SpiNNStorageHandlers Documentation

Jan 29, 2020

Contents

These pages document the python code for the SpiNNStorageHandlers module which is part of the SpiNNaker Project.

This code depends on SpiNNUtils. (Combined_documentation).

Contents:

CHAPTER 1

spinn_storage_handlers package

1.1 Subpackages

1.1.1 spinn_storage_handlers.abstract_classes package

1.1.1.1 Submodules

1.1.1.2 spinn_storage_handlers.abstract_classes.abstract_buffered_data_storage module

class spinn_storage_handlers.abstract_classes.abstract_buffered_data_storage.**AbstractBuffe**

Bases: object

An object that can store and read back buffered data

close()

Closes the data storage

Return type None

Raises *spinn_storage_handlers.exceptions.DataReadException* – If the data storage cannot be closed

eof()

Check if the read pointer is a the end of the data storage

Returns True if the read pointer is at the end of the data storage, False otherwise

Return type bool

read(*data_size*)

Read data from the data storage from the position indicated by the read pointer index

Parameters **data_size** (*int*) – number of bytes to be read

Returns a bytearray containing the data read

Return type bytearray

read_all()
Read all the data contained in the data storage starting from position 0 to the end

Returns a bytearray containing the data read

Return type bytearray

readinto(*data*)
Read some bytes of data from the underlying storage into a pre-defined array. Will block until some bytes are available, but may not fill the array completely.

Parameters **data** (*bytearray*) – The place where the data is to be stored

Returns The number of bytes stored in data

Return type int

Raises **IOError** – If an error occurs reading from the underlying storage

seek_read(*offset*, *whence=0*)
Set the data storage's current read position to the offset

Parameters

- **offset** (*int*) – Position of the read pointer within the buffer
- **whence** – One of: * os.SEEK_SET which means absolute buffer positioning (default) * os.SEEK_CUR which means seek relative to the current read position * os.SEEK_END which means seek relative to the buffer's end

Return type None

seek_write(*offset*, *whence=0*)
Set the data storage's current write position to the offset

Parameters

- **offset** (*int*) – Position of the write pointer within the buffer
- **whence** – One of: * os.SEEK_SET which means absolute buffer positioning (default) * os.SEEK_CUR which means seek relative to the current write position * os.SEEK_END which means seek relative to the buffer's end

Return type None

tell_read()
The current position of the read pointer

Returns The current position of the read pointer

Return type int

tell_write()
The current position of the write pointer

Returns The current position of the write pointer

Return type int

write(*data*)
Store data in the data storage in the position indicated by the write pointer index

Parameters **data** (*bytearray*) – the data to be stored

Return type None

1.1.1.3 spinn_storage_handlers.abstract_classes.abstract_byte_reader module

class spinn_storage_handlers.abstract_classes.abstract_byte_reader.**AbstractByteReader**
Bases: object

An abstract reader of bytes. Note that due to endianness concerns, the methods of this reader should be used directly for the appropriate data type being read; e.g. an int should be written using read_int rather than calling read_byte 4 times unless a specific endianness is being achieved.

is_at_end()
returns true if the reader is currently at the end of the byte reader

Returns returns true if the reader is currently at the end of the byte reader false otherwise

Return type bool

read_byte()
Reads the next byte

Returns A byte

Return type int

Raises

- **IOError** – If there is an error reading from the stream
- **EOFError** – If there are no more bytes to read

read_bytes(*size=None*)
Reads an array of bytes

Parameters **size** (*int*) – The number of bytes to read, or None to read all of the remaining bytes

Returns An array of bytes

Return type bytearray

Raises

- **IOError** – If there is an error reading from the stream
- **EOFError** – If there are too few bytes to read the requested size. Note that if there are no more bytes and size is None, an empty array will be returned

read_int()
Read the next four bytes as in int value

Returns An int

Return type int

Raises

- **IOError** – If there is an error reading from the stream
- **EOFError** – If there are too few bytes to read an int

read_long()
Reads the next eight bytes as a long value

Returns A long

Return type long

Raises

- **IOError** – If there is an error reading from the stream
- **EOFError** – If there are too few bytes to read a long

read_short()
Reads the next two bytes as a short value

Returns A short

Return type int

Raises

- **IOError** – If there is an error reading from the stream
- **EOFError** – If there are too few bytes to read a short

1.1.1.4 spinn_storage_handlers.abstract_classes.abstract_byte_writer module

class spinn_storage_handlers.abstract_classes.abstract_byte_writer.**AbstractByteWriter**
Bases: object

An abstract writer of bytes. Note that due to endianness concerns, the methods of this writer should be used directly for the appropriate data type being written; e.g. an int should be written using write_int rather than calling write_byte 4 times with the parts of the int unless a specific endianness is being achieved.

get_n_bytes_written()
Determines how many bytes have been written in total

Returns The number of bytes written

Return type int

Raises **None** – No known exception is raised

write_byte(*byte_value*)
Writes the lowest order byte of the given value

Parameters **byte_value** (*int*) – The byte to write

Returns Nothing is returned

Return type None

Raises **IOError** – If there is an error writing to the stream

write_bytes(*byte_iterable*)
Writes a set of bytes

Parameters **byte_iterable** (*iterable of bytes*) – The bytes to write

Returns Nothing is returned

Return type None

Raises **IOError** – If there is an error writing to the stream

write_int(*int_value*)
Writes a four byte value

Parameters **int_value** (*int*) – The integer to write

Returns Nothing is returned

Return type None

Raises **IOError** – If there is an error writing to the stream

write_long(*long_value*)
Writes an eight byte value

Parameters **long_value** (*long*) – The long to write

Returns Nothing is returned

Return type None

Raises **IOError** – If there is an error writing to the stream

write_short(*short_value*)
Writes the two lowest order bytes of the given value

Parameters **short_value** (*int*) – The short to write

Returns Nothing is returned

Return type None

Raises **IOError** – If there is an error writing to the stream

1.1.1.5 spinn_storage_handlers.abstract_classes.abstract_context_manager module

class spinn_storage_handlers.abstract_classes.abstract_context_manager.**AbstractContextManag**
Bases: object

Closeable class that supports being used as a simple context manager.

close()

1.1.1.6 spinn_storage_handlers.abstract_classes.abstract_data_reader module

class spinn_storage_handlers.abstract_classes.abstract_data_reader.**AbstractDataReader**
Bases: object

Abstract reader used to read data from somewhere

read(*n_bytes*)
Read some bytes of data from the underlying storage. Will block until some bytes are available, but might not return the full n_bytes. The size of the returned array indicates how many bytes were read.

Parameters **n_bytes** (*int*) – The number of bytes to read

Returns An array of bytes

Return type bytearray

Raises **IOError** – If an error occurs reading from the underlying storage

readall()
Read the rest of the bytes from the underlying stream

Returns The bytes read

Return type bytearray

Raises **IOError** – If there is an error obtaining the bytes

readinto(*data*)
Read some bytes of data from the underlying storage into a pre-defined array. Will block until some bytes are available, but may not fill the array completely.

Parameters **data** (*bytearray*) – The place where the data is to be stored

Returns The number of bytes stored in data

Return type int

Raises **IOError** – If an error occurs reading from the underlying storage

tell()
Returns the position of the file cursor

Returns Position of the file cursor

Return type int

1.1.1.7 spinn_storage_handlers.abstract_classes.abstract_data_writer module

class spinn_storage_handlers.abstract_classes.abstract_data_writer.**AbstractDataWriter**
Bases: object

Abstract writer used to write data somewhere

tell()
Returns the position of the file cursor

Returns Position of the file cursor

Return type int

write(*data*)
Write some bytes of data to the underlying storage. Does not return until all the bytes have been written.

Parameters **data** (*bytearray*) – The data to write

Returns Nothing is returned

Return type None

Raises **IOError** – If an error occurs writing to the underlying storage

1.1.1.8 Module contents

class spinn_storage_handlers.abstract_classes.**AbstractBufferedDataStorage**
Bases: object

An object that can store and read back buffered data

close()
Closes the data storage

Return type None

Raises ***spinn_storage_handlers.exceptions.DataReadException*** – If the data storage cannot be closed

eof()
Check if the read pointer is a the end of the data storage

Returns True if the read pointer is at the end of the data storage, False otherwise

Return type bool

read(*data_size*)
Read data from the data storage from the position indicated by the read pointer index

Parameters **data_size** (*int*) – number of bytes to be read

Returns a bytearray containing the data read

Return type bytearray

read_all()
Read all the data contained in the data storage starting from position 0 to the end

Returns a bytearray containing the data read

Return type bytearray

readinto(*data*)
Read some bytes of data from the underlying storage into a pre-defined array. Will block until some bytes are available, but may not fill the array completely.

Parameters **data** (*bytearray*) – The place where the data is to be stored

Returns The number of bytes stored in data

Return type int

Raises **IOError** – If an error occurs reading from the underlying storage

seek_read(*offset*, *whence=0*)
Set the data storage's current read position to the offset

Parameters

- **offset** (*int*) – Position of the read pointer within the buffer
- **whence** – One of: * os.SEEK_SET which means absolute buffer positioning (default) * os.SEEK_CUR which means seek relative to the current read position * os.SEEK_END which means seek relative to the buffer's end

Return type None

seek_write(*offset*, *whence=0*)
Set the data storage's current write position to the offset

Parameters

- **offset** (*int*) – Position of the write pointer within the buffer
- **whence** – One of: * os.SEEK_SET which means absolute buffer positioning (default) * os.SEEK_CUR which means seek relative to the current write position * os.SEEK_END which means seek relative to the buffer's end

Return type None

tell_read()
The current position of the read pointer

Returns The current position of the read pointer

Return type int

tell_write()
The current position of the write pointer

Returns The current position of the write pointer

Return type int

write(*data*)
Store data in the data storage in the position indicated by the write pointer index

Parameters **data** (*bytearray*) – the data to be stored

Return type None

class spinn_storage_handlers.abstract_classes.**AbstractByteReader**
Bases: object

An abstract reader of bytes. Note that due to endianness concerns, the methods of this reader should be used directly for the appropriate data type being read; e.g. an int should be written using read_int rather than calling read_byte 4 times unless a specific endianness is being achieved.

is_at_end()
returns true if the reader is currently at the end of the byte reader

Returns returns true if the reader is currently at the end of the byte reader false otherwise

Return type bool

read_byte()
Reads the next byte

Returns A byte

Return type int

Raises

- **IOError** – If there is an error reading from the stream
- **EOFError** – If there are no more bytes to read

read_bytes(*size=None*)
Reads an array of bytes

Parameters **size** (*int*) – The number of bytes to read, or None to read all of the remaining bytes

Returns An array of bytes

Return type bytearray

Raises

- **IOError** – If there is an error reading from the stream
- **EOFError** – If there are too few bytes to read the requested size. Note that if there are no more bytes and size is None, an empty array will be returned

read_int()
Read the next four bytes as in int value

Returns An int

Return type int

Raises

- **IOError** – If there is an error reading from the stream
- **EOFError** – If there are too few bytes to read an int

read_long()
Reads the next eight bytes as a long value

Returns A long

Return type long

Raises

- **IOError** – If there is an error reading from the stream
- **EOFError** – If there are too few bytes to read a long

read_short()
Reads the next two bytes as a short value

Returns A short

Return type int

Raises

- **IOError** – If there is an error reading from the stream
- **EOFError** – If there are too few bytes to read a short

class spinn_storage_handlers.abstract_classes.**AbstractByteWriter**
Bases: object

An abstract writer of bytes. Note that due to endianness concerns, the methods of this writer should be used directly for the appropriate data type being written; e.g. an int should be written using write_int rather than calling write_byte 4 times with the parts of the int unless a specific endianness is being achieved.

get_n_bytes_written()
Determines how many bytes have been written in total

Returns The number of bytes written

Return type int

Raises **None** – No known exception is raised

write_byte(*byte_value*)
Writes the lowest order byte of the given value

Parameters **byte_value** (*int*) – The byte to write

Returns Nothing is returned

Return type None

Raises **IOError** – If there is an error writing to the stream

write_bytes(*byte_iterable*)
Writes a set of bytes

Parameters **byte_iterable** (*iterable of bytes*) – The bytes to write

Returns Nothing is returned

Return type None

Raises **IOError** – If there is an error writing to the stream

write_int(*int_value*)
Writes a four byte value

Parameters **int_value** (*int*) – The integer to write

Returns Nothing is returned

Return type None

Raises **IOError** – If there is an error writing to the stream

write_long(*long_value*)
Writes an eight byte value

Parameters **long_value** (*long*) – The long to write

Returns Nothing is returned

Return type None

Raises **IOError** – If there is an error writing to the stream

write_short(*short_value*)
Writes the two lowest order bytes of the given value

Parameters **short_value** (*int*) – The short to write

Returns Nothing is returned

Return type None

Raises **IOError** – If there is an error writing to the stream

class spinn_storage_handlers.abstract_classes.**AbstractContextManager**
Bases: object

Closeable class that supports being used as a simple context manager.

close()

class spinn_storage_handlers.abstract_classes.**AbstractDataReader**
Bases: object

Abstract reader used to read data from somewhere

read(*n_bytes*)
Read some bytes of data from the underlying storage. Will block until some bytes are available, but might not return the full n_bytes. The size of the returned array indicates how many bytes were read.

Parameters **n_bytes** (*int*) – The number of bytes to read

Returns An array of bytes

Return type bytearray

Raises **IOError** – If an error occurs reading from the underlying storage

readall()
Read the rest of the bytes from the underlying stream

Returns The bytes read

Return type bytearray

Raises **IOError** – If there is an error obtaining the bytes

readinto(*data*)
Read some bytes of data from the underlying storage into a pre-defined array. Will block until some bytes are available, but may not fill the array completely.

Parameters **data** (*bytearray*) – The place where the data is to be stored

Returns The number of bytes stored in data

Return type int

Raises **IOError** – If an error occurs reading from the underlying storage

tell()
Returns the position of the file cursor

Returns Position of the file cursor

Return type int

class spinn_storage_handlers.abstract_classes.**AbstractDataWriter**
Bases: object

Abstract writer used to write data somewhere

tell()
Returns the position of the file cursor

Returns Position of the file cursor

Return type int

write(*data*)
Write some bytes of data to the underlying storage. Does not return until all the bytes have been written.

Parameters **data** (*bytearray*) – The data to write

Returns Nothing is returned

Return type None

Raises **IOError** – If an error occurs writing to the underlying storage

1.2 Submodules

1.3 spinn_storage_handlers.buffered_bytearray_data_storage module

class spinn_storage_handlers.buffered_bytearray_data_storage.**BufferedBytearrayDataStorage**
Bases: *spinn_storage_handlers.abstract_classes.abstract_buffered_data_storage.AbstractBufferedDataStorage*, *spinn_storage_handlers.abstract_classes.abstract_context_manager.AbstractContextManager*

Data storage based on a bytearray buffer with two pointers, one for reading and one for writing.

close()
Closes the data storage

Return type None

Raises ***spinn_storage_handlers.exceptions.DataReadException*** – If the data storage cannot be closed

eof()
Check if the read pointer is a the end of the data storage

Returns True if the read pointer is at the end of the data storage, False otherwise

Return type bool

read(*data_size*)
Read data from the data storage from the position indicated by the read pointer index

Parameters **data_size** (*int*) – number of bytes to be read

Returns a bytearray containing the data read

Return type bytearray

read_all()
Read all the data contained in the data storage starting from position 0 to the end

Returns a bytearray containing the data read

Return type bytearray

readinto(*data*)
Read some bytes of data from the underlying storage into a pre-defined array. Will block until some bytes are available, but may not fill the array completely.

Parameters **data** (*bytearray*) – The place where the data is to be stored

Returns The number of bytes stored in data

Return type int

Raises **IOError** – If an error occurs reading from the underlying storage

seek_read(*offset*, *whence=0*)
Set the data storage's current read position to the offset

Parameters

- **offset** (*int*) – Position of the read pointer within the buffer
- **whence** – One of: * os.SEEK_SET which means absolute buffer positioning (default) * os.SEEK_CUR which means seek relative to the current read position * os.SEEK_END which means seek relative to the buffer's end

Return type None

seek_write(*offset*, *whence=0*)
Set the data storage's current write position to the offset

Parameters

- **offset** (*int*) – Position of the write pointer within the buffer
- **whence** – One of: * os.SEEK_SET which means absolute buffer positioning (default) * os.SEEK_CUR which means seek relative to the current write position * os.SEEK_END which means seek relative to the buffer's end

Return type None

tell_read()
The current position of the read pointer

Returns The current position of the read pointer

Return type int

tell_write()
The current position of the write pointer

Returns The current position of the write pointer

Return type int

write(*data*)
Store data in the data storage in the position indicated by the write pointer index

Parameters **data** (*bytearray*) – the data to be stored

Return type None

1.4 spinn_storage_handlers.buffered_file_data_storage module

class spinn_storage_handlers.buffered_file_data_storage.**BufferedFileDataStorage**(*filename*, *mode*)

Bases: *spinn_storage_handlers.abstract_classes.abstract_buffered_data_storage.AbstractBufferedDataStorage*, *spinn_storage_handlers.abstract_classes.abstract_context_manager.AbstractContextManager*

Data storage based on a temporary file with two pointers, one for reading and one for writing.

close()
Closes the data storage

Return type None

Raises ***spinn_storage_handlers.exceptions.DataReadException*** – If the data storage cannot be closed

eof()
Check if the read pointer is a the end of the data storage

Returns True if the read pointer is at the end of the data storage, False otherwise

Return type bool

filename
property method

read(*data_size*)
Read data from the data storage from the position indicated by the read pointer index

Parameters **data_size** (*int*) – number of bytes to be read

Returns a bytearray containing the data read

Return type bytearray

read_all()
Read all the data contained in the data storage starting from position 0 to the end

Returns a bytearray containing the data read

Return type bytearray

readinto(*data*)
See *spinn_storage_handlers.abstract_classes.AbstractBufferedDataStorage.readinto()*

seek_read(*offset*, *whence=0*)
Set the data storage's current read position to the offset

Parameters

- **offset** (*int*) – Position of the read pointer within the buffer
- **whence** – One of: * os.SEEK_SET which means absolute buffer positioning (default) * os.SEEK_CUR which means seek relative to the current read position * os.SEEK_END which means seek relative to the buffer's end

Return type None

seek_write(*offset*, *whence=0*)
Set the data storage's current write position to the offset

Parameters

- **offset** (*int*) – Position of the write pointer within the buffer
- **whence** – One of: * os.SEEK_SET which means absolute buffer positioning (default) * os.SEEK_CUR which means seek relative to the current write position * os.SEEK_END which means seek relative to the buffer's end

Return type None

tell_read()
The current position of the read pointer

Returns The current position of the read pointer

Return type int

tell_write()
The current position of the write pointer

Returns The current position of the write pointer

Return type int

write(*data*)
Store data in the data storage in the position indicated by the write pointer index

Parameters **data** (*bytearray*) – the data to be stored

Return type None

1.5 spinn_storage_handlers.buffered_tempfile_data_storage module

class spinn_storage_handlers.buffered_tempfile_data_storage.**BufferedTempfileDataStorage**
Bases: *spinn_storage_handlers.abstract_classes.abstract_buffered_data_storage.AbstractBufferedDataStorage*, *spinn_storage_handlers.abstract_classes.abstract_context_manager.AbstractContextManager*

Data storage based on a temporary file with two pointers, one for reading and one for writing.

close()
Closes the data storage

Return type None

Raises ***spinn_storage_handlers.exceptions.DataReadException*** – If the data storage cannot be closed

eof()
Check if the read pointer is a the end of the data storage

Returns True if the read pointer is at the end of the data storage, False otherwise

Return type bool

read(*data_size*)
Read data from the data storage from the position indicated by the read pointer index

Parameters **data_size** (*int*) – number of bytes to be read

Returns a bytearray containing the data read

Return type bytearray

read_all()
Read all the data contained in the data storage starting from position 0 to the end

Returns a bytearray containing the data read

Return type bytearray

readinto(*data*)
Read some bytes of data from the underlying storage into a pre-defined array. Will block until some bytes are available, but may not fill the array completely.

Parameters **data** (*bytearray*) – The place where the data is to be stored

Returns The number of bytes stored in data

Return type int

Raises **IOError** – If an error occurs reading from the underlying storage

seek_read(*offset*, *whence=0*)
Set the data storage's current read position to the offset

Parameters

- **offset** (*int*) – Position of the read pointer within the buffer
- **whence** – One of: * os.SEEK_SET which means absolute buffer positioning (default) * os.SEEK_CUR which means seek relative to the current read position * os.SEEK_END which means seek relative to the buffer's end

Return type None

seek_write(*offset*, *whence=0*)
Set the data storage's current write position to the offset

Parameters

- **offset** (*int*) – Position of the write pointer within the buffer
- **whence** – One of: * os.SEEK_SET which means absolute buffer positioning (default) * os.SEEK_CUR which means seek relative to the current write position * os.SEEK_END which means seek relative to the buffer's end

Return type None

tell_read()
The current position of the read pointer

Returns The current position of the read pointer

Return type int

tell_write()
The current position of the write pointer

Returns The current position of the write pointer

Return type int

write(*data*)
Store data in the data storage in the position indicated by the write pointer index

Parameters **data** (*bytearray*) – the data to be stored

Return type None

1.6 spinn_storage_handlers.exceptions module

exception spinn_storage_handlers.exceptions.**BufferedBytearrayOperationNotImplemented**(*message*
Bases: exceptions.Exception

An exception that denotes that the operation required is unavailable for a byteArray buffer

Parameters **message** (*str*) – A message to indicate what when wrong

exception spinn_storage_handlers.exceptions.**DataReadException**(*message*)
Bases: exceptions.Exception

An exception that indicates that there was an error reading from the underlying medium

Parameters **message** (*str*) – A message to indicate what when wrong

exception spinn_storage_handlers.exceptions.**DataWriteException**(*message*)
Bases: exceptions.Exception

An exception that indicates that there was an error writing to the underlying medium

Parameters **message** (*str*) – A message to indicate what when wrong

1.7 spinn_storage_handlers.file_data_reader module

class spinn_storage_handlers.file_data_reader.**FileDataReader**(*filename*)
Bases: *spinn_storage_handlers.abstract_classes.abstract_data_reader.AbstractDataReader*, *spinn_storage_handlers.abstract_classes.abstract_context_manager.AbstractContextManager*

A reader that can read data from a file

Parameters **filename** (*str*) – The file to read

Raises ***spinn_storage_handlers.exceptions.DataReadException*** – If the file cannot found or opened for reading

close()
Closes the file

Return type None

Raises ***spinn_storage_handlers.exceptions.DataReadException*** – If the file cannot be closed

read(*n_bytes*)
See data_specification.abstract_data_reader.AbstractDataReader.read()

readall()
See data_specification.abstract_data_reader.AbstractDataReader.readall()

readinto(*data*)
See data_specification.abstract_data_reader.AbstractDataReader.readinto()

tell()
Returns the position of the file cursor

Returns Position of the file cursor

Return type int

1.8 spinn_storage_handlers.file_data_writer module

class spinn_storage_handlers.file_data_writer.**FileDataWriter**(*filename*)

Bases: *spinn_storage_handlers.abstract_classes.abstract_data_writer.AbstractDataWriter*, *spinn_storage_handlers.abstract_classes.abstract_context_manager.AbstractContextManager*

Parameters **filename** (*str*) – The file to write to

Raises ***spinn_storage_handlers.exceptions.DataWriteException*** – If the file cannot found or opened for writing

close()

Closes the file

Return type None

Raises ***spinn_storage_handlers.exceptions.DataWriteException*** – If the file cannot be closed

filename

property method

tell()

Returns the position of the file cursor

Returns Position of the file cursor

Return type int

write(*data*)

See data_specification.abstract_data_writer.AbstractDataWriter.write()

1.9 Module contents

class spinn_storage_handlers.**BufferedBytearrayDataStorage**

Bases: *spinn_storage_handlers.abstract_classes.abstract_buffered_data_storage.AbstractBufferedDataStorage*, *spinn_storage_handlers.abstract_classes.abstract_context_manager.AbstractContextManager*

Data storage based on a bytearray buffer with two pointers, one for reading and one for writing.

close()

Closes the data storage

Return type None

Raises ***spinn_storage_handlers.exceptions.DataReadException*** – If the data storage cannot be closed

eof()

Check if the read pointer is a the end of the data storage

Returns True if the read pointer is at the end of the data storage, False otherwise

Return type bool

read(*data_size*)

Read data from the data storage from the position indicated by the read pointer index

Parameters **data_size** (*int*) – number of bytes to be read

Returns a bytearray containing the data read

Return type bytearray

read_all()
Read all the data contained in the data storage starting from position 0 to the end

Returns a bytearray containing the data read

Return type bytearray

readinto(*data*)
Read some bytes of data from the underlying storage into a pre-defined array. Will block until some bytes are available, but may not fill the array completely.

Parameters **data** (*bytearray*) – The place where the data is to be stored

Returns The number of bytes stored in data

Return type int

Raises **IOError** – If an error occurs reading from the underlying storage

seek_read(*offset*, *whence=0*)
Set the data storage's current read position to the offset

Parameters

- **offset** (*int*) – Position of the read pointer within the buffer
- **whence** – One of: * os.SEEK_SET which means absolute buffer positioning (default) * os.SEEK_CUR which means seek relative to the current read position * os.SEEK_END which means seek relative to the buffer's end

Return type None

seek_write(*offset*, *whence=0*)
Set the data storage's current write position to the offset

Parameters

- **offset** (*int*) – Position of the write pointer within the buffer
- **whence** – One of: * os.SEEK_SET which means absolute buffer positioning (default) * os.SEEK_CUR which means seek relative to the current write position * os.SEEK_END which means seek relative to the buffer's end

Return type None

tell_read()
The current position of the read pointer

Returns The current position of the read pointer

Return type int

tell_write()
The current position of the write pointer

Returns The current position of the write pointer

Return type int

write(*data*)
Store data in the data storage in the position indicated by the write pointer index

Parameters **data** (*bytearray*) – the data to be stored

Return type None

class spinn_storage_handlers.**BufferedFileDataStorage**(*filename*, *mode*)

Bases: *spinn_storage_handlers.abstract_classes.abstract_buffered_data_storage.AbstractBufferedDataStorage*, *spinn_storage_handlers.abstract_classes.abstract_context_manager.AbstractContextManager*

Data storage based on a temporary file with two pointers, one for reading and one for writing.

close()

Closes the data storage

Return type None

Raises ***spinn_storage_handlers.exceptions.DataReadException*** – If the data storage cannot be closed

eof()

Check if the read pointer is a the end of the data storage

Returns True if the read pointer is at the end of the data storage, False otherwise

Return type bool

filename

property method

read(*data_size*)

Read data from the data storage from the position indicated by the read pointer index

Parameters **data_size** (*int*) – number of bytes to be read

Returns a bytearray containing the data read

Return type bytearray

read_all()

Read all the data contained in the data storage starting from position 0 to the end

Returns a bytearray containing the data read

Return type bytearray

readinto(*data*)

See *spinn_storage_handlers.abstract_classes.AbstractBufferedDataStorage.readinto()*

seek_read(*offset*, *whence=0*)

Set the data storage's current read position to the offset

Parameters

- **offset** (*int*) – Position of the read pointer within the buffer
- **whence** – One of: * os.SEEK_SET which means absolute buffer positioning (default) * os.SEEK_CUR which means seek relative to the current read position * os.SEEK_END which means seek relative to the buffer's end

Return type None

seek_write(*offset*, *whence=0*)

Set the data storage's current write position to the offset

Parameters

- **offset** (*int*) – Position of the write pointer within the buffer

- **whence** – One of: * os.SEEK_SET which means absolute buffer positioning (default) * os.SEEK_CUR which means seek relative to the current write position * os.SEEK_END which means seek relative to the buffer's end

Return type None

tell_read()
The current position of the read pointer

Returns The current position of the read pointer

Return type int

tell_write()
The current position of the write pointer

Returns The current position of the write pointer

Return type int

write(*data*)
Store data in the data storage in the position indicated by the write pointer index

Parameters **data** (*bytearray*) – the data to be stored

Return type None

class spinn_storage_handlers.**BufferedTempfileDataStorage**
Bases: *spinn_storage_handlers.abstract_classes.abstract_buffered_data_storage.AbstractBufferedDataStorage*, *spinn_storage_handlers.abstract_classes.abstract_context_manager.AbstractContextManager*

Data storage based on a temporary file with two pointers, one for reading and one for writing.

close()
Closes the data storage

Return type None

Raises ***spinn_storage_handlers.exceptions.DataReadException*** – If the data storage cannot be closed

eof()
Check if the read pointer is a the end of the data storage

Returns True if the read pointer is at the end of the data storage, False otherwise

Return type bool

read(*data_size*)
Read data from the data storage from the position indicated by the read pointer index

Parameters **data_size** (*int*) – number of bytes to be read

Returns a bytearray containing the data read

Return type bytearray

read_all()
Read all the data contained in the data storage starting from position 0 to the end

Returns a bytearray containing the data read

Return type bytearray

readinto(*data*)
Read some bytes of data from the underlying storage into a pre-defined array. Will block until some bytes are available, but may not fill the array completely.

Parameters **data** (*bytearray*) – The place where the data is to be stored

Returns The number of bytes stored in data

Return type int

Raises **IOError** – If an error occurs reading from the underlying storage

seek_read(*offset*, *whence=0*)
Set the data storage's current read position to the offset

Parameters

- **offset** (*int*) – Position of the read pointer within the buffer
- **whence** – One of: * os.SEEK_SET which means absolute buffer positioning (default) * os.SEEK_CUR which means seek relative to the current read position * os.SEEK_END which means seek relative to the buffer's end

Return type None

seek_write(*offset*, *whence=0*)
Set the data storage's current write position to the offset

Parameters

- **offset** (*int*) – Position of the write pointer within the buffer
- **whence** – One of: * os.SEEK_SET which means absolute buffer positioning (default) * os.SEEK_CUR which means seek relative to the current write position * os.SEEK_END which means seek relative to the buffer's end

Return type None

tell_read()
The current position of the read pointer

Returns The current position of the read pointer

Return type int

tell_write()
The current position of the write pointer

Returns The current position of the write pointer

Return type int

write(*data*)
Store data in the data storage in the position indicated by the write pointer index

Parameters **data** (*bytearray*) – the data to be stored

Return type None

class spinn_storage_handlers.**FileDataReader**(*filename*)
Bases: *spinn_storage_handlers.abstract_classes.abstract_data_reader.AbstractDataReader*, *spinn_storage_handlers.abstract_classes.abstract_context_manager.AbstractContextManager*

A reader that can read data from a file

Parameters **filename** (*str*) – The file to read

Raises *spinn_storage_handlers.exceptions.DataReadException* – If the file cannot found or opened for reading

close()
Closes the file

Return type None

Raises *spinn_storage_handlers.exceptions.DataReadException* – If the file cannot be closed

read(*n_bytes*)
See data_specification.abstract_data_reader.AbstractDataReader.read()

readall()
See data_specification.abstract_data_reader.AbstractDataReader.readall()

readinto(*data*)
See data_specification.abstract_data_reader.AbstractDataReader.readinto()

tell()
Returns the position of the file cursor

Returns Position of the file cursor

Return type int

class spinn_storage_handlers.**FileDataWriter**(*filename*)
Bases: *spinn_storage_handlers.abstract_classes.abstract_data_writer.AbstractDataWriter*, *spinn_storage_handlers.abstract_classes.abstract_context_manager.AbstractContextManager*

Parameters **filename** (*str*) – The file to write to

Raises *spinn_storage_handlers.exceptions.DataWriteException* – If the file cannot found or opened for writing

close()
Closes the file

Return type None

Raises *spinn_storage_handlers.exceptions.DataWriteException* – If the file cannot be closed

filename
property method

tell()
Returns the position of the file cursor

Returns Position of the file cursor

Return type int

write(*data*)
See data_specification.abstract_data_writer.AbstractDataWriter.write()

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index

D

E

F

G

I

R

S

T

W