HEAD

RemoteSha fe5d7a7052fb4b1383cffe875bd8aadfa5b697c4

Contents

`rOmniDriver-package` *rOmniDriver: Omni Driver R wrapper*

Description

This package is a wrapper of the OmniDriver java driver for Ocean Optics spectrometers.

Details

OmniDriver is a proprietary library provided by Ocean Insight to allow the control of the spectrometers they sell through third party software.

Although this package is under GPL, Omnidriver is not, it is copyrighted commercial software from Ocean Insight http://www.oceanoptics.com

Ocean Insight distributes the OmniDriver runtime free of charge. This can be downloaded from Ocean Insight web site. It can be used only with spectrometers sold by Ocean Insight. Ocean Insight has supported the development of this R package by providing a free licence to the OmniDriver SDP, which gave me access to documentation.

Ocean Insight will not provide support for the use of this package, unless the user buys a licence for OmniDriver SDP (spectroscopy development plattform). All support requests about this package should be addressed to the maintainer of the package or public R user forums.

Warning!

We have access to only three spectrometers for testing, a Maya2000Pro, a two-chennal Jaz and a Flame. So, the software is supplied without any assurance that it will work with a particular model and configuration of spectrometer. Please, do report to the maintainer any problems and also if you encounter no problems when using this package. Always supply detailed information about the spectrometer used when contacting the maintainer of the package.

Note

Many methods in OmniDriver are available in two versions, one for multichannel spectrometers and one for single channels ones. In this package each such pair of similar methods are wrapped into a single R method. The default channel index, `ch.index`, in the R implementation defaults to the first available channel, so it does not need to be explicitly supplied for single channel instruments. The spectrometer index, `sr.index`, in the R implementation defaults to the first available spectrometer, so it does not need to be explicitly supplied when a single instrument is connected to the host computer.

Author(s)

Maintainer: Pedro J. Aphalo <pedro.aphalo@helsinki.fi> (ORCID)

References

http://www.oceanoptics.com/Products/omnidriver.asp

http://oceanoptics.com/api/omnidriver/overview-summary.html

See Also

Useful links:

- http://www.r4photobiology.info
- https://github.com/aphalo/romnidriver
- Report bugs at https://github.com/aphalo/romnidriver/issues

---

close_all_spectrometers

close all open connections to spectrometers

Description

close all open connections to spectrometers

Usage

```
close_all_spectrometers(jwrapper)

srs_close(jwrapper)
```

Arguments

jwrapper	a open Wrapper object from Omnidriver

Value

a numeric value

---

get_api_version *Get OmniDriver API version*

Description

A function to query the version of OmniDriver.

Usage

```
get_api_version(jwrapper)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver

Value

A character string value of form "1.01.01" or similar.

---

get_bench *Get optical properties of optical bench*

Description

A function to query the characteristics of the optical bench from a spectrometer.

Usage

```
get_bench(jwrapper, sr.index = 0L, ch.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer
ch.index	an index to address the channel in a spectrometer with more than one channel.

Value

a value

get_boxcar_width *Get current setting of boxcar width*

Description

Function to query the boxcar width being used (the number of pixels averaged).

Usage

```
get_boxcar_width(jwrapper, sr.index = 0L, ch.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer
ch.index	an index to address the channel in a spectrometer with more than one channel.

Value

a numeric value

get_calibration_coefficients_from_buffer *Get calibration coefficients from buffer*

Description

Function to query calibration coefficients from memory buffer of a spectrometer.

Usage

```
get_calibration_coefficients_from_buffer(
  jwrapper,
  sr.index = 0L,
  ch.index = 0L
)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer
ch.index	an index to address the channel in a spectrometer with more than one channel.

Value

A `Coefficients` Java object containing all of the calibration coefficients..

`get_calibration_coefficients_from_eeprom`
Get calibration coefficients from EEPROM

Description

Function to query calibration coefficients from the EEPROM of a spectrometer.

Usage

```
get_calibration_coefficients_from_eeprom(
  jwrapper,
  sr.index = 0L,
  ch.index = 0L
)
```

Arguments

`jwrapper`	an open Wrapper object from Omnidriver
`sr.index`	an index to address the spectrometer
`ch.index`	an index to address the channel in a spectrometer with more than one channel.

Value

A `Coefficients` Java object containing all of the calibration coefficients..

`get_correct_for_detector_nonlineary`
Get current setting of linearity correction (enabled or not)

Description

Get current setting of linearity correction (enabled or not)

Usage

```
get_correct_for_detector_nonlineary(jwrapper, sr.index = 0L, ch.index = 0L)
```

Arguments

`jwrapper`	an open Wrapper object from Omnidriver
`sr.index`	an index to address the spectrometer
`ch.index`	an index to address the channel in a spectrometer with more than one channel.

Value

a numeric value

get_correct_for_electrical_dark

Get current setting for electrical dark correction

Description

Get current setting for electrical dark correction

Usage

```
get_correct_for_electrical_dark(jwrapper, sr.index = 0L, ch.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer
ch.index	an index to address the channel in a spectrometer with more than one channel.

Value

a numeric value

get_correct_for_stray_light

Get current setting of stray light correction

Description

Get current setting of stray light correction

Usage

```
get_correct_for_stray_light(jwrapper, sr.index = 0L, ch.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer
ch.index	an index to address the channel

get_detector *Get description of detector*

Description

Get description of detector

Usage

```
get_detector(jwrapper, sr.index = 0L, ch.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer
ch.index	an index to address the channel in a spectrometer with more than one channel.

Value

a Detector object (Java)

Note

This method seems to be only of use for multichannel spectrometers. It is not documented in the OmniDriver manual and the Java documentation is extremely terse so it is difficult to know what it does and/or how to use it.

get_feature_controller_saturation_threshold *Get feature "saturation threshold"*

Description

A function to retrieve an interface that allows to control the saturation threshold.

Usage

```
get_feature_controller_saturation_threshold(jwrapper, sr.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer

Value

an object which provides the desired interface, or null if this feature is not available for this spectrometer.

Note

Before calling this function you first need to check that the feature is supported by the spectrometer in use by calling `is_feature_supported_saturation_threshold`.

`get_feature_I2C_bus` *Get feature controller "I2C Bus"*

Description

Get a wrapper for accessing this optional feature.

Usage

```
get_feature_I2C_bus(jwrapper, sr.index = 0L)
```

Arguments

`jwrapper`	an open Wrapper object from Omnidriver
`sr.index`	an index to address the spectrometer

Value

an object which provides the desired interface, or null if this feature is not available for this spectrometer.

Note

Before calling this function you first need to check that the feature is supported by the spectrometer in use by calling `is_feature_supported_I2C_bus`.

See Also

Other Spectrometer I2C- and SPI-bus functions.: `get_I2C_bus()`, `get_SPI_bytes()`, `get_feature_SPI_bus()`, `is_feature_supported_I2C_bus()`, `is_feature_supported_SPI_bus()`

get_feature_pixel_binning

Get feature "pixel binning"

Description

Obtain an inferface to the "pixel binning" function of a spectrometer if available.

Usage

```
get_feature_pixel_binning(jwrapper, sr.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer

Value

an object which provides the desired interface, or null if this feature is not available for this spectrometer.

Note

Before calling this function you first need to check that the feature is supported by the spectrometer in use by calling is_feature_supported_pixel_binning.

get_feature_spectrum_type

Get feature "spectrum type"

Description

Get a wrapper for accesing this optional feature.

Usage

```
get_feature_spectrum_type(jwrapper, sr.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer

Value

an object which provides the desired interface, or null if this feature is not available for this spectrometer.

Note

Before calling this function you first need to check that the feature is supported by the spectrometer in use by calling `is_feature_supported_spectrum_type`.

`get_feature_SPI_bus` *Get feature controller "SPI Bus"*

Description

Get a wrapper for accessing this optional feature.

Usage

```
get_feature_SPI_bus(jwrapper, sr.index = 0L)
```

Arguments

`jwrapper`	an open Wrapper object from Omnidriver
`sr.index`	an index to address the spectrometer

Value

an object which provides the desired interface, or null if this feature is not available for this spectrometer.

Note

Before calling this function you first need to check that the feature is supported by the spectrometer in use by calling `is_feature_supported_SPI_bus`.

See Also

Other Spectrometer I2C- and SPI-bus functions.: `get_I2C_bus()`, `get_SPI_bytes()`, `get_feature_I2C_bus()`, `is_feature_supported_I2C_bus()`, `is_feature_supported_SPI_bus()`

get_firmware_model *Get name of spectrometer*

Description

A function to query the version of the firmware from a spectrometer.

Usage

```
get_firmware_model(jwrapper, sr.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer

Value

A character string value of form "USB2000" or similar.

Note

Only supported in API version >= 2.40, in earlier API versions an error is triggered if use is attempted.

get_firmware_version *Get spectrometer firmware version*

Description

A function to query the version of the firmware from a spectrometer.

Usage

```
get_firmware_version(jwrapper, sr.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer

Value

A character string value of form "1.01.01" or similar.

get_I2C_bus *Talk to "I2C bus"*

Description

Send a packet of bytes over the I2C bus and read the result, or simply read from an address.

Usage

```
get_I2C_bus(I2C.bus.wrapper, I2C.address, num.bytes)

set_I2C_bus(I2C.bus.wrapper, I2C.address, num.bytes, message)
```

Arguments

I2C.bus.wrapper	feature wrapper as returned by function rOmniDriver::get_feature_controller_I2C_bus
I2C.address	byte, address to select device in I2C bus.
num.bytes	numeric Number of bytes to be transferred or read, 1..61.
message	byte, array of bytes with maximum length of 61.

Details

Several spectrometers have an I2C bus accesible through the auxiliary connector which can be used to control accessories. The I2C bus in Ocean Insight spectrometers is not intended to be used to control the spectrometer or retrieve data but instead to control and access other accessories with the spectrometer acting as middle man.

Function get_I2C_bus() performs a general purpose read on the I2C pins for interfacing to attached peripherals. The time to complete the command is determined by the amount of data transferred and the response time of the peripheral. The I2C bus runs at 400KHz. The maximum number of bytes that can be read is 61.

Function set_I2C_bus() performs a general-purpose write on the I2C pins for interfacing to attached peripherals. The time to complete the command is determined by the amount of data transferred and the response time of the peripheral. The I2C bus runs at 400KHz.

Value

The result Array of bytes containing the read data. May be null.

The result of the write to the I2C device.

See Also

Other Spectrometer I2C- and SPI-bus functions.: get_SPI_bytes(), get_feature_I2C_bus(), get_feature_SPI_bus(), is_feature_supported_I2C_bus(), is_feature_supported_SPI_bus()

get_integration_time *Get integration time*

Description

A function to query the integration time in use from a spectrometer.

Usage

```
get_integration_time(jwrapper, sr.index = 0L, ch.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer
ch.index	an index to address the channel in a spectrometer with more than one channel.

Value

current integration time setting, in units of microseconds.

get_last_exception *Get OmniDriver last exception*

Description

A function to retrieve the most recent Java exception triggered by OmniDrive as a character string.

Usage

```
get_last_exception(jwrapper)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver

Value

A character string String with a description of the most recent error exception, or "no exception" if no error has been encountered..

get_maximum_integration_time *Get maximum integration time*

Description

A function to query the maximum integration time that can be used from a spectrometer.

Usage

```
get_maximum_integration_time(jwrapper, sr.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer

Value

maximum integration time setting, in units of microseconds.

get_maximum_intensity *Get maximum intensity*

Description

A function to query the maximum integration time that can be used from a spectrometer.

Usage

```
get_maximum_intensity(jwrapper, sr.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer

Value

maximum value per pixel, in units of counts???.

get_minimum_integration_time *Get minimum integration time*

Description

A function to query the minimum integration time that can be used from a spectrometer.

Usage

```
get_minimum_integration_time(jwrapper, sr.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer

Value

minimum integration time setting, in units of microseconds.

get_name *Get name of spectrometer*

Description

A function to query the name (e.g. USB2000) from a spectrometer.

Usage

```
get_name(jwrapper, sr.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer

Value

a value

get_number_of_channels

Get number of channels

Description

A function to query the number of channels from a spectrometer.

Usage

```
get_number_of_channels(jwrapper, sr.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer

Value

a numeric value

get_number_of_dark_pixels

Get number of dark pixels

Description

Function to query the number of pixels in the CCD array.

Usage

```
get_number_of_dark_pixels(jwrapper, sr.index = 0L, ch.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer
ch.index	an index to address the channel in a spectrometer with more than one channel.

Value

a numeric value with the total number of pixels (ie. CCD elements) provided by this spectrometer, including any dark or bevel (unused) pixels.

get_number_of_enabled_channels *Get number of enabled channels*

Description

A function to query the number of enabled channels from a spectrometer.

Usage

```
get_number_of_enabled_channels(jwrapper, sr.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer

Value

a numeric value

get_number_of_pixels *Get total number of pixels in CCD array*

Description

Function to query the total number of pixels in a spectrometer.

Usage

```
get_number_of_pixels(jwrapper, sr.index = 0L, ch.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer
ch.index	an index to address the channel in a spectrometer with more than one channel.

Value

a numeric value with the total number of pixels (ie. CCD elements) provided by this spectrometer, including any dark or bevel (unused) pixels.

get_scans_to_average *Get "number of scans to average"*

Description

Get the setting "number of scans to average" currently in use by the addressed spectrometer channel.

Usage

```
get_scans_to_average(jwrapper, sr.index = 0L, ch.index = 0L)

get_scans_to_avg(jwrapper, sr.index = 0L, ch.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer
ch.index	an index to address the channel in a spectrometer with more than one channel.

Value

a numeric value

get_seconds_time_delta *Get time difference between time stamps*

Description

Compute the time difference between two high speed time stamps.

Usage

```
get_seconds_time_delta(before, after)

get_nano_time_delta(before, after)

get_micro_time_delta(before, after)

get_milli_time_delta(before, after)
```

Arguments

before, after	HighResTimeStamp defined in OmniDriver API and retrieved or constructed with other OmniDriver API methods. Not vectorized!

Value

a numeric vector of length one giving the time difference in seconds, milliseconds, microseconds or nanoseconds.

Note

Nanosecond timing and normal timing are not coincident, but as both are stored in parallel, the differences should differ only in their resolution.

See Also

Other high speed acquisition functions: `get_seconds_time_delta_since()`, `highSpdAcq_allocate_buffer()`, `highSpdAcq_get_number_of_spectra_acquired()`, `highSpdAcq_get_spectrum()`, `highSpdAcq_get_time_stamp()`, `highSpdAcq_start_acquisition()`, `oo_timestamp_to_string()`

`get_seconds_time_delta_since` *Get time difference since a time stamp*

Description

Compute the time difference between a high speed time stamp and current time.

Usage

```
get_seconds_time_delta_since(then)

get_nano_time_delta_since(then)

get_micro_time_delta_since(then)

get_milli_time_delta_since(then)
```

Arguments

`then`	HighResTimeStamp defined in OmniDriver API and retrieved or constructed with other OmniDriver API methods. Not vectorized!

Value

a numeric vector of length one giving the time difference in seconds, milliseconds, microseconds or nanoseconds.

Note

Nanosecond timing and normal timing are not coincident, but as both are stored in parallel, the differences should differ only in their resolution.

See Also

Other high speed acquisition functions: get_seconds_time_delta(), highSpdAcq_allocate_buffer(), highSpdAcq_get_number_of_spectra_acquired(), highSpdAcq_get_spectrum(), highSpdAcq_get_time_stamp(), highSpdAcq_start_acquisition(), oo_timestamp_to_string()

---

get_serial_number *Get serial number of spectrometer*

Description

A function to query the serial number from a spectrometer.

Usage

```
get_serial_number(jwrapper, sr.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer

Value

a value

---

get_spectrum *Get a spectrum from the spectrometer*

Description

Get a spectrum from the spectrometer

Usage

```
get_spectrum(jwrapper, sr.index = 0L, ch.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer
ch.index	an index to address the channel in a spectrometer with more than one channel.

Value

a numeric value

get_SPI_bytes *Talk to "SPI bus"*

Description

Send a packet of bytes over the SPI bus and read the result.

Usage

```
get_SPI_bytes(SPI.bus.wrapper, message, num.bytes)
```

Arguments

SPI.bus.wrapper	feature wrapper as returned by function rOmniDriver::get_feature_controller_SPI_bus
message	bytes, array of bytes with maximum length of 62.
num.bytes	numeric Number of bytes, 1..62.

Details

Several spectrometers have an I2C bus accesible through the auxiliary connector which can be used to control accesories. The I2C bus in Ocean Insight spectrometers is not intended to be used to control the spectrometer or retrieve data but instead to control and access other accessories with the spectrometer acting as middle man.

Value

After writing to SPI, the spectrometer immediately reads from it. The resulting Byte array is returned.

See Also

Other Spectrometer I2C- and SPI-bus functions.: get_I2C_bus(), get_feature_I2C_bus(), get_feature_SPI_bus(), is_feature_supported_I2C_bus(), is_feature_supported_SPI_bus()

get_wavelengths *Get wavelengths from spectrometer*

Description

Function to get wavelengths corresponding to the pixels in the CCD array.

Usage

```
get_wavelengths(jwrapper, sr.index = 0L, ch.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer
ch.index	an index to address the channel in a spectrometer with more than one channel.

Value

A numeric (double) vector with the calculated wavelength at each pixel.

---

get_wrapper_extensions

Get OmniDriver wrapper extensions

Description

A function to query enable the WrapperExtensions class.

Usage

```
get_wrapper_extensions(jwrapper)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver

Value

A character string value of form "1.01.01" or similar.

---

highSpdAcq_allocate_buffer

Allocate memory buffer for high speed acquisition

Description

Pre-allocates a buffer. A call to this function should always precede a call to `highSpdAcq_start_acquisition()`

Usage

```
highSpdAcq_allocate_buffer(jwrapper, sr.index = 0L, number.of.spectra = 100L)
```

Arguments

`jwrapper`	an open Wrapper object from Omnidriver
`sr.index`	an index to address the spectrometer
`number.of.spectra`	integer The number of spectra to be acquired.

Value

a numeric value

See Also

Other high speed acquisition functions: `get_seconds_time_delta()`, `get_seconds_time_delta_since()`, `highSpdAcq_get_number_of_spectra_acquired()`, `highSpdAcq_get_spectrum()`, `highSpdAcq_get_time_stamp()`, `highSpdAcq_start_acquisition()`, `oo_timestamp_to_string()`

---

highSpdAcq_get_number_of_spectra_acquired

Get the number of spectra acquired at high speed

Description

The number of spectra acquired can be smaller than the intended one, and this function retrieved the true number of spectra available in the buffer after the acquisition has finished. A call to this function can be used after a call to `highSpdAcq_start_acquisition()` returns.

Usage

```
highSpdAcq_get_number_of_spectra_acquired(jwrapper)
```

Arguments

`jwrapper`	an open Wrapper object from Omnidriver

Value

a numeric value

See Also

Other high speed acquisition functions: `get_seconds_time_delta()`, `get_seconds_time_delta_since()`, `highSpdAcq_allocate_buffer()`, `highSpdAcq_get_spectrum()`, `highSpdAcq_get_time_stamp()`, `highSpdAcq_start_acquisition()`, `oo_timestamp_to_string()`

highSpdAcq_get_spectrum *Get one spectrum acquired at high speed*

Description

Retrieve a single, previously measured, spectrum from the buffer. A call to this function can be used after a call to highSpdAcq_start_acquisition() returns.

Usage

```
highSpdAcq_get_spectrum(jwrapper, spectrum.number = 1)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
spectrum.number	integer The index into the sequence of spectra most recently acquired at high speed and available in the memory buffer.

Value

a numeric vector

See Also

Other high speed acquisition functions: get_seconds_time_delta(), get_seconds_time_delta_since(), highSpdAcq_allocate_buffer(), highSpdAcq_get_number_of_spectra_acquired(), highSpdAcq_get_time_stamp(), highSpdAcq_start_acquisition(), oo_timestamp_to_string()

highSpdAcq_get_time_stamp *Get the time stamp of one spectrum acquired at high speed*

Description

Retrieve the time stamp for a single, previously measured, spectrum from the buffer. A call to this function can be used after a call to highSpdAcq_start_acquisition() returns.

Usage

```
highSpdAcq_get_time_stamp(jwrapper, spectrum.number = 1)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
spectrum.number	integer The index into the sequence of spectra most recently acquired at high speed and available in the memory buffer.

Value

A HighResTimeStamp object.

See Also

Other high speed acquisition functions: get_seconds_time_delta(), get_seconds_time_delta_since(), highSpdAcq_allocate_buffer(), highSpdAcq_get_number_of_spectra_acquired(), highSpdAcq_get_spectrum(), highSpdAcq_start_acquisition(), oo_timestamp_to_string()

---

highSpdAcq_start_acquisition

Start high speed acquisition

Description

Acquires a series of spectra into a preallocated buffer. A call to this function should be always preceded by a call to highSpdAcq_allocate_buffer().

Usage

```
highSpdAcq_start_acquisition(jwrapper, sr.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer

Value

a numeric value

See Also

Other high speed acquisition functions: get_seconds_time_delta(), get_seconds_time_delta_since(), highSpdAcq_allocate_buffer(), highSpdAcq_get_number_of_spectra_acquired(), highSpdAcq_get_spectrum(), highSpdAcq_get_time_stamp(), oo_timestamp_to_string()

`init_api` *Initialize the connection to the driver*

Description

Function `init_api()` (and its synonim `init_srs()`) create a Wrapper object for calling Omnidriver functions. This Java object of class `com.oceanoptics.omnidriver.api.wrapper.Wrapper` is later used by all other functions in the rOmniDriver to communicate with the spectrometers through the OmniDriver API from Ocen Insight.

Only one wrapper object can be active at a time. You should call this function only once and save the returned object as it is used to access the methods implementing communication with the spectrometer. Clean up by use of `close_srs()` before exiting or before initializing another wrapper to the same driver.

Function `is_api_enableb()` tests if the argument passed to `jwrapper` has been initialized and if it is connected to an active instance of the Java wrapper class.

Function `init_api_extensions()` enables API extensions and returns a wrapper that makes it possible to call additional functions.

Function `is_extended_api_enabled()` tests if the argument passed to `jwrapperext` has been initialized and if it is connected to an active instance of the Java wrapper class.

Function `init_api_extensions()` enables API extensions and returns a wrapper that makes it possible to call additional functions.

Function `is_highres_time_api_enabled()` tests if wrapper has been initialized and saved to a private environment of the package.

Usage

```
init_api()

init_srs()

is_api_enabled(jwrapper)

init_extended_api()

is_extended_api_enabled(jwrapperext)

init_highres_time_api()

is_highres_time_api_enabled()
```

Arguments

`jwrapper`	jobjRef A Java Wrapper object from OmniDriver.
`jwrapperext`	jobjRef An "extension" Java Wrapper object from OmniDriver API.

Value

a wrapper to an instance of a Java object of class "wrapper"

logical

logical

`is_channel_enabled` *Informs whether a channel is enbled or not*

Description

Informs whether a channel of a multichannel spectrometer is enbled or not.

Usage

```
is_channel_enabled(jwrapper, sr.index = 0L, ch.index = 0L)
```

Arguments

`jwrapper`	an open Wrapper object from Omnidriver
`sr.index`	an index to address the spectrometer
`ch.index`	an index to address the channel in a spectrometer with more than one channel.

Value

a numeric value

`is_feature_supported_board_temperature`
Feature "Board temperature"

Description

Check whether feature "board temperature" is available in the spectrometer addressed. Construct a wrapper on the Java controller for the feature. Use a wrapped controller to query the temperature.

Usage

```
is_feature_supported_board_temperature(jwrapper, sr.index = 0L)

get_feature_controller_board_temperature(jwrapper, sr.index = 0L)

get_board_temperature(brd.wrapper)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer
brd.wrapper	an open controller wrapper object obtained with get_feature_controller_board_temperature()

Value

a numeric value

an object which provides the desired interface, or null if this feature is not available for this spectrometer.

Note

Before calling get_feature_controller_board_temperature you f irst need to check that the feature is supported by the spectrometer in use by calling is_feature_supported_board_temperature.

---

is_feature_supported_detector_temperature

Feature "detector temperature"

Description

Check whether feature "detector temperature" is available in the spectrometer addressed. Construct a wrapper on the Java controller for the feature. Use a wrapped controller to query the detector temperature.

Usage

```
is_feature_supported_detector_temperature(jwrapper, sr.index = 0L)

get_feature_detector_temperature(jwrapper, sr.index = 0L)

get_detector_temperature(dtr.wrapper)
```

Arguments

jwrapper	an open Wrapper object from OmniDriver
sr.index	an index to address the spectrometer
dtr.wrapper	an open detector wrapper object obtained with get_feature_detector_temperature()

Value

a numeric value

an object which provides the desired interface, or null if this feature is not available for this spectrometer.

Note

Before calling `get_feature_detector_temperature` you first need to check that the feature is supported by the spectrometer in use#' by calling `is_feature_supported_detector_temperature`.

`is_feature_supported_I2C_bus`
Is feature "I2C Bus" supported?

Description

Checks whether feature "I2C Bus" is available in the spectrometer addressed.

Usage

```
is_feature_supported_I2C_bus(jwrapper, sr.index = 0L)
```

Arguments

`jwrapper`	an open Wrapper object from Omnidriver
`sr.index`	an index to address the spectrometer

Value

a logical value

See Also

Other Spectrometer I2C- and SPI-bus functions.: `get_I2C_bus()`, `get_SPI_bytes()`, `get_feature_I2C_bus()`, `get_feature_SPI_bus()`, `is_feature_supported_SPI_bus()`

`is_feature_supported_internal_trigger`
Feature "ìnternal trigger"

Description

Check whether feature "ìnternal trigger" is available in the spectrometer addressed. Retrieve an interface that allows to control the ìnternal trigger. Control internal trigger of light sources such as Jaz PX.

Usage

```
is_feature_supported_internal_trigger(jwrapper, sr.index = 0L)

get_feature_controller_internal_trigger(jwrapper, sr.index = 0L)

get_trigger_period_valid_range(tgr.wrapper)

get_trigger_period(tgr.wrapper)

set_trigger_period(tgr.wrapper, period)

set_trigger_source(tgr.wrapper, source)

get_trigger_source(tgr.wrapper, char.out = TRUE)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer
tgr.wrapper	feature wrapper as returned by function rOmniDriver::get_feature_controller_internal_trigger
period	numeric Internal trigger period in seconds.
source	character, "on", "off", "external", "internal" or integer [0..3] or logical.
char.out	logical If TRUE return a character string, otherwise return integer as returned by the API call.

Value

an integer value

an object which provides the desired interface, or null if this feature is not available for this spectrometer.

numeric vector of length three with times in seconds for "min", "max" and "step".

Note

Before calling get_feature_controller_internal_trigger you first need to check that the feature is supported by the spectrometer in use by calling is_feature_supported_internal_trigger.

---

is_feature_supported_irradiance_calibration_factor

Feature "Irradiance Calibration Factor"

Description

Check whether feature "Irradiance Calibration Factor" is available in the spectrometer addressed. Construct a wrapper on the Java controller for the feature. Use a wrapped controller to retrieve the calibration factors.

Usage

```
is_feature_supported_irradiance_calibration_factor(jwrapper, sr.index = 0L)

get_feature_irradiance_calibration_factor(jwrapper, sr.index = 0L)

get_irradiance_calibration_factors(irrad.cal.wrapper)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer
irrad.cal.wrapper	An open wrapper for the feature object.

Value

a logical value

A wrapper on a Java object which provides the interface, or null if this feature is not available for this spectrometer.

A numeric vector of calibration factors, as returned by the spectrometer.

Note

Before calling `get_feature_irradiance_calibration_factor` you first need to check that the feature is supported by the spectrometer in use by calling `is_feature_supported_irradiance_calibration_factor`.

An attempt to retrieve calibration factors may fail and not return unless a USB timeout is set with function `set_USB_timeout()`.

---

is_feature_supported_pixel_binning

Is feature "pixel binning" supported?

Description

Checks whether feature "pixel binning" is available in the spectrometer addressed.

Usage

```
is_feature_supported_pixel_binning(jwrapper, sr.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer

Value

a numeric value

---

is_feature_supported_saturation_threshold

Is feature "saturation threshold" supported?

Description

Checks whether feature "saturation threshold" is available in the spectrometer addressed.

Usage

```
is_feature_supported_saturation_threshold(jwrapper, sr.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer

Value

a logical? value

---

is_feature_supported_spectrum_type

Is feature "spectrum type" supported?

Description

Checks whether feature "spectrum type" is available in the spectrometer addressed.

Usage

```
is_feature_supported_spectrum_type(jwrapper, sr.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer

Value

a logical value

is_feature_supported_SPI_bus *Is feature "SPI Bus" supported?*

Description

Checks whether feature "SPI Bus" is available in the spectrometer addressed.

Usage

```
is_feature_supported_SPI_bus(jwrapper, sr.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer

Value

a logical value

See Also

Other Spectrometer I2C- and SPI-bus functions.: get_I2C_bus(), get_SPI_bytes(), get_feature_I2C_bus(), get_feature_SPI_bus(), is_feature_supported_I2C_bus()

is_saturated *Is the most recent spectrum acquired saturated?*

Description

Checks whether the most recently acquired spectrum is saturated (= signal clipping occured).

Usage

```
is_saturated(jwrapper, sr.index = 0L, ch.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer
ch.index	an index to address the channel in a spectrometer with more than one channel.

Value

a numeric value

Note

Be aware that clipping outside the wavelength range of interest for a given measuremt may not be important and can be ignored. This method tests for clipping at any pixel of the array.

---

is_spectrum_valid *Is the most recent spectrum acquired valid?*

Description

Informs whether the most recently acquired spectrum is valid (no communication or similar errors have occurred during acquisition).

Usage

```
is_spectrum_valid(jwrapper, sr.index = 0L, ch.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer
ch.index	an index to address the channel in a spectrometer with more than one channel.

Value

a numeric value

---

is_timeout *Did the last operation time out?*

Description

Checks whether the last operation has timed out. This is useful when working with triggers together with signals used for triggering that may fail or be delayed.

Usage

```
is_timeout(jwrapper, sr.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer

Value

a numeric value

is_USB_timeout *Did the last USB operation time out?*

Description

Checks whether the last USB operation has timed out. This is useful when working with triggers together with signals used for triggering that may fail or be delayed.

Usage

```
is_USB_timeout(jwrapper, sr.index = 0L, ch.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer
ch.index	an index to address the channel in a spectrometer with more than one channel.

Value

a numeric value

oo_timestamp_to_string *Convert time stamp to string*

Description

Convert a time stamp into a character string. Times with millisecond resolution and nanosecond resolutions are separate, with the nanosecond component useful for computing time differences only.

Usage

```
oo_timestamp_to_string(jwrapper, oo_timestamp)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
oo_timestamp	HighResTimeStamp defined in OmniDriver API and retrieved or constructed with other OmniDriver API methods. Not vectorized!

Value

a character string.

Note

Nanosecond timing and normal timing are not coincident.

See Also

Other high speed acquisition functions: `get_seconds_time_delta()`, `get_seconds_time_delta_since()`, `highSpdAcq_allocate_buffer()`, `highSpdAcq_get_number_of_spectra_acquired()`, `highSpdAcq_get_spectrum()`, `highSpdAcq_get_time_stamp()`, `highSpdAcq_start_acquisition()`

`open_all_spectrometers` *Opens connections to all spectrometers*

Description

Opens connections to communicate with all Ocean Insight spectrometers connected through USB and returns the number of spectrometers that can be addressed.

Usage

```
open_all_spectrometers(jwrapper)

number_srs(jwrapper)
```

Arguments

`jwrapper`	an open Wrapper object from Omnidriver

Value

A numeric value giving the number of USB-connected spectrometers.

`set_boxcar_width` *Set "boxcar width"*

Description

Set setting "boxcar width" (the number of adjacent pixels averaged)

Usage

```
set_boxcar_width(jwrapper, half.width.px = 0L, sr.index = 0L, ch.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
half.width.px	the number of pixels in one half of the bocar window (an integer number)
sr.index	an index to address the spectrometer
ch.index	an index to address the channel in a spectrometer with more than one channel.

Value

a numeric value

Note

The signature of this function is different to that of the Java equivalent, so that as with other functions in the package indexes for addressing spectrometer and channel can have default values.

---

set_correct_for_detector_nonlinearity

Set "correct for detector nonlinearity"

Description

Set setting "correct for detector nonlinearity" for the addressed spectrometer.

Usage

```
set_correct_for_detector_nonlinearity(
  jwrapper,
  correct = 0L,
  sr.index = 0L,
  ch.index = 0L
)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
correct	integer or logical; 0, FALSE: apply correction; 1, TRUE: do not apply correction.
sr.index	an index to address the spectrometer
ch.index	an index to address the channel in a spectrometer with more than one channel.

Value

a logical value that the user SHOULD test as the method may fail.

Note

The signature of this function is different to that of the Java equivalent, so that as with other functions in the package indexes for addressing spectrometer and channel can have default values.

`set_correct_for_electrical_dark` *Set "correct for electrical dark signal"*

Description

Set setting "correct for electrical dark signal" for the addressed spectrometer channel

Usage

```
set_correct_for_electrical_dark(
  jwrapper,
  enable = 0L,
  sr.index = 0L,
  ch.index = 0L
)
```

Arguments

`jwrapper`	an open Wrapper object from Omnidriver
`enable`	integer or logical; 0, FALSE: disable; 1, TRUE: enable.
`sr.index`	an index to address the spectrometer
`ch.index`	an index to address the channel in a spectrometer with more than one channel.

Value

a logical value

Note

The signature of this function is different to that of the Java equivalent, so that as with other functions in the package indexes for addressing spectrometer and channel can have default values.

set_integration_time *Set "integration time"*

Description

Set "integration time" for the addressed spectrometer channel

Usage

```
set_integration_time(jwrapper, time.usec = 100L, sr.index = 0L, ch.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
time.usec	integartion time in microseconds (an integer)
sr.index	an index to address the spectrometer
ch.index	an index to address the channel in a spectrometer with more than one channel.

Value

a numeric value

Note

The signature of this function is different to that of the Java equivalent, so that as with other functions in the package indexes for addressing spectrometer and channel can have default values.

set_scans_to_average *Set "number of scans to average"*

Description

Set "number of scans to average"

Usage

```
set_scans_to_average(jwrapper, n.scans = 1L, sr.index = 0L, ch.index = 0L)

set_scans_to_avg(jwrapper, n.scans = 1L, sr.index = 0L, ch.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
n.scans	number of scans (an integer)
sr.index	an index to address the spectrometer
ch.index	an index to address the channel in a spectrometer with more than one channel.

Value

a numeric value

Note

The signature of this function is different to that of the Java equivalent, so that as with other functions in the package indexes for addressing spectrometer and channel can have default values.

---

set_spectrum_type *Set "spectrum type"*

Description

Set the spectrum type to one of normal or raw.

Usage

```
set_spectrum_type(spct.type.wrapper, spct.type = "normal")
```

Arguments

spct.type.wrapper	feature wrapper as returned by function rOmniDriver::get_feature_controller_internal_trigger
spct.type	character, "normal" or "raw" or interger, 0 or 1.

Value

logical TRUE = success, FALSE = failure.

---

set_timeout *Set (trigger) "timeout"*

Description

Setting a "timeout" for the addressed spectrometer is useful when used together with triggers to avoid endless waiting in case the trigger input does not take place.

Usage

```
set_timeout(jwrapper, time.millisec = 1000L, sr.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
time.millisec	time to wait in milliseconds (an integer)
sr.index	an index to address the spectrometer

Value

a numeric value

Note

This method is applied to all channels.

`set_USB_timeout` *Set USB "timeout"*

Description

Setting a "timeout" for the addressed spectrometer is useful when used together with triggers to avoid endless waiting. This function is supported only by USB spectrometers. A time of 0L disables the timeout.

Usage

```
set_USB_timeout(jwrapper, time.millisec = 1000L, sr.index = 0L)
```

Arguments

`jwrapper`	an open Wrapper object from Omnidriver
`time.millisec`	time to wait in milliseconds (an integer)
`sr.index`	an index to address the spectrometer

Value

a numeric value

Note

This method is applied to all channels.

spectrum_flush *Flush the most recent spectrum*

Description

Flush the most recent spectrum acquired or currently being acquired

Usage

```
spectrum_flush(jwrapper, sr.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer

Value

a numeric value

stop_averaging *Stop averaging of scans*

Description

Stop averaging for addressed spectrometer probably not very useful in "vanilla" R as this aborts the current averaging of spectra, but needs a multithreaded application to be able to call it before get.spectrum returns.

Usage

```
stop_averaging(jwrapper, sr.index = 0L, ch.index = 0L)
```

Arguments

jwrapper	an open Wrapper object from Omnidriver
sr.index	an index to address the spectrometer for the time being not exported
ch.index	an index to address the channel in a spectrometer with more than one channel.

Value

a numeric value

Note

The signature of this function is different to that of the Java equivalent, so that as with other functions in the package indexes for addressing spectrometer and channel can have default values.

Index