O2x5xx Sensors Library for Python 3.x

Table of Contents

Description

A Python 3 library for ifm efector O2x5xx 2D sensors (O2D5xx / O2I5xx).

Contact

In case of any issues or if you want to report a bug please contact our support team.

Features

- PCIC V3 client for result data transfer
- O2X5xxDevice client for PCIC command usage

Prerequisites

Usage of examples requires packages listed in the requirements.txt file. Install the package with

```
$ pip install -r requirements.txt
```

Installation

Install the package with

```
$ python setup.py install
```

Examples

For a quick start, go to the `examples` folder and run

```
$ python output_recorder.py 192.168.0.69 myFile.txt 3600
```

with your device's IP address to record the asynchronous PCIC output for 3600 seconds and save the output into myFile.txt

```
$ python image_viewer.py 192.168.0.69
```

to view the image(s) data coming from the camera (requires matplotlib). Each image will be show in an own window.

Usage

PCIC client

The library currently provides three basic clients:

A simple PCIC V3 client

- Create it with `pcic = o3d3xx.PCICV3Client("192.168.0.69", 50010)` providing the device's address and PCIC port.
- Send PCIC commands with e.g. `answer = pcic.sendCommand("G?")` . All asnychronous PCIC messages are discarded while waiting for the answer to the command.
- Read back the next PCIC for a particular ticket number. This can be used to read asynchronously sent results (ticket number "0000"):

  ```
  answer = pcic.readAnswer("0000")
  ```

- Read back any answer coming from the device:
  `ticket, answer = pcic.readNextAnswer()`

A simple O2x5xx client (inheriting PCIC V3 client)

- Create it with `device = o2x5xx.O2x5xxDevice("192.168.0.69", 50010)` providing the device's address and PCIC port.
- Send PCIC commands wrapped into functions with e.g. `answer = device.occupancy_of_application_list()` . All asynchronous PCIC messages are discarded while waiting for the answer to the command.
- Upload PCIC configurations with e.g. `device = o2x5xx.O2x5xxDevice("192.168.0.69", 50010)` The PCIC configuration is valid for the instanced device (session).
- Complete function documentation as docstring.

A PCIC client for asynchronous image retrieval (inheriting O2x5xx client)

- Create it with `image_viewer = o2x5xx.ImageClient("192.168.0.69", 50010)` .
- It configures a PCIC connection to receive all images from the application.
- Read back the next result (a list with header information and dictionary containing all the images) with `result = pcic.readNextFrame()`
- Read back the next result (a list with header information and all images with datatype numpy.ndarray) with `result = pcic.readNextFrame()`

Function Description

For a more detailed explanation of the function take a look on the docstring documentation for each function.

class Client

recv(number_bytes)

```
Read the next bytes of the answer with a defined length.

:param number_bytes: (int) length of bytes
:return: the data as bytearray
```

close()

```
Close the socket session with the device.

:return: None
```

class PCICV3Client (inheriting class Client)

read_next_answer()

```
Read next available answer.

:return: None
```

read_answer(ticket)

```
Read the next available answer with a defined ticket number.

:param ticket: (string) ticket number
:return: answer of the device as a string
```

send_command(cmd)

```
Send a command to the device with 1000 as default ticket number. The length and syntax
of the command is calculated and generated automatically.

:param cmd: (string) Command which you want to send to the device.
:return: answer of the device as a string
```

class O2x5xxDevice (inheriting class PCICV3Client)

activate_application(number)

```
Activates the selected application.

:param number: (int) 2 digits for the application number as decimal value
:return: - * Command was successful
         - ! Application not available
           | <application number> contains wrong value
           | External application switching activated
           | Device is in an invalid state for the command, e.g. configuration mode
         - ? Invalid command length
```

occupancy_of_application_list()

```
Requests the occupancy of the application list.

:return: Syntax: <amount><t><number active application><t> ... <number><t><number>
         e.g. 015    15  01  02     03      04      05      06      07      08      09      10
         - <amount> char string with 3 digits for the amount of applications
           saved on the device as decimal number
         - <t> tabulator (0x09)
```

```
          - <number active application> 2 digits for the active application
          - <number> 2 digits for the application number
          - ! Application not available
            | <application number> contains wrong value
            | External application switching activated
            | Device is in an invalid state for the command, e.g. configuration mode
          - ? Invalid command length
```

upload_pcic_output_configuration(config)

```
    Uploads a Process interface output configuration lasting this session.

    :param config: (dict) configuration data
    :return: - * Command was successful
             - ! Error in configuration
               | Wrong data length
             - ? Invalid command length
```

retrieve_current_process_interface_configuration()

```
    Retrieves the current Process interface configuration.

    :return: Syntax: <length><configuration>
             - <length> 9 digits as decimal value for the data length
             - <configuration> configuration data
             - ? Invalid command length
```

request_current_error_state()

```
    Requests the current error state.

    :return: Syntax: <code>
             - <code> Error code with 8 digits as a decimal value. It contains leading zeros.
             - ! Invalid state (e.g. configuration mode)
             - ? Invalid command length
             - $ Error code unknown
```

request_current_error_state_decoded()

```
    Requests the current error state and error message as a tuple.

    :return: Syntax: [<code>,<error_message>]
             - <code> Error code with 8 digits as a decimal value. It contains leading zeros.
             - <error_message> The corresponding error message to the error code.
             - ! Invalid state (e.g. configuration mode)
             - ? Invalid command length
             - $ Error code unknown
```

gated_software_trigger_on_or_off(state)

```
Turn gated software trigger on or off.

:param state: (int) 1 digit
              "0": turn gated software trigger off
              "1": turn gated software trigger on
:return: - * Trigger could be executed
         - ! Invalid argument, invalid state, trigger already executed
         - ? Something else went wrong
```

request_device_information()

```
Requests device information.

:return: Syntax:
         <vendor><t><article number><t><name><t><location><t>
         <description><t><ip><subnet mask><t><gateway><t><MAC><t>
         <DHCP><t><port number>
         - <vendor>            IFM ELECTRONIC
         - <t>                 Tabulator (0x09)
         - <article number>    e.g. O2D500
         - <name>              UTF8 Unicode string
         - <location>          UTF8 Unicode string
         - <description>       UTF8 Unicode string
         - <ip>                IP address of the device as ASCII character sting e.g. 192.168.
         - <port number>       port number of the XML-RPC
         - <subnet mask>       subnet mask of the device as ASCIIe.g. 192.168.0.69
         - <gateway>           gateway of the device as ASCIIe.g 192.168.0.69
         - <MAC>               MAC address of the device as ASCIIe.g. AA:AA:AA:AA:AA:AA
         - <DHCP>              ASCII string "0" for off and "1" for on
```

return_a_list_of_available_commands()

```
Returns a list of available commands.

:return: - H?                       show this list
         - t                        execute Trigger
         - T?                       execute Trigger and wait for data
         - g<state>              turn gated software trigger on or off
         - o<io-id><io-state>    set IO state
         - O<io-id>?             get IO state
         - I<image-id>?          get last image of defined type
         - A?                       get application list
         - p<state>              activate / deactivate data output
         - a<application number> set active application
         - E?                       get last Error
         - V?                       get current protocol version
         - v<version>            get protocol version
         - c<length of configuration file><configuration file>
                                    configure process data formatting
         - C?                       show current configuration
         - G?                       show device information
         - S?                       show statistics
```

```
        - L?                        retrieves the connection id
        - j<id><length><data> sets string data under specific ID
        - J<id>?                reads string defined under specific ID
        - d<on-off state of view indicator><duration> turn the view indicators on
                                    (permanently or for a defined time) or off
```

request_last_image_taken(image_id)

```
    Request last image taken.

    :param image_id: (int) 2 digits for the image type
                     1: all JPEG images
                     2: all uncompressed images
    :return: Syntax: <length><image data>
             - <length> (int) char string with exactly 9 digits as decimal number for the image da
             - <image data> (bytearray) image data / result data. The data is encapsulated in an i
             - ! No image available
               | Wrong ID
             - ? Invalid command length
```

request_last_image_taken_deserialized(image_id, datatype)

```
    Request last image taken deserialized in image header and image data. Image data can requested
    or decoded as ndarray datatype.

    :param image_id: (int) 2 digits for the image type
                     1: all JPEG images
                     2: all uncompressed images
    :param datatype: (str) image output as hex or ndarray datatype
                     bytes: image(s) as bytes datatype
                     ndarray: image(s) as ndarray datatype
    :return: Syntax: [<header>,<image data>]
             - <header> (dict) header of the image deserialized as dict object
             - <image data> image data / result data. The data is encapsulated
             in an image chunk if bytes as datatype is selected.
             - ! No image available
               | Wrong ID
             - ? Invalid command length
```

overwrite_data_of_a_string(container_id, data)

```
    Overwrites the string data of a specific (ID) string container used in the logic layer.

    :param container_id: (int) number from 00 to 09
    :param data: (string) string of a maximum size of 256 Bytes
    :return: - * Command was successful
             - ! Invalid argument or invalid state (other than run mode)
               | Not existing element with input-container-ID in logic layer
             - ? Syntax error
```

read_string_from_defined_container(container_id)

```
Read the current defined string from the defined input string container.
The string is represented as byte array.

:param container_id: (int) number from 00 to 09
:return: Syntax: <length><data>
         - <length>: 9 digits as decimal value for the data length
         - <data>: content of byte array
         - ! Invalid argument or invalid state (other than run mode)
           | Not existing element with input-container-ID in logic layer
         - ? Syntax error
```

return_the_current_session_id()

```
Returns the current session ID.

:return: 3 digits with leading "0"
```

set_logic_state_of_a_id(io_id, state)

```
Sets the logic state of a specific ID.

:param io_id: (int) 2 digits for digital output
              1: IO1
              "02": IO2
:param state: (int) 1 digit for the state
              "0": logic state low
              "1": logic state high
:return: Syntax: <IO-ID><IO-state>
         - <IO-ID> 2 digits for digital output
         "01": IO1
         "02": IO2
         - <IO-state> 1 digit for the state
         "0": logic state low
         "1": logic state high
         - ! Invalid state (e.g. configuration mode)
           | Wrong ID
           | Element PCIC Output not connected to DIGITAL_OUT element in logic layer
         - ? Invalid command length
```

request_state_of_a_id(io_id)

```
Requests the state of a specific ID.

:param io_id: 2 digits for digital output
              "01": IO1
              "02": IO2
:return: Syntax: <IO-ID><IO-state>
         - <IO-ID> 2 digits for digital output
         "01": IO1
         "02": IO2
         - <IO-state> 1 digit for the state
         "0": logic state low
```

```
          "1": logic state high
          - ! Invalid state (e.g. configuration mode)
            | Wrong ID
            | Element PCIC Output not connected to DIGITAL_OUT element in logic layer
          - ? Invalid command length
```

turn_process_interface_output_on_or_off(state)

```
    Turns the Process interface output on or off. Be aware that this modification only
    affects the own session and is not considered to be a global parameter.

    :param state: (int) 1 digit
                  0: deactivates all asynchronous output
                  1: activates asynchronous result output
                  2: activates asynchronous error output
                  3: activates asynchronous error and data output
                  4: activates asynchronous notifications
                  5: activates asynchronous notifications and asynchronous result
                  6: activates asynchronous notifications and asynchronous error output
                  7: activates all outputs
    :return: - * Command was successful
             - ! <state>: contains wrong value
             - ? Invalid command length
```

request_current_decoding_statistics()

```
    Requests current decoding statistics.

    :return: Syntax: <number of results><t><number of positive decodings><t><number of false
                     decodings>
             - <t> tabulator (0x09)
             - <number of results> Images taken since application start. 10 digits decimal value w
               leading "0"
             - <number of positive decodings> Number of decodings leading to a positive result. 10
               decimal value with leading "0"
             - <number of false decodings> Number of decodings leading to a negative result. 10 di
               decimal value with leading "0"
             - ! No application active
```

execute_asynchronous_trigger()

```
   Executes trigger. The result data is send asynchronously.
    Only compatible with configured trigger source "Process Interface" on the sensor.

    :return: - * Trigger was executed, the device captures an image and evaluates the result.
             - ! Device is busy with an evaluation
               | Device is in an invalid state for the command, e.g. configuration mode
               | Device is set to a different trigger source
               | No active application
```

execute_synchronous_trigger()

```
Executes trigger. The result data is send synchronously.
Only compatible with configured trigger source "Process Interface" on the sensor.

:return: - (str) decoded data output of process interface
         - ! Device is busy with an evaluation
           | Device is in an invalid state for the command, e.g. configuration mode
           | Device is set to a different trigger source
           | No active application
```

set_current_protocol_version(version)

```
Sets the current protocol version. The device configuration is not affected.

:param version: 2 digits for the protocol version. Only protocol version V3 is supported.
:return: - * Command was successful
         - ! Invalid version
         - ? Invalid command length
```

request_current_protocol_version()

```
Requests current protocol version.

:return: Syntax: <current version><empty><min version><empty><max version>
         - <current version> 2 digits for the currently set version
         - <empty> space sign 0x20
         - <min/max version> 2 digits for the available min and max version that can be set
```

turn_state_of_view_indicator_on_or_off(state, duration)

```
Turn the view indicators on (permanently or for a defined time) or off.

Syntax: d<on-off state of view indicator><duration>

:param state: (int) duration time in seconds
              0: turn the view indicators off
              1: turn the view indicators on
:param duration: (int) duration time in seconds (parameter has no impact if you turn indicator
:return: - * Command was successful
         - ! Invalid state (e.g. configuration mode)
           | Wrong state
           | Wrong duration
         - ? Invalid command length
```

class ImageClient (inheriting class O2x5xxDevice)

number_images()

```
A function for which returns the number of images from application.

:return: (int) number of images
```

read_image_ids()

```
A function for reading the PCIC image output and parsing the image IDs.
The image IDs are stored in property self.image_IDs

:return: list of image ids
```

read_next_frames()

```
Function for reading next asynchronous frames.
Frames are stored in property self.frames

:return: None
```

make_figure()

```
Function for making figure object and using parsed image ID as subtitle.

:param idx: (int) list index of self.image.IDs property
                        e.g. idx == 0 would read string value '2' from self.image.IDs = ['

:return: Syntax: [<fig>, <ax>, <im>]
                                  - <fig>: matplotlib figure object
                                  - <ax>: AxesSubplot instance of figure object
                                  - <im>: AxesImage instance of figure object
```

Unit Tests

FW version: 1.22.9323

O2D5xx	O2I5xx	tested function	comment
x	x	activate_application	-
x	x	application_list	-

O2D5xx	O2I5xx	tested function	comment
x	x	upload_pcic_output_configuration	-
x	x	retrieve_current_process_interface_configuration	-
x	x	request_current_error_state	-
x	x	request_current_error_state_decoded	-
x	x	gated_software_trigger_on_or_off	-
x	x	request_device_information	-
x	x	return_a_list_of_available_commands	-
x	x	request_last_image_taken	uncompressed images not available due to high data size with 5 images
x	x	request_last_image_taken_deserialized	-
x	x	overwrite_data_of_a_string	-
x	x	read_string_from_defined_container	-
x	x	return_the_current_session_id	-
x	x	set_logic_state_of_a_id	-
x	x	request_state_of_a_id	Reading io states requires element in logic layer
x	x	turn_process_interface_output_on_or_off	-
x	x	request_current_decoding_statistics	-
x	x	execute_asynchronous_trigger	-
x	x	execute_synchronous_trigger	-
x	x	set_current_protocol_version	-
x	x	request_current_protocol_version	-
x	x	turn_state_of_view_indicator_on_or_off	only available for O2I5xx sensors

FW version: 1.22.9009

O2D5xx	O2I5xx	tested function	comment

O2D5xx	O2I5xx	tested function	comment
x	x	activate_application	-
x	x	application_list	-
x	x	upload_pcic_output_configuration	-
x	x	retrieve_current_process_interface_configuration	-
x	x	request_current_error_state	-
x	x	request_current_error_state_decoded	-
x	x	gated_software_trigger_on_or_off	-
x	x	request_device_information	-
x	x	return_a_list_of_available_commands	-
x	x	request_last_image_taken	uncompressed images not available due to high data size with 5 images
x	x	request_last_image_taken_deserialized	-
x	x	overwrite_data_of_a_string	-
x	x	read_string_from_defined_container	-
x	x	return_the_current_session_id	-
x	x	set_logic_state_of_a_id	-
x	x	request_state_of_a_id	Reading io states requires element in logic layer
x	x	turn_process_interface_output_on_or_off	-
x	x	request_current_decoding_statistics	-
x	x	execute_asynchronous_trigger	-
x	x	execute_synchronous_trigger	-
x	x	set_current_protocol_version	-
x	x	request_current_protocol_version	-
x	x	turn_state_of_view_indicator_on_or_off	only available for O2I5xx sensors

Source README.md Styleguide

https://github.com/amontalenti/elements-of-python-style/blob/master/README.md