OCI L1a

Release 0.1.0

Christopher Field

Nov 02, 2020

CONTENTS:

CHAPTER

ONE

METHODS DEFINED BY READ_OCI_1A_CLASS

This module provides a class to read PACE OCI L1a spatial/spectral data from one or more netCDF4 files. It also provides a method to return the data spectral bands.

For a description of the L1A file format see: OCI L1A data format TDMS Document number **OCI-SCI-SPEC-0079**.

A more descriptive document on the L1A format is: PACE OCI Level-1A Data Product User's Guide by Fred Patt

For information on the thermal variables in the L1A files, see: OCI-THRM-SPEC-0108 and **OCI-SYS-TN-0270**

read_oci_1a_class.**CCD_DIM = 512**
Number of pixels in each direction of the CCD. Constant.

class read_oci_1a_class.**READ_OCI_SPA_SPE_L1A**(*path=None*, *mask_dark=False*)
Class to read the OCI l1a spatial/spectral data and assign spectral and pixel labels.

Parameters

- **path** – File path, either a Path object or string. Default (None)
- **mask_dark** (*bool*) – Set True to mask dark data in returned CCD image. Default is False.

If **path** is None, then an empty object is created to which files can be appended by using the **append** method.

The goal of class is to create data attributes of the OCI RED CCD output and BLUE CCD output. The CCD outputs are **N_s** x **N_w** x **N_p** arrays, where **N_s** is the number of scans, **N_w** is the number of spectral bands (**N_wr** for the red CCD and **N_wb** for the blue CCD), and **N_p** is the number of spatial pixels.

Data attributes created by this class fall into two groups (explained below) and are:

Group 1:

1. **aux_param_table**: The aux_param_table read from the file.
2. **blue_spectral_mode**: The blue_spectral_mode values read from the file.
3. **red_spectral_mode**: The red_spectral_mode values read from the file.
4. **spatial_aggregation**: The spatial_aggregation values read from the file.
5. **spatial_zone_data_type**: The spatial_zone_data_type values read from the file.
6. **spatial_zone_lines**: The spatial_zone_lines values read from the file.
7. **red_wavelengths**: The center wavelength of each red CCD spectral band (nm) (Length N_wr).
8. **blue_wavelengths**: The center wavelength of each blue CCD spectral band (nm) (Length N_wb).
9. **pixel_angle**: The scan angle from the start marker of each spatial pixel (deg) (Length N_p).

10. **pixel_data_type**: The spatial zone type of each spatial pixel (unitless) (Length N_p).

Group 2:

1. **pixel_data_type_red**: Same as pixel_data_type broadcast to shape **N_s** x **N_wr** x **N_p**. Note that due to the magic of numpy indexing this array uses only the memory pixel_data_type does.
2. **pixel_data_type_blue**: Same as pixel_data_type broadcast to shape **N_s** x **N_wr** x **N_p**. Note that due to the magic of numpy indexing this array uses only the memory as pixel_data_type does.
3. **DC_swir_data**: The dark count data read from DC_swir (**N_s** x **N_wr** x **N_d**)?, where **N_d** is the number of dark pixels recorded.
4. **DC_red_data**: The dark count data read from DC_red (**N_s** x **N_wr** x **N_d**), where **N_d** is the number of dark pixels recorded. Values have been divided by 16.
5. **DC_blue_data**: The dark count data read from DC_blue (**N_s** x **N_wb** x **N_d**). Values have been divided by 16.
6. **Sci_swir_data**: The raw counts read from sci_swir (**N_s** x **N_wr** x **N_d**)?. If **mask_dark** was True, then any pixels not in zone 9 are masked. (NOT YET.)
7. **Sci_red_data**: The raw counts read from sci_red (**N_s** x **N_wr** x **N_d**). If **mask_dark** was True, then any pixels not in zone 9 are masked.
8. **Sci_blue_data**: The raw counts read from sci_blue (**N_s** x **N_wb** x **N_d**). If **mask_dark** was True, then any pixels not in zone 9 are masked.
9. **scan_start_time**: The values read from scan_start_time (**N_s**).
10. **scan_start_CCSDS_sec**: The values read from scan_start_CCSDS_sec (**N_s**).
11. **scan_start_CCSDS_usec**: The values read from scan_start_CCSDS_usec (**N_s**).
12. **scan_start_TIA**: Scan line start time in TAI (int64 in us) (**N_s**).

In addition, **path** is defined, the path to a single input file or a list of paths to each input file when **append()** is used.

When data files are accumulated by the append method, the first 6 data attributes in group 1 are tested for a match between the previously read file and the current one. Any mismatch will raise a ValueError indicating which was the first one to not match. Data attributes in Group 1 are all taken from the current file and are always valid. However, when appending, data attributes in Group 2 are accumulated in lists and don't take the form of numpy masked arrays until after the **append_conclude()** method has been called.

Note: Some methods permanently modify the data attributes in place. To preserve a version of the original use **copy.deepcopy()**. For example:

```
from copy import deepcopy
oci.append(read_oci_1a_class(file))
oci2 = deepcopy(oci)
oci2.subtract_dark()
oci2 = deepcopy(oci)     # Reset oci2 to the original values.
```

---

append(*other*)

Append the CCD data from "other" to this object.

Parameters **other** – A READ_OCI_SPA_SPE_L1A object

Returns None

Raises ValueError if any of the spatial/spectral aggregation are different.

Typical use:

```
oci_ccd = READ_OCI_SPA_SPE_L1A()
for file in files:
    oci_ccd.append(READ_OCI_SPA_SPE_L1A(file))
oci_ccd.append_conclude()
```

append_conclude()
Convert lists accumulated by **append** into masked arrays in group 2.

Returns None

Until this method is called, the data attributes in Group 2 are not valid.

select__data_type(*data_type*)
Trim data attributes to those spatial pixels with the type **data_type**.

Parameters **data_type** – Scalar data type to keep in the trimmed data.

Returns None

Note: This method permanently modifies the data attributes in place.

subtract_dark(*start=40*, *dev=3*)
Average the dark counts for each scan and subtract from each scan

Parameters

- **start** – First index to use when computing the dark average (Default=40)
- **dev** – Multiple of sigma to mask when computing dark level (Default=3)

Returns

Note: This method permanently modifies **Sci_red_data** and **Sci_blue_data** in memory.

class read_oci_1a_class.**READ_OCI_THERMAL_L1A**(*path=None*, *mask_dark=False*)
Class to read the OCI l1a thermal data.

Parameters

- **path** – File path, either a Path object or string. Default (None)
- **mask_dark** (*bool*) – Set True to mask dark data in returned CCD image. Default is False.

If **path** is None, then an empty object is created to which files can be appended by using the **append** method.

Data attributes created by this class are:

1. **aob_temperatures**
2. **blue_fpa_temperatures**
3. **dau_temperatures**
4. **dau_tlm_time**
5. **icdu_temperatures**. Not implemented.
6. **red_fpa_temperatures**
7. **sds_det_temperatures**

8. **tc_tlm_time**

In addition, **path** is defined, the path to a single input file or a list of paths to each input file when **append()** is used.

append(*other*)
Append the CCD data from "other" to this object.

Parameters **other** – A read_oci_1a_class object

Returns None

Raises ValueError if any of the spatial/spectral aggregation are different.

Typical use:

```
oci_therm = READ_OCI_THERMAL_L1A()
for file in files:
    oci_therm.append(READ_OCI_THERMAL_L1A(file))
oci_therm.append_conclude()
```

append_conclude()
Convert lists accumulated by **append** into masked arrays in group 2.

Returns None

Until this method is called, the thermal numpy arrays are not defined.

plot(*variable*, *ax=None*, *plt_type='3d'*)
Method doesn't work. Just return

read_oci_1a_class.**getCCDbands3**(*spectral_mode*, *band*)
compute PACE OCI CCD wavebands for given input aggregation spectral_mode.

Parameters

- **spectral_mode** – vector of spectral aggregation factors, one for each tap
- **band** – "blue" or "red"

Returns output vector of aggregated pixel center wavelengths, max(spectral_mode)

Raises ValueError **band** is not one of the valid strings.

read_oci_1a_class.**get_CCD_pixel_type**(*spatial_zone_lines*, *spatial_aggregation*, *spatial_zone_data_type*, *num_spatial_pixels*)
Use spatial zone line, aggregation, and type to generate scanner angle for each pixel.

Parameters

- **spatial_zone_lines** – Values from file "spatial_spectral_modes"
- **spatial_aggregation** – Values from file "spatial_spectral_modes"
- **spatial_zone_data_type** – Values from file "spatial_spectral_modes"
- **num_spatial_pixels** – Number of spatial pixels in the scan (CCD pixels).

Returns 1darray pixel_angle, 1darray pixel_data_type, spatial aggregation

read_oci_1a_class.**get_pace_oci_l1a_files**(*path*, *pattern=''*)
Return from path a sorted list of PACE OCI L1A files with that match the pattern.

Parameters

- **path** – Path to a folder with OCI L1A files.
- **pattern** – Center of pattern to match

Returns Sorted list of files that match pattern.

If pattern == "" (default), the file match is made against "PACE_OCI_*.L1A.nc". If pattern starts with "/", the file match is made against pattern[1:]. Otherwise, the file match is made against "PACE_OCI_*" + pattern + "*.L1A.nc"

read_oci_1a_class.**main**(*args*)

Read the PACE OCI L1A netCDF files in args.paths and plot termperatures and spectrum.

Parameters **args** – Name space from argparse.

Returns None

CHAPTER

TWO

ETU_THERMAL

This file contains (some of) the conversations of OCI L1A thermal data to deg C.

Attempts to describe conversations for L1A engineering_data: AOB_temperatures, blue_FPA_temperatures, ICDU_thermisters, red_FPA_temperatures, and SDS_det_temperatures.

The temperature conversions are taken from **OCI-SYS-TN-0270_OCI Temperature Sensor List for K3 Temperature Calibration Revision G**, source file, **pace_oci_thermistor_table.txt** from Gene Eplee (21 October 2020, and code samples **read_pace_oci_icdu_temps.pro** and **read_pace_full_oci_temps.pro** by Gene Eplee. Note that these files don't currently include the **Vmeas,gnd** correction shown for some channels in the Excel file **OCI-SYS-TN-0270_OCI**.

Note that the **engineering_data** group contains three time variables: **DAU_tlm_time**, **DDC_tlm_time**, and **TC_tlm_time**. It is not clear to me which ones go with which data sets.

- uint16 blue_FPA_temperatures(tlm_packets, DAU_FPA_temps) 'DAU_FPA_temps': <class 'netCDF4._netCDF4.Dimension'>: name = 'DAU_FPA_temps', size = 14,
- uint16 red_FPA_temperatures(tlm_packets, DAU_FPA_temps) "
- uint16 SDS_det_temperatures(tlm_packets, DAU_SDS_det_temps) 'DAU_SDS_det_temps': <class 'netCDF4._netCDF4.Dimension'>: name = 'DAU_SDS_det_temps', size = 16,
- uint16 AOB_temperatures(tlm_packets, DAU_AOB_temps) 'DAU_AOB_temps': <class 'netCDF4._netCDF4.Dimension'>: name = 'DAU_AOB_temps', size = 9,
- uint16 DAU_temperatures(tlm_packets, DAU_temps) 'DAU_temps': <class 'netCDF4._netCDF4.Dimension'>: name = 'DAU_temps', size = 14,

Some references https://www.earthinversion.com/utilities/reading-NetCDF4-data-in-python/

https://www.unidata.ucar.edu/software/netcdf/documentation/NUG/_best_practices.html#bp_Conventions

ToDo. These conversions have not been checked that they generate correct temperatures.

etu_thermal.**DAU_FPA_temps**
: Dictionary of names (not yet defined) and conversions for dimension **DAU_FPA_temps**.

  Used by the variables **blue_FPA_temperatures** and **red_FPA_temperatures**. This is a dictionary with key "name" a list of sensor names and key "convert" a list of list of polynomial coefficients from highest to lowest power to convert volts to deg C (after conversion from raw counts to volts..

etu_thermal.**DAU_SDS_det_temps**
: Dictionary of names and conversions for dimension **DAU_SDS_det_temps**.

  Used by the variable **SDS_det_temperatures**. This is a dictionary with key "name" a list of sensor names and key "convert" a list of list of polynomial coefficients from highest to lowest power to convert volts to deg C (after conversion from raw counts to volts..

etu_thermal.**DAU_AOB_temps**
Dictionary of names and conversions for dimension **DAU_AOB_temps**.

Used by the variable **AOB_temperatures**. This is a dictionary with key "name" a list of sensor names and key "convert" a list of list of polynomial coefficients from highest to lowest power to convert volts to deg C (after conversion from raw counts to volts..

etu_thermal.**DAU_temps**
Dictionary of names and conversions for dimension **DAU_temps**.

Used by the variable **DAU_temperatures**. This is a dictionary with key "name" a list of sensor names and key "convert" a list of list of polynomial coefficients from highest to lowest power to convert volts to deg C (after conversion from raw counts to volts..

etu_thermal.**thermal_dims**
Dictionary of thermal channel dimensions. Each dimension previously defined in this module can be accessed by a string key of its name permitting looking it up from the netCDF variable data structure.

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX