

---

# **mdfreader Documentation**

***Release 2.8***

**Aymeric Rateau**

**Oct 31, 2018**



# CONTENTS

<b>1</b>	<b>mdfreader module documentation</b>	<b>3</b>
1.1	Platform and python version . . . . .	3
1.2	Dependencies . . . . .	3
1.3	mdfreader . . . . .	3
<b>2</b>	<b>mdf module documentation</b>	<b>11</b>
2.1	Dependencies . . . . .	11
2.2	mdf . . . . .	11
<b>3</b>	<b>mdf3reader module documentation</b>	<b>17</b>
3.1	mdf3reader . . . . .	17
<b>4</b>	<b>mdfinfo3 module documentation</b>	<b>23</b>
4.1	Dependencies . . . . .	23
4.2	mdfinfo3 . . . . .	23
<b>5</b>	<b>mdf4reader module documentation</b>	<b>25</b>
5.1	Dependencies . . . . .	25
5.2	mdf4reader . . . . .	25
<b>6</b>	<b>mdfinfo4 module documentation</b>	<b>33</b>
6.1	mdfinfo4 . . . . .	33
<b>7</b>	<b>channel module documentation</b>	<b>39</b>
<b>8</b>	<b>Indices and tables</b>	<b>47</b>
	<b>Python Module Index</b>	<b>49</b>
	<b>Index</b>	<b>51</b>



Contents:



## MDFREADER MODULE DOCUMENTATION

Measured Data Format file reader main module

### 1.1 Platform and python version

With Unix and Windows for python 2.6+ and 3.2+

**Author** Aymeric Rateau

Created on Sun Oct 10 12:57:28 2010

### 1.2 Dependencies

- Python >2.6, >3.2 <<http://www.python.org>>
- Numpy >1.6 <<http://numpy.scipy.org>>
- Sympy to convert channels with formula
- bitarray for not byte aligned data parsing
- Matplotlib >1.0 <<http://matplotlib.sourceforge.net>>
- NetCDF
- h5py for the HDF5 export
- xlwt for the excel export (not existing for python3)
- openpyxl for the excel 2007 export
- scipy for the Matlab file conversion
- zlib to uncompress data block if needed

`mdfreader.mdfreader.PythonVersion`

*float* – Python version currently running, needed for compatibility of both python 2.6+ and 3.2+

### 1.3 mdfreader

**class** `mdfreader.mdfreader.Mdf` (*file\_name=None, channel\_list=None, convert\_after\_read=True, filter\_channel\_names=False, no\_data\_loading=False, compression=False, convert\_tables=False, metadata=2*)

Bases: `mdfreader.mdf3reader.Mdf3`, `mdfreader.mdf4reader.Mdf4`

Mdf class

**fileName**

*str* – file name

**MDFVersionNumber**

*int* – mdf file version number

**masterChannelList**

*dict* – Represents data structure: a key per master channel with corresponding value containing a list of channels One key or master channel represents then a data group having same sampling interval.

**multiProc**

*bool* – Flag to request channel conversion multi processed for performance improvement. One thread per data group.

**file\_metadata**

*dict* – file metadata with minimum keys : author, organisation, project, subject, comment, time, date

**read**( *file\_name = None, multi\_processed = False, channel\_list=None, convert\_after\_read*

*filter\_channel\_names=False, no\_data\_loading=False, compression=False*)

reads mdf file version 3.x and 4.x

**write** (*file\_name=None*)

writes simple mdf file

**get\_channel\_data** (*channel\_name*)

returns channel numpy array

**convert\_all\_channel** ()

converts all channel data according to CCBLOCK information

**get\_channel\_unit** (*channel\_name*)

returns channel unit

**plot** (*channels*)

Plot channels with Matplotlib

**resample** (*sampling\_time = 0.1, master\_channel=None*)

Resamples all data groups

**export\_to\_csv** (*file\_name = None, sampling = 0.1*)

Exports mdf data into CSV file

**export\_to\_NetCDF** (*file\_name = None, sampling = None*)

Exports mdf data into netcdf file

**export\_to\_hdf5** (*file\_name = None, sampling = None*)

Exports mdf class data structure into hdf5 file

**export\_to\_matlab** (*file\_name = None*)

Exports mdf class data structure into Matlab file

**export\_to\_excel** (*file\_name = None*)

Exports mdf data into excel 95 to 2003 file

**export\_to\_xlsx** (*file\_name=None*)

Exports mdf data into excel 2007 and 2010 file

**convert\_to\_pandas** (*sampling=None*)

converts mdf data structure into pandas dataframe(s)

**keep\_channels** (*channel\_list*)

keeps only list of channels and removes the other channels



**merge\_mdf( mdf\_class ) :**  
 Merges data of 2 mdf classes

## Notes

mdf class is a nested dict Channel name is the primary dict key of mdf class At a higher level, each channel includes the following keys :

- 'data' : containing vector of data (numpy)
- 'unit' : unit (string)
- 'master' : master channel of channel (time, crank angle, etc.)
- 'description' : Description of channel
- **'conversion': mdfinfo nested dict for CCBLOCK.** Exist if channel not converted, used to convert with getChannelData method

## Examples

```
>>> import mdfreader
>>> yop=mdfreader.Mdf('NameOfFile')
>>> yop.keys() # list channels names
# list channels grouped by raster or master channel
>>> yop.masterChannelList
>>> yop.plot('channelName') or yop.plot({'channel1','channel2'})
>>> yop.resample(0.1) or yop.resample()
>>> yop.export_to_csv(sampling=0.01)
>>> yop.export_to_NetCDF()
>>> yop.export_to_hdf5()
>>> yop.export_to_matlab()
>>> yop.export_to_excel()
>>> yop.export_to_xlsx()
>>> yop.convert_to_pandas() # converts data groups into pandas dataframes
>>> yop.write() # writes mdf file
# drops all the channels except the one in argument
>>> yop.keep_channels({'channel1','channel2','channel3'})
>>> yop.get_channel_data('channelName') # returns channel numpy array
```

### convert\_all\_channel()

Converts all channels from raw data to converted data according to CCBLOCK information Converted data will take more memory.

### convert\_to\_pandas(sampling=None)

converts mdf data structure into pandas dataframe(s)

**Parameters** *sampling* (float, optional) – resampling interval

## Notes

One pandas dataframe is converted per data group Not adapted yet for mdf4 as it considers only time master channels

### cut(begin=None, end=None)

Cut data

**Parameters**

- **begin** (*float*) – beginning value in master channel from which to start cutting in all channels
- **end** (*float*) – ending value in master channel from which to start cutting in all channels

### Notes

Use this method if whole data in mdf are using same physical or type of master channel (for instance time).

**export\_to\_NetCDF** (*file\_name=None, sampling=None*)

Exports mdf data into netcdf file

#### Parameters

- **file\_name** (*str, optional*) – file name. If no name defined, it will use original mdf name and path
- **sampling** (*float, optional*) – sampling interval.

### Notes

Dependency: scipy

**export\_to\_csv** (*file\_name=None, sampling=None*)

Exports mdf data into CSV file

#### Parameters

- **file\_name** (*str, optional*) – file name. If no name defined, it will use original mdf name and path
- **sampling** (*float, optional*) – sampling interval. None by default

### Notes

Data saved in CSV file be automatically resampled as it is difficult to save in this format data not sharing same master channel Warning: this can be slow for big data, CSV is text format after all

**export\_to\_excel** (*file\_name=None*)

Exports mdf data into excel 95 to 2003 file

**Parameters** **file\_name** (*str, optional*) – file name. If no name defined, it will use original mdf name and path

### Notes

xlwt is not fast even for small files, consider other binary formats like HDF5 or Matlab If there are more than 256 channels, data will be saved over different worksheets Also Excel 2003 is becoming rare these days, prefer using exportToXlsx Dependencies: xlwt for python 2.6+, xlwt3 for python 3.2+

**export\_to\_hdf5** (*file\_name=None, sampling=None, compression=None, compression\_opts=None*)

Exports mdf class data structure into hdf5 file

#### Parameters

- **file\_name** (*str, optional*) – file name. If no name defined, it will use original mdf name and path
- **sampling** (*float, optional*) – sampling interval.

- **compression** (*str*, *optional*) – HDF5 compression algorithm. Valid options are ‘gzip’, ‘lzf’. gzip compression recommended for portability. szip compression not supported due to legal reasons.
- **compression\_opts** (*int*, *optional*) – HDF5 gzip compression level, 0-9. Only valid if gzip compression is used. Level 4 (default) recommended for best balance between compression and time.

### Notes

The maximum attributes will be stored Data structure will be similar has it is in masterChannelList attribute  
Dependency: h5py

**export\_to\_matlab** (*file\_name=None*)

Export mdf data into Matlab file format 5, tentatively compressed

**Parameters** **file\_name** (*str*, *optional*) – file name. If no name defined, it will use original mdf name and path

### Notes

This method will dump all data into Matlab file but you will loose below information: - unit and descriptions of channel - data structure, what is corresponding master channel to a channel. Channels might have then different lengths  
Dependency: scipy

**export\_to\_xlsx** (*file\_name=None*)

Exports mdf data into excel 2007 and 2010 file

**Parameters** **file\_name** (*str*, *optional*) – file name. If no name defined, it will use original mdf name and path

### Notes

It is recommended to export resampled data for performances  
Dependency: openpyxl

**get\_channel\_data** (*channel\_name*, *raw\_data=False*)

Return channel numpy array

**Parameters**

- **channel\_name** (*str*) – channel name
- **raw\_data** (*bool*) – flag to return non converted data

**Returns** converted, if not already done, data corresponding to channel name

**Return type** numpy array

### Notes

This method is the safest to get channel data as numpy array from ‘data’ dict key might contain raw data

**keep\_channels** (*channel\_list*)

keeps only list of channels and removes the other channels

**Parameters** **channel\_list** (*list of str*) – list of channel names

**merge\_mdf** (*mdf\_class*)

Merges data of 2 mdf classes

**Parameters** **mdf\_class** (*Mdf*) – mdf class instance to be merge with self

## Notes

both classes must have been resampled, otherwise, impossible to know master channel to match create union of both channel lists and fill with Nan for unknown sections in channels

**plot** (*channel\_name\_list\_of\_list*)

Plot channels with Matplotlib

**Parameters** **channel\_name\_list\_of\_list** (*str or list of str or list of list of str*) – channel name or list of channel names or list of list of channel names list of list will create multiplots

## Notes

Channel description and unit will be tentatively displayed with axis labels

**plot\_all** ()

**read** (*file\_name=None, multi\_processed=False, channel\_list=None, convert\_after\_read=True, filter\_channel\_names=False, no\_data\_loading=False, compression=False, metadata=2*)  
reads mdf file version 3.x and 4.x

### Parameters

- **file\_name** (*str, optional*) – file name
- **multi\_processed** (*bool*) – flag to activate multiprocessing of channel data conversion
- **channel\_list** (*list of str, optional*) – list of channel names to be read If you use channelList, reading might be much slower but it will save you memory. Can be used to read big files
- **convert\_after\_read** (*bool, optional*) – flag to convert channel after read, True by default If you use convertAfterRead by setting it to false, all data from channels will be kept raw, no conversion applied. If many float are stored in file, you can gain from 3 to 4 times memory footprint To calculate value from channel, you can then use method `.getChannelData()`
- **filter\_channel\_names** (*bool, optional*) – flag to filter long channel names from its module names separated by ‘.’
- **no\_data\_loading** (*bool, optional*) – Flag to read only file info but no data to have minimum memory use
- **compression** (*bool or str, optional*) – To compress data in memory using `blosc` or `bcolz`, takes cpu time if `compression = int(1 to 9)`, uses `bcolz` for compression if `compression = ‘blosc’`, uses `blosc` for compression Choice given, efficiency depends of data
- **metadata** (*int, optional, default = 2*) – Reading metadata has impact on performance, especially for mdf 4.x using xml. 2: minimal metadata reading (mostly channel blocks) 1: used for `noDataLoading` 0: all metadata reading, including Source Information, Attachment, etc..

## Notes

If you keep `convertAfterRead` to true, you can set attribute `mdf.multiProc` to activate channel conversion in multiprocessing. Gain in reading time can be around 30% if file is big and using a lot of float channels

**Warning:** MultiProc use should be avoided when reading several files in a batch, it is not thread safe. You should better multi process instances of mdf rather than using multiproc in mdf class (see implementation of mdfconverter)

**resample** (*sampling\_time=None, master\_channel=None*)

Resamples all data groups into one data group having defined sampling interval or sharing same master channel

#### Parameters

- **sampling\_time** (*float, optional*) – resampling interval, None by default. If None, will merge all datagroups into a unique datagroup having the highest sampling rate from all datagroups
- **\*\*or\*\*** –
- **master\_channel** (*str, optional*) – master channel name to be used for all channels

#### Notes

1. resampling is relatively safe for mdf3 as it contains only time series. However, mdf4 can contain also distance, angle, etc. It might make not sense to apply one resampling to several data groups that do not share same kind of master channel (like time resampling to distance or angle data groups) If several kind of data groups are used, you should better use pandas to resample

2. resampling will convert all your channels so be careful for big files and memory consumption

**write** (*file\_name=None, compression=False*)

Writes simple mdf file, same format as originally read, default is 4.x

#### Parameters

- **file\_name** (*str, optional*) – Name of file If file name is not input, written file name will be the one read with appended ‘\_new’ string before extension
- **compression** (*bool*) – Flag to store data compressed (from mdf version 4.1) If activated, will write in version 4.1 even if original file is in version 3.x

#### Notes

All channels will be converted, so size might be bigger than original file

**class** mdfreader.mdfreader.**MdfInfo**() -> new empty dictionary dict(mapping) -> new dictionary initialized from a mapping object's (key, value) pairs  
dict(iterable) -> new dictionary initialized as if via: d = {}  
for k, v in iterable: d[k] = v dict(\*\*kwargs) -> new dictionary initialized with the name=value pairs in the keyword argument list. For example: dict(one=1, two=2)

Bases: dict

**fid**

**fileName**

**filterChannelNames**

**list\_channels** (*file\_name=None*)

Read MDF file blocks and returns a list of contained channels

**Parameters** `file_name` (*string*) – file name

**Returns** `nameList` – list of channel names

**Return type** list of string

**mdfversion**

**read\_info** (*file\_name=None, fid=None, minimal=0*)

Reads MDF file and extracts its complete structure

**Parameters**

- **file\_name** (*str, optional*) – file name. If not input, uses fileName attribute
- **fid** (*file identifier, optional*) –
- **minimal** (*int*) – 0 will load every metadata 1 will load DG, CG, CN and CC 2 will load only DG

**zipfile**

## MDF MODULE DOCUMENTATION

mdf\_skeleton module describing basic mdf structure and methods

Created on Thu Sept 24 2015

**Author** Aymeric Rateau

### 2.1 Dependencies

- Python >2.6, >3.2 <<http://www.python.org>>
- Numpy >1.6 <<http://numpy.scipy.org>>

### 2.2 mdf

**class** mdfreader.mdf.CompressedData

Bases: object

**compression** (*a*)

data compression method

**Parameters** *a* (*numpy array*) – data to be compresses

**data**

**decompression** ()

data decompression

**Returns**

**Return type** uncompressed numpy array

**dtype**

**class** mdfreader.mdf.MdfSkeleton () -> new empty dictionary dict(mapping) -> new dictionary initialized from a mapping object's (key, value) pairs dict(iterable) -> new dictionary initialized as if via: *d = {}* for *k, v* in iterable: *d[k] = v* dict(\*\*kwargs) -> new dictionary initialized with the name=value pairs in the keyword argument list. For example: dict(one=1, two=2)

Bases: dict

**MDFVersionNumber**

**add\_channel** (*channel\_name*, *data*, *master\_channel*, *master\_type=1*, *unit=""*, *description=""*, *conversion=None*, *info=None*, *compression=False*, *identifier=None*)  
adds channel to mdf dict.

**Parameters**

- **channel\_name** (*str*) – channel name
- **data** (*numpy array*) – numpy array of channel's data
- **master\_channel** (*str*) – master channel name
- **master\_type** (*int, optional*) – master channel type : 0=None, 1=Time, 2=Angle, 3=Distance, 4=index
- **unit** (*str, optional*) – unit description
- **description** (*str, optional*) – channel description
- **conversion** (*info class, optional*) – conversion description from info class
- **info** (*info class for CNBlock, optional*) – used for CABlock axis creation and channel conversion
- **compression** (*bool*) – flag to ask for channel data compression
- **identifier** (*tuple*) – tuple of int and str following below structure: (data group number, channel group number, channel number), (channel name, channel source, channel path), (group name, group source, group path)

**add\_metadata** (*author=”, organisation=”, project=”, subject=”, comment=”, date=”, time=”)*  
adds basic metadata to mdf class

**Parameters**

- **author** (*str*) – author of file
- **organisation** (*str*) – organisation of author
- **project** (*str*) –
- **subject** (*str*) –
- **comment** (*str*) –
- **date** (*str*) –
- **time** (*str*) –

---

**Note:** All fields are optional, default being empty string

---

**convertAfterRead**

**convert\_tables**

**copy()**

copy a mdf class

**Returns** **mdf\_skeleton** – copy of a mdf\_skeleton class

**Return type** class instance

**fid**

**fileName**

**file\_metadata**

**filterChannelNames**



**get\_channel** (*channel\_name*)

Extract channel dict from mdf structure

**Parameters** **channel\_name** (*str*) – channel name

**Returns**

**Return type** channel dictionary containing data, description, unit, etc.

**get\_channel\_conversion** (*channel\_name*)

Extract channel conversion dict from mdf structure

**Parameters** **channel\_name** (*str*) – channel name

**Returns**

**Return type** channel conversion dict

**get\_channel\_desc** (*channel\_name*)

Extract channel description information from mdf structure

**Parameters** **channel\_name** (*str*) – channel name

**Returns**

**Return type** channel description string

**get\_channel\_master** (*channel\_name*)

Extract channel master name from mdf structure

**Parameters** **channel\_name** (*str*) – channel name

**Returns**

**Return type** channel master name string

**get\_channel\_master\_type** (*channel\_name*)

Extract channel master type information from mdf structure

**Parameters** **channel\_name** (*str*) – channel name

**Returns** channel mater type integer

**Return type** 0=None, 1=Time, 2=Angle, 3=Distance, 4=index

**get\_channel\_unit** (*channel\_name*)

Returns channel unit string Implemented for a future integration of pint

**Parameters** **channel\_name** (*str*) – channel name

**Returns** unit string description

**Return type** str

**get\_invalid\_bit** (*channel\_name*)

**get\_invalid\_channel** (*channel\_name*)

**info**

**masterChannelList**

**multiProc**

**remove\_channel** (*channel\_name*)

removes channel from mdf dict.

**Parameters** **channel\_name** (*str*) – channel name

**Returns**

**Return type** value of mdf dict key=channel\_name

**remove\_channel\_conversion** (*channel\_name*)

removes conversion key from mdf channel dict.

**Parameters** **channel\_name** (*str*) – channel name

**Returns**

**Return type** removed value from dict

**rename\_channel** (*channel\_name*, *new\_name*)

Modifies name of channel

**Parameters**

- **channel\_name** (*str*) – channel name
- **new\_name** (*str*) – new channel name

**set\_channel\_attachment** (*channel\_name*, *attachment*)

Modifies channel attachment

**Parameters**

- **channel\_name** (*str*) – channel name
- **attachment** – channel attachment

**set\_channel\_conversion** (*channel\_name*, *conversion*)

Modifies conversion dict of channel

**Parameters**

- **channel\_name** (*str*) – channel name
- **conversion** (*dict*) – conversion dictionary

**set\_channel\_data** (*channel\_name*, *data*, *compression=False*)

Modifies data of channel

**Parameters**

- **channel\_name** (*str*) – channel name
- **data** (*numpy array*) – channel data
- **compression** (*bool or str*) – trigger for data compression

**set\_channel\_desc** (*channel\_name*, *desc*)

Modifies description of channel

**Parameters**

- **channel\_name** (*str*) – channel name
- **desc** (*str*) – channel description

**set\_channel\_master** (*channel\_name*, *master*)

Modifies channel master name

**Parameters**

- **channel\_name** (*str*) – channel name
- **master** (*str*) – master channel name

**set\_channel\_master\_type** (*channel\_name*, *master\_type*)

Modifies master channel type

**Parameters**

- **channel\_name** (*str*) – channel name
- **master\_type** (*int*) – master channel type

**set\_channel\_unit** (*channel\_name*, *unit*)

Modifies unit of channel

**Parameters**

- **channel\_name** (*str*) – channel name
- **unit** (*str*) – channel unit

**set\_invalid\_bit** (*channel\_name*, *bit\_position*)

**set\_invalid\_channel** (*channel\_name*, *invalid\_channel*)

**zipfile**



## MDF3READER MODULE DOCUMENTATION

Measured Data Format file reader module for version 3.x

**Author** Aymeric Rateau

Created on Sun Oct 10 12:57:28 2010

`mdfreader.mdf3reader.PythonVersion`

*float* – Python version currently running, needed for compatibility of both python 2.6+ and 3.2+

### 3.1 mdf3reader

**class** `mdfreader.mdf3reader.DATA` (*fid, pointer*)

Bases: dict

DATA class is organizing record classes itself made of channel. This class inherits from dict. Keys are corresponding to channel group recordID. A DATA class corresponds to a data block, a dict of record classes (one per channel group). Each record class contains a list of channel class representing the structure of channel record.

**fid**

*io.open* – file identifier

**pointerToData**

*int* – position of Data block in mdf file

**BlockLength**

*int* – total size of data block

**add\_record** (*record*)

Adds a new record in DATA class dict

**read** (*channelSet*)

Reads data block

**load\_sorted** (*record, nameList=None*)

Reads sorted data block from record definition

**load\_unsorted** (*nameList=None*)

Reads unsorted data block, not yet implemented

**add\_record** (*record*)

Adds a new record in DATA class dict

**Parameters** **record** (*class*) – channel group definition listing record channel classes

**load\_sorted** (*record, name\_list=None*)

Reads sorted data block from record definition

**Parameters**

- **record** (*class*) – channel group definition listing record channel classes
- **name\_list** (*set of str, optional*) – list of channel names

**Returns**

**Return type** numpy recarray of data

**load\_unsorted** (*name\_list=None*)

Reads unsorted data block from record definition

**Parameters**

- **record** (*class*) – channel group definition listing record channel classes
- **name\_list** (*set of str, optional*) – list of channel names

**Returns**

**Return type** numpy recarray of data

**read** (*channel\_set, file\_name*)

Reads data block

**Parameters**

- **channel\_set** (*set of str, optional*) – list of channel names
- **file\_name** (*str*) – name of file

```
class mdfreader.mdf3reader.Mdf3 (file_name=None, channel_list=None, con-  
                                vert_after_read=True, filter_channel_names=False,  
                                no_data_loading=False, compression=False, con-  
                                vert_tables=False, metadata=2)
```

Bases: *mdfreader.mdf.MdfSkeleton*

mdf file version 3.0 to 3.3 class

**fileName**

*str* – file name

**MDFVersionNumber**

*int* – mdf file version number

**masterChannelList**

*dict* – Represents data structure: a key per master channel with corresponding value containing a list of channels One key or master channel represents then a data group having same sampling interval.

**multiProc**

*bool* – Flag to request channel conversion multi processed for performance improvement. One thread per data group.

**convertAfterRead**

*bool* – flag to convert raw data to physical just after read

**filterChannelNames**

*bool* – flag to filter long channel names from its module names separated by ‘.’

**file\_metadata**

*dict* – file metadata with minimum keys: author, organisation, project, subject, comment, time, date

**read3** (*fileName=None, info=None, multiProc=False, channelList=None, convertAfterRead=True*)

Reads mdf 3.x file data and stores it in dict

**\_get\_channel\_data3** (*channelName*)

Returns channel numpy array

**\_convert\_channel3** (*channelName*)

converts specific channel from raw to physical data according to CCBlock information

**\_convert\_all\_channel3** ()

Converts all channels from raw data to converted data according to CCBlock information

**write3** (*fileName=None*)

Writes simple mdf 3.3 file

**read3** (*file\_name=None, info=None, multi\_processed=False, channel\_list=None, convert\_after\_read=True, filter\_channel\_names=False, compression=False, metadata=2*)

Reads mdf 3.x file data and stores it in dict

#### Parameters

- **file\_name** (*str, optional*) – file name
- **info** (*mdfinfo3.info3 class*) – info3 class containing all MDF Blocks
- **multi\_processed** (*bool*) – flag to activate multiprocessing of channel data conversion
- **channel\_list** (*list of str, optional*) – list of channel names to be read If you use channelList, reading might be much slower but it will save you memory. Can be used to read big files
- **convert\_after\_read** (*bool, optional*) – flag to convert channel after read, True by default If you use convertAfterRead by setting it to false, all data from channels will be kept raw, no conversion applied. If many float are stored in file, you can gain from 3 to 4 times memory footprint To calculate value from channel, you can then use method `.get_channel_data()`
- **filter\_channel\_names** (*bool, optional*) – flag to filter long channel names from its module names separated by ‘.’
- **compression** (*bool, optional*) – flag to activate data compression with blosc
- **metadata** (*int, optional, default = 2*) – Reading metadata has impact on performance, especially for mdf 4.x using xml. 2: minimal metadata reading (mostly channel blocks) 1: used for noDataLoading 0: all metadata reading

**write3** (*file\_name=None*)

Writes simple mdf 3.3 file

**Parameters file\_name** (*str, optional*) – Name of file If file name is not input, written file name will be the one read with appended ‘\_new’ string before extension

#### Notes

All channels will be converted to physical data, so size might be bigger than original file

**class** mdfreader.mdf3reader.**Record** (*data\_group, channel\_group*)

Bases: list

**record class lists Channel classes**, it is representing a channel group

**CGrecordLength**

*int* – length of record from channel group block information in Byte

**recordLength**

*int* – length of record from channels information in Byte

**numberOfRecords**

*int* – number of records in data block

**recordID**

*int* – recordID corresponding to channel group

**recordIDnumber**

*int* – size of recordID

**dataGroup**

*int*: – data group number

**channelGroup**

*int* – channel group number

**numpyDataRecordFormat**

*list* – list of numpy (dtype) for each channel

**dataRecordName**

*list* – list of channel names used for recarray attribute definition

**master**

*dict* – define name and number of master channel

**recordToChannelMatching**

*dict* – helps to identify nested bits in byte

**channelNames**

*set* – channel names to be stored, useful for low memory consumption but slow

**hiddenBytes**

*Bool, False by default* – flag in case of non declared channels in record

**byte\_aligned**

*Bool, True by default* – flag for byte aligned record

**addChannel** (*info*, *channelNumber*)

**loadInfo** (*info*)

**readSortedRecord** (*fid*, *pointer*, *channelSet=None*)

**readRecordBuf** (*buf*, *channelSet=None*)

**readRecordBits** (*bita*, *channelSet=None*)

**add\_channel** (*info*, *channel\_number*)

add a channel in class

**Parameters**

- **info** (*mdfinfo3.info3 class*) –
- **channel\_number** (*int*) – channel number in mdfinfo3.info3 class

**load\_info** (*info*)

gathers records related from info class

**Parameters** **info** (*mdfinfo3.info3 class*) –

**read\_record\_bits** (*bit\_stream*, *channel\_set=None*)

read stream of record bits by bits in case of not aligned or hidden bytes

**Parameters**

- **bit\_stream** (*stream*) – stream of bytes read in file
- **channel\_set** (*Set of str, optional*) – list of channel to read

**Returns** **rec** – returns dictionary of channel with its corresponding values



**Return type** dict

**read\_record\_buf** (*buf*, *channel\_set=None*)

read stream of record bytes

**Parameters**

- **buf** (*stream*) – stream of bytes read in file
- **channel\_set** (*Set of str, optional*) – list of channel to read

**Returns** **rec** – returns dictionary of channel with its corresponding values

**Return type** dict

**read\_sorted\_record** (*fid*, *pointer*, *channel\_set=None*)

reads record, only one channel group per data group

**Parameters**

- **fid** (*float*) – file identifier
- **pointer** – position in file of data block beginning
- **channel\_set** (*Set of str, optional*) – list of channel to read

**Returns** **rec** – contains a matrix of raw data in a recarray (attributes corresponding to channel name)

**Return type** numpy recarray

## Notes

If channelSet is None, read data using `numpy.core.records.fromfile` that is rather quick. However, in case of large file, you can use channelSet to load only interesting channels or only one channel on demand, but be aware it might be much slower.



## MDFINFO3 MODULE DOCUMENTATION

Measured Data Format blocks parser for version 3.x

Created on Thu Dec 9 12:57:28 2014

**Author** Aymeric Rateau

### 4.1 Dependencies

- Python >2.6, >3.2 <<http://www.python.org>>
- Numpy >1.6 <<http://numpy.scipy.org>>

`mdfreader.mdfinfo3.PythonVersion`

*float* – Python version currently running, needed for compatibility of both python 2.6+ and 3.2+

### 4.2 mdfinfo3

**class** `mdfreader.mdfinfo3.Info3()` -> new empty dictionary *dict(mapping)* -> new dictionary initialized from a mapping object's (key, value) pairs *dict(iterable)* -> new dictionary initialized as if via: *d = {}* for *k, v* in *iterable*: *d[k] = v* *dict(\*\*kwargs)* -> new dictionary initialized with the *name=value* pairs in the keyword argument list. For example: *dict(one=1, two=2)*

Bases: `dict`

**clean\_dg\_info** (*dg*)

delete CN,CC and CG blocks related to data group

**Parameters** *dg* (*int*) – data group number

**fid**

**fileName**

**filterChannelNames**

**list\_channels3** (*file\_name=None, fid=None*)

reads data, channel group and channel blocks to list channel names

**file\_name**

*str* – file name

**Returns**

**Return type** list of channel names

**read\_cg\_block** (*fid*, *dg*, *minimal=0*)  
read all CG blocks and relying CN & CC

**Parameters**

- **fid** (*float*) – file identifier
- **dg** (*int*) – data group number
- **minimal** (*int*) – 0 will load every metadata 1 will load DG, CG, CN and CC 2 will load only DG

**read\_info3** (*fid*, *minimal=0*)  
read all file blocks except data

**Parameters**

- **fid** (*float*) – file identifier
- **minimal** (*int*) – 0 will load every metadata 1 will load DG, CG, CN and CC 2 will load only DG

`mdfreader.mdinfo3.read_cc_block` (*fid*, *pointer*)  
channel conversion block reading

`mdfreader.mdinfo3.read_ce_block` (*fid*, *pointer*)  
reads source block

`mdfreader.mdinfo3.read_cg_block` (*fid*, *pointer*)  
channel block reading

`mdfreader.mdinfo3.read_cn_block` (*fid*, *pointer*)  
channel block reading

`mdfreader.mdinfo3.read_dg_block` (*fid*, *pointer*)  
data group block reading

`mdfreader.mdinfo3.read_hd_block` (*fid*, *pointer*, *version=0*)  
header block reading

`mdfreader.mdinfo3.read_tx_block` (*fid*, *pointer*)  
reads text block

## MDF4READER MODULE DOCUMENTATION

Measured Data Format file reader module for version 4.x.

**Author** Aymeric Rateau

Created on Thu Dec 10 12:57:28 2013

### 5.1 Dependencies

- Python >2.6, >3.2 <<http://www.python.org>>
- Numpy >1.6 <<http://numpy.scipy.org>>
- bitarray to parse bits in not aligned bytes
- Sympy to convert channels with formula if needed
- zlib to uncompress data block if needed

`mdfreader.mdf4reader.PythonVersion`

*float* – Python version currently running, needed for compatibility of both python 2.6+ and 3.2+

### 5.2 mdf4reader

**class** `mdfreader.mdf4reader.Data()` -> new empty dictionary *dict(mapping)* -> new dictionary initialized from a mapping object's (key, value) pairs *dict(iterable)* -> new dictionary initialized as if via: *d = {}* for *k, v* in *iterable*: *d[k] = v* *dict(\*\*kwargs)* -> new dictionary initialized with the *name=value* pairs in the keyword argument list. For example: *dict(one=1, two=2)*

Bases: `dict`

**add\_record** (*record*)

Adds a new record in Data class dict.

**Parameters** **record** (*class*) – channel group definition listing record channel classes

**fid**

**load** (*record*, *info*, *name\_list=None*, *sorted\_flag=True*, *vlsd=False*)

Reads data block from record definition

**Parameters**

- **record** (*class*) – channel group definition listing record channel classes
- **info** (*class*) – contains blocks

- **name\_list** (*list of str, optional*) – list of channel names
- **sorted\_flag** (*bool, optional*) – flag to know if data block is sorted (only one Channel Group in block) or unsorted (several Channel Groups identified by a recordID). As unsorted block can contain CG records in random order, block is processed iteratively, not in raw like sorted -> much slower reading
- **vlsd** (*bool*) – indicate a sd block, compressed (DZ) or not (SD)

#### Returns

**Return type** numpy recarray of data

#### pointer\_to\_data

**read** (*channel\_set, info, filename*)

Reads data block

#### Parameters

- **channel\_set** (*set of str*) – set of channel names
- **info** (*info object*) – contains blocks structures
- **filename** – name of file ot read

**read\_record** (*record\_id, info, buf*)

read record from a buffer

#### Parameters

- **record\_id** (*int*) – record identifier
- **info** (*class*) – contains blocks
- **buf** (*str*) – buffer of data from file to be converted to channel raw data

#### type

```
class mdfreader.mdf4reader.Mdf4 (file_name=None,          channel_list=None,          con-  
                                vert_after_read=True,      filter_channel_names=False,  
                                no_data_loading=False,       compression=False,      con-  
                                vert_tables=False, metadata=2)
```

Bases: `mdfreader.mdf.MdfSkeleton`

mdf file reader class from version 4.0 to 4.1.1

#### fileName

*str* – file name

#### MDFVersionNumber

*int* – mdf file version number

#### masterChannelList

*dict* – Represents data structure: a key per master channel with corresponding value containing a list of channels One key or master channel represents then a data group having same sampling interval.

#### multiProc

*bool* – Flag to request channel conversion multi processed for performance improvement. One thread per data group.

#### convertAfterRead

*bool* – flag to convert raw data to physical just after read

#### filterChannelNames

*bool* – flag to filter long channel names from its module names separated by ‘.’

**file\_metadata**

*dict* – file metadata with minimum keys : author, organisation, project, subject, comment, time, date

**read4** (*fileName=None, info=None, multiProc=False, channelList=None, convertAfterRead=True*)

Reads mdf 4.x file data and stores it in dict

**\_get\_channel\_data\_4** (*channelName*)

Returns channel numpy array

**\_convert\_channel\_data\_4** (*channel, channel\_name, convert\_tables, multiProc=False, Q=None*)

select right conversion and calculates it

**\_convert\_channel\_4** (*channelName*)

converts specific channel from raw to physical data according to CCBlock information

**\_convert\_all\_channel\_4** ()

Converts all channels from raw data to converted data according to CCBlock information

**write4** (*file\_name=None, compression=False*)

writes mdf 4.1 file

**apply\_invalid\_bit** (*channel\_name*)

mask data from invalid bit channel if existing

**get\_channel\_name\_4** (*name, path*)

returns a list of tuples

**apply\_invalid\_bit** (*channel\_name*)

Mask data of channel based on its invalid bit definition if there is

**Parameters** **channel\_name** (*str*) – Name of channel

**get\_channel\_name\_4** (*name, path*)

finds mdf channel name from name and path

**Parameters**

- **name** (*str*) – channel name
- **path** (*str*) – source path or name, or channel group name, source name or path

**Returns**

**Return type** list of tuples (channel\_name, (ndg, ncg, ncn))

**read4** (*file\_name=None, info=None, multi\_processed=False, channel\_list=None, convert\_after\_read=True, compression=False, metadata=2*)

Reads mdf 4.x file data and stores it in dict

**Parameters**

- **file\_name** (*str, optional*) – file name
- **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks
- **multi\_processed** (*bool*) – flag to activate multiprocessing of channel data conversion
- **channel\_list** (*list of str, optional*) – list of channel names to be read If you use channelList, reading might be much slower but it will save you memory. Can be used to read big files
- **convert\_after\_read** (*bool, optional*) – flag to convert channel after read, True by default If you use convertAfterRead by setting it to false, all data from channels will be kept raw, no conversion applied. If many float are stored in file, you can gain

from 3 to 4 times memory footprint To calculate value from channel, you can then use method `.get_channel_data()`

- **compression** (*bool, optional*) – flag to activate data compression with `blosc`
- **metadata** (*int, optional, default = 2*) – Reading metadata has impact on performance, especially for mdf 4.x using xml. 2: minimal metadata reading (mostly channel blocks) 1: used for `noDataLoading` 0: all metadata reading, including Source Information, Attachment, etc..

**write4** (*file\_name=None, compression=False*)

Writes simple mdf 4.1 file

#### Parameters

- **file\_name** (*str, optional*) – Name of file If file name is not input, written file name will be the one read with appended ‘\_new’ string before extension
- **compression** (*bool*) – flag to store data compressed

#### Notes

All channels will be converted to physical data, so size might be bigger than original file

**class** `mdfreader.mdf4reader.Record` (*data\_group, channel\_group*)

Bases: `list`

**CANOpen**

**CGrecordLength**

**Flags**

**MLSD**

**VLSD**

**VLSD\_CG**

**add\_channel** (*info, channel\_number*)

add a channel in class

#### Parameters

- **info** (*mdfinfo4.info4 class*) –
- **channel\_number** (*int*) – channel number in `mdfinfo4.info4` class

**byte\_aligned**

**channelGroup**

**channelNames**

**dataGroup**

**dataRecordName**

**generate\_chunks** ()

calculate data split

#### Returns

**Return type** (*n\_record\_chunk, chunk\_size*)

**hiddenBytes**



**initialise\_reccarray** (*info*, *channel\_set*, *n\_records*, *dtype=None*, *channels\_indexes=None*)

Initialise reccarray

**Parameters**

- **info** (*info class*) –
- **channel\_set** (*set of str, optional*) – set of channel to read
- **n\_records** (*int*) – number of records
- **dtype** (*numpy dtype, optional*) –
- **channels\_indexes** (*list of int, optional*) –

**Returns** **rec** – contains a matrix of raw data in a reccarray (attributes corresponding to channel name)

**Return type** numpy reccarray

**invalid\_channel**

**load\_info** (*info*)

gathers records related from info class

**Parameters** **info** (*mdfinfo4.info4 class*) –

**master**

**numberOfRecords**

**numpyDataRecordFormat**

**read\_all\_channels\_sorted\_record** (*fid*)

reads all channels from file using numpy fromstring, chunk by chunk

**Parameters** **fid** – file identifier

**Returns** **rec** – contains a matrix of raw data in a reccarray (attributes corresponding to channel name)

**Return type** numpy reccarray

**read\_channels\_from\_bytes** (*bit\_stream*, *info*, *channel\_set=None*, *n\_records=None*, *dtype=None*, *channels\_indexes=None*)

reads stream of record bytes using dataRead module if available otherwise bitarray

**Parameters**

- **bit\_stream** (*stream*) – stream of bytes
- **info** (*info class*) –
- **channel\_set** (*set of str, optional*) – set of channel to read
- **n\_records** (*int*) – number of records
- **dtype** (*numpy dtype*) –
- **channels\_indexes** (*list of int*) –

**Returns** **rec** – contains a matrix of raw data in a reccarray (attributes corresponding to channel name)

**Return type** numpy reccarray

**read\_channels\_from\_bytes\_fallback** (*bit\_stream*, *info*, *channel\_set=None*, *n\_records=None*, *dtype=None*, *channels\_indexes=None*)

reads stream of record bytes using bitarray in case no dataRead available

**Parameters**

- **bit\_stream** (*stream*) – stream of bytes
- **info** (*info class*) –
- **channel\_set** (*set of str, optional*) – set of channel to read
- **n\_records** (*int*) – number of records
- **dtype** (*numpy dtype*) –
- **channels\_indexes** (*list of int*) –

**Returns** **rec** – contains a matrix of raw data in a recarray (attributes corresponding to channel name)

**Return type** numpy recarray

**read\_not\_all\_channels\_sorted\_record** (*fid, info, channel\_set*)

reads channels from file listed in channelSet

**Parameters**

- **fid** – file identifier
- **info** (*info class*) –
- **channel\_set** (*set of str, optional*) – set of channel to read

**Returns** **rec** – contains a matrix of raw data in a recarray (attributes corresponding to channel name)

**Return type** numpy recarray

**read\_record\_buf** (*buf, info, channel\_set=None*)

read stream of record bytes

**Parameters**

- **buf** (*stream*) – stream of bytes read in file
- **info** (*class*) – contains blocks structure
- **channel\_set** (*set of str, optional*) – set of channel to read

**Returns** **rec** – returns dictionary of channel with its corresponding values

**Return type** dict

**read\_sorted\_record** (*fid, info, channel\_set=None*)

reads record, only one channel group per datagroup

**Parameters**

- **fid** – file identifier
- **info** – info class
- **channel\_set** (*set of str, optional*) – set of channel to read

**Returns** **rec** – contains a matrix of raw data in a recarray (attributes corresponding to channel name)

**Return type** numpy recarray

## Notes

If channelSet is None, read data using `numpy.core.records.fromfile` that is rather quick. However, in case of large file, you can use channelSet to load only interesting channels or only one channel on demand, but be aware it might be much slower.

**recordID**

**recordIDCFormat**

**recordIDsize**

**recordLength**

**recordToChannelMatching**



## MDFINFO4 MODULE DOCUMENTATION

Measured Data Format blocks parser for version 4.x

Created on Sun Dec 15 12:57:28 2013

**Author** Aymeric Rateau

`mdfreader.mdfinfo4.PythonVersion`

*float* – Python version currently running, needed for compatibility of both python 2.6+ and 3.2+

### 6.1 mdfinfo4

**class** `mdfreader.mdfinfo4.ATBlock` (*fid, pointer*)

Bases: dict

reads Attachment block and saves in class dict

**class** `mdfreader.mdfinfo4.CABlock`

Bases: dict

reads Channel Array block and saves in class dict

**load** (*byte\_offset\_base*)

**read** (*fid, pointer*)

**write** (*fid*)

**class** `mdfreader.mdfinfo4.CCBlock`

Bases: dict

reads Channel Conversion block and saves in class dict

**read\_cc** (*fid, pointer*)

**class** `mdfreader.mdfinfo4.CGBlock` (*fid=None, pointer=None*)

Bases: dict

reads Channel Group block and saves in class dict

**read** (*fid, pointer*)

**write** (*fid*)

**class** `mdfreader.mdfinfo4.CHBlock` (*fid, pointer*)

Bases: dict

reads Channel Hierarchy block and saves in class dict

```
class mdfreader.mdinfo4.CNBlock
```

```
    Bases: dict
```

```
    reads Channel block and saves in class dict
```

```
    read_cn (**kargs)
```

```
    write (fid)
```

```
class mdfreader.mdinfo4.CommentBlock
```

```
    Bases: dict
```

```
    reads or writes Comment block and saves in class dict
```

```
    load (data, MDType)
```

```
    read_cm (**kargs)
```

```
        reads Comment block and saves in class dict
```

#### Parameters

- **fid** – file identifier
- **pointer** (*int*) – position in file
- **MDType** (*str*) – describes metadata type, ('CN', 'unit', 'FH', 'SI', 'HD', 'CC', 'EV')

#### Notes

Can read xml (MD metadata) or text (TX) comments from several kind of blocks

```
    write (fid)
```

```
class mdfreader.mdinfo4.DGBlock (fid=None, pointer=None)
```

```
    Bases: dict
```

```
    reads Data Group block and saves in class dict
```

```
    read (fid, pointer)
```

```
    write (fid)
```

```
class mdfreader.mdinfo4.DLBlock
```

```
    Bases: dict
```

```
    reads Data List block
```

```
    read (fid, link_count)
```

```
    write (fid, chunks, position)
```

```
class mdfreader.mdinfo4.DTBlock () -> new empty dictionary dict(mapping) -> new dictionary initialized from a mapping object's (key, value) pairs dict(iterable) -> new dictionary initialized as if via: d = {} for k, v in iterable: d[k] = v dict(**kwargs) -> new dictionary initialized with the name=value pairs in the keyword argument list. For example: dict(one=1, two=2)
```

```
    Bases: dict
```

```
    load (record_byte_offset, nRecords, pointer)
```

```
    write (fid, data)
```

```
class mdfreader.mdinfo4.DZBlock
```

```
    Bases: dict
```

```
    reads Data List block
```

**static decompress\_data\_block** (*block*, *zip\_type*, *zip\_parameter*, *org\_data\_length*)  
decompress datablock.

#### Parameters

- **block** (*bytes*) – raw data compressed
- **zip\_type** (*int*) – 0 for non transposed, 1 for transposed data
- **zip\_parameter** (*int*) – first dimension of matrix to be transposed
- **org\_data\_length** (*int*) – uncompressed data length

#### Returns

**Return type** uncompressed raw data

**read** (*fid*)

**write** (*fid*, *data*, *record\_length*)

**class** mdfreader.mdinfo4.**EVBlock** (*fid*, *pointer*)

Bases: dict

reads Event block and saves in class dict

**class** mdfreader.mdinfo4.**FHBlock** (*fid=None*, *pointer=None*)

Bases: dict

reads File History block and save in class dict

**read** (*fid*, *pointer*)

**write** (*fid*)

**class** mdfreader.mdinfo4.**HDBlock** (*fid=None*)

Bases: dict

reads Header block and save in class dict

**read** (*fid=None*)

**write** (*fid*)

**class** mdfreader.mdinfo4.**HLBlock**

Bases: dict

reads Header List block

**load** (*record\_byte\_offset*, *n\_records*, *position*)

**read** (*fid*)

**write** (*fid*, *data*)

**class** mdfreader.mdinfo4.**IDBlock** (*fid=None*)

Bases: dict

reads or writes ID Block

**read** (*fid*)

reads IDBlock

**static write** (*fid*)

Writes IDBlock

**class** mdfreader.mdinfo4.**Info4**() -> new empty dictionary dict(mapping) -> new dictionary initialized from a mapping object's (key, value) pairs dict(iterable) -> new dictionary initialized as if via: `d = {}` for `k, v` in iterable: `d[k] = v` dict(\*\*kwargs) -> new dictionary initialized with the name=value pairs in the keyword argument list. For example: `dict(one=1, two=2)`

Bases: dict

**clean\_dg\_info**(dg)  
delete CN,CC and CG blocks related to data group

**Parameters** dg (int) – data group number

**fid**

**fileName**

**list\_channels\_4**(file\_name=None, fid=None)  
Read MDF file and extract its complete structure

**Parameters**

- **file\_name** (str) – file name
- **fid** –

**Returns**

**Return type** list of channel names contained in file

**static read\_at\_block**(fid, pointer)  
reads Attachment blocks

**Parameters**

- **fid** (float) – file identifier
- **pointer** (int) – position of ATBlock in file

**Returns**

**Return type** Attachments Blocks in a dict

**read\_cg\_block**(fid, dg, channel\_name\_list=False, minimal=0)  
reads Channel Group blocks

**Parameters**

- **fid** (float) – file identifier
- **dg** (int) – data group number
- **channel\_name\_list** (bool) – Flag to reads only channel blocks for listChannels4 method
- **minimal** (bool) – to activate minimum content reading for raw data fetching

**read\_cn\_block**(fid, dg, cg, channel\_name\_list=False, minimal=0)  
reads Channel blocks

**Parameters**

- **fid** (float) – file identifier
- **dg** (int) – data group number
- **cg** (int) – channel group number in data group



- **channel\_name\_list** (*bool*) – Flag to reads only channel blocks for listChannels4 method
- **minimal** (*flag*) – to activate minimum content reading for raw data fetching

**read\_composition** (*fid, dg, cg, mlsd\_channels*)  
check for composition of channels, arrays or structures

#### Parameters

- **fid** (*float*) – file identifier
- **dg** (*int*) – data group number
- **cg** (*int*) – channel group number in data group
- **mlsd\_channels** (*list of int*) – channel numbers

#### Returns

**Return type** MLSDChannels list of appended Maximum Length Sampling Data channels

**read\_dg\_block** (*fid, channel\_name\_list=False, minimal=0*)  
reads Data Group Blocks

#### Parameters

- **fid** (*float*) – file identifier
- **channel\_name\_list** (*bool*) – Flag to reads only channel blocks for listChannels4 method
- **minimal** (*flag*) – to activate minimum content reading for raw data fetching

**read\_info** (*fid, minimal*)  
read all file blocks except data

#### Parameters

- **fid** (*float*) – file identifier
- **minimal** (*flag*) – to activate minimum content reading for raw data fetching

**static read\_sr\_block** (*fid, pointer*)  
reads Sample Reduction Blocks

#### Parameters

- **fid** (*float*) – file identifier
- **pointer** (*int*) – position of SRBlock in file

#### Returns

**Return type** Sample Reduction Blocks in a dict

**unique\_id** (*ndg, ncg, ncn*)  
generate unique id tuples

#### Parameters

- **ndg** (*int*) – data group number
- **ncg** (*int*) – channel group number
- **ncn** (*int*) – channel number

**Returns tuples** – (channel name, channel source, channel path), (group name, group source, group path)

**Return type** (data group number, channel group number, channel number),

**zipfile**

**class** mdfreader.mdinfo4.**SIBlock**

Bases: dict

reads Source Information block and saves in class dict

**read\_si** (*fid, pointer*)

**class** mdfreader.mdinfo4.**SRBlock** (*fid, pointer*)

Bases: dict

reads Sample Reduction block and saves in class dict

## CHANNEL MODULE DOCUMENTATION

Measured Data Format file reader module.

**Author** Aymeric Rateau

Created on Wed Oct 04 21:13:28 2017

`mdfreader.channel.PythonVersion`

*float* – Python version currently running, needed for compatibility of both python 2.6+ and 3.4+

`mdfreader.channel.channel`

-----  
**class** `mdfreader.channel.Channel13` (*info, data\_group, channel\_group, channel\_number, record\_id\_number*)

Bases: object

Channel class gathers all about channel structure in a record

**name**

*str* – Name of channel

**unit**

*str, default empty string* – channel unit

**desc**

*str* – channel description

**conversion**

*info class* – conversion dictionary

**channelNumber**

*int* – channel number corresponding to `mdfinfo3.info3` class

**signalDataType**

*int* – signal type according to specification

**bitCount**

*int* – number of bits used to store channel record

**nBytes**

*int* – number of bytes (1 byte = 8 bits) taken by channel record

**dataFormat**

*str* – numpy dtype as string

**CFormat**

*struct class instance* – struct instance to convert from C Format

**byteOffset**

*int* – position of channel record in complete record in bytes

**bitOffset**

*int* – bit position of channel value inside byte in case of channel having bit count below 8

**recAttributeName**

*str* – channel name compliant to a valid python identifier (recarray attribute)

**RecordFormat**

*list of str* – dtype format used for numpy.core.records functions ((name\_title,name),str\_type)

**channelType**

*int* – channel type

**posByteBeg**

*int* – start position in number of bit of channel record in complete record

**posByteEnd**

*int* – end position in number of bit of channel record in complete record

**bit\_masking\_needed**

*bool, default false* – True if bit masking needed after data read

**\_\_init\_\_** (*info, dataGroup, channelGroup, channelNumber, recordIDnumber*)  
constructor

**\_\_str\_\_** ()  
to print class attributes

**change\_channel\_name** (*channel\_group*)  
rename duplicated channel name within unsorted channel groups

**change\_channel\_name** (*channel\_group*)  
In case of duplicate channel names within several channel groups for unsorted data, rename channel name

**Parameters** **channel\_group** (*int*) – channelGroup number

**class** mdfreader.channel.Channel4

Bases: object

**CANOpen\_offset** ()  
CANopen channel bytes offset

**Returns**

**Return type** integer, channel bytes offset

**VLSD\_CG\_Flag**

**attachment** (*fid, info*)  
In case of sync channel attached to channel

**Parameters**

- **fid** (*class*) – file identifier
- **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

**Return type** ATBlock class from mdfinfo4 module

**bit\_count** (*info*)  
calculates channel number of bits

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

**Return type** integer corresponding to channel number of bits

**bit\_masking\_needed** (*info*)

Valid if bit masking need

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

**Return type** boolean True if channel needs bit masking, otherwise False

**bit\_offset** (*info*)

channel data bit offset in record

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

**Return type** integer, channel bit offset

**byteOffset**

**c\_format** (*info*)

channel data C format

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

**Return type** string data C format

**c\_format\_structure** (*info*)

channel data C format struct object

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

**Return type** string data C format struct object

**ca\_block** (*info*)

Extracts channel CA Block from info4

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

**Return type** CABlock object from mdfinfo4 module

**calc\_byte\_offset** (*info*)

channel data bytes offset in record (without record id)

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

**Return type** integer, channel bytes offset

**calc\_bytes** (*info*)

calculates channel bytes number

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

**Return type** number of bytes integer

**change\_channel\_name** (*channel\_group*)

In case of duplicate channel names within several channel groups for unsorted data, rename channel name

**Parameters** **channel\_group** (*int*) – channelGroup number

**channelGroup**

**channelNumber**

**channel\_sync\_type** (*info*)

Extracts channel sync type from info4

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

- *integer corresponding to channel sync type*
- *0 no sync, normal data*
- *1 time*
- *2 angle*
- *3 distance*
- *4 index*

**channel\_type** (*info*)

Extracts channel type from info4

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

- *integer describing channel type*
- *0 normal channel*
- *1 variable length*
- *2 master channel*
- *3 virtual master channel*
- *4 sync channel*
- *5 max length data*
- *6 virtual data channel*

**cn\_block** (*info*)

channel block

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

**Return type** CNBlock class from mdfinfo4 module

**conversion** (*info*)

channel conversion CCBLOCK

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

**Return type** *CCBlock*

**data** (*info*)  
returns data block pointer for VLSD, MLD or sync channels

**dataGroup**

**data\_format** (*info*)  
channel numpy.core.records data format

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

**Return type** string data format

**desc** (*info*)  
channel description

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

**Return type** channel description string

**has\_invalid\_bit** (*info*)

**invalid\_bit** (*info*)  
extracts from info4 the channels valid bits positions

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

**Return type** channel valid bit position

**is\_ca\_block** (*info*)

**isnumeric** (*info*)  
check this is numeric channel from data type

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

**Return type** boolean, true if numeric channel, otherwise false

**little\_endian** (*info*)  
check if channel is little endian

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

**Return type** boolean

**nBytes**

**name**

**native\_data\_format** (*info*)

**numpy\_format** (*info*)  
channel numpy.core.records data format

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns** **endian, dataType**

**Return type** string data format

**pos\_bit\_beg** (*info*)

channel data bit starting position in record

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

**Return type** integer, channel bit starting position

**pos\_bit\_end** (*info*)

channel data bit ending position in record

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

**Return type** integer, channel bit ending position

**pos\_byte\_beg** (*info*)

channel data bytes starting position in record

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

**Return type** integer, channel bytes starting position

**pos\_byte\_end** (*info*)

channel data bytes ending position in record

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

**Return type** integer, channel bytes ending position

**record\_attribute\_name** ()

clean up channel name from unauthorised characters

**Returns**

**Return type** channel name compliant to python attributes names (for recarray)

**record\_id\_size** (*info*)

Extracts record id size from info4

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

- *integer describing record id size*
- *0 no record id used*
- *1 uint8*
- *2 uint16*
- *4 uint32*
- *8 uint64*

**set** (*info, data\_group, channel\_group, channel\_number*)

channel initialisation

**Parameters**

- **info** (*mdfinfo4.info4 class*) –



- **data\_group** (*int*) – data group number in mdinfo4.info4 class
- **channel\_group** (*int*) – channel group number in mdinfo4.info4 class
- **channel\_number** (*int*) – channel number in mdinfo4.info4 class

**set\_CANOpen** (*info, data\_group, channel\_group, channel\_number, name*)

CANOpen channel initialisation

#### Parameters

- **info** (*mdinfo4.info4 class*) –
- **data\_group** (*int*) – data group number in mdinfo4.info4 class
- **channel\_group** (*int*) – channel group number in mdinfo4.info4 class
- **channel\_number** (*int*) – channel number in mdinfo4.info4 class
- **name** (*str*) – name of channel. Should be in ('ms', 'day', 'days', 'hour', 'month', 'minute', 'year')

**set\_invalid\_bytes** (*info, data\_group, channel\_group, channel\_number*)

invalid\_bytes channel initialisation

#### Parameters

- **info** (*mdinfo4.info4 class*) –
- **data\_group** (*int*) – data group number in mdinfo4.info4 class
- **channel\_group** (*int*) – channel group number in mdinfo4.info4 class
- **channel\_number** (*int*) – channel number in mdinfo4.info4 class

**signal\_data\_type** (*info, byte\_aligned=True*)

extract signal data type from info4 class

#### Parameters

- **info** (*mdinfo4.info4 class*) – info4 class containing all MDF Blocks
- **byte\_aligned** (*bool*) – flag activated if channel is part of a record byte aligned

#### Returns

- *integer corresponding to channel data type*
- *0 unsigned integer little endian*
- *1 unsigned integer big endian*
- *2 signed integer little endian*
- *3 signed integer big endian*
- *4 float little endian*
- *5 float big endian*
- *6 string latin*
- *7 string utf-8*
- *9 string utf-16*
- *10 byte array*
- *11 mime sample*

- 12 *mime stream*
- 13 *CANopen date*
- 14 *CANopen time*

**type**

**unit** (*info*)

channel unit

**Parameters** **info** (*mdfinfo4.info4 class*) – info4 class containing all MDF Blocks

**Returns**

**Return type** channel unit string

`mdfreader.channel.array_format4` (*signal\_data\_type, number\_of\_bytes*)

function returning numpy style string from channel data type and number of bits

**Parameters**

- **signal\_data\_type** (*int*) – channel data type according to specification
- **number\_of\_bytes** (*int*) – number of bytes taken by channel data in a record

**Returns** **endian, data\_type** – numpy dtype format used by `numpy.core.records` to read channel raw data

**Return type** str

`mdfreader.channel.data_type_format4` (*signal\_data\_type, number\_of\_bytes*)

function returning C format string from channel data type and number of bits

**Parameters**

- **signal\_data\_type** (*int*) – channel data type according to specification
- **number\_of\_bytes** (*int*) – number of bytes taken by channel data in a record

**Returns** **data\_type** – C format used by `fread` to read channel raw data

**Return type** str

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### m

- `mdfreader.channel`, [39](#)
- `mdfreader.mdf`, [11](#)
- `mdfreader.mdf3reader`, [17](#)
- `mdfreader.mdf4reader`, [25](#)
- `mdfreader.mdffinfo3`, [23](#)
- `mdfreader.mdffinfo4`, [33](#)
- `mdfreader.mdfreader`, [3](#)



## Symbols

[\\_\\_init\\_\\_\(\)](#) (mdfreader.channel.Channel3 method), 40  
[\\_\\_str\\_\\_\(\)](#) (mdfreader.channel.Channel3 method), 40  
[\\_convert\\_all\\_channel3\(\)](#) (mdfreader.mdf3reader.Mdf3 method), 19  
[\\_convert\\_all\\_channel\\_4\(\)](#) (mdfreader.mdf4reader.Mdf4 method), 27  
[\\_convert\\_channel3\(\)](#) (mdfreader.mdf3reader.Mdf3 method), 18  
[\\_convert\\_channel\\_4\(\)](#) (mdfreader.mdf4reader.Mdf4 method), 27  
[\\_convert\\_channel\\_data\\_4\(\)](#) (mdfreader.mdf4reader.Mdf4 method), 27  
[\\_get\\_channel\\_data3\(\)](#) (mdfreader.mdf3reader.Mdf3 method), 18  
[\\_get\\_channel\\_data\\_4\(\)](#) (mdfreader.mdf4reader.Mdf4 method), 27

## A

[add\\_channel\(\)](#) (mdfreader.mdf.MdfSkeleton method), 11  
[add\\_channel\(\)](#) (mdfreader.mdf3reader.Record method), 20  
[add\\_channel\(\)](#) (mdfreader.mdf4reader.Record method), 28  
[add\\_metadata\(\)](#) (mdfreader.mdf.MdfSkeleton method), 12  
[add\\_record\(\)](#) (mdfreader.mdf3reader.DATA method), 17  
[add\\_record\(\)](#) (mdfreader.mdf4reader.Data method), 25  
[addChannel\(\)](#) (mdfreader.mdf3reader.Record method), 20  
[apply\\_invalid\\_bit\(\)](#) (mdfreader.mdf4reader.Mdf4 method), 27  
[array\\_format4\(\)](#) (in module mdfreader.channel), 46  
[ATBlock](#) (class in mdfreader.mdinfo4), 33  
[attachment\(\)](#) (mdfreader.channel.Channel4 method), 40

## B

[bit\\_count\(\)](#) (mdfreader.channel.Channel4 method), 40  
[bit\\_masking\\_needed](#) (mdfreader.channel.Channel3 attribute), 40  
[bit\\_masking\\_needed\(\)](#) (mdfreader.channel.Channel4 method), 41  
[bit\\_offset\(\)](#) (mdfreader.channel.Channel4 method), 41

[bitCount](#) (mdfreader.channel.Channel3 attribute), 39  
[bitOffset](#) (mdfreader.channel.Channel3 attribute), 40  
[BlockLength](#) (mdfreader.mdf3reader.DATA attribute), 17  
[byte\\_aligned](#) (mdfreader.mdf3reader.Record attribute), 20  
[byte\\_aligned](#) (mdfreader.mdf4reader.Record attribute), 28  
[byteOffset](#) (mdfreader.channel.Channel3 attribute), 39  
[byteOffset](#) (mdfreader.channel.Channel4 attribute), 41

## C

[c\\_format\(\)](#) (mdfreader.channel.Channel4 method), 41  
[c\\_format\\_structure\(\)](#) (mdfreader.channel.Channel4 method), 41  
[ca\\_block\(\)](#) (mdfreader.channel.Channel4 method), 41  
[CABlock](#) (class in mdfreader.mdinfo4), 33  
[calc\\_byte\\_offset\(\)](#) (mdfreader.channel.Channel4 method), 41  
[calc\\_bytes\(\)](#) (mdfreader.channel.Channel4 method), 41  
[CANOpen](#) (mdfreader.mdf4reader.Record attribute), 28  
[CANOpen\\_offset\(\)](#) (mdfreader.channel.Channel4 method), 40  
[CCBlock](#) (class in mdfreader.mdinfo4), 33  
[CFormat](#) (mdfreader.channel.Channel3 attribute), 39  
[CGBlock](#) (class in mdfreader.mdinfo4), 33  
[CGrecordLength](#) (mdfreader.mdf3reader.Record attribute), 19  
[CGrecordLength](#) (mdfreader.mdf4reader.Record attribute), 28  
[change\\_channel\\_name\(\)](#) (mdfreader.channel.Channel3 method), 40  
[change\\_channel\\_name\(\)](#) (mdfreader.channel.Channel4 method), 42  
[channel](#) (in module mdfreader.channel), 39  
[Channel3](#) (class in mdfreader.channel), 39  
[Channel4](#) (class in mdfreader.channel), 40  
[channel\\_sync\\_type\(\)](#) (mdfreader.channel.Channel4 method), 42  
[channel\\_type\(\)](#) (mdfreader.channel.Channel4 method), 42  
[channelGroup](#) (mdfreader.channel.Channel4 attribute), 42  
[channelGroup](#) (mdfreader.mdf3reader.Record attribute), 20

channelGroup (mdfreader.mdf4reader.Record attribute), 28

channelNames (mdfreader.mdf3reader.Record attribute), 20

channelNames (mdfreader.mdf4reader.Record attribute), 28

channelNumber (mdfreader.channel.Channel3 attribute), 39

channelNumber (mdfreader.channel.Channel4 attribute), 42

channelType (mdfreader.channel.Channel3 attribute), 40

CHBlock (class in mdfreader.mdfinfo4), 33

clean\_dg\_info() (mdfreader.mdfinfo3.Info3 method), 23

clean\_dg\_info() (mdfreader.mdfinfo4.Info4 method), 36

cn\_block() (mdfreader.channel.Channel4 method), 42

CNBlock (class in mdfreader.mdfinfo4), 33

CommentBlock (class in mdfreader.mdfinfo4), 34

CompressedData (class in mdfreader.mdf), 11

compression() (mdfreader.mdf.CompressedData method), 11

conversion (mdfreader.channel.Channel3 attribute), 39

conversion() (mdfreader.channel.Channel4 method), 42

convert\_all\_channel() (mdfreader.mdfreader.Mdf method), 4, 5

convert\_tables (mdfreader.mdf.MdfSkeleton attribute), 12

convert\_to\_pandas() (mdfreader.mdfreader.Mdf method), 4, 5

convertAfterRead (mdfreader.mdf.MdfSkeleton attribute), 12

convertAfterRead (mdfreader.mdf3reader.Mdf3 attribute), 18

convertAfterRead (mdfreader.mdf4reader.Mdf4 attribute), 26

copy() (mdfreader.mdf.MdfSkeleton method), 12

cut() (mdfreader.mdfreader.Mdf method), 5

## D

DATA (class in mdfreader.mdf3reader), 17

Data (class in mdfreader.mdf4reader), 25

data (mdfreader.mdf.CompressedData attribute), 11

data() (mdfreader.channel.Channel4 method), 42

data\_format() (mdfreader.channel.Channel4 method), 43

data\_type\_format4() (in module mdfreader.channel), 46

dataFormat (mdfreader.channel.Channel3 attribute), 39

dataGroup (mdfreader.channel.Channel4 attribute), 43

dataGroup (mdfreader.mdf3reader.Record attribute), 20

dataGroup (mdfreader.mdf4reader.Record attribute), 28

dataRecordName (mdfreader.mdf3reader.Record attribute), 20

dataRecordName (mdfreader.mdf4reader.Record attribute), 28

decompress\_data\_block() (mdfreader.mdfinfo4.DZBlock static method), 34

decompression() (mdfreader.mdf.CompressedData method), 11

desc (mdfreader.channel.Channel3 attribute), 39

desc() (mdfreader.channel.Channel4 method), 43

DGBlock (class in mdfreader.mdfinfo4), 34

DLBlock (class in mdfreader.mdfinfo4), 34

DTBlock (class in mdfreader.mdfinfo4), 34

dtype (mdfreader.mdf.CompressedData attribute), 11

DZBlock (class in mdfreader.mdfinfo4), 34

## E

EVBlock (class in mdfreader.mdfinfo4), 35

export\_to\_csv() (mdfreader.mdfreader.Mdf method), 4, 6

export\_to\_excel() (mdfreader.mdfreader.Mdf method), 4, 6

export\_to\_hdf5() (mdfreader.mdfreader.Mdf method), 4, 6

export\_to\_matlab() (mdfreader.mdfreader.Mdf method), 4, 7

export\_to\_NetCDF() (mdfreader.mdfreader.Mdf method), 4, 6

export\_to\_xlsx() (mdfreader.mdfreader.Mdf method), 4, 7

## F

FHBlock (class in mdfreader.mdfinfo4), 35

fid (mdfreader.mdf.MdfSkeleton attribute), 12

fid (mdfreader.mdf3reader.DATA attribute), 17

fid (mdfreader.mdf4reader.Data attribute), 25

fid (mdfreader.mdfinfo3.Info3 attribute), 23

fid (mdfreader.mdfinfo4.Info4 attribute), 36

fid (mdfreader.mdfreader.MdfInfo attribute), 9

file\_metadata (mdfreader.mdf.MdfSkeleton attribute), 12

file\_metadata (mdfreader.mdf3reader.Mdf3 attribute), 18

file\_metadata (mdfreader.mdf4reader.Mdf4 attribute), 26

file\_metadata (mdfreader.mdfreader.Mdf attribute), 4

file\_name (mdfreader.mdfinfo3.Info3 attribute), 23

fileName (mdfreader.mdf.MdfSkeleton attribute), 12

fileName (mdfreader.mdf3reader.Mdf3 attribute), 18

fileName (mdfreader.mdf4reader.Mdf4 attribute), 26

fileName (mdfreader.mdfinfo3.Info3 attribute), 23

fileName (mdfreader.mdfinfo4.Info4 attribute), 36

fileName (mdfreader.mdfreader.Mdf attribute), 3

fileName (mdfreader.mdfreader.MdfInfo attribute), 9

filterChannelNames (mdfreader.mdf.MdfSkeleton attribute), 12

filterChannelNames (mdfreader.mdf3reader.Mdf3 attribute), 18

filterChannelNames (mdfreader.mdf4reader.Mdf4 attribute), 26

filterChannelNames (mdfreader.mdfinfo3.Info3 attribute), 23

filterChannelNames (mdfreader.mdfreader.MdfInfo attribute), 9

Flags (mdfreader.mdf4reader.Record attribute), 28



## G

generate\_chunks() (mdfreader.mdf4reader.Record method), 28

get\_channel() (mdfreader.mdf.MdfSkeleton method), 12

get\_channel\_conversion() (mdfreader.mdf.MdfSkeleton method), 13

get\_channel\_data() (mdfreader.mdfreader.Mdf method), 4, 7

get\_channel\_desc() (mdfreader.mdf.MdfSkeleton method), 13

get\_channel\_master() (mdfreader.mdf.MdfSkeleton method), 13

get\_channel\_master\_type() (mdfreader.mdf.MdfSkeleton method), 13

get\_channel\_name\_4() (mdfreader.mdf4reader.Mdf4 method), 27

get\_channel\_unit() (mdfreader.mdf.MdfSkeleton method), 13

get\_channel\_unit() (mdfreader.mdfreader.Mdf method), 4

get\_invalid\_bit() (mdfreader.mdf.MdfSkeleton method), 13

get\_invalid\_channel() (mdfreader.mdf.MdfSkeleton method), 13

## H

has\_invalid\_bit() (mdfreader.channel.Channel4 method), 43

HDBlock (class in mdfreader.mdfinfo4), 35

hiddenBytes (mdfreader.mdf3reader.Record attribute), 20

hiddenBytes (mdfreader.mdf4reader.Record attribute), 28

HLBlock (class in mdfreader.mdfinfo4), 35

## I

IDBlock (class in mdfreader.mdfinfo4), 35

info (mdfreader.mdf.MdfSkeleton attribute), 13

Info3 (class in mdfreader.mdfinfo3), 23

Info4 (class in mdfreader.mdfinfo4), 35

initialise\_reccarray() (mdfreader.mdf4reader.Record method), 28

invalid\_bit() (mdfreader.channel.Channel4 method), 43

invalid\_channel (mdfreader.mdf4reader.Record attribute), 29

is\_ca\_block() (mdfreader.channel.Channel4 method), 43

isnumeric() (mdfreader.channel.Channel4 method), 43

## K

keep\_channels() (mdfreader.mdfreader.Mdf method), 4, 7

## L

list\_channels() (mdfreader.mdfreader.MdfInfo method), 9

list\_channels3() (mdfreader.mdfinfo3.Info3 method), 23

list\_channels\_4() (mdfreader.mdfinfo4.Info4 method), 36

little\_endian() (mdfreader.channel.Channel4 method), 43

load() (mdfreader.mdf4reader.Data method), 25

load() (mdfreader.mdfinfo4.CABlock method), 33

load() (mdfreader.mdfinfo4.CommentBlock method), 34

load() (mdfreader.mdfinfo4.DTBBlock method), 34

load() (mdfreader.mdfinfo4.HLBlock method), 35

load\_info() (mdfreader.mdf3reader.Record method), 20

load\_info() (mdfreader.mdf4reader.Record method), 29

load\_sorted() (mdfreader.mdf3reader.DATA method), 17

load\_unsorted() (mdfreader.mdf3reader.DATA method), 17, 18

loadInfo() (mdfreader.mdf3reader.Record method), 20

## M

master (mdfreader.mdf3reader.Record attribute), 20

master (mdfreader.mdf4reader.Record attribute), 29

masterChannelList (mdfreader.mdf.MdfSkeleton attribute), 13

masterChannelList (mdfreader.mdf3reader.Mdf3 attribute), 18

masterChannelList (mdfreader.mdf4reader.Mdf4 attribute), 26

masterChannelList (mdfreader.mdfreader.Mdf attribute), 4

Mdf (class in mdfreader.mdfreader), 3

Mdf3 (class in mdfreader.mdf3reader), 18

Mdf4 (class in mdfreader.mdf4reader), 26

MdfInfo (class in mdfreader.mdfreader), 9

mdfreader.channel (module), 39

mdfreader.mdf (module), 11

mdfreader.mdf3reader (module), 17

mdfreader.mdf4reader (module), 25

mdfreader.mdfinfo3 (module), 23

mdfreader.mdfinfo4 (module), 33

mdfreader.mdfreader (module), 3

MdfSkeleton (class in mdfreader.mdf), 11

mdfversion (mdfreader.mdfreader.MdfInfo attribute), 10

MDFVersionNumber (mdfreader.mdf.MdfSkeleton attribute), 11

MDFVersionNumber (mdfreader.mdf3reader.Mdf3 attribute), 18

MDFVersionNumber (mdfreader.mdf4reader.Mdf4 attribute), 26

MDFVersionNumber (mdfreader.mdfreader.Mdf attribute), 4

merge\_mdf() (mdfreader.mdfreader.Mdf method), 7

MLSD (mdfreader.mdf4reader.Record attribute), 28

multiProc (mdfreader.mdf.MdfSkeleton attribute), 13

multiProc (mdfreader.mdf3reader.Mdf3 attribute), 18

multiProc (mdfreader.mdf4reader.Mdf4 attribute), 26

multiProc (mdfreader.mdfreader.Mdf attribute), 4

N

name (mdfreader.channel.Channel3 attribute), 39

name (mdfreader.channel.Channel4 attribute), 43

`native_data_format()` (mdfreader.channel.Channel4 method), 43

`nBytes` (mdfreader.channel.Channel3 attribute), 39

`nBytes` (mdfreader.channel.Channel4 attribute), 43

`numberOfRecords` (mdfreader.mdf3reader.Record attribute), 19

`numberOfRecords` (mdfreader.mdf4reader.Record attribute), 29

`numpy_format()` (mdfreader.channel.Channel4 method), 43

`numpyDataRecordFormat` (mdfreader.mdf3reader.Record attribute), 20

`numpyDataRecordFormat` (mdfreader.mdf4reader.Record attribute), 29

## P

`plot()` (mdfreader.mdfreader.Mdf method), 4, 8

`plot_all()` (mdfreader.mdfreader.Mdf method), 8

`pointer_to_data` (mdfreader.mdf4reader.Data attribute), 26

`pointerToData` (mdfreader.mdf3reader.DATA attribute), 17

`pos_bit_beg()` (mdfreader.channel.Channel4 method), 43

`pos_bit_end()` (mdfreader.channel.Channel4 method), 44

`pos_byte_beg()` (mdfreader.channel.Channel4 method), 44

`pos_byte_end()` (mdfreader.channel.Channel4 method), 44

`posByteBeg` (mdfreader.channel.Channel3 attribute), 40

`posByteEnd` (mdfreader.channel.Channel3 attribute), 40

`PythonVersion` (in module mdfreader.channel), 39

`PythonVersion` (in module mdfreader.mdf3reader), 17

`PythonVersion` (in module mdfreader.mdf4reader), 25

`PythonVersion` (in module mdfreader.mdfinfo3), 23

`PythonVersion` (in module mdfreader.mdfinfo4), 33

`PythonVersion` (in module mdfreader.mdfreader), 3

## R

`read()` (mdfreader.mdf3reader.DATA method), 17, 18

`read()` (mdfreader.mdf4reader.Data method), 26

`read()` (mdfreader.mdfinfo4.CABlock method), 33

`read()` (mdfreader.mdfinfo4.CGBlock method), 33

`read()` (mdfreader.mdfinfo4.DGBlock method), 34

`read()` (mdfreader.mdfinfo4.DLBlock method), 34

`read()` (mdfreader.mdfinfo4.DZBlock method), 35

`read()` (mdfreader.mdfinfo4.FHBlock method), 35

`read()` (mdfreader.mdfinfo4.HDBlock method), 35

`read()` (mdfreader.mdfinfo4.HLBlock method), 35

`read()` (mdfreader.mdfinfo4.IDBlock method), 35

`read()` (mdfreader.mdfreader.Mdf method), 8

`read3()` (mdfreader.mdf3reader.Mdf3 method), 18, 19

`read4()` (mdfreader.mdf4reader.Mdf4 method), 27

`read_all_channels_sorted_record()` (mdfreader.mdf4reader.Record method), 29

`read_at_block()` (mdfreader.mdfinfo4.Info4 static method), 36

`read_cc()` (mdfreader.mdfinfo4.CCBBlock method), 33

`read_cc_block()` (in module mdfreader.mdfinfo3), 24

`read_ce_block()` (in module mdfreader.mdfinfo3), 24

`read_cg_block()` (in module mdfreader.mdfinfo3), 24

`read_cg_block()` (mdfreader.mdfinfo3.Info3 method), 23

`read_cg_block()` (mdfreader.mdfinfo4.Info4 method), 36

`read_channels_from_bytes()` (mdfreader.mdf4reader.Record method), 29

`read_channels_from_bytes_fallback()` (mdfreader.mdf4reader.Record method), 29

`read_cm()` (mdfreader.mdfinfo4.CommentBlock method), 34

`read_cn()` (mdfreader.mdfinfo4.CNBlock method), 34

`read_cn_block()` (in module mdfreader.mdfinfo3), 24

`read_cn_block()` (mdfreader.mdfinfo4.Info4 method), 36

`read_composition()` (mdfreader.mdfinfo4.Info4 method), 37

`read_dg_block()` (in module mdfreader.mdfinfo3), 24

`read_dg_block()` (mdfreader.mdfinfo4.Info4 method), 37

`read_hd_block()` (in module mdfreader.mdfinfo3), 24

`read_info()` (mdfreader.mdfinfo4.Info4 method), 37

`read_info()` (mdfreader.mdfreader.MdfInfo method), 10

`read_info3()` (mdfreader.mdfinfo3.Info3 method), 24

`read_not_all_channels_sorted_record()` (mdfreader.mdf4reader.Record method), 30

`read_record()` (mdfreader.mdf4reader.Data method), 26

`read_record_bits()` (mdfreader.mdf3reader.Record method), 20

`read_record_buf()` (mdfreader.mdf3reader.Record method), 21

`read_record_buf()` (mdfreader.mdf4reader.Record method), 30

`read_si()` (mdfreader.mdfinfo4.SIBlock method), 38

`read_sorted_record()` (mdfreader.mdf3reader.Record method), 21

`read_sorted_record()` (mdfreader.mdf4reader.Record method), 30

`read_sr_block()` (mdfreader.mdfinfo4.Info4 static method), 37

`read_tx_block()` (in module mdfreader.mdfinfo3), 24

`readRecordBits()` (mdfreader.mdf3reader.Record method), 20

`readRecordBuf()` (mdfreader.mdf3reader.Record method), 20

`readSortedRecord()` (mdfreader.mdf3reader.Record method), 20

`recAttributeName` (mdfreader.channel.Channel3 attribute), 40

`Record` (class in mdfreader.mdf3reader), 19

`Record` (class in mdfreader.mdf4reader), 28

`record_attribute_name()` (mdfreader.channel.Channel4 method), 44

- record\_id\_size() (mdfreader.channel.Channel4 method), 44
- RecordFormat (mdfreader.channel.Channel3 attribute), 40
- recordID (mdfreader.mdf3reader.Record attribute), 19
- recordID (mdfreader.mdf4reader.Record attribute), 31
- recordIDCFormat (mdfreader.mdf4reader.Record attribute), 31
- recordIDnumber (mdfreader.mdf3reader.Record attribute), 20
- recordIDsize (mdfreader.mdf4reader.Record attribute), 31
- recordLength (mdfreader.mdf3reader.Record attribute), 19
- recordLength (mdfreader.mdf4reader.Record attribute), 31
- recordToChannelMatching (mdfreader.mdf3reader.Record attribute), 20
- recordToChannelMatching (mdfreader.mdf4reader.Record attribute), 31
- remove\_channel() (mdfreader.mdf.MdfSkeleton method), 13
- remove\_channel\_conversion() (mdfreader.mdf.MdfSkeleton method), 14
- rename\_channel() (mdfreader.mdf.MdfSkeleton method), 14
- resample() (mdfreader.mdfreader.Mdf method), 4, 9
- ## S
- set() (mdfreader.channel.Channel4 method), 44
- set\_CANOpen() (mdfreader.channel.Channel4 method), 45
- set\_channel\_attachment() (mdfreader.mdf.MdfSkeleton method), 14
- set\_channel\_conversion() (mdfreader.mdf.MdfSkeleton method), 14
- set\_channel\_data() (mdfreader.mdf.MdfSkeleton method), 14
- set\_channel\_desc() (mdfreader.mdf.MdfSkeleton method), 14
- set\_channel\_master() (mdfreader.mdf.MdfSkeleton method), 14
- set\_channel\_master\_type() (mdfreader.mdf.MdfSkeleton method), 14
- set\_channel\_unit() (mdfreader.mdf.MdfSkeleton method), 15
- set\_invalid\_bit() (mdfreader.mdf.MdfSkeleton method), 15
- set\_invalid\_bytes() (mdfreader.channel.Channel4 method), 45
- set\_invalid\_channel() (mdfreader.mdf.MdfSkeleton method), 15
- SIBlock (class in mdfreader.mdfinfo4), 38
- signal\_data\_type() (mdfreader.channel.Channel4 method), 45
- signalDataType (mdfreader.channel.Channel3 attribute), 39
- SRBlock (class in mdfreader.mdfinfo4), 38
- ## T
- type (mdfreader.channel.Channel4 attribute), 46
- type (mdfreader.mdf4reader.Data attribute), 26
- ## U
- unique\_id() (mdfreader.mdfinfo4.Info4 method), 37
- unit (mdfreader.channel.Channel3 attribute), 39
- unit() (mdfreader.channel.Channel4 method), 46
- ## V
- VLSD (mdfreader.mdf4reader.Record attribute), 28
- VLSD\_CG (mdfreader.mdf4reader.Record attribute), 28
- VLSD\_CG\_Flag (mdfreader.channel.Channel4 attribute), 40
- ## W
- write() (mdfreader.mdfinfo4.CABlock method), 33
- write() (mdfreader.mdfinfo4.CGBlock method), 33
- write() (mdfreader.mdfinfo4.CNBlock method), 34
- write() (mdfreader.mdfinfo4.CommentBlock method), 34
- write() (mdfreader.mdfinfo4.DGBlock method), 34
- write() (mdfreader.mdfinfo4.DLBlock method), 34
- write() (mdfreader.mdfinfo4.DTBlock method), 34
- write() (mdfreader.mdfinfo4.DZBlock method), 35
- write() (mdfreader.mdfinfo4.FHBlock method), 35
- write() (mdfreader.mdfinfo4.HDBlock method), 35
- write() (mdfreader.mdfinfo4.HLBlock method), 35
- write() (mdfreader.mdfinfo4.IDBlock static method), 35
- write() (mdfreader.mdfreader.Mdf method), 4, 9
- write3() (mdfreader.mdf3reader.Mdf3 method), 19
- write4() (mdfreader.mdf4reader.Mdf4 method), 27, 28
- ## Z
- zipfile (mdfreader.mdf.MdfSkeleton attribute), 15
- zipfile (mdfreader.mdfinfo4.Info4 attribute), 38
- zipfile (mdfreader.mdfreader.MdfInfo attribute), 10