mdfreader Documentation

Release 2.8

Aymeric Rateau

Oct 06, 2018

CONTENTS

1 mdf module documentation **3**
- 1.1 Platform and python version . . . 3
- 1.2 Dependencies . . . 3
- 1.3 mdf module . . . 3

2 mdfreader module documentation **9**
- 2.1 Platform and python version . . . 9
- 2.2 Dependencies . . . 9

3 mdf3reader module documentation **17**
- 3.1 Platform and python version . . . 17
- 3.2 Dependencies . . . 17

4 mdfinfo3 module documentation **25**
- 4.1 Platform and python version . . . 25
- 4.2 Dependencies . . . 25

5 mdf4reader module documentation **27**
- 5.1 Platform and python version . . . 27
- 5.2 Dependencies . . . 27

6 mdfinfo4 module documentation **37**
- 6.1 Platform and python version . . . 37
- 6.2 Dependencies . . . 37

7 channel module documentation **43**
- 7.1 Platform and python version . . . 43
- 7.2 Dependencies . . . 43

8 Indices and tables **51**

Python Module Index **53**

Index **55**

Contents:

CHAPTER

ONE

MDF MODULE DOCUMENTATION

mdf_skeleton module describing basic mdf structure and methods

Created on Thu Sept 24 2015

1.1 Platform and python version

With Unix and Windows for python 2.6+ and 3.2+

Author Aymeric Rateau

1.2 Dependencies

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

1.3 mdf module

class `mdfreader.mdf.`**`compressed_data`**

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.**mdf_skeleton**() *-> 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(*dataGroup*, *channel_name*, *data*, *master_channel*, *master_type=1*, *unit="*, *description="*, *conversion=None*, *info=None*, *compression=False*, *id=None*)

adds channel to mdf dict.

Parameters

- **dataGroup** (*int*) – dataGroup number. Is appended to master name for non unique channel names
- **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
- **id** (*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

fid

fileName

file_metadata

filterChannelNames

getChannel(*channelName*)
Extract channel dict from mdf structure

Parameters **channelName** (*str*) – channel name

Returns

Return type channel dictionnary containing data, description, unit, etc.

getChannelConversion(*channelName*)
Extract channel conversion dict from mdf structure

Parameters **channelName** (*str*) – channel name

Returns

Return type channel conversion dict

getChannelDesc(*channelName*)
Extract channel description information from mdf structure

Parameters **channelName** (*str*) – channel name

Returns

Return type channel description string

getChannelMaster(*channelName*)
Extract channel master name from mdf structure

Parameters **channelName** (*str*) – channel name

Returns

Return type channel master name string

getChannelMasterType(*channelName*)
Extract channel master type information from mdf structure

Parameters **channelName** (*str*) – channel name

Returns **channel mater type integer**

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

getChannelUnit(*channelName*)
Returns channel unit string Implemented for a future integration of pint

Parameters **channelName** (*str*) – channel name

Returns unit string description

Return type str

getInvalidBit(*channelName*)

getInvalidChannel(*channelName*)

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(*channelName*)
removes conversion key from mdf channel dict.

Parameters **channelName** (*str*) – channel name

Returns

Return type removed value from dict

rename_channel(*channelName*, *newname*)
Modifies name of channel

Parameters

- **channelName** (*str*) – channel name
- **newname** (*str*) – new channel name

setChannelAttachment(*channelName*, *attachment*)
Modifies channel attachment

Parameters

- **channelName** (*str*) – channel name
- **attachment** – channel attachment

setChannelConversion(*channelName*, *conversion*)
Modifies conversion dict of channel

Parameters

- **channelName** (*str*) – channel name
- **conversion** (*dict*) – conversion dictionnary

setChannelData(*channelName*, *data*, *compression=False*)
Modifies data of channel

Parameters

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

setChannelDesc(*channelName*, *desc*)
Modifies description of channel

Parameters

- **`channelName`** (*`str`*) – channel name
- **`desc`** (*`str`*) – channel description

`setChannelMaster`(*channelName*, *master*)
Modifies channel master name

Parameters

- **`channelName`** (*`str`*) – channel name
- **`master`** (*`str`*) – master channel name

`setChannelMasterType`(*channelName*, *masterType*)
Modifies master channel type

Parameters

- **`channelName`** (*`str`*) – channel name
- **`masterType`** (*`int`*) – master channel type

`setChannelUnit`(*channelName*, *unit*)
Modifies unit of channel

Parameters

- **`channelName`** (*`str`*) – channel name
- **`unit`** (*`str`*) – channel unit

`setInvalidBit`(*channelName*, *bit_position*)

`setInvalidChannel`(*channelName*, *invalid_channel*)

`zipfile`

CHAPTER

TWO

MDFREADER MODULE DOCUMENTATION

Measured Data Format file reader main module

2.1 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

2.2 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+

`mdfreader module`

`--------------------------`

`class` `mdfreader.mdfreader.`**`mdf`**(*fileName=None*, *channelList=None*, *convertAfterRead=True*, *filterChannelNames=False*, *noDataLoading=False*, *compression=False*, *convertTables=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( fileName = None, multiProc = False, channelList=None, convertAfterRead=True, fil`

noDataLoading=False, compression=False) reads mdf file version 3.x and 4.x

`write`(*fileName=None*)
writes simple mdf file

`getChannelData`(*channelName*)
returns channel numpy array

`convertAllChannel`()
converts all channel data according to CCBlock information

`getChannelUnit`(*channelName*)
returns channel unit

`plot`(*channels*)
Plot channels with Matplotlib

`resample`(*samplingTime = 0.1*, *masterChannel=None*)
Resamples all data groups

`exportToCSV`(*filename = None*, *sampling = 0.1*)
Exports mdf data into CSV file

`exportToNetCDF`(*filename = None*, *sampling = None*)
Exports mdf data into netcdf file

`exportToHDF5`(*filename = None*, *sampling = None*)
Exports mdf class data structure into hdf5 file

`exportToMatlab`(*filename = None*)
Exports mdf class data structure into Matlab file

`exportToExcel`(*filename = None*)
Exports mdf data into excel 95 to 2003 file

`exportToXlsx`(*filename=None*)
Exports mdf data into excel 2007 and 2010 file

`convertToPandas`(*sampling=None*)
converts mdf data structure into pandas dataframe(s)

`keepChannels`(*channelList*)
keeps only list of channels and removes the other channels

mergeMdf(mdfClass):
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(channelName='master3')
>>> yop.exportoCSV(sampling=0.01)
>>> yop.exportNetCDF()
>>> yop.exporttoHDF5()
>>> yop.exporttoMatlab()
>>> yop.exporttoExcel()
>>> yop.exporttoXlsx()
>>> yop.convertToPandas() # converts data groups into pandas dataframes
>>> yop.write() # writes mdf file
# drops all the channels except the one in argument
>>> yop.keepChannels({'channel1','channel2','channel3'})
>>> yop.getChannelData('channelName') # returns channel numpy array
```

allPlot()

convertAllChannel()
Converts all channels from raw data to converted data according to CCBlock information Converted data will take more memory.

convertToPandas(*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

copy()
make a shallow copy a mdf class

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).

exportToCSV(*filename=None*, *sampling=None*)
Exports mdf data into CSV file

Parameters

- **filename** (*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 fille 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

exportToExcel(*filename=None*)
Exports mdf data into excel 95 to 2003 file

Parameters **filename** (*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+

exportToHDF5(*filename=None*, *sampling=None*, *compression=None*, *compression_opts=None*)
Exports mdf class data structure into hdf5 file

Parameters

- **filename** (*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

`exportToMatlab` (*filename=None*)
Export mdf data into Matlab file format 5, tentatively compressed

Parameters **`filename`** (*`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

`exportToNetCDF` (*filename=None*, *sampling=None*)
Exports mdf data into netcdf file

Parameters

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

Notes

Dependency: scipy

`exportToXlsx` (*filename=None*)
Exports mdf data into excel 2007 and 2010 file

Parameters **`filename`** (*`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

`getChannelData` (*channelName*, *raw_data=False*)
Return channel numpy array

Parameters

- **`channelName`** (*`str`*) – channel name
- **`raw_data`** (*`bool`*) – flag to return non converted data
- **`Returns`** –
- **`-----------`** –

- **array** (*numpy*) – converted, if not already done, data corresponding to channel name

Notes

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

keepChannels(*channelList*)
keeps only list of channels and removes the other channels

Parameters **channelList** (*list of str*) – list of channel names

mergeMdf(*mdfClass*)
Merges data of 2 mdf classes

Parameters **mdfClass** (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

read(*fileName=None*, *multiProc=False*, *channelList=None*, *convertAfterRead=True*, *filterChannelNames=False*, *noDataLoading=False*, *compression=False*, *metadata=2*)
reads mdf file version 3.x and 4.x

Parameters

- **fileName** (*str, optional*) – file name
- **multiProc** (*bool*) – flag to activate multiprocessing of channel data conversion
- **channelList** (*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
- **convertAfterRead** (*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()
- **filterChannelNames** (*bool, optional*) – flag to filter long channel names from its module names separated by '.'
- **noDataLoading** (*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

`resample`(*samplingTime=None*, *masterChannel=None*)
Resamples all data groups into one data group having defined sampling interval or sharing same master channel

Parameters

- **`samplingTime`** (*`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**`** –
- **`masterChannel`** (*`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`(*fileName=None*, *compression=False*)
Writes simple mdf file, same format as originally read, default is 4.x

Parameters

- **`fileName`** (*`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

listChannels(*fileName=None*)

Read MDF file blocks and returns a list of contained channels

Parameters **fileName** (*string*) – file name

Returns **nameList** – list of channel names

Return type list of string

mdfversion

readinfo(*fileName=None, fid=None, minimal=0*)

Reads MDF file and extracts its complete structure

Parameters

- **fileName** (*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

CHAPTER

THREE

MDF3READER MODULE DOCUMENTATION

Measured Data Format file reader module for version 3.x

3.1 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

3.2 Dependencies

- Python >2.6, >3.2 <http://www.python.org>
- Numpy >1.6 <http://numpy.scipy.org>
- Sympy to convert channels with formula

`mdfreader.mdf3reader.`**`PythonVersion`**

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

`mdf3reader module`

`--------------------------`

`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

`addRecord`(*record*)

Adds a new record in DATA class dict

read(*channelSet*)
Reads data block

loadSorted(*record*, *nameList=None*)
Reads sorted data block from record definition

loadUnSorted(*nameList=None*)
Reads unsorted data block, not yet implemented

addRecord(*record*)
Adds a new record in DATA class dict

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

loadSorted(*record*, *nameList=None*)
Reads sorted data block from record definition

Parameters

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

Returns

Return type numpy recarray of data

loadUnSorted(*nameList=None*)
Reads unsorted data block from record definition

Parameters

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

Returns

Return type numpy recarray of data

read(*channelSet*, *filename*)
Reads data block

Parameters

- **channelSet** (*set of str, optional*) – list of channel names
- **filename** (*str*) – name of file

mdfreader.mdf3reader.**expConv**(*data*, *conv*)
apply exponential conversion to data

Parameters

- **data** (*numpy 1D array*) – raw data to be converted to physical value
- **conv** (*mdfinfo3.info3 conversion block ('CCBlock') dict*) –

Returns

Return type converted data to physical value

mdfreader.mdf3reader.**formulaConv**(*data*, *conv*)
apply formula conversion to data

Parameters

- **data** (*numpy 1D array*) – raw data to be converted to physical value

- **conv** (*mdfinfo3.info3 conversion block ('CCBlock') dict*) –

Returns

Return type converted data to physical value

Notes

Requires sympy module

`mdfreader.mdf3reader.`**`linearConv`**(*data*, *conv*)
apply linear conversion to data

Parameters

- **data** (*numpy 1D array*) – raw data to be converted to physical value
- **conv** (*mdfinfo3.info3 conversion block ('CCBlock') dict*) –

Returns

Return type converted data to physical value

`mdfreader.mdf3reader.`**`logConv`**(*data*, *conv*)
apply logarithmic conversion to data

Parameters

- **data** (*numpy 1D array*) – raw data to be converted to physical value
- **conv** (*mdfinfo3.info3 conversion block ('CCBlock') dict*) –

Returns

Return type converted data to physical value

class `mdfreader.mdf3reader.`**`mdf3`**(*fileName=None*, *channelList=None*, *convertAfterRead=True*, *filterChannelNames=False*, *noDataLoading=False*, *compression=False*, *convertTables=False*, *metadata=2*)
Bases: *mdfreader.mdf.mdf_skeleton*

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

`_getChannelData3` (*channelName*)
Returns channel numpy array

`_convertChannel3` (*channelName*)
converts specific channel from raw to physical data according to CCBlock information

`_convertAllChannel3` ()
Converts all channels from raw data to converted data according to CCBlock information

`write3` (*fileName=None*)
Writes simple mdf 3.3 file

`read3` (*fileName=None*, *info=None*, *multiProc=False*, *channelList=None*, *convertAfterRead=True*, *filterChannelNames=False*, *compression=False*, *metadata=2*)
Reads mdf 3.x file data and stores it in dict

Parameters

- **`fileName`** (*`str, optional`*) – file name
- **`info`** (*`mdfinfo3.info3 class`*) – info3 class containing all MDF Blocks
- **`multiProc`** (*`bool`*) – flag to activate multiprocessing of channel data conversion
- **`channelList`** (*`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
- **`convertAfterRead`** (*`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()
- **`filterChannelNames`** (*`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` (*fileName=None*)
Writes simple mdf 3.3 file

Parameters **`fileName`** (*`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

`mdfreader.mdf3reader.`**`polyConv`** (*data*, *conv*)
apply polynomial conversion to data

Parameters

- **`data`** (*`numpy 1D array`*) – raw data to be converted to physical value

- **conv** (*mdfinfo3.info3 conversion block ('CCBlock') dict*) –

Returns

Return type converted data to physical value

`mdfreader.mdf3reader.`**`rationalConv`**(*data*, *conv*)

apply rational conversion to data

Parameters

- **data** (*numpy 1D array*) – raw data to be converted to physical value
- **conv** (*mdfinfo3.info3 conversion block ('CCBlock') dict*) –

Returns

Return type converted data to physical value

class `mdfreader.mdf3reader.`**`record`**(*dataGroup*, *channelGroup*)

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*)

addChannel(*info*, *channelNumber*)
add a channel in class

Parameters

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

loadInfo(*info*)
gathers records related from info class

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

readRecordBits(*bita*, *channelSet=None*)
read stream of record bits by bits in case of not aligned or hidden bytes

Parameters

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

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

Return type dict

readRecordBuf(*buf*, *channelSet=None*)
read stream of record bytes

Parameters

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

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

Return type dict

readSortedRecord(*fid*, *pointer*, *channelSet=None*)
reads record, only one channel group per datagroup

Parameters

- **fid** (*float*) – file identifier
- **pointer** – position in file of data block beginning
- **channelSet** (*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.

`mdfreader.mdf3reader.`**`tabConv`**(*data*, *conv*)

apply Tabular conversion to data

Parameters

- **`data`** (*`numpy 1D array`*) – raw data to be converted to physical value
- **`conv`** (*`mdfinfo3.info3 conversion block ('CCBlock') dict`*) –

Returns

Return type converted data to physical value

`mdfreader.mdf3reader.`**`tabInterpConv`**(*data*, *conv*)

apply Tabular interpolation conversion to data

Parameters

- **`data`** (*`numpy 1D array`*) – raw data to be converted to physical value
- **`conv`** (*`mdfinfo3.info3 conversion block ('CCBlock') dict`*) –

Returns

Return type converted data to physical value

`mdfreader.mdf3reader.`**`textRangeTableConv`**(*data*, *conv*)

apply text range table conversion to data

Parameters

- **`data`** (*`numpy 1D array`*) – raw data to be converted to physical value
- **`conv`** (*`mdfinfo3.info3 conversion block ('CCBlock') dict`*) –

Returns

Return type converted data to physical value

`mdfreader.mdf3reader.`**`textTableConv`**(*data*, *conv*)

apply text table conversion to data

Parameters

- **`data`** (*`numpy 1D array`*) – raw data to be converted to physical value
- **`conv`** (*`mdfinfo3.info3 conversion block ('CCBlock') dict`*) –

Returns

Return type converted data to physical value

CHAPTER

FOUR

MDFINFO3 MODULE DOCUMENTATION

Measured Data Format blocks parser for version 3.x

Created on Thu Dec 9 12:57:28 2014

4.1 Platform and python version

With Unix and Windows for python 2.6+ and 3.2+

Author Aymeric Rateau

4.2 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+

`mdfinfo3 module`

`--------------------------`

`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`

`cleanDGinfo`(*dg*)

delete CN,CC and CG blocks related to data group

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

`fid`

`fileName`

`filterChannelNames`

`listChannels3`(*fileName=None, fid=None*)

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

fileName
str – file name

Returns

Return type list of channel names

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

Parameters

- **fid** (*float*) – file identifier
- **dg** (*int*) – datagroup number
- **channelSet** (*set*) – set of channel names to read
- **minimal** (*int*) – 0 will load every metadata 1 will load DG, CG, CN and CC 2 will load only DG

readinfo3 (*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.mdfinfo3.**read_cc_block** (*fid*, *pointer*)
channel conversion block reading

mdfreader.mdfinfo3.**read_ce_block** (*fid*, *pointer*)
reads source block

mdfreader.mdfinfo3.**read_cg_block** (*fid*, *pointer*)
channel block reading

mdfreader.mdfinfo3.**read_cn_block** (*fid*, *pointer*)
channel block reading

mdfreader.mdfinfo3.**read_dg_block** (*fid*, *pointer*)
data group block reading

mdfreader.mdfinfo3.**read_hd_block** (*fid*, *pointer*, *version=0*)
header block reading

mdfreader.mdfinfo3.**read_tx_block** (*fid*, *pointer*)
reads text block

CHAPTER

FIVE

MDF4READER MODULE DOCUMENTATION

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

5.1 Platform and python version

With Unix and Windows for python 2.6+ and 3.2+

Author Aymeric Rateau

Created on Thu Dec 10 12:57:28 2013

5.2 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+

`mdf4reader module`

`---------------------------`

`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`

`addRecord`(*record*)
Adds a new record in DATA class dict.

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

`fid`

load(*record*, *info*, *nameList=None*, *sortedFlag=True*, *vlsd=False*)
Reads data block from record definition

Parameters

- **class** (*info*) – channel group definition listing record channel classes
- **class** – contains blocks
- **nameList** (*list of str, optional*) – list of channel names
- **sortedFlag** (*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

pointerTodata

read(*channelSet*, *info*, *filename*)
Reads data block

Parameters

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

readRecord(*recordID*, *info*, *buf*, *channelSet=None*)
read record from a buffer

Parameters

- **recordID** (*int*) – record identifier
- **class** (*info*) – contains blocks
- **buf** (*str*) – buffer of data from file to be converted to channel raw data
- **channelSet** (*set of str*) – setof channel names to be read

type

mdfreader.mdf4reader.**DATABlock**(*record*, *info*, *parent_block*, *channelSet=None*, *nrecords=None*, *sortedFlag=True*, *vlsd=False*)
DATABlock converts raw data into arrays

Parameters

- **record** (*class*) – record class instance describing a channel group record
- **parent_block** (*class*) – MDFBlock class containing at least parent block header
- **channelSet** (*set of str, optional*) – defines set of channels to only read, can be slow but saves memory, for big files
- **nrecords** (*int, optional*) – number of records to read

- **`sortedFlag`** (*`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 a recarray containing the channels data

Notes

This function will read DTBlock, RDBlock, DZBlock (compressed), RDBlock (VLSD), sorted or unsorted

`mdfreader.mdf4reader.`**`equalizeStringLength`**(*buf*)
Makes all strings in a list having same length by appending spaces strings.

Parameters **`buf`** (*`list of str`*) –

Returns

Return type list of str elements all having same length

`mdfreader.mdf4reader.`**`formulaConv`**(*vect*, *formula*)
apply formula conversion to data

Parameters

- **`vect`** (*`numpy 1D array`*) – raw data to be converted to physical value
- **`cc_val`** (*`mdfinfo4.info4 conversion block ('CCBlock') dict`*) –

Returns

Return type converted data to physical value

`mdfreader.mdf4reader.`**`linearConv`**(*vect*, *cc_val*)
apply linear conversion to data

Parameters

- **`vect`** (*`numpy 1D array`*) – raw data to be converted to physical value
- **`cc_val`** (*`mdfinfo4.info4 conversion block ('CCBlock') dict`*) –

Returns

Return type converted data to physical value

`class` `mdfreader.mdf4reader.`**`mdf4`**(*fileName=None*, *channelList=None*, *convertAfterRead=True*, *filterChannelNames=False*, *noDataLoading=False*, *compression=False*, *convertTables=False*, *metadata=2*)
Bases: *`mdfreader.mdf.mdf_skeleton`*

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

_getChannelData4 (*channelName*)
Returns channel numpy array

_convertChannel4 (*channelName*)
converts specific channel from raw to physical data according to CCBlock information

_convertAllChannel4 ()
Converts all channels from raw data to converted data according to CCBlock information

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

getChannelName4 (*name*, *path*)
finds mdf channel name from name and path

Parameters

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

read4 (*fileName=None*, *info=None*, *multiProc=False*, *channelList=None*, *convertAfterRead=True*, *compression=False*, *metadata=2*)
Reads mdf 4.x file data and stores it in dict

Parameters

- **fileName** (`str, optional`) – file name
- **info** (`mdfinfo4.info4 class`) – info4 class containing all MDF Blocks
- **multiProc** (`bool`) – flag to activate multiprocessing of channel data conversion
- **channelList** (`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

- **convertAfterRead** (*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()
- **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 (*fileName=None*, *compression=False*)

Writes simple mdf 4.1 file

Parameters

- **fileName** (*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

mdfreader.mdf4reader.**rationalConv** (*vect*, *cc_val*)

apply rational conversion to data

Parameters

- **vect** (*numpy 1D array*) – raw data to be converted to physical value
- **cc_val** (*mdfinfo4.info4 conversion block ('CCBlock') dict*) –

Returns

Return type converted data to physical value

mdfreader.mdf4reader.**readUnsorted** (*record*, *info*, *parent_block*, *channelSet=None*)

mdfreader.mdf4reader.**read_sdblock** (*signal_data_type*, *sdblock*, *sdblock_length*)

Reads vlsd channel from its SD Block bytes

Parameters

- **signal_data_type** (*int*) –
- **sdblock** (*bytes*) –
- **Block bytes** (*SD*) –
- **sdblock_length** (*int*) –
- **Block data length (header not included)** (*SD*) –

Returns

Return type array

class mdfreader.mdf4reader.**record** (*dataGroup*, *channelGroup*)

Bases: `list`

CANOpen

CGrecordLength

Flags

MLSD

VLSD

VLSD_CG

addChannel (*info*, *channelNumber*)
add a channel in class

Parameters

- **info** (*mdfinfo4.info4 class*) –
- **channelNumber** (*int*) – channel number in mdfinfo4.info4 class

byte_aligned

channelGroup

channelNames

dataGroup

dataRecordName

generate_chunks ()
calculate data split

Returns

Return type (nrecord_chunk, chunk_size)

hiddenBytes

initialise_recarray (*info*, *channelSet*, *nrecords*, *dtype=None*, *channels_indexes=None*)
Initialise recarray

Parameters

- **info** (*info class*) –
- **channelSet** (*set of str, optional*) – set of channel to read
- **nrecords** (*int*) – number of records
- **dtype** (*numpy dtype, optional*) –
- **channels_indexes** (*list of int, optional*) –

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

Return type numpy recarray

invalid_channel

loadInfo (*info*)
gathers records related from info class

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

master

numberOfRecords

numpyDataRecordFormat

readRecordBuf (*buf*, *info*, *channelSet=None*)
read stream of record bytes

Parameters

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

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

Return type dict

readSortedRecord (*fid*, *info*, *channelSet=None*)
reads record, only one channel group per datagroup

Parameters

- **fid** – file identifier
- **pointer** – position in file of data block beginning
- **channelSet** (*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.

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 recarray (attributes corresponding to channel name)

Return type numpy recarray

read_channels_from_bytes (*bita*, *info*, *channelSet=None*, *nrecords=None*, *dtype=None*, *channels_indexes=None*)
reads stream of record bytes using dataRead module if available otherwise bitarray

Parameters

- **bita** (*stream*) – stream of bytes
- **info** (*info class*) –
- **channelSet** (*set of str, optional*) – set of channel to read
- **nrecords** (*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_channels_from_bytes_fallback`(*bita*, *info*, *channelSet=None*, *nrecords=None*, *dtype=None*, *channels_indexes=None*)

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

Parameters

- **`bita`** (*`stream`*) – stream of bytes
- **`info`** (*`info class`*) –
- **`channelSet`** (*`set of str, optional`*) – set of channel to read
- **`nrecords`** (*`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*, *channelSet*)

reads channels from file listed in channelSet

Parameters

- **`fid`** – file identifier
- **`info`** (*`info class`*) –
- **`channelSet`** (*`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

`recordID`

`recordIDCFormat`

`recordIDsize`

`recordLength`

`recordToChannelMatching`

`mdfreader.mdf4reader.`**`textToTextConv`**(*vect*, *cc_ref*)

apply text to text conversion to data

Parameters

- **`vect`** (*`numpy 1D array`*) – raw data to be converted to physical value
- **`cc_ref`** (*`cc_ref from mdfinfo4.info4 conversion block ('CCBlock') dict`*) –

Returns

Return type converted data to physical value

mdfreader.mdf4reader.**textToValueConv**(*vect*, *cc_val*, *cc_ref*)
apply text to value conversion to data

Parameters

- **vect** (*numpy 1D array*) – raw data to be converted to physical value
- **cc_val** (*cc_val from mdfinfo4.info4 conversion block ('CCBlock') dict*) –
- **cc_ref** (*cc_ref from mdfinfo4.info4 conversion block ('CCBlock') dict*) –

Returns

Return type converted data to physical value

mdfreader.mdf4reader.**valueRangeToTextConv**(*vect*, *cc_val*, *cc_ref*)
apply value range to text conversion to data

Parameters

- **vect** (*numpy 1D array*) – raw data to be converted to physical value
- **cc_val** (*cc_val from mdfinfo4.info4 conversion block ('CCBlock') dict*) –
- **cc_ref** (*cc_ref from mdfinfo4.info4 conversion block ('CCBlock') dict*) –

Returns

Return type converted data to physical value

mdfreader.mdf4reader.**valueRangeToValueTableConv**(*vect*, *cc_val*)
apply value range to value table conversion to data

Parameters

- **vect** (*numpy 1D array*) – raw data to be converted to physical value
- **cc_val** (*mdfinfo4.info4 conversion block ('CCBlock') dict*) –

Returns

Return type converted data to physical value

mdfreader.mdf4reader.**valueToTextConv**(*vect*, *cc_val*, *cc_ref*)
apply value to text conversion to data

Parameters

- **vect** (*numpy 1D array*) – raw data to be converted to physical value
- **cc_val** (*cc_val from mdfinfo4.info4 conversion block ('CCBlock') dict*) –
- **cc_ref** (*cc_ref from mdfinfo4.info4 conversion block ('CCBlock') dict*) –

Returns

Return type converted data to physical value

mdfreader.mdf4reader.**valueToValueTableWInterpConv**(*vect*, *cc_val*)
apply value to value table with interpolation conversion to data

Parameters

- **`vect`** (*`numpy 1D array`*) – raw data to be converted to physical value
- **`cc_val`** (*`mdfinfo4.info4 conversion block ('CCBlock') dict`*) –

Returns

Return type converted data to physical value

`mdfreader.mdf4reader.`**`valueToValueTableWOInterpConv`**(*vect*, *cc_val*)

apply value to value table without interpolation conversion to data

Parameters

- **`vect`** (*`numpy 1D array`*) – raw data to be converted to physical value
- **`cc_val`** (*`mdfinfo4.info4 conversion block ('CCBlock') dict`*) –

Returns

Return type converted data to physical value

CHAPTER

SIX

MDFINFO4 MODULE DOCUMENTATION

Measured Data Format blocks paser for version 4.x

6.1 Platform and python version

With Unix and Windows for python 2.6+ and 3.2+

Created on Sun Dec 15 12:57:28 2013

Author Aymeric Rateau

6.2 Dependencies

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

`mdfreader.mdfinfo4.`**`PythonVersion`**

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

`mdfinfo4 module`

`--------------------------`

`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

`readCC`(*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.mdfinfo4.`**`CNBlock`**

Bases: `dict`

reads Channel block and saves in class dict

`readCN`(***kargs*)

`write`(*fid*)

class `mdfreader.mdfinfo4.`**`CommentBlock`**

Bases: `dict`

reads or writes Comment block and saves in class dict

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

`readCM`(***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.mdfinfo4.`**`DGBlock`**(*fid=None*, *pointer=None*)

Bases: `dict`

reads Data Group block and saves in class dict

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

`write`(*fid*)

class `mdfreader.mdfinfo4.`**`DLBlock`**

Bases: `dict`

reads Data List block

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

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

class `mdfreader.mdfinfo4.`**`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.mdfinfo4.`**`DZBlock`**

Bases: `dict`

reads Data List block

static decompress_datablock(*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.mdfinfo4.`**`EVBlock`**(*fid, pointer*)

Bases: `dict`

reads Event block and saves in class dict

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

Bases: `dict`

reads File History block and save in class dict

`read`(*fid, pointer*)

`write`(*fid*)

class `mdfreader.mdfinfo4.`**`HDBlock`**(*fid=None, pointer=64*)

Bases: `dict`

reads Header block and save in class dict

`read`(*fid=None, pointer=64*)

`write`(*fid*)

class `mdfreader.mdfinfo4.`**`HLBlock`**

Bases: `dict`

reads Header List block

`load`(*record_byte_offset, nRecords, position*)

read (*fid*)

write (*fid*, *data*)

class mdfreader.mdfinfo4.**IDBlock** (*fid=None*)

Bases: `dict`

reads or writes ID Block

read (*fid*)

reads IDBlock

write (*fid*)

Writes IDBlock

class mdfreader.mdfinfo4.**SIBlock**

Bases: `dict`

reads Source Information block and saves in class dict

readSI (*fid*, *pointer*)

class mdfreader.mdfinfo4.**SRBlock** (*fid*, *pointer*)

Bases: `dict`

reads Sample Reduction block and saves in class dict

class mdfreader.mdfinfo4.**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`

cleanDGinfo (*dg*)

delete CN,CC and CG blocks related to data group

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

fid

fileName

listChannels4 (*fileName=None*, *fid=None*)

Read MDF file and extract its complete structure

Parameters **fileName** (*str*) – file name

Returns

Return type list of channel names contained in file

readATBlock (*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

readCGBlock (*fid*, *dg*, *channelNameList=False*, *minimal=0*)
reads Channel Group blocks

Parameters

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

readCNBlock (*fid*, *dg*, *cg*, *channelNameList=False*, *minimal=0*)
reads Channel blocks

Parameters

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

readComposition (*fid*, *dg*, *cg*, *MLSDChannels*)
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
- **MLSDChannels** (*list of int*) – channel numbers

Returns

Return type MLSDChannels list of appended Maximum Length Sampling Data channels

readDGBlock (*fid*, *channelNameList=False*, *minimal=0*)
reads Data Group Blocks

Parameters

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

readSRBlock (*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

`readinfo` (*fid*, *minimal*)
read all file blocks except data

Parameters

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

`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`

CHAPTER

SEVEN

CHANNEL MODULE DOCUMENTATION

Measured Data Format file reader module.

7.1 Platform and python version

With Unix and Windows for python 2.7 and 3.4+

Author Aymeric Rateau

Created on Wed Oct 04 21:13:28 2017

7.2 Dependencies

- Python >2.6, >3.4 <http://www.python.org>
- Numpy >1.6 <http://numpy.scipy.org>

`mdfreader.channel.`**`PythonVersion`**

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

`channel module`

`--------------------------`

`class` `mdfreader.channel.`**`Channel3`**(*info*, *dataGroup*, *channelGroup*, *channelNumber*, *recordIDnumber*)

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 dictionnary

`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_stype)

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

changeChannelName(*channelGroup*)
In case of duplicate channel names within several channel groups for unsorted data, rename channel name

Parameters **channelGroup** (*int*) – channelGroup bumber

mdfreader.channel.**arrayformat4**(*signalDataType*, *numberOfBytes*)
function returning numpy style string from channel data type and number of bits

Parameters

- **signalDataType** (*int*) – channel data type according to specification
- **numberOfBytes** (*int*) – number of bytes taken by channel data in a record

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

Return type str

class `mdfreader.channel.`**`channel4`**

Bases: `object`

`CABlock`(*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

`CANOpenOffset`()

CANopen channel bytes offset

Returns

Return type integer, channel bytes offset

`CFormat`(*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

`CNBlock`(*info*)

channel block

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

Returns

Return type CNBlock class from mdfinfo4 module

`Format`(*info*)

channel data C format

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

Returns

Return type string data C format

`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

`bitCount`(*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

`bitOffset`(*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

`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

`byteOffset`

`calc_byteOffset`(*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

`changeChannelName`(*channelGroup*)
In case of duplicate channel names within several channel groups for unsorted data, rename channel name

Parameters **`channelGroup`** (*`int`*) – channelGroup bumber

`channelGroup`

`channelNumber`

`channelSyncType`(*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*

`channelType`(*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*

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

dataFormat(*info*)
channel numpy.core.records data format

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

Returns

Return type string data format

dataGroup

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

isCABlock(*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`

`nativedataFormat`(*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

`posBitBeg`(*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

`posBitEnd`(*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

`posByteBeg`(*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

`posByteEnd`(*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

`recAttributeName`(*info*)
clean up channel name from unauthorised characters

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

Returns

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

recordIDsize(*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*, *dataGroup*, *channelGroup*, *channelNumber*)
channel initialisation

Parameters

- **info** (*mdfinfo4.info4 class*) –
- **dataGroup** (*int*) – data group number in mdfinfo4.info4 class
- **channelGroup** (*int*) – channel group number in mdfinfo4.info4 class
- **channelNumber** (*int*) – channel number in mdfinfo4.info4 class

setCANOpen(*info*, *dataGroup*, *channelGroup*, *channelNumber*, *name*)
CANOpen channel intialisation

Parameters

- **info** (*mdfinfo4.info4 class*) –
- **dataGroup** (*int*) – data group number in mdfinfo4.info4 class
- **channelGroup** (*int*) – channel group number in mdfinfo4.info4 class
- **channelNumber** (*int*) – channel number in mdfinfo4.info4 class
- **name** (*str*) – name of channel. Should be in ('ms', 'day', 'days', 'hour', 'month', 'minute', 'year')

setInvalidBytes(*info*, *dataGroup*, *channelGroup*, *channelNumber*)
invalid_bytes channel initialisation

Parameters

- **info** (*mdfinfo4.info4 class*) –
- **dataGroup** (*int*) – data group number in mdfinfo4.info4 class
- **channelGroup** (*int*) – channel group number in mdfinfo4.info4 class
- **channelNumber** (*int*) – channel number in mdfinfo4.info4 class

signalDataType(*info*, *byte_aligned=True*)
extract signal data type from info4 class

Parameters

- **`info`** (*`mdfinfo4.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.`**`datatypeformat4`**(*signalDataType*, *numberOfBytes*)
function returning C format string from channel data type and number of bits

Parameters

- **`signalDataType`** (*`int`*) – channel data type according to specification
- **`numberOfBytes`** (*`int`*) – number of bytes taken by channel data in a record

Returns **dataType** – C format used by fread to read channel raw data

Return type str

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

mdfreader.channel, 43
mdfreader.mdf, 3
mdfreader.mdf3reader, 17
mdfreader.mdf4reader, 27
mdfreader.mdfinfo3, 25
mdfreader.mdfinfo4, 37
mdfreader.mdfreader, 9

INDEX

Symbols

__init__() (mdfreader.channel.Channel3 method), 44
__str__() (mdfreader.channel.Channel3 method), 44
_convertAllChannel3() (mdfreader.mdf3reader.mdf3 method), 20
_convertAllChannel4() (mdfreader.mdf4reader.mdf4 method), 30
_convertChannel3() (mdfreader.mdf3reader.mdf3 method), 20
_convertChannel4() (mdfreader.mdf4reader.mdf4 method), 30
_getChannelData3() (mdfreader.mdf3reader.mdf3 method), 20
_getChannelData4() (mdfreader.mdf4reader.mdf4 method), 30

A

add_channel() (mdfreader.mdf.mdf_skeleton method), 4
add_metadata() (mdfreader.mdf.mdf_skeleton method), 4
addChannel() (mdfreader.mdf3reader.record method), 21, 22
addChannel() (mdfreader.mdf4reader.record method), 32
addRecord() (mdfreader.mdf3reader.DATA method), 17, 18
addRecord() (mdfreader.mdf4reader.DATA method), 27
allPlot() (mdfreader.mdfreader.mdf method), 11
apply_invalid_bit() (mdfreader.mdf4reader.mdf4 method), 30
arrayformat4() (in module mdfreader.channel), 44
ATBlock (class in mdfreader.mdfinfo4), 37
attachment() (mdfreader.channel.channel4 method), 45

B

bit_masking_needed (mdfreader.channel.Channel3 attribute), 44
bit_masking_needed() (mdfreader.channel.channel4 method), 46
bitCount (mdfreader.channel.Channel3 attribute), 44
bitCount() (mdfreader.channel.channel4 method), 45
bitOffset (mdfreader.channel.Channel3 attribute), 44
bitOffset() (mdfreader.channel.channel4 method), 45
BlockLength (mdfreader.mdf3reader.DATA attribute), 17
byte_aligned (mdfreader.mdf3reader.record attribute), 21
byte_aligned (mdfreader.mdf4reader.record attribute), 32
byteOffset (mdfreader.channel.Channel3 attribute), 44
byteOffset (mdfreader.channel.channel4 attribute), 46

C

CABlock (class in mdfreader.mdfinfo4), 37
CABlock() (mdfreader.channel.channel4 method), 45
calc_byteOffset() (mdfreader.channel.channel4 method), 46
calc_bytes() (mdfreader.channel.channel4 method), 46
CANOpen (mdfreader.mdf4reader.record attribute), 31
CANOpenOffset() (mdfreader.channel.channel4 method), 45
CCBlock (class in mdfreader.mdfinfo4), 37
CFormat (mdfreader.channel.Channel3 attribute), 44
CFormat() (mdfreader.channel.channel4 method), 45
CGBlock (class in mdfreader.mdfinfo4), 37
CGrecordLength (mdfreader.mdf3reader.record attribute), 21
CGrecordLength (mdfreader.mdf4reader.record attribute), 31
changeChannelName() (mdfreader.channel.Channel3 method), 44
changeChannelName() (mdfreader.channel.channel4 method), 46
Channel3 (class in mdfreader.channel), 43
channel4 (class in mdfreader.channel), 44
channelGroup (mdfreader.channel.channel4 attribute), 46
channelGroup (mdfreader.mdf3reader.record attribute), 21
channelGroup (mdfreader.mdf4reader.record attribute), 32
channelNames (mdfreader.mdf3reader.record attribute), 21
channelNames (mdfreader.mdf4reader.record attribute), 32
channelNumber (mdfreader.channel.Channel3 attribute), 43
channelNumber (mdfreader.channel.channel4 attribute), 46

channelSyncType() (mdfreader.channel.channel4 method), 46
channelType (mdfreader.channel.Channel3 attribute), 44
channelType() (mdfreader.channel.channel4 method), 46
CHBlock (class in mdfreader.mdfinfo4), 38
cleanDGinfo() (mdfreader.mdfinfo3.info3 method), 25
cleanDGinfo() (mdfreader.mdfinfo4.info4 method), 40
CNBlock (class in mdfreader.mdfinfo4), 38
CNBlock() (mdfreader.channel.channel4 method), 45
CommentBlock (class in mdfreader.mdfinfo4), 38
compressed_data (class in mdfreader.mdf), 3
compression() (mdfreader.mdf.compressed_data method), 3
conversion (mdfreader.channel.Channel3 attribute), 43
conversion() (mdfreader.channel.channel4 method), 47
convert_tables (mdfreader.mdf.mdf_skeleton attribute), 4
convertAfterRead (mdfreader.mdf.mdf_skeleton attribute), 4
convertAfterRead (mdfreader.mdf3reader.mdf3 attribute), 19
convertAfterRead (mdfreader.mdf4reader.mdf4 attribute), 30
convertAllChannel() (mdfreader.mdfreader.mdf method), 10, 11
convertToPandas() (mdfreader.mdfreader.mdf method), 10, 11
copy() (mdfreader.mdf.mdf_skeleton method), 5
copy() (mdfreader.mdfreader.mdf method), 11
cut() (mdfreader.mdfreader.mdf method), 12

D

DATA (class in mdfreader.mdf3reader), 17
DATA (class in mdfreader.mdf4reader), 27
data (mdfreader.mdf.compressed_data attribute), 3
data() (mdfreader.channel.channel4 method), 47
DATABlock() (in module mdfreader.mdf4reader), 28
dataFormat (mdfreader.channel.Channel3 attribute), 44
dataFormat() (mdfreader.channel.channel4 method), 47
dataGroup (mdfreader.channel.channel4 attribute), 47
dataGroup (mdfreader.mdf3reader.record attribute), 21
dataGroup (mdfreader.mdf4reader.record attribute), 32
dataRecordName (mdfreader.mdf3reader.record attribute), 21
dataRecordName (mdfreader.mdf4reader.record attribute), 32
datatypeformat4() (in module mdfreader.channel), 50
decompress_datablock() (mdfreader.mdfinfo4.DZBlock static method), 39
decompression() (mdfreader.mdf.compressed_data method), 3
desc (mdfreader.channel.Channel3 attribute), 43
desc() (mdfreader.channel.channel4 method), 47
DGBlock (class in mdfreader.mdfinfo4), 38
DLBlock (class in mdfreader.mdfinfo4), 38
DTBlock (class in mdfreader.mdfinfo4), 38
dtype (mdfreader.mdf.compressed_data attribute), 3
DZBlock (class in mdfreader.mdfinfo4), 39

E

equalizeStringLength() (in module mdfreader.mdf4reader), 29
EVBlock (class in mdfreader.mdfinfo4), 39
expConv() (in module mdfreader.mdf3reader), 18
exportToCSV() (mdfreader.mdfreader.mdf method), 10, 12
exportToExcel() (mdfreader.mdfreader.mdf method), 10, 12
exportToHDF5() (mdfreader.mdfreader.mdf method), 10, 12
exportToMatlab() (mdfreader.mdfreader.mdf method), 10, 13
exportToNetCDF() (mdfreader.mdfreader.mdf method), 10, 13
exportToXlsx() (mdfreader.mdfreader.mdf method), 10, 13

F

FHBlock (class in mdfreader.mdfinfo4), 39
fid (mdfreader.mdf.mdf_skeleton attribute), 5
fid (mdfreader.mdf3reader.DATA attribute), 17
fid (mdfreader.mdf4reader.DATA attribute), 27
fid (mdfreader.mdfinfo3.info3 attribute), 25
fid (mdfreader.mdfinfo4.info4 attribute), 40
fid (mdfreader.mdfreader.mdfinfo attribute), 16
file_metadata (mdfreader.mdf.mdf_skeleton attribute), 5
file_metadata (mdfreader.mdf3reader.mdf3 attribute), 19
file_metadata (mdfreader.mdf4reader.mdf4 attribute), 30
file_metadata (mdfreader.mdfreader.mdf attribute), 10
fileName (mdfreader.mdf.mdf_skeleton attribute), 5
fileName (mdfreader.mdf3reader.mdf3 attribute), 19
fileName (mdfreader.mdf4reader.mdf4 attribute), 29
fileName (mdfreader.mdfinfo3.info3 attribute), 25
fileName (mdfreader.mdfinfo4.info4 attribute), 40
fileName (mdfreader.mdfreader.mdf attribute), 10
fileName (mdfreader.mdfreader.mdfinfo attribute), 16
filterChannelNames (mdfreader.mdf.mdf_skeleton attribute), 5
filterChannelNames (mdfreader.mdf3reader.mdf3 attribute), 19
filterChannelNames (mdfreader.mdf4reader.mdf4 attribute), 30
filterChannelNames (mdfreader.mdfinfo3.info3 attribute), 25
filterChannelNames (mdfreader.mdfreader.mdfinfo attribute), 16
Flags (mdfreader.mdf4reader.record attribute), 32
Format() (mdfreader.channel.channel4 method), 45
formulaConv() (in module mdfreader.mdf3reader), 18

formulaConv() (in module mdfreader.mdf4reader), 29

G

generate_chunks() (mdfreader.mdf4reader.record method), 32
getChannel() (mdfreader.mdf.mdf_skeleton method), 5
getChannelConversion() (mdfreader.mdf.mdf_skeleton method), 5
getChannelData() (mdfreader.mdfreader.mdf method), 10, 13
getChannelDesc() (mdfreader.mdf.mdf_skeleton method), 5
getChannelMaster() (mdfreader.mdf.mdf_skeleton method), 5
getChannelMasterType() (mdfreader.mdf.mdf_skeleton method), 5
getChannelName4() (mdfreader.mdf4reader.mdf4 method), 30
getChannelUnit() (mdfreader.mdf.mdf_skeleton method), 5
getChannelUnit() (mdfreader.mdfreader.mdf method), 10
getInvalidBit() (mdfreader.mdf.mdf_skeleton method), 5
getInvalidChannel() (mdfreader.mdf.mdf_skeleton method), 5

H

has_invalid_bit() (mdfreader.channel.channel4 method), 47
HDBlock (class in mdfreader.mdfinfo4), 39
hiddenBytes (mdfreader.mdf3reader.record attribute), 21
hiddenBytes (mdfreader.mdf4reader.record attribute), 32
HLBlock (class in mdfreader.mdfinfo4), 39

I

IDBlock (class in mdfreader.mdfinfo4), 40
info (mdfreader.mdf.mdf_skeleton attribute), 6
info3 (class in mdfreader.mdfinfo3), 25
info4 (class in mdfreader.mdfinfo4), 40
initialise_recarray() (mdfreader.mdf4reader.record method), 32
invalid_bit() (mdfreader.channel.channel4 method), 47
invalid_channel (mdfreader.mdf4reader.record attribute), 32
isCABlock() (mdfreader.channel.channel4 method), 47
isnumeric() (mdfreader.channel.channel4 method), 47

K

keepChannels() (mdfreader.mdfreader.mdf method), 10, 14

L

linearConv() (in module mdfreader.mdf3reader), 19
linearConv() (in module mdfreader.mdf4reader), 29
listChannels() (mdfreader.mdfreader.mdfinfo method), 16
listChannels3() (mdfreader.mdfinfo3.info3 method), 25
listChannels4() (mdfreader.mdfinfo4.info4 method), 40
little_endian() (mdfreader.channel.channel4 method), 48
load() (mdfreader.mdf4reader.DATA method), 27
load() (mdfreader.mdfinfo4.CABlock method), 37
load() (mdfreader.mdfinfo4.CommentBlock method), 38
load() (mdfreader.mdfinfo4.DTBlock method), 39
load() (mdfreader.mdfinfo4.HLBlock method), 39
loadInfo() (mdfreader.mdf3reader.record method), 22
loadInfo() (mdfreader.mdf4reader.record method), 32
loadSorted() (mdfreader.mdf3reader.DATA method), 18
loadUnSorted() (mdfreader.mdf3reader.DATA method), 18
logConv() (in module mdfreader.mdf3reader), 19

M

master (mdfreader.mdf3reader.record attribute), 21
master (mdfreader.mdf4reader.record attribute), 32
masterChannelList (mdfreader.mdf.mdf_skeleton attribute), 6
masterChannelList (mdfreader.mdf3reader.mdf3 attribute), 19
masterChannelList (mdfreader.mdf4reader.mdf4 attribute), 29
masterChannelList (mdfreader.mdfreader.mdf attribute), 10
mdf (class in mdfreader.mdfreader), 9
mdf3 (class in mdfreader.mdf3reader), 19
mdf4 (class in mdfreader.mdf4reader), 29
mdf_skeleton (class in mdfreader.mdf), 3
mdfinfo (class in mdfreader.mdfreader), 15
mdfreader.channel (module), 43
mdfreader.mdf (module), 3
mdfreader.mdf3reader (module), 17
mdfreader.mdf4reader (module), 27
mdfreader.mdfinfo3 (module), 25
mdfreader.mdfinfo4 (module), 37
mdfreader.mdfreader (module), 9
mdfversion (mdfreader.mdfreader.mdfinfo attribute), 16
MDFVersionNumber (mdfreader.mdf.mdf_skeleton attribute), 4
MDFVersionNumber (mdfreader.mdf3reader.mdf3 attribute), 19
MDFVersionNumber (mdfreader.mdf4reader.mdf4 attribute), 29
MDFVersionNumber (mdfreader.mdfreader.mdf attribute), 10
mergeMdf() (mdfreader.mdfreader.mdf method), 14
MLSD (mdfreader.mdf4reader.record attribute), 32
multiProc (mdfreader.mdf.mdf_skeleton attribute), 6
multiProc (mdfreader.mdf3reader.mdf3 attribute), 19
multiProc (mdfreader.mdf4reader.mdf4 attribute), 30
multiProc (mdfreader.mdfreader.mdf attribute), 10

N

name (mdfreader.channel.Channel3 attribute), 43
name (mdfreader.channel.channel4 attribute), 48
nativedataFormat() (mdfreader.channel.channel4 method), 48
nBytes (mdfreader.channel.Channel3 attribute), 44
nBytes (mdfreader.channel.channel4 attribute), 48
numberOfRecords (mdfreader.mdf3reader.record attribute), 21
numberOfRecords (mdfreader.mdf4reader.record attribute), 32
numpy_format() (mdfreader.channel.channel4 method), 48
numpyDataRecordFormat (mdfreader.mdf3reader.record attribute), 21
numpyDataRecordFormat (mdfreader.mdf4reader.record attribute), 32

P

plot() (mdfreader.mdfreader.mdf method), 10, 14
pointerToData (mdfreader.mdf3reader.DATA attribute), 17
pointerTodata (mdfreader.mdf4reader.DATA attribute), 28
polyConv() (in module mdfreader.mdf3reader), 20
posBitBeg() (mdfreader.channel.channel4 method), 48
posBitEnd() (mdfreader.channel.channel4 method), 48
posByteBeg (mdfreader.channel.Channel3 attribute), 44
posByteBeg() (mdfreader.channel.channel4 method), 48
posByteEnd (mdfreader.channel.Channel3 attribute), 44
posByteEnd() (mdfreader.channel.channel4 method), 48
PythonVersion (in module mdfreader.channel), 43
PythonVersion (in module mdfreader.mdf3reader), 17
PythonVersion (in module mdfreader.mdf4reader), 27
PythonVersion (in module mdfreader.mdfinfo3), 25
PythonVersion (in module mdfreader.mdfinfo4), 37
PythonVersion (in module mdfreader.mdfreader), 9

R

rationalConv() (in module mdfreader.mdf3reader), 21
rationalConv() (in module mdfreader.mdf4reader), 31
read() (mdfreader.mdf3reader.DATA method), 17, 18
read() (mdfreader.mdf4reader.DATA method), 28
read() (mdfreader.mdfinfo4.CABlock method), 37
read() (mdfreader.mdfinfo4.CGBlock method), 38
read() (mdfreader.mdfinfo4.DGBlock method), 38
read() (mdfreader.mdfinfo4.DLBlock method), 38
read() (mdfreader.mdfinfo4.DZBlock method), 39
read() (mdfreader.mdfinfo4.FHBlock method), 39
read() (mdfreader.mdfinfo4.HDBlock method), 39
read() (mdfreader.mdfinfo4.HLBlock method), 39
read() (mdfreader.mdfinfo4.IDBlock method), 40
read() (mdfreader.mdfreader.mdf method), 14
read3() (mdfreader.mdf3reader.mdf3 method), 20
read4() (mdfreader.mdf4reader.mdf4 method), 30
read_all_channels_sorted_record() (mdfreader.mdf4reader.record method), 33
read_cc_block() (in module mdfreader.mdfinfo3), 26
read_ce_block() (in module mdfreader.mdfinfo3), 26
read_cg_block() (in module mdfreader.mdfinfo3), 26
read_channels_from_bytes() (mdfreader.mdf4reader.record method), 33
read_channels_from_bytes_fallback() (mdfreader.mdf4reader.record method), 34
read_cn_block() (in module mdfreader.mdfinfo3), 26
read_dg_block() (in module mdfreader.mdfinfo3), 26
read_hd_block() (in module mdfreader.mdfinfo3), 26
read_not_all_channels_sorted_record() (mdfreader.mdf4reader.record method), 34
read_sdblock() (in module mdfreader.mdf4reader), 31
read_tx_block() (in module mdfreader.mdfinfo3), 26
readATBlock() (mdfreader.mdfinfo4.info4 method), 40
readCC() (mdfreader.mdfinfo4.CCBlock method), 37
readCGBlock() (mdfreader.mdfinfo3.info3 method), 26
readCGBlock() (mdfreader.mdfinfo4.info4 method), 40
readCM() (mdfreader.mdfinfo4.CommentBlock method), 38
readCN() (mdfreader.mdfinfo4.CNBlock method), 38
readCNBlock() (mdfreader.mdfinfo4.info4 method), 41
readComposition() (mdfreader.mdfinfo4.info4 method), 41
readDGBlock() (mdfreader.mdfinfo4.info4 method), 41
readinfo() (mdfreader.mdfinfo4.info4 method), 42
readinfo() (mdfreader.mdfreader.mdfinfo method), 16
readinfo3() (mdfreader.mdfinfo3.info3 method), 26
readRecord() (mdfreader.mdf4reader.DATA method), 28
readRecordBits() (mdfreader.mdf3reader.record method), 22
readRecordBuf() (mdfreader.mdf3reader.record method), 22
readRecordBuf() (mdfreader.mdf4reader.record method), 32
readSI() (mdfreader.mdfinfo4.SIBlock method), 40
readSortedRecord() (mdfreader.mdf3reader.record method), 22
readSortedRecord() (mdfreader.mdf4reader.record method), 33
readSRBlock() (mdfreader.mdfinfo4.info4 method), 41
readUnsorted() (in module mdfreader.mdf4reader), 31
recAttributeName (mdfreader.channel.Channel3 attribute), 44
recAttributeName() (mdfreader.channel.channel4 method), 48
record (class in mdfreader.mdf3reader), 21
record (class in mdfreader.mdf4reader), 31
RecordFormat (mdfreader.channel.Channel3 attribute), 44
recordID (mdfreader.mdf3reader.record attribute), 21

recordID (mdfreader.mdf4reader.record attribute), 34
recordIDCFormat (mdfreader.mdf4reader.record attribute), 34
recordIDnumber (mdfreader.mdf3reader.record attribute), 21
recordIDsize (mdfreader.mdf4reader.record attribute), 34
recordIDsize() (mdfreader.channel.channel4 method), 49
recordLength (mdfreader.mdf3reader.record attribute), 21
recordLength (mdfreader.mdf4reader.record attribute), 34
recordToChannelMatching (mdfreader.mdf3reader.record attribute), 21
recordToChannelMatching (mdfreader.mdf4reader.record attribute), 34
remove_channel() (mdfreader.mdf.mdf_skeleton method), 6
remove_channel_conversion() (mdfreader.mdf.mdf_skeleton method), 6
rename_channel() (mdfreader.mdf.mdf_skeleton method), 6
resample() (mdfreader.mdfreader.mdf method), 10, 15

S

set() (mdfreader.channel.channel4 method), 49
setCANOpen() (mdfreader.channel.channel4 method), 49
setChannelAttachment() (mdfreader.mdf.mdf_skeleton method), 6
setChannelConversion() (mdfreader.mdf.mdf_skeleton method), 6
setChannelData() (mdfreader.mdf.mdf_skeleton method), 6
setChannelDesc() (mdfreader.mdf.mdf_skeleton method), 6
setChannelMaster() (mdfreader.mdf.mdf_skeleton method), 7
setChannelMasterType() (mdfreader.mdf.mdf_skeleton method), 7
setChannelUnit() (mdfreader.mdf.mdf_skeleton method), 7
setInvalidBit() (mdfreader.mdf.mdf_skeleton method), 7
setInvalidBytes() (mdfreader.channel.channel4 method), 49
setInvalidChannel() (mdfreader.mdf.mdf_skeleton method), 7
SIBlock (class in mdfreader.mdfinfo4), 40
signalDataType (mdfreader.channel.Channel3 attribute), 43
signalDataType() (mdfreader.channel.channel4 method), 49
SRBlock (class in mdfreader.mdfinfo4), 40

T

tabConv() (in module mdfreader.mdf3reader), 23
tabInterpConv() (in module mdfreader.mdf3reader), 23
textRangeTableConv() (in module mdfreader.mdf3reader), 23
textTableConv() (in module mdfreader.mdf3reader), 23
textToTextConv() (in module mdfreader.mdf4reader), 34
textToValueConv() (in module mdfreader.mdf4reader), 34
type (mdfreader.channel.channel4 attribute), 50
type (mdfreader.mdf4reader.DATA attribute), 28

U

unique_id() (mdfreader.mdfinfo4.info4 method), 42
unit (mdfreader.channel.Channel3 attribute), 43
unit() (mdfreader.channel.channel4 method), 50

V

valueRangeToTextConv() (in module mdfreader.mdf4reader), 35
valueRangeToValueTableConv() (in module mdfreader.mdf4reader), 35
valueToTextConv() (in module mdfreader.mdf4reader), 35
valueToValueTableWInterpConv() (in module mdfreader.mdf4reader), 35
valueToValueTableWOInterpConv() (in module mdfreader.mdf4reader), 36
VLSD (mdfreader.mdf4reader.record attribute), 32
VLSD_CG (mdfreader.mdf4reader.record attribute), 32
VLSD_CG_Flag (mdfreader.channel.channel4 attribute), 45

W

write() (mdfreader.mdfinfo4.CABlock method), 37
write() (mdfreader.mdfinfo4.CGBlock method), 38
write() (mdfreader.mdfinfo4.CNBlock method), 38
write() (mdfreader.mdfinfo4.CommentBlock method), 38
write() (mdfreader.mdfinfo4.DGBlock method), 38
write() (mdfreader.mdfinfo4.DLBlock method), 38
write() (mdfreader.mdfinfo4.DTBlock method), 39
write() (mdfreader.mdfinfo4.DZBlock method), 39
write() (mdfreader.mdfinfo4.FHBlock method), 39
write() (mdfreader.mdfinfo4.HDBlock method), 39
write() (mdfreader.mdfinfo4.HLBlock method), 40
write() (mdfreader.mdfinfo4.IDBlock method), 40
write() (mdfreader.mdfreader.mdf method), 10, 15
write3() (mdfreader.mdf3reader.mdf3 method), 20
write4() (mdfreader.mdf4reader.mdf4 method), 31

Z

zipfile (mdfreader.mdf.mdf_skeleton attribute), 7
zipfile (mdfreader.mdfinfo4.info4 attribute), 42
zipfile (mdfreader.mdfreader.mdfinfo attribute), 16