Arduino Gyroscope Driver

Generated by Doxygen 1.8.9.1

Tue Aug 18 2015 22:52:34

Contents

1 Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

2 Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

3 File Index

3.1 File List

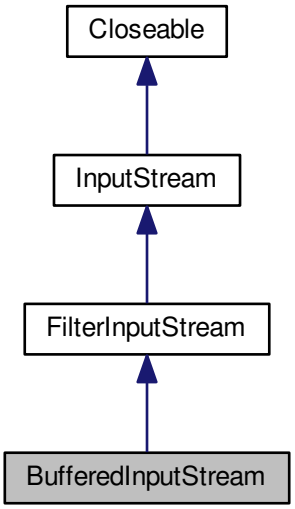
Here is a list of all files with brief descriptions:

4 Class Documentation

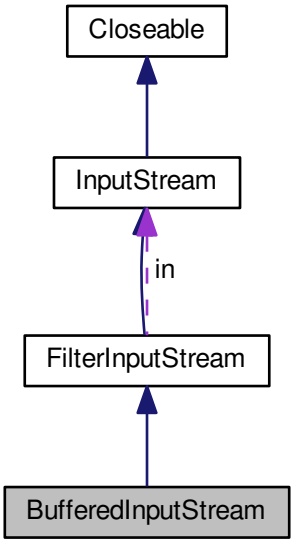
4.1 BufferedInputStream Class Reference

```
#include <BufferedInputStream.h>
```

Inheritance diagram for BufferedInputStream:



Collaboration diagram for BufferedInputStream:



Public Member Functions

- BufferedInputStream (InputStream *in, unsigned char *buf, int size)
- virtual int available ()

- virtual void close ()
- virtual void mark ()
- virtual bool markSupported ()
- virtual int read ()
- virtual int read (unsigned char *b, int len)
- virtual int read (unsigned char *b, int off, int len)
- virtual void reset ()
- virtual unsigned int skip (unsigned int n)

Protected Attributes

- unsigned char * buf
- int count
- int pos
- int markpos
- bool marked

Private Member Functions

- void realineBufferContent ()
- void fill (int startPos)

Private Attributes

- unsigned int size

Additional Inherited Members

4.1.1 Detailed Description

Raspberry IO.

BufferedInputStream

A `BufferedInputStream` adds functionality to another input stream-namely, the ability to buffer the input and to support the `mark` and `reset` methods. When the `BufferedInputStream` is created, an internal buffer array is passed. As bytes from the stream are read or skipped, the internal buffer is refilled as necessary from the contained input stream, many bytes at a time. The `mark` operation remembers a point in the input stream and the `reset` operation causes all the bytes read since the most recent `mark` operation to be reread before new bytes are taken from the contained input stream.

Definition at line 29 of file BufferedInputStream.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 BufferedInputStream::BufferedInputStream (InputStream * *in*, unsigned char * *buf*, int *size*)

Public constructor.

Parameters

in	

buf	
size	

Definition at line 29 of file BufferedInputStream.cpp.

4.1.3 Member Function Documentation

4.1.3.1 int BufferedInputStream::available ( ) [virtual]

Returns the number of bytes that can be read(or skipped over) from this input stream without blocking by the next caller of a method for this input stream.

Reimplemented from FilterInputStream.

Definition at line 39 of file BufferedInputStream.cpp.

4.1.3.2 void BufferedInputStream::close ( ) [virtual]

Closes this input stream and releases any system resources associated with the stream.

Reimplemented from FilterInputStream.

Definition at line 43 of file BufferedInputStream.cpp.

4.1.3.3 void BufferedInputStream::fill (int *startPos*) [private]

Fills the buffer.

Parameters

startPos	

Definition at line 128 of file BufferedInputStream.cpp.

4.1.3.4 void BufferedInputStream::mark ( ) [virtual]

Marks the current position in this input stream.

Reimplemented from FilterInputStream.

Definition at line 140 of file BufferedInputStream.cpp.

4.1.3.5 bool BufferedInputStream::markSupported ( ) [virtual]

Tests if this input stream supports the mark and reset methods.

Reimplemented from FilterInputStream.

Definition at line 147 of file BufferedInputStream.cpp.

4.1.3.6 int BufferedInputStream::read ( ) [virtual]

Reads the next unsigned char of data from the input stream.

Reimplemented from FilterInputStream.

Definition at line 100 of file BufferedInputStream.cpp.

4.1.3.7 int BufferedInputStream::read (unsigned char * *b*, int *len*) [virtual]

Reads some number of bytes from the input stream and stores them into the buffer array b.

Parameters

b	
len	

Returns

Reimplemented from FilterInputStream.

Definition at line 53 of file BufferedInputStream.cpp.

4.1.3.8 int BufferedInputStream::read (unsigned char * *b*, int *off*, int *len*) [virtual]

Reads some number of bytes from the input stream and stores them into the buffer array b.

Reimplemented from FilterInputStream.

Definition at line 57 of file BufferedInputStream.cpp.

4.1.3.9 void BufferedInputStream::realineBufferContent ( ) [private]

Moves the valid bytes on the buffer to the left side of the buffer.

Definition at line 116 of file BufferedInputStream.cpp.

4.1.3.10 void BufferedInputStream::reset ( ) [virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

Reimplemented from FilterInputStream.

Definition at line 47 of file BufferedInputStream.cpp.

4.1.3.11 unsigned int BufferedInputStream::skip (unsigned int *n*) [virtual]

Skips over and discards n bytes of data from this input stream.

Reimplemented from FilterInputStream.

Definition at line 151 of file BufferedInputStream.cpp.

4.1.4 Member Data Documentation

4.1.4.1 unsigned char* BufferedInputStream::buf [protected]

The internal buffer array where the data is stored.

Definition at line 41 of file BufferedInputStream.h.

4.1.4.2 int BufferedInputStream::count [protected]

The index one greater than the index of the last valid unsigned char in the buffer.

This value is always in the range `0` through `size`; elements `buf[0]` through `buf[count-1]` contain buffered input data obtained from the underlying input stream.

Definition at line 52 of file BufferedInputStream.h.

4.1.4.3 bool BufferedInputStream::marked [protected]

Flag to determine if there is a marker on this input stream.

Definition at line 98 of file BufferedInputStream.h.

4.1.4.4 int BufferedInputStream::markpos [protected]

The value of the `pos` field at the time the last `mark` method was called.

This value is always in the range `0` through `pos`. If there is no marked position in the input stream, this field is `-1`. If there is a marked position in the input stream, then `buf[markpos]` is the first unsigned char to be supplied as input after a `reset` operation. If `markpos` is not `-1`, then all bytes from positions `buf[markpos]` through `buf[pos-1]` must remain in the buffer array (though they may be moved to another place in the buffer array, with suitable adjustments to the values of `count`, `pos`, and `markpos`); they may not be discarded unless and until the difference between `pos` and `markpos` exceeds `marklimit`.

Definition at line 93 of file BufferedInputStream.h.

4.1.4.5 int BufferedInputStream::pos [protected]

The current position in the buffer.

This is the index of the next character to be read from the `buf` array.

This value is always in the range `0` through `count`. If it is less than `count`, then `buf[pos]` is the next unsigned char to be supplied as input; if it is equal to `count`, then the next `read` or `skip` operation will require more bytes to be read from the contained input stream.

Definition at line 67 of file BufferedInputStream.h.

4.1.4.6 unsigned int BufferedInputStream::size [private]

The size of the buffer.

Definition at line 34 of file BufferedInputStream.h.

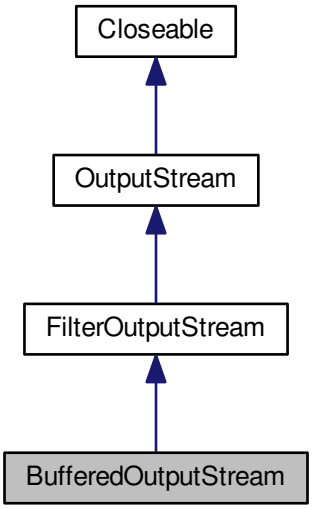
The documentation for this class was generated from the following files:

- BufferedInputStream.h
- BufferedInputStream.cpp

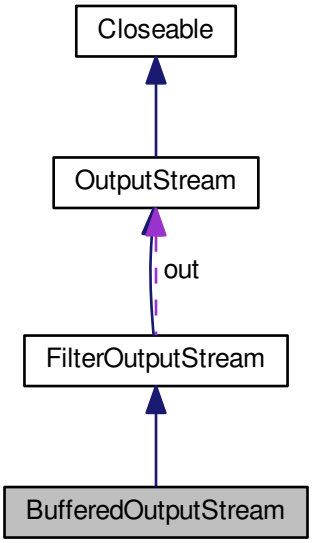
4.2 BufferedOutputStream Class Reference

```
#include <BufferedOutputStream.h>
```

Inheritance diagram for BufferedOutputStream:



Collaboration diagram for BufferedOutputStream:



Public Member Functions

- BufferedOutputStream (OutputStream ∗out, unsigned char ∗buf, int size)
- void write (unsigned char b)

- virtual void write (unsigned char *b, int len)
- virtual void write (unsigned char *b, int off, int len)
- virtual void flush ()
- virtual void close ()

Protected Attributes

- unsigned char * buf
- int size
- int count

Private Member Functions

- void flushBuffer ()

4.2.1 Detailed Description

Raspberry IO.

BufferedOutputStream

The class implements a buffered output stream. By setting up such an output stream, an application can write bytes to the underlying output stream without necessarily causing a call to the underlying system for each unsigned char written.

Definition at line 17 of file BufferedOutputStream.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 BufferedOutputStream::BufferedOutputStream (OutputStream * *out*, unsigned char * *buf*, int *size*)

Creates a new buffered output stream to write data to the specified underlying output stream with the specified buffer size.

Parameters

out	the underlying output stream.
size	the buffer size.

Definition at line 17 of file BufferedOutputStream.cpp.

4.2.3 Member Function Documentation

4.2.3.1 void BufferedOutputStream::close ( ) [virtual]

Closes this output stream and releases any system resources associated with the stream.

The `close` method of `FilterOutputStream` calls its `flush` method, and then calls the `close` method of its underlying output stream.

Reimplemented from FilterOutputStream.

Definition at line 60 of file BufferedOutputStream.cpp.

4.2.3.2 void BufferedOutputStream::flush ( ) [virtual]

Flushes this buffered output stream.

This forces any buffered output bytes to be written out to the underlying output stream.

Reimplemented from FilterOutputStream.

Definition at line 55 of file BufferedOutputStream.cpp.

4.2.3.3 void BufferedOutputStream::flushBuffer ( ) [private]

Flush the internal buffer.

Definition at line 65 of file BufferedOutputStream.cpp.

4.2.3.4 void BufferedOutputStream::write (unsigned char *b*) [virtual]

Writes the specified unsigned char to this buffered output stream.

Parameters

b	the unsigned char to be written.

Exceptions

IOException	if an I/O error occurs.

Reimplemented from FilterOutputStream.

Definition at line 24 of file BufferedOutputStream.cpp.

4.2.3.5 void BufferedOutputStream::write (unsigned char * *b*, int *len*) [virtual]

Writes len bytes from the specified unsigned char array to this output stream.

The general contract for write(b, len) is that it should have exactly the same effect as the call write(b, 0, len).

Parameters

b	
len	

Reimplemented from FilterOutputStream.

Definition at line 31 of file BufferedOutputStream.cpp.

4.2.3.6 void BufferedOutputStream::write (unsigned char * *b*, int *off*, int *len*) [virtual]

Writes `len` bytes from the specified unsigned char array starting at offset `off` to this buffered output stream.

Ordinarily this method stores bytes from the given array into this stream's buffer, flushing the buffer to the underlying output stream as needed. If the requested length is at least as large as this stream's buffer, however, then this method will flush the buffer and write the bytes directly to the underlying output stream. Thus redundant `BufferedOutputStream`s will not copy data unnecessarily.

Parameters

b	the data.
off	the start offset in the data.
len	the number of bytes to write.

Reimplemented from FilterOutputStream.

Definition at line 35 of file BufferedOutputStream.cpp.

4.2.4 Member Data Documentation

4.2.4.1 unsigned char* BufferedOutputStream::buf [protected]

The internal buffer where data is stored.

Definition at line 23 of file BufferedOutputStream.h.

4.2.4.2 int BufferedOutputStream::count [protected]

The number of valid bytes in the buffer.

This value is always in the range `0` through `len`; elements `buf[0]` through `buf[count-1]` contain valid unsigned char data.

Definition at line 36 of file BufferedOutputStream.h.

4.2.4.3 int BufferedOutputStream::size [protected]

The size of the buffer where data is stored.

Definition at line 28 of file BufferedOutputStream.h.

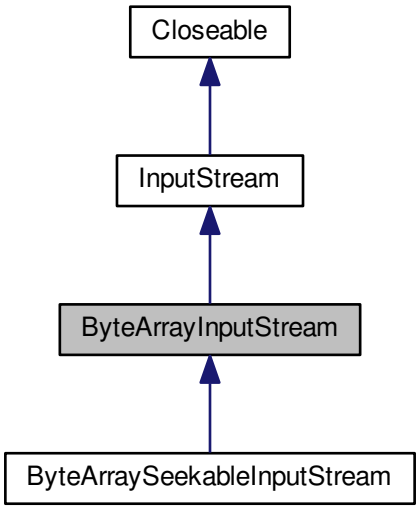
The documentation for this class was generated from the following files:

- BufferedOutputStream.h
- BufferedOutputStream.cpp

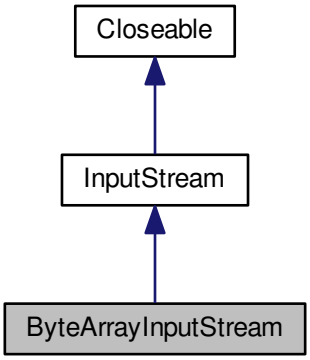
4.3 ByteArrayInputStream Class Reference

`#include <ByteArrayInputStream.h>`

Inheritance diagram for ByteArrayInputStream:



Collaboration diagram for ByteArrayInputStream:



Public Member Functions

- ByteArrayInputStream (unsigned char *buf, unsigned int count)
- virtual int available ()
- virtual void mark ()
- virtual bool markSupported ()
- virtual int read ()
- virtual void reset ()

Protected Attributes

- unsigned char * buf
- unsigned int count
- unsigned int pos
- unsigned int markpos

4.3.1 Detailed Description

Raspberry IO.

ByteArrayInputStream

A ByteArrayInputStream contains an internal buffer that contains bytes that may be read from the stream.

Definition at line 15 of file ByteArrayInputStream.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 ByteArrayInputStream::ByteArrayInputStream (unsigned char * *buf*, unsigned int *count*)

Definition at line 15 of file ByteArrayInputStream.cpp.

4.3.3 Member Function Documentation

4.3.3.1 int ByteArrayInputStream::available ( ) [virtual]

Returns the number of bytes that can be read(or skipped over) from this input stream without blocking by the next caller of a method for this input stream.

Returns

Reimplemented from InputStream.

Definition at line 22 of file ByteArrayInputStream.cpp.

4.3.3.2 void ByteArrayInputStream::mark ( ) [virtual]

Marks the current position in this input stream.

Reimplemented from InputStream.

Definition at line 29 of file ByteArrayInputStream.cpp.

4.3.3.3 bool ByteArrayInputStream::markSupported ( ) [virtual]

Tests if this input stream supports the mark and reset methods.

Returns

Reimplemented from InputStream.

Definition at line 33 of file ByteArrayInputStream.cpp.

4.3.3.4 int ByteArrayInputStream::read ( ) [virtual]

Reads the next unsigned char of data from the input stream.

Returns

Implements InputStream.

Definition at line 37 of file ByteArrayInputStream.cpp.

4.3.3.5 void ByteArrayInputStream::reset ( ) [virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

Reimplemented from InputStream.

Definition at line 41 of file ByteArrayInputStream.cpp.

4.3.4 Member Data Documentation

4.3.4.1 unsigned char* ByteArrayInputStream::buf [protected]

Definition at line 21 of file ByteArrayInputStream.h.

4.3.4.2 unsigned int ByteArrayInputStream::count [protected]

Definition at line 26 of file ByteArrayInputStream.h.

4.3.4.3 unsigned int ByteArrayInputStream::markpos [protected]

Definition at line 36 of file ByteArrayInputStream.h.

4.3.4.4 unsigned int ByteArrayInputStream::pos [protected]

Definition at line 31 of file ByteArrayInputStream.h.

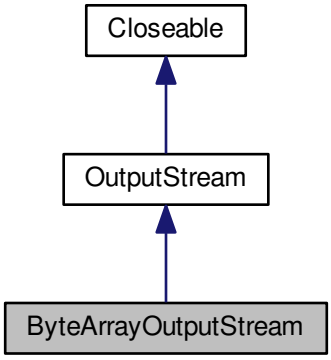
The documentation for this class was generated from the following files:

- ByteArrayInputStream.h
- ByteArrayInputStream.cpp

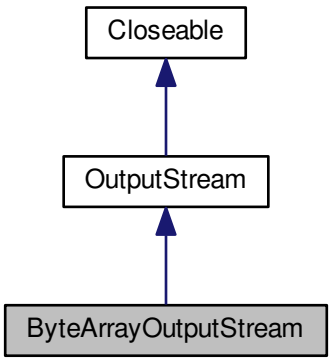
4.4 ByteArrayOutputStream Class Reference

`#include <ByteArrayOutputStream.h>`

Inheritance diagram for ByteArrayOutputStream:



Collaboration diagram for ByteArrayOutputStream:



Public Member Functions

- ByteArrayOutputStream (unsigned char *buf, unsigned int count)
- void reset ()
- unsigned int size ()
- unsigned char * toByteArray ()
- virtual void write (unsigned char b)

Protected Attributes

- unsigned char * buf
- unsigned int count
- unsigned int pos

4.4.1 Detailed Description

Raspberry IO.

ByteArrayOutputStream

This class implements an output stream in which the data is written into a unsigned char array.

Definition at line 15 of file ByteArrayOutputStream.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 ByteArrayOutputStream::ByteArrayOutputStream (unsigned char * *buf*, unsigned int *count*)

Public constructor.

Parameters

buf	
count	

Definition at line 15 of file ByteArrayOutputStream.cpp.

4.4.3 Member Function Documentation

4.4.3.1 void ByteArrayOutputStream::reset ( )

Resets the count field of this unsigned char array output stream to zero.

Definition at line 21 of file ByteArrayOutputStream.cpp.

4.4.3.2 unsigned int ByteArrayOutputStream::size ( )

Returns the current size of the buffer.

Returns

unsigned int The size of the stream.

Definition at line 25 of file ByteArrayOutputStream.cpp.

4.4.3.3 unsigned char * ByteArrayOutputStream::toByteArray ( )

Creates a newly allocated unsigned char array.

Returns

unsigned char* The unsigned char array.

Definition at line 29 of file ByteArrayOutputStream.cpp.

4.4.3.4 void ByteArrayOutputStream::write (unsigned char *b*) [virtual]

Writes the specified unsigned char to this output stream.

Parameters

b	The unsigned char to be written.

Implements OutputStream.

Definition at line 33 of file ByteArrayOutputStream.cpp.

4.4.4 Member Data Documentation

4.4.4.1 unsigned char* ByteArrayOutputStream::buf [protected]

Definition at line 21 of file ByteArrayOutputStream.h.

4.4.4.2 unsigned int ByteArrayOutputStream::count [protected]

Definition at line 26 of file ByteArrayOutputStream.h.

4.4.4.3 unsigned int ByteArrayOutputStream::pos [protected]

Definition at line 31 of file ByteArrayOutputStream.h.

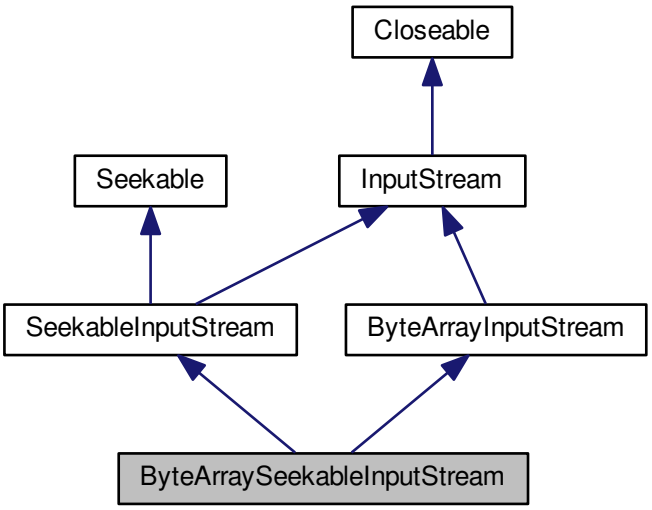
The documentation for this class was generated from the following files:

- ByteArrayOutputStream.h
- ByteArrayOutputStream.cpp

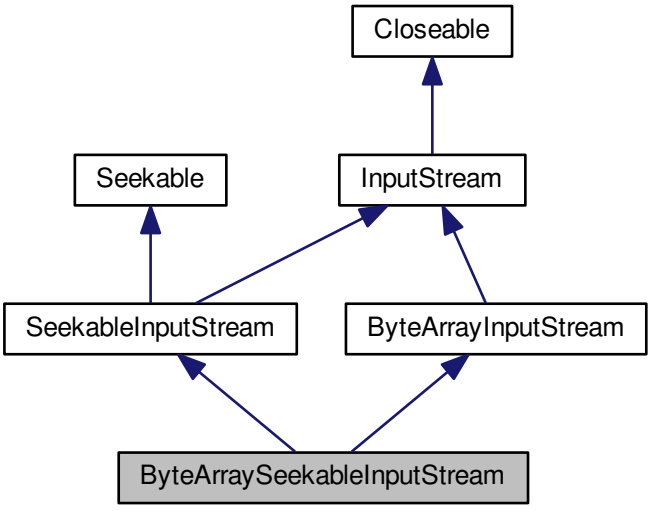
4.5 ByteArraySeekableInputStream Class Reference

```
#include <ByteArraySeekableInputStream.h>
```

Inheritance diagram for ByteArraySeekableInputStream:



Collaboration diagram for ByteArraySeekableInputStream:



Public Member Functions

- ByteArraySeekableInputStream (unsigned char *buf, unsigned int count)
- virtual void seek (unsigned int pos)

Additional Inherited Members

4.5.1 Detailed Description

Raspberry IO.

ByteArraySeekableInputStream

A ByteArraySeekableInputStream obtains input bytes from a resource in a file system that implements SeekableInputStream interface.

Definition at line 16 of file ByteArraySeekableInputStream.h.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 ByteArraySeekableInputStream::ByteArraySeekableInputStream (unsigned char * *buf,* unsigned int *count*)

Definition at line 15 of file ByteArraySeekableInputStream.cpp.

4.5.3 Member Function Documentation

4.5.3.1 void ByteArraySeekableInputStream::seek (unsigned int *pos*) [virtual]

Implements Seekable.

Definition at line 20 of file ByteArraySeekableInputStream.cpp.

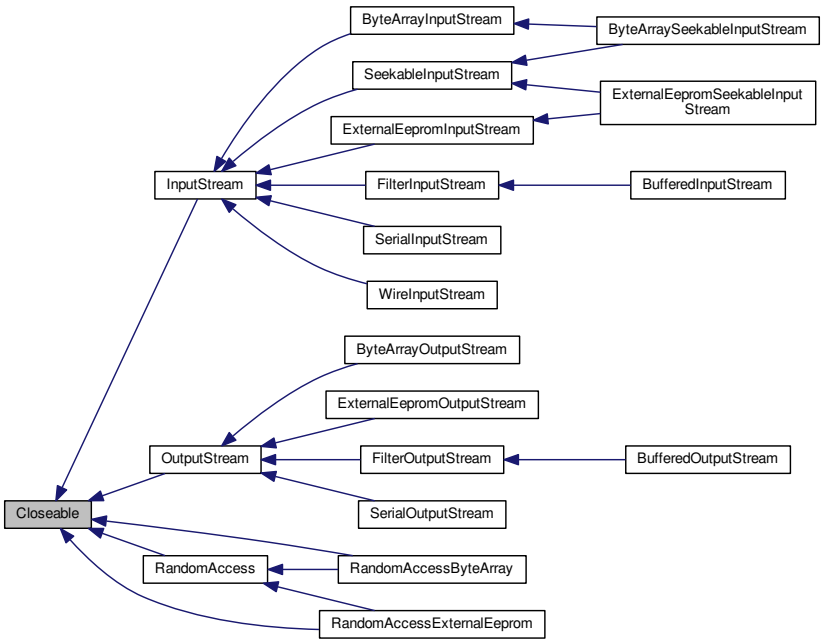
The documentation for this class was generated from the following files:

- ByteArraySeekableInputStream.h
- ByteArraySeekableInputStream.cpp

4.6 Closeable Class Reference

```
#include <Closeable.h>
```

Inheritance diagram for Closeable:



Public Member Functions

- virtual ~Closeable ()
- virtual void close ()=0

4.6.1 Detailed Description

Raspberry IO.

Closeable

A Closeable is a source or destination of data that can be closed.

Definition at line 12 of file Closeable.h.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 virtual Closeable::~Closeable ( ) [inline], [virtual]

Definition at line 15 of file Closeable.h.

4.6.3 Member Function Documentation

4.6.3.1 virtual void Closeable::close ( ) [pure virtual]

Implemented in BufferedInputStream, FilterInputStream, FilterOutputStream, BufferedOutputStream, RandomAccessByteArray, RandomAccessExternalEeprom, InputStream, and OutputStream.

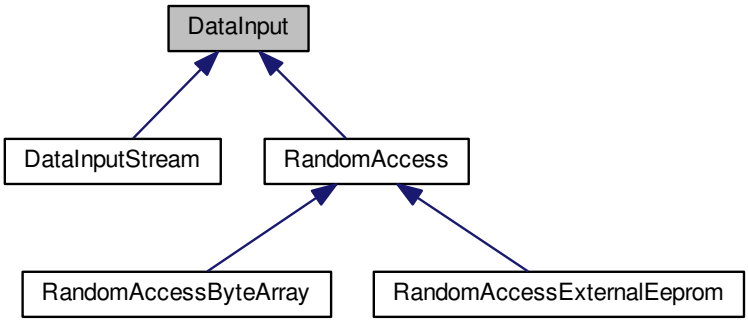
The documentation for this class was generated from the following file:

- Closeable.h

4.7 DataInput Class Reference

`#include <DataInput.h>`

Inheritance diagram for DataInput:



Public Member Functions

- virtual ~DataInput ()
- virtual unsigned char readByte ()=0
- virtual bool readBoolean ()=0
- virtual char readChar ()=0
- virtual unsigned char readUnsignedChar ()=0
- virtual int readInt ()=0
- virtual unsigned int readUnsignedInt ()=0
- virtual long readLong ()=0
- virtual unsigned long readUnsignedLong ()=0
- virtual float readFloat ()=0
- virtual double readDouble ()=0
- virtual void readFully (unsigned char *b, int len)=0
- virtual unsigned int skipBytes (unsigned int n)=0

4.7.1 Detailed Description

Raspberry IO.

DataInput

The DataInput interface provides for reading bytes from a binary stream and reconstructing from them data in any of the primitive arduino types.

Definition at line 14 of file DataInput.h.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 virtual DataInput::~DataInput ( ) [inline], [virtual]

Definition at line 17 of file DataInput.h.

4.7.3 Member Function Documentation

4.7.3.1 virtual bool DataInput::readBoolean ( ) [pure virtual]

Reads a bool from the stream.

Returns

bool

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.

4.7.3.2 virtual unsigned char DataInput::readByte ( ) [pure virtual]

Reads a unsigned char from the stream.

Returns

unsigned char

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.

4.7.3.3 virtual char DataInput::readChar ( ) [pure virtual]

Reads a char from the stream.

Returns

char

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.

4.7.3.4 virtual double DataInput::readDouble ( ) [pure virtual]

Reads a double from the stream.

Returns

double

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.

4.7.3.5 virtual float DataInput::readFloat ( ) [pure virtual]

Reads a float from the stream.

Returns

float

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.

4.7.3.6 virtual void DataInput::readFully (unsigned char * *b*, int *len*) [pure virtual]

Reads a array of bytes from the stream.

Parameters

b	
len	

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.

4.7.3.7 virtual int DataInput::readInt ( ) [pure virtual]

Reads an int from the stream.

Returns

int

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.

4.7.3.8 virtual long DataInput::readLong ( ) [pure virtual]

Reads a long from the stream.

Returns

long

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.

4.7.3.9 virtual unsigned char DataInput::readUnsignedChar ( ) [pure virtual]

Reads an unsigned char from the stream.

Returns

unsigned char

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.

4.7.3.10 virtual unsigned int DataInput::readUnsignedInt ( ) [pure virtual]

Reads an unsigned int from the stream.

Returns

unsigned int

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.

4.7.3.11 virtual unsigned long DataInput::readUnsignedLong ( ) [pure virtual]

Reads a unsigned long from the stream.

Returns

unsigned long

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.

4.7.3.12 virtual unsigned int DataInput::skipBytes (unsigned int *n*) [pure virtual]

Skips n bytes of the stream.

Parameters

n	

Returns

unsigned int The number of skipped bytes.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.

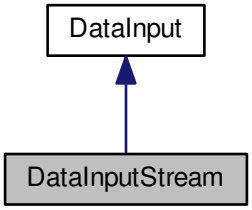
The documentation for this class was generated from the following file:

- DataInput.h

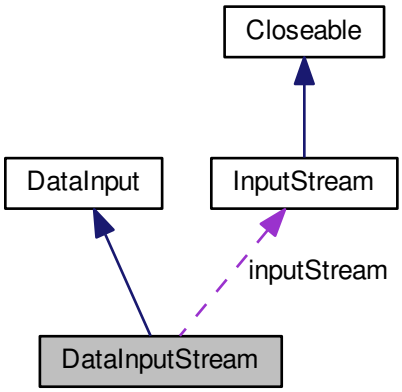
4.8 DataInputStream Class Reference

`#include <DataInputStream.h>`

Inheritance diagram for DataInputStream:



Collaboration diagram for DataInputStream:



Public Member Functions

- DataInputStream (InputStream *inputStream)
- virtual unsigned char readByte ()
- virtual bool readBoolean ()
- virtual char readChar ()
- virtual unsigned char readUnsignedChar ()
- virtual int readInt ()
- virtual unsigned int readUnsignedInt ()
- virtual long readLong ()
- virtual unsigned long readUnsignedLong ()
- virtual float readFloat ()
- virtual double readDouble ()
- virtual void readFully (unsigned char *b, int len)
- virtual unsigned int skipBytes (unsigned int n)

Private Attributes

- InputStream * inputStream

4.8.1 Detailed Description

Raspberry IO.

DataInputStream

A data input stream lets an application read data from a InputStream.

Definition at line 15 of file DataInputStream.h.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 DataInputStream::DataInputStream (InputStream * *inputStream*)

Public constructor.

Parameters

inputStream	

Definition at line 14 of file DataInputStream.cpp.

4.8.3 Member Function Documentation

4.8.3.1 bool DataInputStream::readBoolean ( ) [virtual]

Reads a bool from the stream.

Returns

bool

Implements DataInput.

Definition at line 22 of file DataInputStream.cpp.

4.8.3.2 unsigned char DataInputStream::readByte ( ) [virtual]

Reads a unsigned char from the stream.

Returns

unsigned char

Implements DataInput.

Definition at line 18 of file DataInputStream.cpp.

4.8.3.3 char DataInputStream::readChar ( ) [virtual]

Reads a char from the stream.

Returns

char

Implements DataInput.

Definition at line 26 of file DataInputStream.cpp.

4.8.3.4 double DataInputStream::readDouble ( ) [virtual]

Reads a double from the stream.

Returns

double

Implements DataInput.

Definition at line 66 of file DataInputStream.cpp.

4.8.3.5 float DataInputStream::readFloat ( ) [virtual]

Reads a float from the stream.

Returns

float

Implements DataInput.

Definition at line 62 of file DataInputStream.cpp.

4.8.3.6 void DataInputStream::readFully (unsigned char * *b*, int *len*) [virtual]

Reads a array of bytes from the stream.

Parameters

b	
len	

Implements DataInput.

Definition at line 70 of file DataInputStream.cpp.

4.8.3.7 int DataInputStream::readInt ( ) [virtual]

Reads an int from the stream.

Returns

int

Implements DataInput.

Definition at line 34 of file DataInputStream.cpp.

4.8.3.8 long DataInputStream::readLong ( ) [virtual]

Reads a long from the stream.

Returns

long

Implements DataInput.

Definition at line 46 of file DataInputStream.cpp.

4.8.3.9 unsigned char DataInputStream::readUnsignedChar ( ) [virtual]

Reads an unsigned char from the stream.

Returns

unsigned char

Implements DataInput.

Definition at line 30 of file DataInputStream.cpp.

4.8.3.10 unsigned int DataInputStream::readUnsignedInt ( ) [virtual]

Reads an unsigned int from the stream.

Returns

unsigned int

Implements DataInput.

Definition at line 42 of file DataInputStream.cpp.

4.8.3.11 unsigned long DataInputStream::readUnsignedLong ( ) [virtual]

Reads a unsigned long from the stream.

Returns

unsigned long

Implements DataInput.

Definition at line 58 of file DataInputStream.cpp.

4.8.3.12 unsigned int DataInputStream::skipBytes (unsigned int *n*) [virtual]

Skips n bytes of the stream.

Parameters

n	

Returns

unsigned int The number of skipped bytes.

Implements DataInput.

Definition at line 76 of file DataInputStream.cpp.

4.8.4 Member Data Documentation

4.8.4.1 InputStream* DataInputStream::inputStream [private]

The used input stream.

Definition at line 20 of file DataInputStream.h.

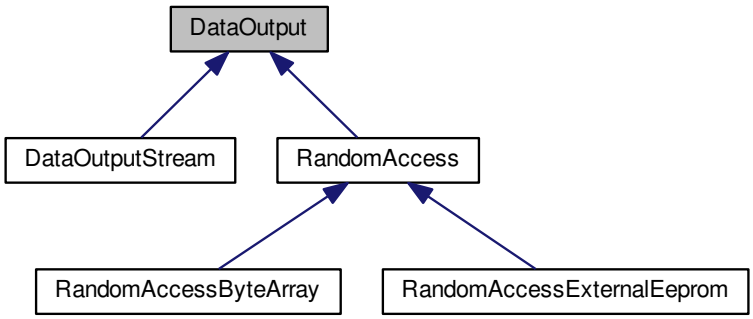
The documentation for this class was generated from the following files:

- DataInputStream.h
- DataInputStream.cpp

4.9 DataOutput Class Reference

`#include <DataOutput.h>`

Inheritance diagram for DataOutput:



Public Member Functions

- virtual void write (unsigned char *b, int len)=0
- virtual void write (unsigned char b)=0
- virtual void writeByte (unsigned char b)=0
- virtual void writeBytes (unsigned char *b, int len)=0
- virtual void writeBoolean (bool v)=0
- virtual void writeChar (char c)=0
- virtual void writeUnsignedChar (unsigned char c)=0

- virtual void writeInt (int v)=0
- virtual void writeUnsignedInt (unsigned int v)=0
- virtual void writeLong (long v)=0
- virtual void writeUnsignedLong (unsigned long v)=0
- virtual void writeFloat (float v)=0
- virtual void writeDouble (double v)=0

4.9.1 Detailed Description

Raspberry IO.

DataOutput

The DataOutput interface provides for converting data from any of the primitive types to a series of bytes and writing these bytes to a binary stream.

Definition at line 13 of file DataOutput.h.

4.9.2 Member Function Documentation

4.9.2.1 virtual void DataOutput::write (unsigned char * *b*, int *len*) [pure virtual]

Writes an array of bytes into the stream.

Parameters

b	The array of bytes.
len	The length of such array.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

4.9.2.2 virtual void DataOutput::write (unsigned char *b*) [pure virtual]

Writes a unsigned char into the stream.

Parameters

b	The unsigned char to be written.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

4.9.2.3 virtual void DataOutput::writeBoolean (bool *v*) [pure virtual]

Writes a bool into the stream.

Parameters

v	The bool to be written.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

4.9.2.4 virtual void DataOutput::writeByte (unsigned char *b*) [pure virtual]

Writes a unsigned char into the stream.

Parameters

b	The unsigned char to be written.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

4.9.2.5 virtual void DataOutput::writeBytes (unsigned char * *b*, int *len*) [pure virtual]

Writes an array of bytes into the stream.

Parameters

b	The array of bytes.
len	The length of such array.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

4.9.2.6 virtual void DataOutput::writeChar (char *c*) [pure virtual]

Writes a char into the stream.

Parameters

c	The char to be written.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

4.9.2.7 virtual void DataOutput::writeDouble (double *v*) [pure virtual]

Writes a double into the stream.

Parameters

v	The double to be written.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

4.9.2.8 virtual void DataOutput::writeFloat (float *v*) [pure virtual]

Writes a float into the stream.

Parameters

v	The float to be written.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

4.9.2.9 virtual void DataOutput::writeInt (int *v*) [pure virtual]

Writes an int into the stream.

Parameters

v	The int to be written.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

4.9.2.10 virtual void DataOutput::writeLong (long *v*) [pure virtual]

Writes a long into the stream.

Parameters

v	The long to be written.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

4.9.2.11 virtual void DataOutput::writeUnsignedChar (unsigned char *c*) [pure virtual]

Writes an unsigned char into the stream.

Parameters

c	The unsigned char to be written.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

4.9.2.12 virtual void DataOutput::writeUnsignedInt (unsigned int *v*) [pure virtual]

Writes an unsigned int into the stream.

Parameters

v	The unsigned int to be written.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

4.9.2.13 virtual void DataOutput::writeUnsignedLong (unsigned long *v*) [pure virtual]

Writes a unsigned long into the stream.

Parameters

v	The unsigned long to be written.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

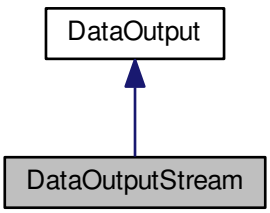
The documentation for this class was generated from the following file:

- DataOutput.h

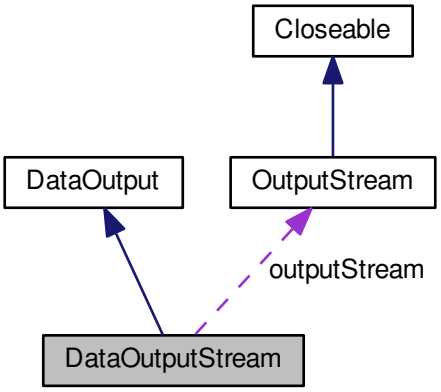
4.10 DataOutputStream Class Reference

`#include <DataOutputStream.h>`

Inheritance diagram for DataOutputStream:



Collaboration diagram for DataOutputStream:



Public Member Functions

- DataOutputStream (OutputStream ∗outputStream)
- virtual void write (unsigned char ∗b, int len)
- virtual void write (unsigned char b)
- virtual void writeByte (unsigned char b)
- virtual void writeBytes (unsigned char ∗b, int len)
- virtual void writeBoolean (bool v)
- virtual void writeChar (char c)
- virtual void writeUnsignedChar (unsigned char c)
- virtual void writeInt (int v)
- virtual void writeUnsignedInt (unsigned int v)
- virtual void writeLong (long v)
- virtual void writeUnsignedLong (unsigned long v)
- virtual void writeFloat (float v)
- virtual void writeDouble (double v)

Private Attributes

- OutputStream ∗ outputStream

4.10.1 Detailed Description

Raspberry IO.

DataOutputStream

A data output stream lets an application write types to an OutputStream.

Definition at line 16 of file DataOutputStream.h.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 DataOutputStream::DataOutputStream (OutputStream * *outputStream*)

Public constructor.

Parameters

outputStream	The stream to be used.

Definition at line 14 of file DataOutputStream.cpp.

4.10.3 Member Function Documentation

4.10.3.1 void DataOutputStream::write (unsigned char * *b*, int *len*) [virtual]

Writes an array of bytes into the stream.

Parameters

b	The array of bytes.
len	The length of such array.

Implements DataOutput.

Definition at line 18 of file DataOutputStream.cpp.

4.10.3.2 void DataOutputStream::write (unsigned char *b*) [virtual]

Writes a unsigned char into the stream.

Parameters

b	The unsigned char to be written.

Implements DataOutput.

Definition at line 22 of file DataOutputStream.cpp.

4.10.3.3 void DataOutputStream::writeBoolean (bool *v*) [virtual]

Writes a bool into the stream.

Parameters

v	The bool to be written.

Implements DataOutput.

Definition at line 36 of file DataOutputStream.cpp.

4.10.3.4 void DataOutputStream::writeByte (unsigned char *b*) [virtual]

Writes a unsigned char into the stream.

Parameters

b	The unsigned char to be written.

Implements DataOutput.

Definition at line 26 of file DataOutputStream.cpp.

4.10.3.5 void DataOutputStream::writeBytes (unsigned char * *b*, int *len*) [virtual]

Writes an array of bytes into the stream.

Parameters

b	The array of bytes.
len	The length of such array.

Implements DataOutput.

Definition at line 30 of file DataOutputStream.cpp.

4.10.3.6 void DataOutputStream::writeChar (char *c*) [virtual]

Writes a char into the stream.

Parameters

c	The char to be written.

Implements DataOutput.

Definition at line 40 of file DataOutputStream.cpp.

4.10.3.7 void DataOutputStream::writeDouble (double *v*) [virtual]

Writes a double into the stream.

Parameters

v	The double to be written.

Implements DataOutput.

Definition at line 72 of file DataOutputStream.cpp.

4.10.3.8 void DataOutputStream::writeFloat (float *v*) [virtual]

Writes a float into the stream.

Parameters

v	The float to be written.

Implements DataOutput.

Definition at line 68 of file DataOutputStream.cpp.

4.10.3.9 void DataOutputStream::writeInt (int *v*) [virtual]

Writes an int into the stream.

Parameters

v	The int to be written.

Implements DataOutput.

Definition at line 48 of file DataOutputStream.cpp.

4.10.3.10 void DataOutputStream::writeLong (long *v*) [virtual]

Writes a long into the stream.

Parameters

v	The long to be written.

Implements DataOutput.

Definition at line 57 of file DataOutputStream.cpp.

4.10.3.11 void DataOutputStream::writeUnsignedChar (unsigned char *c*) [virtual]

Writes an unsigned char into the stream.

Parameters

c	The unsigned char to be written.

Implements DataOutput.

Definition at line 44 of file DataOutputStream.cpp.

4.10.3.12 void DataOutputStream::writeUnsignedInt (unsigned int *v*) [virtual]

Writes an unsigned int into the stream.

Parameters

v	The unsigned int to be written.

Implements DataOutput.

Definition at line 53 of file DataOutputStream.cpp.

4.10.3.13 void DataOutputStream::writeUnsignedLong (unsigned long *v*) [virtual]

Writes a unsigned long into the stream.

Parameters

v	The unsigned long to be written.

Implements DataOutput.

Definition at line 64 of file DataOutputStream.cpp.

4.10.4 Member Data Documentation

4.10.4.1 OutputStream* DataOutputStream::outputStream [private]

The stream to be used.

Definition at line 21 of file DataOutputStream.h.

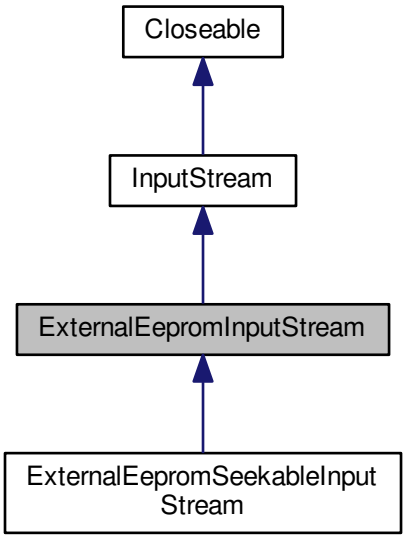
The documentation for this class was generated from the following files:

- DataOutputStream.h
- DataOutputStream.cpp

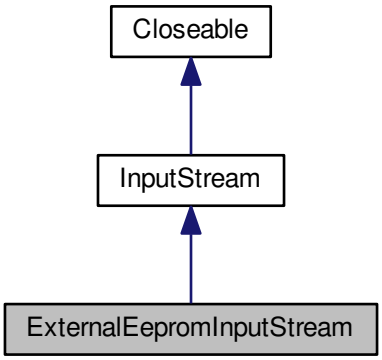
4.11 ExternalEepromInputStream Class Reference

`#include <ExternalEepromInputStream.h>`

Inheritance diagram for ExternalEepromInputStream:



Collaboration diagram for ExternalEepromInputStream:



Public Member Functions

- ExternalEepromInputStream (ExternalEeprom ∗externalEeprom)
- virtual int available ()
- virtual void mark ()
- virtual bool markSupported ()
- virtual int read ()
- virtual int read (unsigned char ∗b, int off, int len)
- virtual void reset ()

Protected Attributes

- ExternalEeprom * externalEeprom
- unsigned int pos
- unsigned int markpos
- unsigned int externalEepromSize

4.11.1 Detailed Description

Raspberry IO.

ExternalEepromInputStream

An ExternalEepromInputStream obtains input bytes from a externalEeprom.

Definition at line 16 of file ExternalEepromInputStream.h.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 ExternalEepromInputStream::ExternalEepromInputStream (ExternalEeprom * *externalEeprom*)

Public constructor.

Parameters

externalEeprom	The externalEeprom where data is stored.

Definition at line 15 of file ExternalEepromInputStream.cpp.

4.11.3 Member Function Documentation

4.11.3.1 int ExternalEepromInputStream::available ( ) [virtual]

Returns the number of bytes that can be read(or skipped over) from this input stream without blocking by the next caller of a method for this input stream.

Returns

int The available number of bytes.

Reimplemented from InputStream.

Definition at line 23 of file ExternalEepromInputStream.cpp.

4.11.3.2 void ExternalEepromInputStream::mark ( ) [virtual]

Marks the current position in this input stream.

Reimplemented from InputStream.

Definition at line 30 of file ExternalEepromInputStream.cpp.

4.11.3.3 bool ExternalEepromInputStream::markSupported ( ) [virtual]

Tests if this input stream supports the mark and reset methods.

Returns

bool

Reimplemented from InputStream.

Definition at line 34 of file ExternalEepromInputStream.cpp.

4.11.3.4 int ExternalEepromInputStream::read ( ) [virtual]

Reads the next unsigned char of data from the input stream.

Returns

int The read unsigned char as an int.

Implements InputStream.

Definition at line 38 of file ExternalEepromInputStream.cpp.

4.11.3.5 int ExternalEepromInputStream::read (unsigned char * *b*, int *off*, int *len*) [virtual]

Reads len of bytes from the input stream.

Parameters

b	
off	
len	

Returns

Reimplemented from InputStream.

Definition at line 45 of file ExternalEepromInputStream.cpp.

4.11.3.6 void ExternalEepromInputStream::reset ( ) [virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

Reimplemented from InputStream.

Definition at line 54 of file ExternalEepromInputStream.cpp.

4.11.4 Member Data Documentation

4.11.4.1 ExternalEeprom* ExternalEepromInputStream::externalEeprom [protected]

Definition at line 22 of file ExternalEepromInputStream.h.

4.11.4.2 unsigned int ExternalEepromInputStream::externalEepromSize [protected]

Definition at line 37 of file ExternalEepromInputStream.h.

4.11.4.3 unsigned int ExternalEepromInputStream::markpos [protected]

Definition at line 32 of file ExternalEepromInputStream.h.

4.11.4.4 unsigned int ExternalEepromInputStream::pos [protected]

Definition at line 27 of file ExternalEepromInputStream.h.

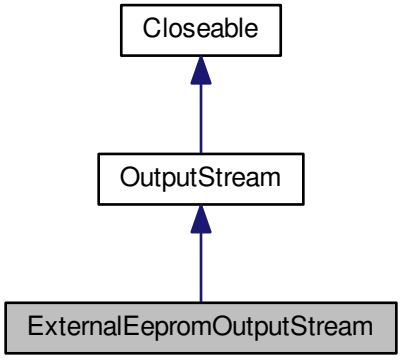
The documentation for this class was generated from the following files:

- ExternalEepromInputStream.h
- ExternalEepromInputStream.cpp

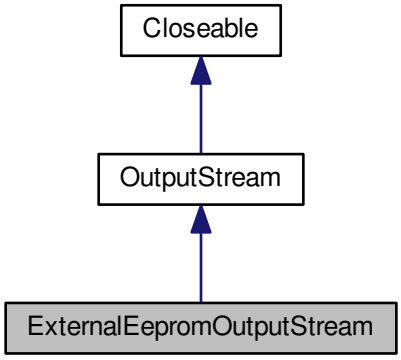
4.12 ExternalEepromOutputStream Class Reference

`#include <ExternalEepromOutputStream.h>`

Inheritance diagram for ExternalEepromOutputStream:



Collaboration diagram for ExternalEepromOutputStream:



Public Member Functions

- ExternalEepromOutputStream (ExternalEeprom ∗externalEeprom)
- virtual void write (unsigned char b)
- virtual void write (unsigned char ∗b, int off, int len)

Private Attributes

- ExternalEeprom ∗ externalEeprom
- unsigned int pos

4.12.1 Detailed Description

Raspberry IO.

ExternalEepromOutputStream

A resource output stream is an output stream for writing data to an ExternalEeprom.

Definition at line 16 of file ExternalEepromOutputStream.h.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 ExternalEepromOutputStream::ExternalEepromOutputStream (ExternalEeprom * *externalEeprom*)

Public constructor.

Parameters

externalEeprom	

Definition at line 14 of file ExternalEepromOutputStream.cpp.

4.12.3 Member Function Documentation

4.12.3.1 void ExternalEepromOutputStream::write (unsigned char *b*) [virtual]

Writes the specified unsigned char to this output stream.

Parameters

b	

Implements OutputStream.

Definition at line 20 of file ExternalEepromOutputStream.cpp.

4.12.3.2 void ExternalEepromOutputStream::write (unsigned char * *b*, int *off*, int *len*) [virtual]

Writes len bytes from the specified unsigned char array starting at offset off to this output stream.

Parameters

b	
off	
len	

Reimplemented from OutputStream.

Definition at line 24 of file ExternalEepromOutputStream.cpp.

4.12.4 Member Data Documentation

4.12.4.1 ExternalEeprom* ExternalEepromOutputStream::externalEeprom [private]

The associated eeprom.

Definition at line 21 of file ExternalEepromOutputStream.h.

4.12.4.2 unsigned int ExternalEepromOutputStream::pos [private]

Current eeprom position.

Definition at line 26 of file ExternalEepromOutputStream.h.

The documentation for this class was generated from the following files:

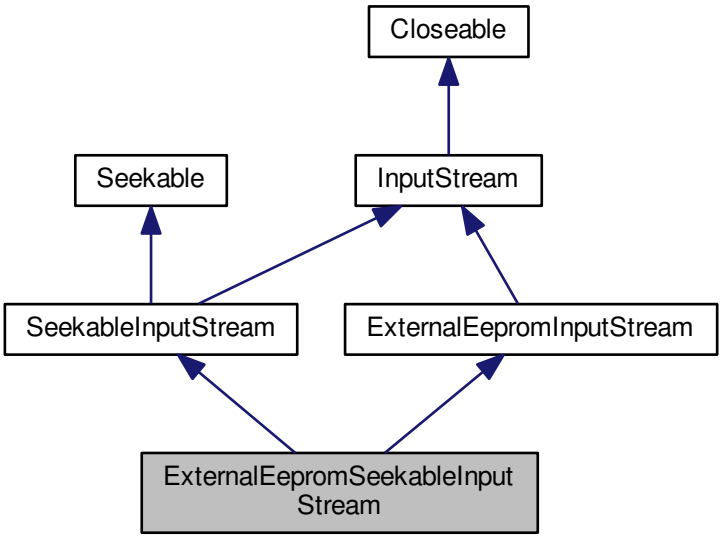
- ExternalEepromOutputStream.h

- ExternalEepromOutputStream.cpp

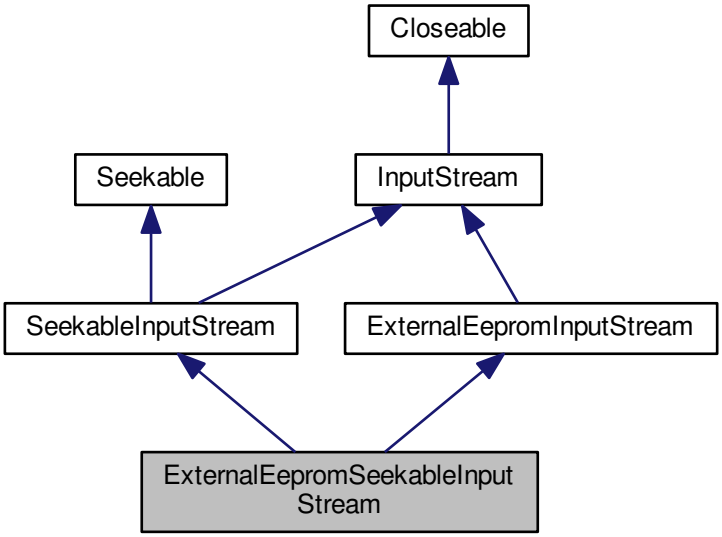
4.13 ExternalEepromSeekableInputStream Class Reference

`#include <ExternalEepromSeekableInputStream.h>`

Inheritance diagram for ExternalEepromSeekableInputStream:



Collaboration diagram for ExternalEepromSeekableInputStream:



Public Member Functions

- ExternalEepromSeekableInputStream (ExternalEeprom *externalEeprom)
- virtual void seek (unsigned int pos)

Additional Inherited Members

4.13.1 Detailed Description

Raspberry IO.

ExternalEepromSeekableInputStream

A ExternalEepromSeekableInputStream obtains input bytes from a external input stream.

Definition at line 17 of file ExternalEepromSeekableInputStream.h.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 ExternalEepromSeekableInputStream::ExternalEepromSeekableInputStream (ExternalEeprom * *externalEeprom*)

Public constructor.

Parameters

resource	The external eeprom to be used.

Definition at line 15 of file ExternalEepromSeekableInputStream.cpp.

4.13.3 Member Function Documentation

4.13.3.1 void ExternalEepromSeekableInputStream::seek (unsigned int *pos*) [virtual]

Seeks this input stream to the position.

Parameters

pos	THe position.

Implements Seekable.

Definition at line 20 of file ExternalEepromSeekableInputStream.cpp.

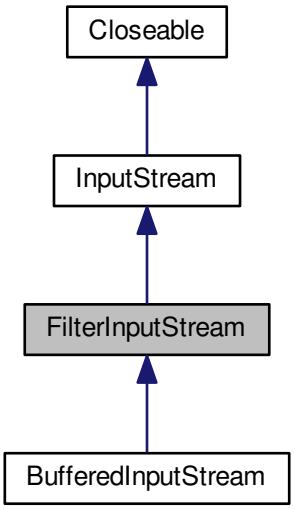
The documentation for this class was generated from the following files:

- ExternalEepromSeekableInputStream.h
- ExternalEepromSeekableInputStream.cpp

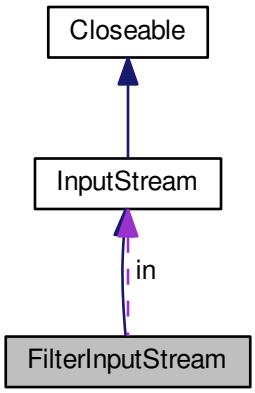
4.14 FilterInputStream Class Reference

`#include <FilterInputStream.h>`

Inheritance diagram for FilterInputStream:



Collaboration diagram for FilterInputStream:



Public Member Functions

- virtual int read ()
- virtual int read (unsigned char *b, int len)
- virtual int read (unsigned char *b, int off, int len)
- virtual unsigned int skip (unsigned int n)
- virtual int available ()
- virtual void close ()
- virtual void mark ()
- virtual void reset ()
- virtual bool markSupported ()

Protected Member Functions

- FilterInputStream (InputStream *in)

Protected Attributes

- InputStream * in

4.14.1 Detailed Description

A `FilterInputStream` contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.

The class `FilterInputStream` itself simply overrides all methods of `InputStream` with versions that pass all requests to the contained input stream. Subclasses of `FilterInputStream` may further override some of these methods and may also provide additional methods and fields.

Definition at line 21 of file FilterInputStream.h.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 FilterInputStream::FilterInputStream (InputStream * *in*) [protected]

Creates a `FilterInputStream` by assigning the argument `in` to the field `this->in` so as to remember it for later use.

Parameters

in	the underlying input stream

Definition at line 21 of file FilterInputStream.cpp.

4.14.3 Member Function Documentation

4.14.3.1 int FilterInputStream::available ( ) [virtual]

Returns an estimate of the number of bytes that can be read (or skipped over) from this input stream without blocking by the next caller of a method for this input stream.

Returns

an estimate of the number of bytes that can be read (or skipped over) from this input stream without blocking.

Reimplemented from InputStream.

Reimplemented in BufferedInputStream.

Definition at line 41 of file FilterInputStream.cpp.

4.14.3.2 void FilterInputStream::close ( ) [virtual]

Closes this input stream.

This method simply performs `in->close()`.

Reimplemented from InputStream.

Reimplemented in BufferedInputStream.

Definition at line 45 of file FilterInputStream.cpp.

4.14.3.3 void FilterInputStream::mark ( ) [virtual]

Marks the current position in this input stream.

A subsequent call to the `reset` method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

This method simply performs `in->mark()`.

Reimplemented from InputStream.

Reimplemented in BufferedInputStream.

Definition at line 49 of file FilterInputStream.cpp.

4.14.3.4 bool FilterInputStream::markSupported ( ) [virtual]

Tests if this input stream supports the `mark` and `reset` methods.

This method simply performs `in->markSupported()`.

Returns

`true` if this stream type supports the `mark` and `reset` method; `false` otherwise.

Reimplemented from InputStream.

Reimplemented in BufferedInputStream.

Definition at line 57 of file FilterInputStream.cpp.

4.14.3.5 int FilterInputStream::read ( ) [virtual]

Reads the next unsigned char of data from this input stream.

The value unsigned char is returned as an `int` in the range `0` to `255`. If no unsigned char is available because the end of the stream has been reached, the value `-1` is returned.

This method simply performs `in->read()` and returns the result.

Returns

the next unsigned char of data, or `-1` if the end of the stream is reached.

Implements InputStream.

Reimplemented in BufferedInputStream.

Definition at line 25 of file FilterInputStream.cpp.

4.14.3.6 int FilterInputStream::read (unsigned char * *b*, int *len*) [virtual]

Reads up to `len` bytes of data from this input stream into an array of bytes.

This method simply performs the call `read(b, 0, len)` and returns the result. It is important that it does *not* do `in->read(b)` instead; certain subclasses of `FilterInputStream` depend on the implementation strategy actually used.

Parameters

b	the buffer into which the data is read.

Returns

the total number of bytes read into the buffer, or `-1` if there is no more data because the end of the stream has been reached.

Reimplemented from InputStream.

Reimplemented in BufferedInputStream.

Definition at line 29 of file FilterInputStream.cpp.

4.14.3.7 int FilterInputStream::read (unsigned char * *b*, int *off*, int *len*) [virtual]

Reads up to `len` bytes of data from this input stream into an array of bytes.

This method simply performs `in->read(b, off, len)` and returns the result.

Parameters

b	the buffer into which the data is read.
off	the start offset in the destination array `b`
len	the maximum number of bytes read.

Returns

the total number of bytes read into the buffer, or `-1` if there is no more data because the end of the stream has been reached.

Reimplemented from InputStream.

Reimplemented in BufferedInputStream.

Definition at line 33 of file FilterInputStream.cpp.

4.14.3.8 void FilterInputStream::reset ( ) [virtual]

Repositions this stream to the position at the time the `mark` method was last called on this input stream.

This method simply performs `in->reset()`.

Stream marks are intended to be used in situations where you need to read ahead a little to see what's in the stream. Often this is most easily done by invoking some general parser. If the stream is of the type handled by the parse, it just chugs along happily. If the stream is not of that type, the parser should toss an exception when it fails.

Reimplemented from InputStream.

Reimplemented in BufferedInputStream.

Definition at line 53 of file FilterInputStream.cpp.

4.14.3.9 unsigned int FilterInputStream::skip (unsigned int *n*) [virtual]

This method simply performs `in->skip(n)`.

Parameters

Reimplemented from InputStream.

Reimplemented in BufferedInputStream.

Definition at line 37 of file FilterInputStream.cpp.

4.14.4 Member Data Documentation

4.14.4.1 InputStream* FilterInputStream::in [protected]

The input stream to be filtered.

Definition at line 28 of file FilterInputStream.h.

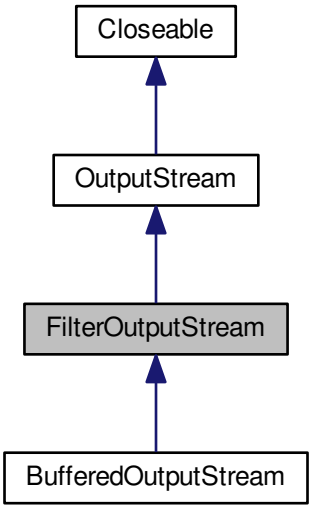
The documentation for this class was generated from the following files:

- FilterInputStream.h
- FilterInputStream.cpp

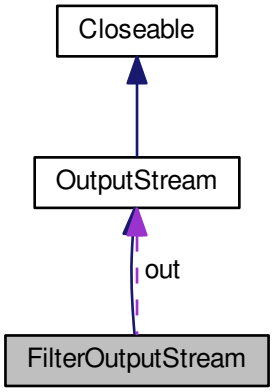
4.15 FilterOutputStream Class Reference

`#include <FilterOutputStream.h>`

Inheritance diagram for FilterOutputStream:



Collaboration diagram for FilterOutputStream:



Public Member Functions

- FilterOutputStream (OutputStream *out)
- virtual void write (unsigned char b)
- virtual void write (unsigned char *b, int len)
- virtual void write (unsigned char *b, int off, int len)
- virtual void flush ()
- virtual void close ()

Protected Attributes

- OutputStream * out

4.15.1 Detailed Description

Raspberry IO.

FilterOutputStream

This class is the superclass of all classes that filter output streams. These streams sit on top of an already existing output stream (the *underlying* output stream) which it uses as its basic sink of data, but possibly transforming the data along the way or providing additional functionality.

The class `FilterOutputStream` itself simply overrides all methods of `OutputStream` with versions that pass all requests to the underlying output stream. Subclasses of `FilterOutputStream` may further override some of these methods as well as provide additional methods and fields.

Definition at line 24 of file FilterOutputStream.h.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 FilterOutputStream::FilterOutputStream (OutputStream * *out*)

Creates an output stream filter built on top of the specified underlying output stream.

Parameters

out	the underlying output stream to be assigned to the field `this->out` for later use.

Definition at line 24 of file FilterOutputStream.cpp.

4.15.3 Member Function Documentation

4.15.3.1 void FilterOutputStream::close ( ) [virtual]

Closes this output stream and releases any system resources associated with the stream.

The `close` method of `FilterOutputStream` calls its `flush` method, and then calls the `close` method of its underlying output stream.

Reimplemented from OutputStream.

Reimplemented in BufferedOutputStream.

Definition at line 44 of file FilterOutputStream.cpp.

4.15.3.2 void FilterOutputStream::flush ( ) [virtual]

Flushes this output stream and forces any buffered output bytes to be written out to the stream.

The `flush` method of `FilterOutputStream` calls the `flush` method of its underlying output stream.

Reimplemented from OutputStream.

Reimplemented in BufferedOutputStream.

Definition at line 40 of file FilterOutputStream.cpp.

4.15.3.3 void FilterOutputStream::write (unsigned char *b*) [virtual]

Writes the specified `unsigned char` to this output stream.

The `write` method of `FilterOutputStream` calls the `write` method of its underlying output stream, that is, it performs `out->write(b)`.

Implements the abstract `write` method of `OutputStream`.

Parameters

b	the `unsigned char`.

Implements OutputStream.

Reimplemented in BufferedOutputStream.

Definition at line 28 of file FilterOutputStream.cpp.

4.15.3.4 void FilterOutputStream::write (unsigned char * *b*, int *len*) [virtual]

Writes `len` bytes to this output stream.

The `write` method of `FilterOutputStream` calls its `write` method of two arguments with the arguments `b` and <codelen.

Parameters

b	the data to be written.
len	the length

Reimplemented from OutputStream.

Reimplemented in BufferedOutputStream.

Definition at line 32 of file FilterOutputStream.cpp.

4.15.3.5 void FilterOutputStream::write (unsigned char * *b*, int *off*, int *len*) [virtual]

Writes `len` bytes from the specified `unsigned char` array starting at offset `off` to this output stream.

Parameters

b	the data.
off	the start offset in the data.
len	the number of bytes to write.

Reimplemented from OutputStream.

Reimplemented in BufferedOutputStream.

Definition at line 36 of file FilterOutputStream.cpp.

4.15.4 Member Data Documentation

4.15.4.1 OutputStream* FilterOutputStream::out [protected]

The underlying output stream to be filtered.

Definition at line 30 of file FilterOutputStream.h.

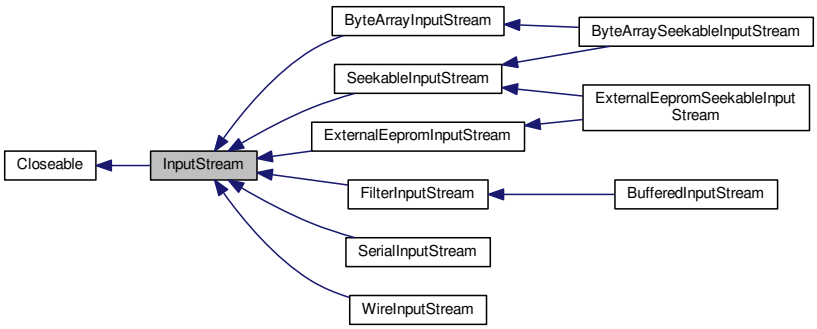
The documentation for this class was generated from the following files:

- FilterOutputStream.h
- FilterOutputStream.cpp

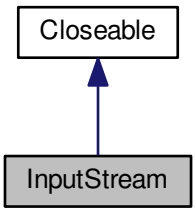
4.16 InputStream Class Reference

`#include <InputStream.h>`

Inheritance diagram for InputStream:



Collaboration diagram for InputStream:



Public Member Functions

- virtual ~InputStream ()
- virtual int available ()
- virtual void close ()
- virtual void mark ()
- virtual bool markSupported ()
- virtual int read ()=0
- virtual int read (unsigned char *b, int len)
- virtual int read (unsigned char *b, int off, int len)
- virtual void reset ()
- virtual unsigned int skip (unsigned int n)

4.16.1 Detailed Description

Raspberry IO.

InputStream

This abstract class is the superclass of all classes representing an input stream of bytes.

Applications that need to define a subclass of InputStream must always provide a method that returns the next unsigned char of input.

Definition at line 18 of file InputStream.h.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 virtual InputStream::~InputStream ( ) [inline], [virtual]

Definition at line 21 of file InputStream.h.

4.16.3 Member Function Documentation

4.16.3.1 int InputStream::available ( ) [virtual]

Returns the number of bytes that can be read(or skipped over) from this input stream without blocking by the next caller of a method for this input stream.

Reimplemented in BufferedInputStream, FilterInputStream, SerialInputStream, ExternalEepromInputStream, ByteArrayInputStream, and WireInputStream.

Definition at line 18 of file InputStream.cpp.

4.16.3.2 void InputStream::close ( ) [virtual]

Closes this input stream and releases any system resources associated with the stream.

Implements Closeable.

Reimplemented in BufferedInputStream, and FilterInputStream.

Definition at line 22 of file InputStream.cpp.

4.16.3.3 void InputStream::mark ( ) [virtual]

Marks the current position in this input stream.

Reimplemented in BufferedInputStream, FilterInputStream, ExternalEepromInputStream, and ByteArrayInputStream.

Definition at line 25 of file InputStream.cpp.

4.16.3.4 bool InputStream::markSupported ( ) [virtual]

Tests if this input stream supports the mark and reset methods.

Reimplemented in FilterInputStream, BufferedInputStream, ExternalEepromInputStream, and ByteArrayInputStream.

Definition at line 28 of file InputStream.cpp.

4.16.3.5 virtual int InputStream::read ( ) [pure virtual]

Reads the next unsigned char of data from the input stream.

Implemented in BufferedInputStream, SerialInputStream, ExternalEepromInputStream, ByteArrayInputStream, FilterInputStream, and WireInputStream.

4.16.3.6 int InputStream::read (unsigned char * *b*, int *len*) [virtual]

Reads some number of bytes from the input stream and stores them into the buffer array b.

Reimplemented in BufferedInputStream, SerialInputStream, and FilterInputStream.

Definition at line 32 of file InputStream.cpp.

4.16.3.7 int InputStream::read (unsigned char * *b*, int *off*, int *len*) [virtual]

Writes len of bytes into the stream.

Parameters

b	
off	
len	

Returns

Reimplemented in BufferedInputStream, FilterInputStream, ExternalEepromInputStream, and WireInputStream.

Definition at line 36 of file InputStream.cpp.

4.16.3.8 void InputStream::reset ( ) [virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream.

Reimplemented in BufferedInputStream, FilterInputStream, ExternalEepromInputStream, and ByteArrayInputStream.

Definition at line 56 of file InputStream.cpp.

4.16.3.9 unsigned int InputStream::skip (unsigned int *n*) [virtual]

Skips over and discards n bytes of data from this input stream.

Reimplemented in BufferedInputStream, and FilterInputStream.

Definition at line 59 of file InputStream.cpp.

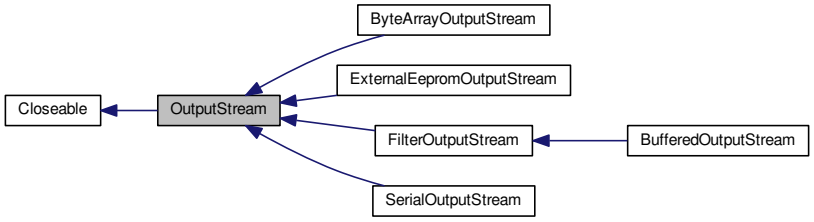
The documentation for this class was generated from the following files:

- InputStream.h
- InputStream.cpp

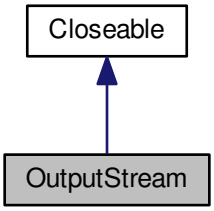
4.17 OutputStream Class Reference

`#include <OutputStream.h>`

Inheritance diagram for OutputStream:



Collaboration diagram for OutputStream:



Public Member Functions

- virtual ~OutputStream ()
- virtual void close ()
- virtual void flush ()
- virtual void write (unsigned char b)=0
- virtual void write (unsigned char *b, int len)
- virtual void write (unsigned char *b, int off, int len)

4.17.1 Detailed Description

Raspberry IO.

OutputStream

This abstract class is the superclass of all classes representing an output stream of bytes. An output stream accepts output bytes and sends them to some sink.

Applications that need to define a subclass of `OutputStream` must always provide at least a method that writes one unsigned char of output.

Definition at line 20 of file OutputStream.h.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 virtual OutputStream::~OutputStream ( ) [inline], [virtual]

Definition at line 23 of file OutputStream.h.

4.17.3 Member Function Documentation

4.17.3.1 void OutputStream::close ( ) [virtual]

Closes this output stream and releases any system resources associated with this stream.

Implements Closeable.

Reimplemented in FilterOutputStream, and BufferedOutputStream.

Definition at line 32 of file OutputStream.cpp.

4.17.3.2 void OutputStream::flush ( ) [virtual]

Flushes this output stream and forces any buffered output bytes to be written out.

Reimplemented in BufferedOutputStream, and FilterOutputStream.

Definition at line 29 of file OutputStream.cpp.

4.17.3.3 virtual void OutputStream::write (unsigned char *b*) [pure virtual]

Writes the specified unsigned char to this output stream.

Implemented in ByteArrayOutputStream, BufferedOutputStream, FilterOutputStream, ExternalEepromOutputStream, and SerialOutputStream.

4.17.3.4 void OutputStream::write (unsigned char * *b*, int *len*) [virtual]

Writes len bytes from the specified unsigned char array to this output stream.

Parameters

b	
len	

Reimplemented in BufferedOutputStream, and FilterOutputStream.

Definition at line 16 of file OutputStream.cpp.

4.17.3.5 void OutputStream::write (unsigned char * *b*, int *off*, int *len*) [virtual]

Writes len bytes from the specified unsigned char array starting at offset off to this output stream.

Parameters

b	
off	
len	

Reimplemented in BufferedOutputStream, FilterOutputStream, and ExternalEepromOutputStream.

Definition at line 20 of file OutputStream.cpp.

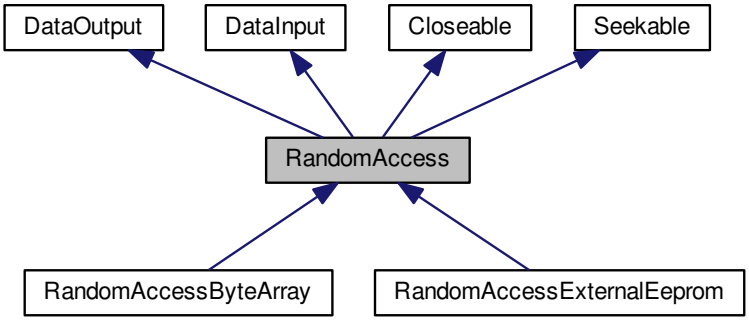
The documentation for this class was generated from the following files:

- OutputStream.h
- OutputStream.cpp

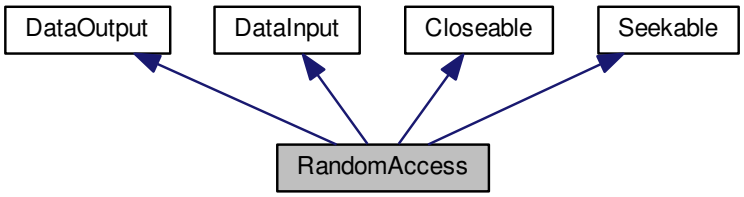
4.18 RandomAccess Class Reference

`#include <RandomAccess.h>`

Inheritance diagram for RandomAccess:



Collaboration diagram for RandomAccess:



Additional Inherited Members

4.18.1 Detailed Description

Raspberry IO.

RandomAccess

Interface derived from DataInput, DataOutput, Closeable and Seekable.

Definition at line 17 of file RandomAccess.h.

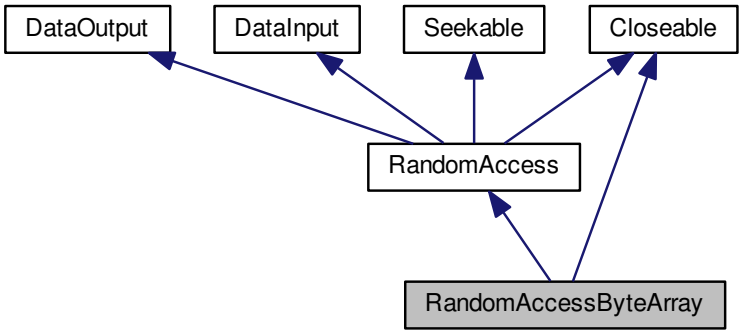
The documentation for this class was generated from the following file:

- RandomAccess.h

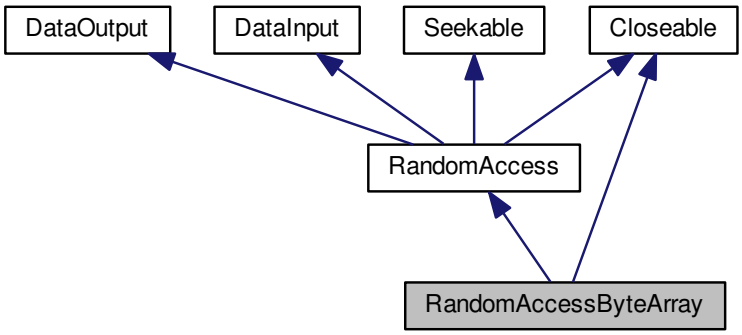
4.19 RandomAccessByteArray Class Reference

```
#include <RandomAccessByteArray.h>
```

Inheritance diagram for RandomAccessByteArray:



Collaboration diagram for RandomAccessByteArray:



Public Member Functions

- RandomAccessByteArray (unsigned char ∗buf, unsigned int count)
- virtual void seek (unsigned int pos)
- unsigned int length ()
- virtual void close ()
- virtual void write (unsigned char ∗b, int len)
- virtual void write (unsigned char b)
- virtual void writeByte (unsigned char b)
- virtual void writeBytes (unsigned char ∗b, int len)
- virtual void writeBoolean (bool v)
- virtual void writeChar (char c)
- virtual void writeUnsignedChar (unsigned char c)
- virtual void writeInt (int v)
- virtual void writeUnsignedInt (unsigned int v)
- virtual void writeLong (long v)

- virtual void writeUnsignedLong (unsigned long v)
- virtual void writeFloat (float v)
- virtual void writeDouble (double v)
- virtual unsigned char readByte ()
- virtual bool readBoolean ()
- virtual char readChar ()
- virtual unsigned char readUnsignedChar ()
- virtual int readInt ()
- virtual unsigned int readUnsignedInt ()
- virtual long readLong ()
- virtual unsigned long readUnsignedLong ()
- virtual float readFloat ()
- virtual double readDouble ()
- virtual void readFully (unsigned char *b, int len)
- virtual unsigned int skipBytes (unsigned int n)

Private Attributes

- unsigned char * buf
- unsigned int count
- unsigned int pos

4.19.1 Detailed Description

Raspberry IO.

RandomAccessByteArray

Instances of this class support both reading and writing to a random access unsigned char array.

Definition at line 16 of file RandomAccessByteArray.h.

4.19.2 Constructor & Destructor Documentation

4.19.2.1 RandomAccessByteArray::RandomAccessByteArray (unsigned char * *buf*, unsigned int *count*)

Public constructor.

Parameters

buf	The unsigned char array.
count	The size of such unsigned char array.

Definition at line 15 of file RandomAccessByteArray.cpp.

4.19.3 Member Function Documentation

4.19.3.1 void RandomAccessByteArray::close ( ) [virtual]

Closing a unsigned char array has no effect.

Implements Closeable.

Definition at line 29 of file RandomAccessByteArray.cpp.

4.19.3.2 unsigned int RandomAccessByteArray::length ( )

Returns the length of the stream.

Returns

The length.

Definition at line 21 of file RandomAccessByteArray.cpp.

4.19.3.3 bool RandomAccessByteArray::readBoolean ( ) [virtual]

Reads a bool from the stream.

Returns

bool

Implements DataInput.

Definition at line 94 of file RandomAccessByteArray.cpp.

4.19.3.4 unsigned char RandomAccessByteArray::readByte ( ) [virtual]

Reads a unsigned char from the stream.

Returns

unsigned char

Implements DataInput.

Definition at line 90 of file RandomAccessByteArray.cpp.

4.19.3.5 char RandomAccessByteArray::readChar ( ) [virtual]

Reads a char from the stream.

Returns

char

Implements DataInput.

Definition at line 98 of file RandomAccessByteArray.cpp.

4.19.3.6 double RandomAccessByteArray::readDouble ( ) [virtual]

Reads a double from the stream.

Returns

double

Implements DataInput.

Definition at line 138 of file RandomAccessByteArray.cpp.

4.19.3.7 float RandomAccessByteArray::readFloat ( ) [virtual]

Reads a float from the stream.

Returns

float

Implements DataInput.

Definition at line 134 of file RandomAccessByteArray.cpp.

4.19.3.8 void RandomAccessByteArray::readFully (unsigned char * *b*, int *len*) [virtual]

Reads a array of bytes from the stream.

Parameters

b	
len	

Implements DataInput.

Definition at line 142 of file RandomAccessByteArray.cpp.

4.19.3.9 int RandomAccessByteArray::readInt ( ) [virtual]

Reads an int from the stream.

Returns

int

Implements DataInput.

Definition at line 106 of file RandomAccessByteArray.cpp.

4.19.3.10 long RandomAccessByteArray::readLong ( ) [virtual]

Reads a long from the stream.

Returns

long

Implements DataInput.

Definition at line 118 of file RandomAccessByteArray.cpp.

4.19.3.11 unsigned char RandomAccessByteArray::readUnsignedChar ( ) [virtual]

Reads an unsigned char from the stream.

Returns

unsigned char

Implements DataInput.

Definition at line 102 of file RandomAccessByteArray.cpp.

4.19.3.12 unsigned int RandomAccessByteArray::readUnsignedInt ( ) [virtual]

Reads an unsigned int from the stream.

Returns

unsigned int

Implements DataInput.

Definition at line 114 of file RandomAccessByteArray.cpp.

4.19.3.13 unsigned long RandomAccessByteArray::readUnsignedLong ( ) [virtual]

Reads a unsigned long from the stream.

Returns

unsigned long

Implements DataInput.

Definition at line 130 of file RandomAccessByteArray.cpp.

4.19.3.14 void RandomAccessByteArray::seek (unsigned int *pos*) [virtual]

Seeks the stream at the position.

Parameters

pos	The position.

Implements Seekable.

Definition at line 25 of file RandomAccessByteArray.cpp.

4.19.3.15 unsigned int RandomAccessByteArray::skipBytes (unsigned int *n*) [virtual]

Skips n bytes of the stream.

Parameters

n	

Returns

unsigned int The number of skipped bytes.

Implements DataInput.

Definition at line 148 of file RandomAccessByteArray.cpp.

4.19.3.16 void RandomAccessByteArray::write (unsigned char * *b*, int *len*) [virtual]

Writes an array of bytes into the stream.

Parameters

b	The array of bytes.
len	The length of such array.

Implements DataOutput.

Definition at line 32 of file RandomAccessByteArray.cpp.

4.19.3.17 void RandomAccessByteArray::write (unsigned char *b*) [virtual]

Writes a unsigned char into the stream.

Parameters

b	The unsigned char to be written.

Implements DataOutput.

Definition at line 36 of file RandomAccessByteArray.cpp.

4.19.3.18 void RandomAccessByteArray::writeBoolean (bool *v*) [virtual]

Writes a bool into the stream.

Parameters

v	The bool to be written.

Implements DataOutput.

Definition at line 50 of file RandomAccessByteArray.cpp.

4.19.3.19 void RandomAccessByteArray::writeByte (unsigned char *b*) [virtual]

Writes a unsigned char into the stream.

Parameters

b	The unsigned char to be written.

Implements DataOutput.

Definition at line 40 of file RandomAccessByteArray.cpp.

4.19.3.20 void RandomAccessByteArray::writeBytes (unsigned char * *b*, int *len*) [virtual]

Writes an array of bytes into the stream.

Parameters

b	The array of bytes.
len	The length of such array.

Implements DataOutput.

Definition at line 44 of file RandomAccessByteArray.cpp.

4.19.3.21 void RandomAccessByteArray::writeChar (char *c*) [virtual]

Writes a char into the stream.

Parameters

c	The char to be written.

Implements DataOutput.

Definition at line 54 of file RandomAccessByteArray.cpp.

4.19.3.22 void RandomAccessByteArray::writeDouble (double *v*) [virtual]

Writes a double into the stream.

Parameters

v	The double to be written.

Implements DataOutput.

Definition at line 86 of file RandomAccessByteArray.cpp.

4.19.3.23 void RandomAccessByteArray::writeFloat (float *v*) [virtual]

Writes a float into the stream.

Parameters

v	The float to be written.

Implements DataOutput.

Definition at line 82 of file RandomAccessByteArray.cpp.

4.19.3.24 void RandomAccessByteArray::writeInt (int *v*) [virtual]

Writes an int into the stream.

Parameters

v	The int to be written.

Implements DataOutput.

Definition at line 62 of file RandomAccessByteArray.cpp.

4.19.3.25 void RandomAccessByteArray::writeLong (long *v*) [virtual]

Writes a long into the stream.

Parameters

v	The long to be written.

Implements DataOutput.

Definition at line 71 of file RandomAccessByteArray.cpp.

4.19.3.26 void RandomAccessByteArray::writeUnsignedChar (unsigned char *c*) [virtual]

Writes an unsigned char into the stream.

Parameters

c	The unsigned char to be written.

Implements DataOutput.

Definition at line 58 of file RandomAccessByteArray.cpp.

4.19.3.27 void RandomAccessByteArray::writeUnsignedInt (unsigned int *v*) [virtual]

Writes an unsigned int into the stream.

Parameters

v	The unsigned int to be written.

Implements DataOutput.

Definition at line 67 of file RandomAccessByteArray.cpp.

4.19.3.28 void RandomAccessByteArray::writeUnsignedLong (unsigned long *v*) [virtual]

Writes a unsigned long into the stream.

Parameters

v	The unsigned long to be written.

Implements DataOutput.

Definition at line 78 of file RandomAccessByteArray.cpp.

4.19.4 Member Data Documentation

4.19.4.1 unsigned char* RandomAccessByteArray::buf [private]

Buffer used to work.

Definition at line 21 of file RandomAccessByteArray.h.

4.19.4.2 unsigned int RandomAccessByteArray::count [private]

Buffer size.

Definition at line 26 of file RandomAccessByteArray.h.

4.19.4.3 unsigned int RandomAccessByteArray::pos [private]

Current position.

Definition at line 31 of file RandomAccessByteArray.h.

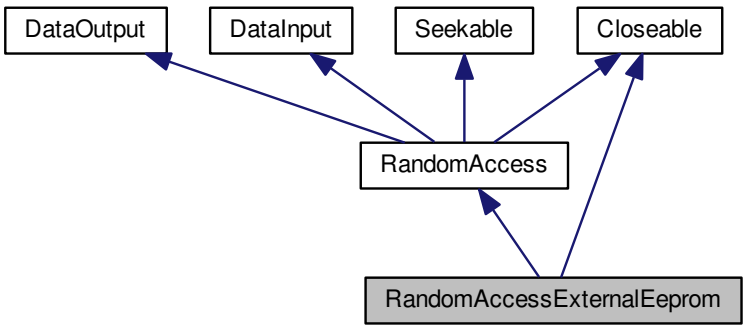
The documentation for this class was generated from the following files:

- RandomAccessByteArray.h
- RandomAccessByteArray.cpp

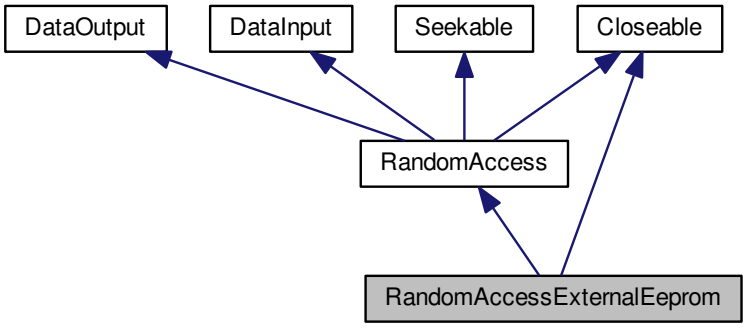
4.20 RandomAccessExternalEeprom Class Reference

`#include <RandomAccessExternalEeprom.h>`

Inheritance diagram for RandomAccessExternalEeprom:



Collaboration diagram for RandomAccessExternalEeprom:



Public Member Functions

- RandomAccessExternalEeprom (ExternalEeprom *externalEeprom)
- virtual void seek (unsigned int pos)
- unsigned int length ()
- virtual void close ()
- virtual void write (unsigned char *b, int len)
- virtual void write (unsigned char b)
- virtual void writeByte (unsigned char b)
- virtual void writeBytes (unsigned char *b, int len)
- virtual void writeBoolean (bool v)
- virtual void writeChar (char c)
- virtual void writeUnsignedChar (unsigned char c)
- virtual void writeInt (int v)
- virtual void writeUnsignedInt (unsigned int v)
- virtual void writeLong (long v)
- virtual void writeUnsignedLong (unsigned long v)
- virtual void writeFloat (float v)
- virtual void writeDouble (double v)
- virtual unsigned char readByte ()
- virtual bool readBoolean ()
- virtual char readChar ()
- virtual unsigned char readUnsignedChar ()
- virtual int readInt ()
- virtual unsigned int readUnsignedInt ()
- virtual long readLong ()
- virtual unsigned long readUnsignedLong ()
- virtual float readFloat ()
- virtual double readDouble ()
- virtual void readFully (unsigned char *b, int len)
- virtual unsigned int skipBytes (unsigned int n)

Private Attributes

- ExternalEeprom * externalEeprom
- unsigned int pos

4.20.1 Detailed Description

Raspberry IO.

RandomAccessExternalEeprom

Instances of this class support both reading and writing to a random access externalEeprom. A random access externalEeprom behaves like a large array of bytes stored in the externalEeprom system.

Definition at line 18 of file RandomAccessExternalEeprom.h.

4.20.2 Constructor & Destructor Documentation

4.20.2.1 RandomAccessExternalEeprom::RandomAccessExternalEeprom (ExternalEeprom * *externalEeprom*)

Public constructor.

Parameters

externalEeprom	The external eeprom instance to be used.

Definition at line 17 of file RandomAccessExternalEeprom.cpp.

4.20.3 Member Function Documentation

4.20.3.1 void RandomAccessExternalEeprom::close ( ) [virtual]

Closing a external eeprom has no effect.

Implements Closeable.

Definition at line 31 of file RandomAccessExternalEeprom.cpp.

4.20.3.2 unsigned int RandomAccessExternalEeprom::length ( )

Returns the length of the stream.

Returns

The length.

Definition at line 23 of file RandomAccessExternalEeprom.cpp.

4.20.3.3 bool RandomAccessExternalEeprom::readBoolean ( ) [virtual]

Reads a bool from the stream.

Returns

bool

Implements DataInput.

Definition at line 96 of file RandomAccessExternalEeprom.cpp.

4.20.3.4 unsigned char RandomAccessExternalEeprom::readByte ( ) [virtual]

Reads a unsigned char from the stream.

Returns

unsigned char

Implements DataInput.

Definition at line 92 of file RandomAccessExternalEeprom.cpp.

4.20.3.5 char RandomAccessExternalEeprom::readChar ( ) [virtual]

Reads a char from the stream.

Returns

char

Implements DataInput.

Definition at line 100 of file RandomAccessExternalEeprom.cpp.

4.20.3.6 double RandomAccessExternalEeprom::readDouble ( ) [virtual]

Reads a double from the stream.

Returns

double

Implements DataInput.

Definition at line 140 of file RandomAccessExternalEeprom.cpp.

4.20.3.7 float RandomAccessExternalEeprom::readFloat ( ) [virtual]

Reads a float from the stream.

Returns

float

Implements DataInput.

Definition at line 136 of file RandomAccessExternalEeprom.cpp.

4.20.3.8 void RandomAccessExternalEeprom::readFully (unsigned char * *b*, int *len*) [virtual]

Reads a array of bytes from the stream.

Parameters

b	
len	

Implements DataInput.

Definition at line 144 of file RandomAccessExternalEeprom.cpp.

4.20.3.9 int RandomAccessExternalEeprom::readInt ( ) [virtual]

Reads an int from the stream.

Returns

int

Implements DataInput.

Definition at line 108 of file RandomAccessExternalEeprom.cpp.

4.20.3.10 long RandomAccessExternalEeprom::readLong ( ) [virtual]

Reads a long from the stream.

Returns

long

Implements DataInput.

Definition at line 120 of file RandomAccessExternalEeprom.cpp.

4.20.3.11 unsigned char RandomAccessExternalEeprom::readUnsignedChar ( ) [virtual]

Reads an unsigned char from the stream.

Returns

unsigned char

Implements DataInput.

Definition at line 104 of file RandomAccessExternalEeprom.cpp.

4.20.3.12 unsigned int RandomAccessExternalEeprom::readUnsignedInt ( ) [virtual]

Reads an unsigned int from the stream.

Returns

unsigned int

Implements DataInput.

Definition at line 116 of file RandomAccessExternalEeprom.cpp.

4.20.3.13 unsigned long RandomAccessExternalEeprom::readUnsignedLong ( ) [virtual]

Reads a unsigned long from the stream.

Returns

unsigned long

Implements DataInput.

Definition at line 132 of file RandomAccessExternalEeprom.cpp.

4.20.3.14 void RandomAccessExternalEeprom::seek (unsigned int *pos*) [virtual]

Seeks the stream at the position.

Parameters

pos	The position.

Implements Seekable.

Definition at line 27 of file RandomAccessExternalEeprom.cpp.

4.20.3.15 unsigned int RandomAccessExternalEeprom::skipBytes (unsigned int *n*) [virtual]

Skips n bytes of the stream.

Parameters

n	

Returns

unsigned int The number of skipped bytes.

Implements DataInput.

Definition at line 150 of file RandomAccessExternalEeprom.cpp.

4.20.3.16 void RandomAccessExternalEeprom::write (unsigned char * *b,* int *len*) [virtual]

Writes an array of bytes into the stream.

Parameters

b	The array of bytes.
len	The length of such array.

Implements DataOutput.

Definition at line 34 of file RandomAccessExternalEeprom.cpp.

4.20.3.17 void RandomAccessExternalEeprom::write (unsigned char *b*) [virtual]

Writes a unsigned char into the stream.

Parameters

b	The unsigned char to be written.

Implements DataOutput.

Definition at line 38 of file RandomAccessExternalEeprom.cpp.

4.20.3.18 void RandomAccessExternalEeprom::writeBoolean (bool *v*) [virtual]

Writes a bool into the stream.

Parameters

v	The bool to be written.

Implements DataOutput.

Definition at line 52 of file RandomAccessExternalEeprom.cpp.

4.20.3.19 void RandomAccessExternalEeprom::writeByte (unsigned char *b*) [virtual]

Writes a unsigned char into the stream.

Parameters

b	The unsigned char to be written.

Implements DataOutput.

Definition at line 42 of file RandomAccessExternalEeprom.cpp.

4.20.3.20 void RandomAccessExternalEeprom::writeBytes (unsigned char * *b*, int *len*) [virtual]

Writes an array of bytes into the stream.

Parameters

b	The array of bytes.
len	The length of such array.

Implements DataOutput.

Definition at line 46 of file RandomAccessExternalEeprom.cpp.

4.20.3.21 void RandomAccessExternalEeprom::writeChar (char *c*) [virtual]

Writes a char into the stream.

Parameters

c	The char to be written.

Implements DataOutput.

Definition at line 56 of file RandomAccessExternalEeprom.cpp.

4.20.3.22 void RandomAccessExternalEeprom::writeDouble (double *v*) [virtual]

Writes a double into the stream.

Parameters

v	The double to be written.

Implements DataOutput.

Definition at line 88 of file RandomAccessExternalEeprom.cpp.

4.20.3.23 void RandomAccessExternalEeprom::writeFloat (float *v*) [virtual]

Writes a float into the stream.

Parameters

v	The float to be written.

Implements DataOutput.

Definition at line 84 of file RandomAccessExternalEeprom.cpp.

4.20.3.24 void RandomAccessExternalEeprom::writeInt (int *v*) [virtual]

Writes an int into the stream.

Parameters

v	The int to be written.

Implements DataOutput.

Definition at line 64 of file RandomAccessExternalEeprom.cpp.

4.20.3.25 void RandomAccessExternalEeprom::writeLong (long *v*) [virtual]

Writes a long into the stream.

Parameters

v	The long to be written.

Implements DataOutput.

Definition at line 73 of file RandomAccessExternalEeprom.cpp.

4.20.3.26 void RandomAccessExternalEeprom::writeUnsignedChar (unsigned char *c*) [virtual]

Writes an unsigned char into the stream.

Parameters

c	The unsigned char to be written.

Implements DataOutput.

Definition at line 60 of file RandomAccessExternalEeprom.cpp.

4.20.3.27 void RandomAccessExternalEeprom::writeUnsignedInt (unsigned int *v*) [virtual]

Writes an unsigned int into the stream.

Parameters

v	The unsigned int to be written.

Implements DataOutput.

Definition at line 69 of file RandomAccessExternalEeprom.cpp.

4.20.3.28 void RandomAccessExternalEeprom::writeUnsignedLong (unsigned long *v*) [virtual]

Writes a unsigned long into the stream.

Parameters

v	The unsigned long to be written.

Implements DataOutput.

Definition at line 80 of file RandomAccessExternalEeprom.cpp.

4.20.4 Member Data Documentation

4.20.4.1 ExternalEeprom* RandomAccessExternalEeprom::externalEeprom [private]

The external eeprom to be used.

Definition at line 23 of file RandomAccessExternalEeprom.h.

4.20.4.2 unsigned int RandomAccessExternalEeprom::pos [private]

Current position.

Definition at line 28 of file RandomAccessExternalEeprom.h.

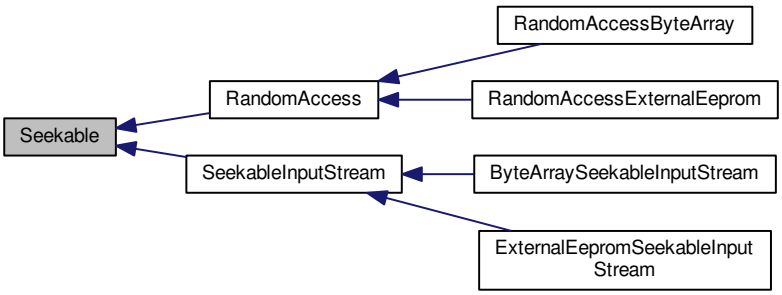
The documentation for this class was generated from the following files:

- RandomAccessExternalEeprom.h
- RandomAccessExternalEeprom.cpp

4.21 Seekable Class Reference

`#include <Seekable.h>`

Inheritance diagram for Seekable:



Public Member Functions

- virtual ~Seekable ()
- virtual void seek (unsigned int pos)=0

4.21.1 Detailed Description

Raspberry IO.

Seekable

Definition at line 10 of file Seekable.h.

4.21.2 Constructor & Destructor Documentation

4.21.2.1 virtual Seekable::~Seekable ( ) [inline], [virtual]

Definition at line 13 of file Seekable.h.

4.21.3 Member Function Documentation

4.21.3.1 virtual void Seekable::seek (unsigned int *pos*) [pure virtual]

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, ExternalEepromSeekableInputStream, and ByteArraySeekableInputStream.

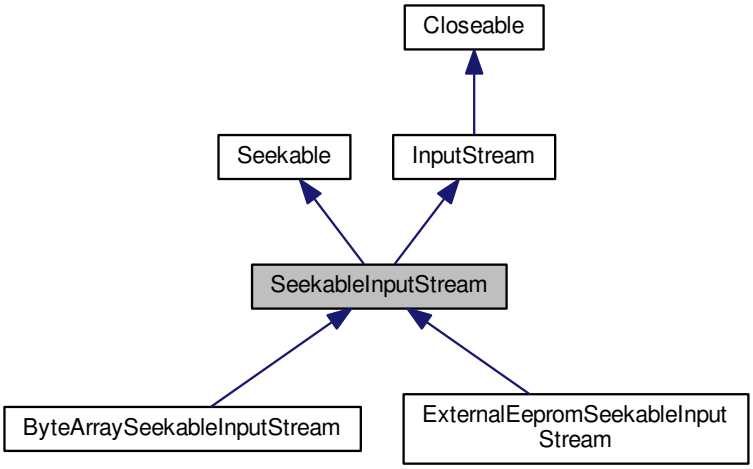
The documentation for this class was generated from the following file:

- Seekable.h

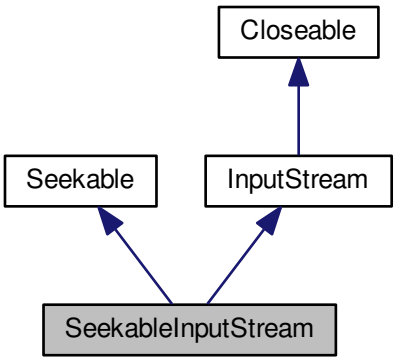
4.22 SeekableInputStream Class Reference

`#include <SeekableInputStream.h>`

Inheritance diagram for SeekableInputStream:



Collaboration diagram for SeekableInputStream:



Additional Inherited Members

4.22.1 Detailed Description

Raspberry IO.

SeekableInputStream

Definition at line 13 of file SeekableInputStream.h.

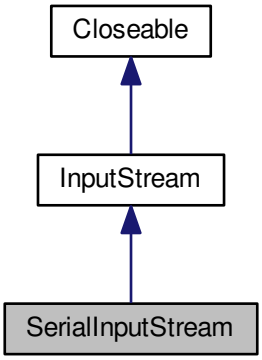
The documentation for this class was generated from the following file:

- SeekableInputStream.h

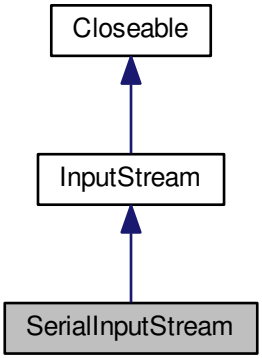
4.23 SerialInputStream Class Reference

```
#include <SerialInputStream.h>
```

Inheritance diagram for SerialInputStream:



Collaboration diagram for SerialInputStream:



Public Types

- enum BoudRate {
  BR_50 = B50, BR_75 = B75, BR_110 = B110, BR_134 = B134,
  BR_150 = B150, BR_200 = B200, BR_300 = B300, BR_600 = B600,
  BR_1200 = B1200, BR_1800 = B1800, BR_2400 = B2400, BR_4800 = B4800,
  BR_9600 = B9600, BR_19200 = B19200, BR_38400 = B38400, BR_57600 = B57600,
  BR_115200 = B115200, BR_230400 = B230400, BR_460800 = B460800, BR_500000 = B500000,
  BR_576000 = B576000, BR_921600 = B921600, BR_1000000 = B1000000, BR_1152000 = B1152000,
  BR_1500000 = B1500000, BR_2000000 = B2000000, BR_2500000 = B2500000, BR_3000000 = B3000000,
  BR_3500000 = B3500000, BR_4000000 = B4000000 }

Public Member Functions

- SerialInputStream (const char *dev, BoudRate boundRate)
- virtual int available ()
- virtual int read ()
- virtual int read (unsigned char *b, int len)

Private Attributes

- int fd
- int tmp

4.23.1 Detailed Description

Raspberry IO.

SerialInputStream

A SerialInputStream obtains input bytes from a serial port.

Definition at line 22 of file SerialInputStream.h.

4.23.2 Member Enumeration Documentation

4.23.2.1 enum SerialInputStream::BoudRate

Enumerator

BR_50

BR_75

BR_110

BR_134

BR_150

BR_200

BR_300

BR_600

BR_1200

BR_1800

BR_2400

BR_4800

BR_9600

BR_19200

BR_38400

BR_57600

BR_115200

BR_230400

BR_460800

BR_500000

BR_576000

BR_921600

BR_1000000

BR_1152000

BR_1500000

BR_2000000

BR_2500000

BR_3000000

BR_3500000

BR_4000000

Definition at line 37 of file SerialInputStream.h.

4.23.3 Constructor & Destructor Documentation

4.23.3.1 SerialInputStream::SerialInputStream (const char * *dev*, BoudRate *boundRate*)

Public constructor.

Parameters

boundRate	

Definition at line 14 of file SerialInputStream.cpp.

4.23.4 Member Function Documentation

4.23.4.1 int SerialInputStream::available ( ) [virtual]

Returns the number of bytes that can be read(or skipped over) from this input stream without blocking by the next caller of a method for this input stream.

Reimplemented from InputStream.

Definition at line 36 of file SerialInputStream.cpp.

4.23.4.2 int SerialInputStream::read ( ) [virtual]

Reads the next unsigned char of data from the input stream.

Implements InputStream.

Definition at line 46 of file SerialInputStream.cpp.

4.23.4.3 int SerialInputStream::read (unsigned char * *b*, int *len*) [virtual]

Reads some number of bytes from the input stream and stores them into the buffer array b.

Reimplemented from InputStream.

Definition at line 60 of file SerialInputStream.cpp.

4.23.5 Member Data Documentation

4.23.5.1 int SerialInputStream::fd [private]

File description.

Definition at line 28 of file SerialInputStream.h.

4.23.5.2 int SerialInputStream::tmp [private]

Internal control.

Definition at line 33 of file SerialInputStream.h.

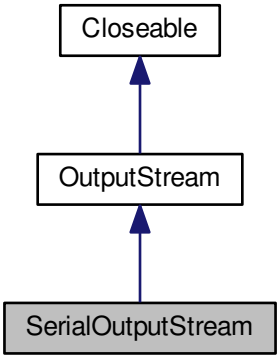
The documentation for this class was generated from the following files:

- SerialInputStream.h
- SerialInputStream.cpp

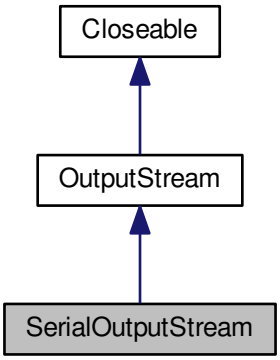
4.24 SerialOutputStream Class Reference

`#include <SerialOutputStream.h>`

Inheritance diagram for SerialOutputStream:



Collaboration diagram for SerialOutputStream:



Private Member Functions

- SerialOutputStream (unsigned int boudRate)
- virtual void write (unsigned char b)

Additional Inherited Members

4.24.1 Detailed Description

Raspberry IO.

SerialOutputStream

A serial output stream is a output stream to write in a serial port.

Definition at line 14 of file SerialOutputStream.h.

4.24.2 Constructor & Destructor Documentation

4.24.2.1 SerialOutputStream::SerialOutputStream (unsigned int *boudRate*) [private]

Public constructor.

Parameters

boundRate	

Definition at line 14 of file SerialOutputStream.cpp.

4.24.3 Member Function Documentation

4.24.3.1 void SerialOutputStream::write (unsigned char *b*) [private], [virtual]

Writes the specified unsigned char to this output stream.

Implements OutputStream.

Definition at line 17 of file SerialOutputStream.cpp.

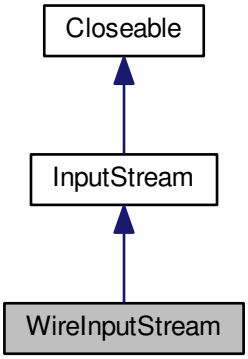
The documentation for this class was generated from the following files:

- SerialOutputStream.h
- SerialOutputStream.cpp

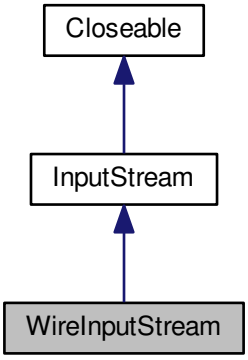
4.25 WireInputStream Class Reference

```
#include <WireInputStream.h>
```

Inheritance diagram for WireInputStream:



Collaboration diagram for WireInputStream:



Public Member Functions

- WireInputStream (unsigned char addredd)
- virtual int available ()
- virtual int read ()
- virtual int read (unsigned char ∗b, int off, int len)

Protected Attributes

- unsigned char address

4.25.1 Detailed Description

Raspberry IO.

WireInputStream

A WireInputStream obtains input bytes from the wire bus.

Definition at line 15 of file WireInputStream.h.

4.25.2 Constructor & Destructor Documentation

4.25.2.1 WireInputStream::WireInputStream (unsigned char *addredd*)

Public constructor.

Parameters

address	

Definition at line 14 of file WireInputStream.cpp.

4.25.3 Member Function Documentation

4.25.3.1 int WireInputStream::available ( ) [virtual]

Returns the number of bytes that can be read(or skipped over) from this input stream without blocking by the next caller of a method for this input stream.

Reimplemented from InputStream.

Definition at line 19 of file WireInputStream.cpp.

4.25.3.2 int WireInputStream::read ( ) [virtual]

Reads the next unsigned char of data from the input stream.

Implements InputStream.

Definition at line 23 of file WireInputStream.cpp.

4.25.3.3 int WireInputStream::read (unsigned char * *b*, int *off*, int *len*) [virtual]

Writes len of bytes into the stream.

Parameters

b	
off	
len	

Returns

Reimplemented from InputStream.

Definition at line 33 of file WireInputStream.cpp.

4.25.4 Member Data Documentation

4.25.4.1 unsigned char WireInputStream::address [protected]

The wire device address.

Definition at line 21 of file WireInputStream.h.

The documentation for this class was generated from the following files:

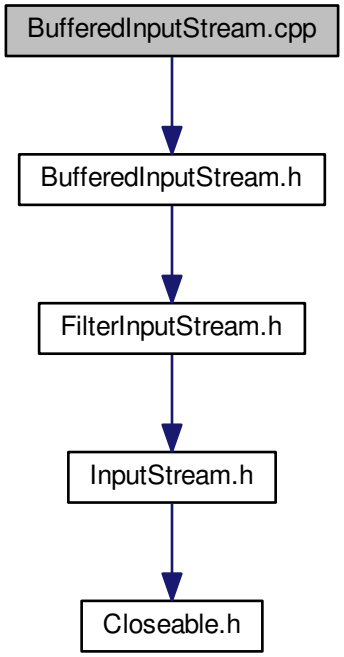
- WireInputStream.h
- WireInputStream.cpp

5 File Documentation

5.1 BufferedInputStream.cpp File Reference

```
#include "BufferedInputStream.h"
```

Include dependency graph for BufferedInputStream.cpp:



Macros

- #define __RASPBERRY_IO_BUFFERED_INPUT_STREAM_CPP__ 1

5.1.1 Macro Definition Documentation

5.1.1.1 #define __RASPBERRY_IO_BUFFERED_INPUT_STREAM_CPP__ 1

Raspberry IO.

BufferedInputStream

A `BufferedInputStream` adds functionality to another input stream-namely, the ability to buffer the input and to support the `mark` and `reset` methods. When the `BufferedInputStream` is created, an internal buffer

array is passed. As bytes from the stream are read or skipped, the internal buffer is refilled as necessary from the contained input stream, many bytes at a time. The `mark` operation remembers a point in the input stream and the `reset` operation causes all the bytes read since the most recent `mark` operation to be reread before new bytes are taken from the contained input stream.

Definition at line 25 of file BufferedInputStream.cpp.

5.2 BufferedInputStream.cpp

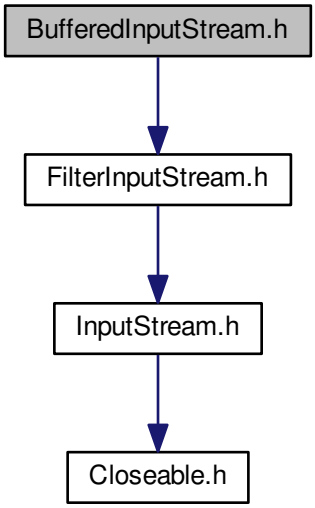
```
00001
00024 #ifndef __RASPBERRY_IO_BUFFERED_INPUT_STREAM_CPP__
00025 #define __RASPBERRY_IO_BUFFERED_INPUT_STREAM_CPP__ 1
00026
00027 #include "BufferedInputStream.h"
00028
00029 BufferedInputStream::BufferedInputStream(
      InputStream* in, unsigned char* buf,
00030         int size) :
00031         FilterInputStream(in), buf(buf) {
00032     this->size = size;
00033     count = 0;
00034     pos = 0;
00035     marked = false;
00036     markpos = 0;
00037 }
00038
00039 int BufferedInputStream::available() {
00040     return in->available() + (count - pos);
00041 }
00042
00043 void BufferedInputStream::close() {
00044     in->close();
00045 }
00046
00047 void BufferedInputStream::reset() {
00048     if (marked) {
00049         pos = markpos;
00050     }
00051 }
00052
00053 int BufferedInputStream::read(unsigned char* b, int len) {
00054     return read(b, 0, len);
00055 }
00056
00057 int BufferedInputStream::read(unsigned char* b, int off, int len) {
00058     int cnt, available;
00059     available = count - pos;
00060
00061     /*
00062      * The needed data are already in the buffer?
00063      */
00064     if (available >= len) {
00065         for (int i = 0; i < len; i++) {
00066             b[off + i] = buf[pos + i];
00067         }
00068         pos += len;
00069         return len;
00070     }
00071
00072     /*
00073      * The buffer data is not enough, but is necessary.
00074      */
00075     for (int i = 0; i < available; i++) {
00076         b[off + i] = buf[pos + i];
00077     }
00078     marked = false;
00079     pos = 0;
00080     count = 0;
00081
00082     /*
00083      * Reads the rest from the stream.
00084      */
00085     cnt = in->read(b, off + available, len - available);
00086
00087     /*
00088      * Tests if we had enough data.
00089      */
00090     if (cnt < 0) {
00091         return available;
00092     } else if (cnt < (len - available)) {
00093         return available + cnt;
00094     } else {
00095         fill(0);
```

```
00096     }
00097     return len;
00098 }
00099
00100 int BufferedInputStream::read() {
00101
00102     /*
00103      * Tests if the buffer is completely used.
00104      */
00105     if (pos >= count) {
00106         marked = false;
00107         fill(0);
00108         if (count == 0) {
00109             return -1;
00110         }
00111         pos = 0;
00112     }
00113     return (int) buf[pos++];
00114 }
00115
00116 void BufferedInputStream::realineBufferContent() {
00117     int n;
00118     if (pos > 0) {
00119         n = count - pos;
00120         for (int i = 0; i < n; i++) {
00121             buf[i] = buf[pos + i];
00122         }
00123         count -= pos;
00124         pos = 0;
00125     }
00126 }
00127
00128 void BufferedInputStream::fill(int startPos) {
00129     int n, needed;
00130     needed = size - startPos;
00131     if (needed <= 0) {
00132         return;
00133     }
00134     n = in->read(buf, startPos, needed);
00135     if (n > 0) {
00136         count = startPos + n;
00137     }
00138 }
00139
00140 void BufferedInputStream::mark() {
00141     realineBufferContent();
00142     fill(count);
00143     markpos = 0;
00144     marked = true;
00145 }
00146
00147 bool BufferedInputStream::markSupported() {
00148     return true;
00149 }
00150
00151 unsigned int BufferedInputStream::skip(unsigned int n) {
00152     unsigned int buffered, skiped;
00153     buffered = count - pos;
00154     if (buffered >= n) {
00155         pos += n;
00156         return n;
00157     }
00158     pos = 0;
00159     count = 0;
00160     marked = false;
00161     skiped = buffered + in->skip(n - buffered);
00162     return skiped;
00163 }
00164
00165 #endif /* __RASPBERRY_IO_BUFFERED_INPUT_STREAM_CPP__ */
```

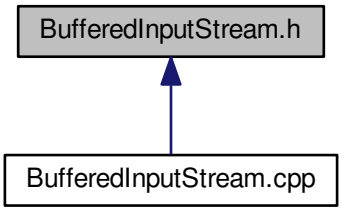
5.3 BufferedInputStream.h File Reference

```
#include <FilterInputStream.h>
```

Include dependency graph for BufferedInputStream.h:



This graph shows which files directly or indirectly include this file:



Classes

- class BufferedInputStream

5.4 BufferedInputStream.h

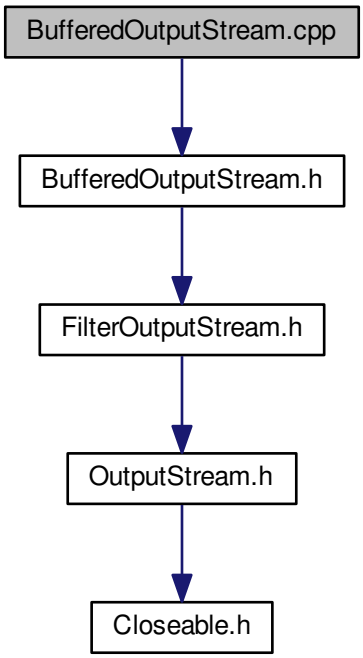
```
00001
00024 #ifndef __RASPBERRY_IO_BUFFERED_INPUT_STREAM_H__
00025 #define __RASPBERRY_IO_BUFFERED_INPUT_STREAM_H__ 1
00026
00027 #include <FilterInputStream.h>
00028
00029 class BufferedInputStream: public FilterInputStream {
00030
00034     unsigned int size;
00035
```

```
00036 protected:
00037
00041     unsigned char* buf;
00042
00052     int count;
00053
00067     int pos;
00068
00093     int markpos;
00094
00098     bool marked;
00099
00100 public:
00101
00109     BufferedInputStream(InputStream* in, unsigned char* buf, int size);
00110
00116     virtual int available();
00117
00122     virtual void close();
00123
00127     virtual void mark();
00128
00132     virtual bool markSupported();
00133
00137     virtual int read();
00138
00147     virtual int read(unsigned char* b, int len);
00148
00153     virtual int read(unsigned char* b, int off, int len);
00154
00159     virtual void reset();
00160
00164     virtual unsigned int skip(unsigned int n);
00165
00166 private:
00167
00171     void realineBufferContent();
00172
00178     void fill(int startPos);
00179 };
00180
00181 #endif /* __RASPBERRY_IO_BUFFERED_INPUT_STREAM_H__ */
```

5.5 BufferedOutputStream.cpp File Reference

```
#include "BufferedOutputStream.h"
```

Include dependency graph for BufferedOutputStream.cpp:



Macros

- #define __RASPBERRY_IO_BUFFERED_OUTPUT_STREAM_CPP__ 1

5.5.1 Macro Definition Documentation

5.5.1.1 #define __RASPBERRY_IO_BUFFERED_OUTPUT_STREAM_CPP__ 1

Raspberry IO.

BufferedOutputStream

The class implements a buffered output stream. By setting up such an output stream, an application can write bytes to the underlying output stream without necessarily causing a call to the underlying system for each unsigned char written.

Definition at line 13 of file BufferedOutputStream.cpp.

5.6 BufferedOutputStream.cpp

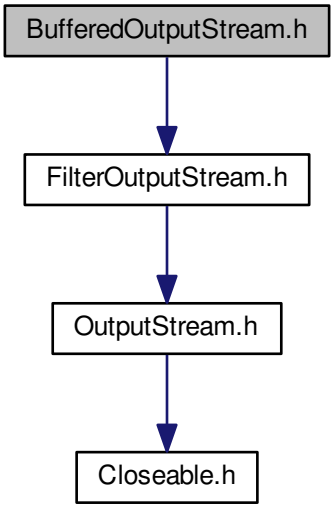
```
00001
00012 #ifndef __RASPBERRY_IO_BUFFERED_OUTPUT_STREAM_CPP__
00013 #define __RASPBERRY_IO_BUFFERED_OUTPUT_STREAM_CPP__ 1
00014
00015 #include "BufferedOutputStream.h"
00016
00017 BufferedOutputStream::BufferedOutputStream(
      OutputStream* out,
00018         unsigned char* buf, int size) :
00019         FilterOutputStream(out), buf(buf) {
```

```
00020     this->size = size;
00021     count = 0;
00022 }
00023
00024 void BufferedOutputStream::write(unsigned char b) {
00025     if (count >= size) {
00026         flushBuffer();
00027     }
00028     buf[count++] = b;
00029 }
00030
00031 void BufferedOutputStream::write(unsigned char* b, int len) {
00032     write(b, 0, len);
00033 }
00034
00035 void BufferedOutputStream::write(unsigned char* b, int off, int len) {
00036     /*
00037      * If the request length exceeds the size of the output buffer,
00038      * flush the output buffer and then write the data directly.
00039      * In this way buffered streams will cascade harmlessly.
00040      */
00041     if (len >= size) {
00042         flushBuffer();
00043         out->write(b, off, len);
00044         return;
00045     }
00046     if (len > size - count) {
00047         flushBuffer();
00048     }
00049     for (int i = 0; i < len; i++) {
00050         buf[count + i] = b[off + i];
00051     }
00052     count += len;
00053 }
00054
00055 void BufferedOutputStream::flush() {
00056     flushBuffer();
00057     out->flush();
00058 }
00059
00060 void BufferedOutputStream::close() {
00061     flush();
00062     out->close();
00063 }
00064
00065 void BufferedOutputStream::flushBuffer() {
00066     if (count > 0) {
00067         out->write(buf, 0, count);
00068         count = 0;
00069     }
00070 }
00071
00072 #endif /* __RASPBERRY_IO_BUFFERED_OUTPUT_STREAM_CPP__ */
```

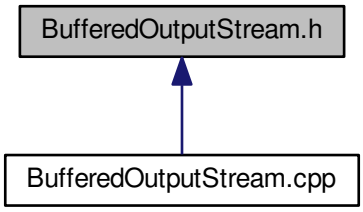
5.7 BufferedOutputStream.h File Reference

```
#include <FilterOutputStream.h>
```

Include dependency graph for BufferedOutputStream.h:



This graph shows which files directly or indirectly include this file:



Classes

- class BufferedOutputStream

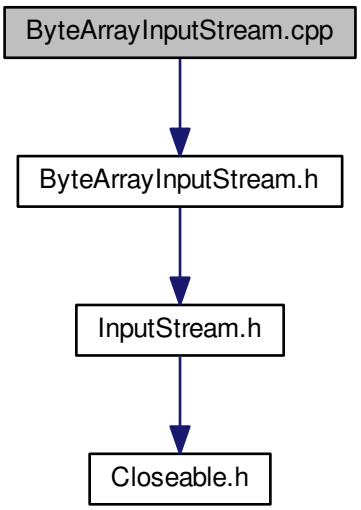
5.8 BufferedOutputStream.h

```
00001
00012 #ifndef __RASPBERRY_IO_BUFFERED_OUTPUT_STREAM_H__
00013 #define __RASPBERRY_IO_BUFFERED_OUTPUT_STREAM_H__ 1
00014
00015 #include <FilterOutputStream.h>
00016
00017 class BufferedOutputStream: public FilterOutputStream {
00018 protected:
00019
00023     unsigned char* buf;
```

```
00024
00028     int size;
00029
00036     int count;
00037
00038 public:
00039
00048     BufferedOutputStream(OutputStream* out, unsigned char* buf, int size
      );
00049
00056     void write(unsigned char b);
00057
00066     virtual void write(unsigned char* b, int len);
00067
00083     virtual void write(unsigned char* b, int off, int len);
00084
00089     virtual void flush();
00090
00091     virtual void close();
00092
00093 private:
00094
00098     void flushBuffer();
00099 };
00100
00101 #endif /* __RASPBERRY_IO_BUFFERED_OUTPUT_STREAM_H__ */
```

5.9 ByteArrayInputStream.cpp File Reference

`#include "ByteArrayInputStream.h"`
Include dependency graph for ByteArrayInputStream.cpp:



Macros

- #define __RASPBERRY_IO_BYTE_ARRAY_INPUT_STREAM_CPP__ 1

5.9.1 Macro Definition Documentation

5.9.1.1 #define __RASPBERRY_IO_BYTE_ARRAY_INPUT_STREAM_CPP__ 1

Raspberry IO.

ByteArrayInputStream

A ByteArrayInputStream contains an internal buffer that contains bytes that may be read from the stream.

Definition at line 11 of file ByteArrayInputStream.cpp.

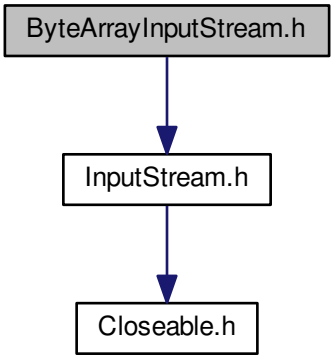
5.10 ByteArrayInputStream.cpp

```
00001
00010 #ifndef __RASPBERRY_IO_BYTE_ARRAY_INPUT_STREAM_CPP__
00011 #define __RASPBERRY_IO_BYTE_ARRAY_INPUT_STREAM_CPP__ 1
00012
00013 #include "ByteArrayInputStream.h"
00014
00015 ByteArrayInputStream::ByteArrayInputStream(unsigned char* buf,
00016         unsigned int count) :
00017         buf(buf), count(count) {
00018     markpos = 0;
00019     pos = 0;
00020 }
00021
00022 int ByteArrayInputStream::available() {
00023     if ((count - pos) > 0) {
00024         return 1;
00025     }
00026     return 0;
00027 }
00028
00029 void ByteArrayInputStream::mark() {
00030     markpos = pos;
00031 }
00032
00033 bool ByteArrayInputStream::markSupported() {
00034     return true;
00035 }
00036
00037 int ByteArrayInputStream::read() {
00038     return buf[pos++];
00039 }
00040
00041 void ByteArrayInputStream::reset() {
00042     pos = markpos;
00043 }
00044
00045 #endif /* __RASPBERRY_IO_BYTE_ARRAY_INPUT_STREAM_CPP__ */
```

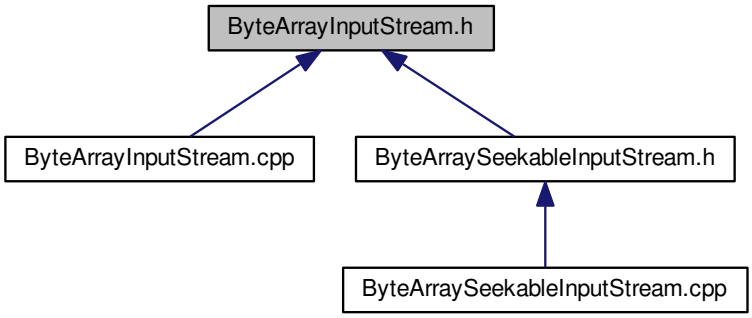
5.11 ByteArrayInputStream.h File Reference

```
#include <InputStream.h>
```

Include dependency graph for ByteArrayInputStream.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ByteArrayInputStream

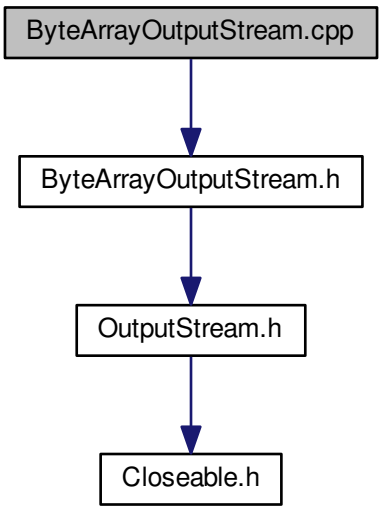
5.12 ByteArrayInputStream.h

```
00001
00010 #ifndef __RASPBERRY_IO_BYTE_ARRAY_INPUT_STREAM_H__
00011 #define __RASPBERRY_IO_BYTE_ARRAY_INPUT_STREAM_H__ 1
00012
00013 #include <InputStream.h>
00014
00015 class ByteArrayInputStream: public virtual InputStream {
00016 protected:
00017
00018     /*
00019      * The buffer where data is stored.
00020      */
00021     unsigned char* buf;
00022
```

```
00023     /*
00024      * The number of valid bytes in the buffer.
00025      */
00026     unsigned int count;
00027
00028     /*
00029      * Current position
00030      */
00031     unsigned int pos;
00032
00033     /*
00034      * The currently marked position in the stream.
00035      */
00036     unsigned int markpos;
00037
00038 public:
00039
00040     ByteArrayInputStream(unsigned char* buf, unsigned int count);
00041
00048     virtual int available();
00049
00053     virtual void mark();
00054
00060     virtual bool markSupported();
00061
00065     using InputStream::read;
00066
00072     virtual int read();
00073
00078     virtual void reset();
00079 };
00080
00081 #endif /* __RASPBERRY_IO_BYTE_ARRAY_INPUT_STREAM_H__ */
```

5.13 ByteArrayOutputStream.cpp File Reference

`#include "ByteArrayOutputStream.h"`
Include dependency graph for ByteArrayOutputStream.cpp:



Macros

- #define __RASPBERRY_IO_BYTE_ARRAY_OUTPUT_STREAM_CPP__ 1

5.13.1 Macro Definition Documentation

5.13.1.1 #define __RASPBERRY_IO_BYTE_ARRAY_OUTPUT_STREAM_CPP__ 1

Raspberry IO.

ByteArrayOutputStream

This class implements an output stream in which the data is written into a unsigned char array.

Definition at line 11 of file ByteArrayOutputStream.cpp.

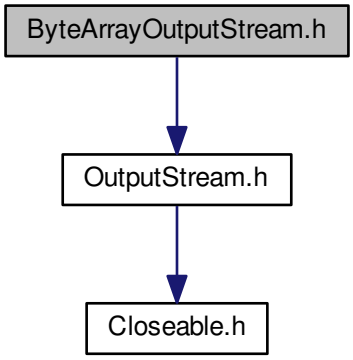
5.14 ByteArrayOutputStream.cpp

```
00001
00010 #ifndef __RASPBERRY_IO_BYTE_ARRAY_OUTPUT_STREAM_CPP__
00011 #define __RASPBERRY_IO_BYTE_ARRAY_OUTPUT_STREAM_CPP__ 1
00012
00013 #include "ByteArrayOutputStream.h"
00014
00015 ByteArrayOutputStream::ByteArrayOutputStream(unsigned char* buf
      ,
00016         unsigned int count) :
00017         buf(buf), count(count) {
00018     pos = 0;
00019 }
00020
00021 void ByteArrayOutputStream::reset() {
00022     pos = 0;
00023 }
00024
00025 unsigned int ByteArrayOutputStream::size() {
00026     return count;
00027 }
00028
00029 unsigned char* ByteArrayOutputStream::toByteArray() {
00030     return buf;
00031 }
00032
00033 void ByteArrayOutputStream::write(unsigned char b) {
00034     buf[pos++] = b;
00035 }
00036
00037 #endif /* __RASPBERRY_IO_BYTE_ARRAY_OUTPUT_STREAM_CPP__ */
```

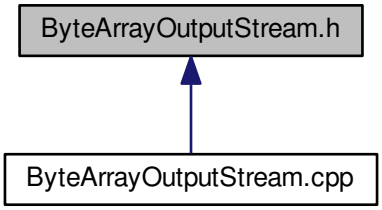
5.15 ByteArrayOutputStream.h File Reference

```
#include <OutputStream.h>
```

Include dependency graph for ByteArrayOutputStream.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ByteArrayOutputStream

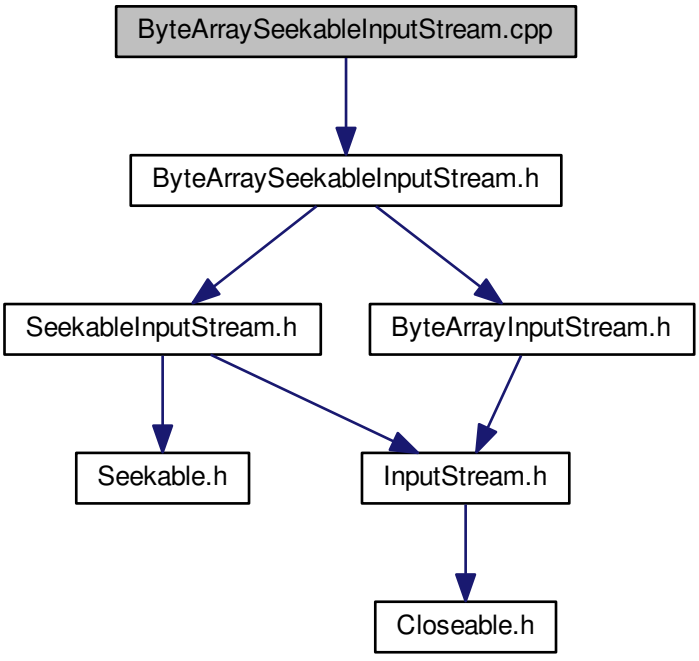
5.16 ByteArrayOutputStream.h

```
00001
00010 #ifndef __RASPBERRY_IO_BYTE_ARRAY_OUTPUT_STREAM_H__
00011 #define __RASPBERRY_IO_BYTE_ARRAY_OUTPUT_STREAM_H__ 1
00012
00013 #include <OutputStream.h>
00014
00015 class ByteArrayOutputStream: public OutputStream {
00016 protected:
00017
00018     /*
00019      * The buffer where data is stored.
00020      */
00021     unsigned char* buf;
00022
00023     /*
00024      * The number of valid bytes in the buffer.
00025      */
```

```
00026     unsigned int count;
00027
00028     /*
00029      * Current position
00030      */
00031     unsigned int pos;
00032
00033 public:
00034
00041     ByteArrayOutputStream(unsigned char* buf, unsigned int count);
00042
00046     void reset();
00047
00053     unsigned int size();
00054
00060     unsigned char* toByteArray();
00061
00065     using OutputStream::write;
00066
00072     virtual void write(unsigned char b);
00073 };
00074
00075 #endif /* __RASPBERRY_IO_BYTE_ARRAY_OUTPUT_STREAM_H__ */
```

5.17 ByteArraySeekableInputStream.cpp File Reference

`#include "ByteArraySeekableInputStream.h"`
Include dependency graph for ByteArraySeekableInputStream.cpp:



Macros

- #define __RASPBERRY_IO_BYTE_ARRAY_SEEKABLE_INPUT_STREAM_CPP__ 1

5.17.1 Macro Definition Documentation

5.17.1.1 #define __RASPBERRY_IO_BYTE_ARRAY_SEEKABLE_INPUT_STREAM_CPP__ 1

Raspberry IO.

ByteArraySeekableInputStream

A ByteArraySeekableInputStream obtains input bytes from a resource in a file system that implements SeekableInputStream interface.

Definition at line 11 of file ByteArraySeekableInputStream.cpp.

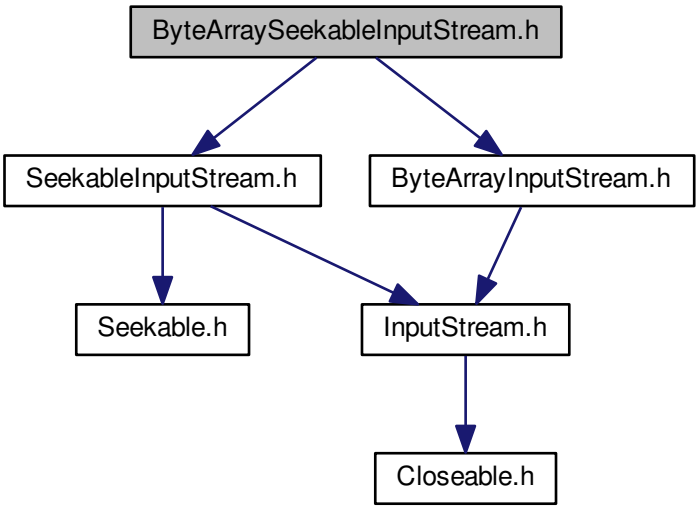
5.18 ByteArraySeekableInputStream.cpp

```
00001
00010 #ifndef __RASPBERRY_IO_BYTE_ARRAY_SEEKABLE_INPUT_STREAM_CPP__
00011 #define __RASPBERRY_IO_BYTE_ARRAY_SEEKABLE_INPUT_STREAM_CPP__ 1
00012
00013 #include "ByteArraySeekableInputStream.h"
00014
00015 ByteArraySeekableInputStream::ByteArraySeekableInputStream
      (unsigned char* buf,
00016         unsigned int count) :
00017         ByteArrayInputStream(buf, count) {
00018 }
00019
00020 void ByteArraySeekableInputStream::seek(unsigned int pos) {
00021     this->pos = pos;
00022 }
00023
00024 #endif /* __RASPBERRY_IO_BYTE_ARRAY_SEEKABLE_INPUT_STREAM_CPP__ */
```

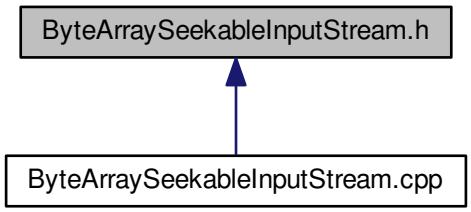
5.19 ByteArraySeekableInputStream.h File Reference

```
#include <SeekableInputStream.h>
#include <ByteArrayInputStream.h>
```

Include dependency graph for ByteArraySeekableInputStream.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ByteArraySeekableInputStream

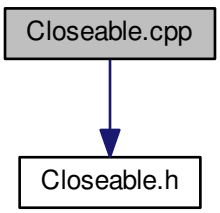
5.20 ByteArraySeekableInputStream.h

```
00001
00010 #ifndef __RASPBERRY_IO_BYTE_ARRAY_SEEKABLE_INPUT_STREAM_H__
00011 #define __RASPBERRY_IO_BYTE_ARRAY_SEEKABLE_INPUT_STREAM_H__ 1
00012
00013 #include <SeekableInputStream.h>
00014 #include <ByteArrayInputStream.h>
00015
00016 class ByteArraySeekableInputStream: public
      SeekableInputStream,
00017         public ByteArrayInputStream {
00018 public:
00019
00020     ByteArraySeekableInputStream(unsigned char* buf, unsigned int
      count);
00021
00022     virtual void seek(unsigned int pos);
00023 };
00024
00025 #endif /* __RASPBERRY_IO_BYTE_ARRAY_SEEKABLE_INPUT_STREAM_H__ */
```

5.21 Closeable.cpp File Reference

```
#include "Closeable.h"
```

Include dependency graph for Closeable.cpp:



Macros

- #define __RASPBERRY_IO_CLOSEABLE_CPP__ 1

5.21.1 Macro Definition Documentation

5.21.1.1 #define __RASPBERRY_IO_CLOSEABLE_CPP__ 1

Raspberry IO.

Closeable

A Closeable is a source or destination of data that can be closed.

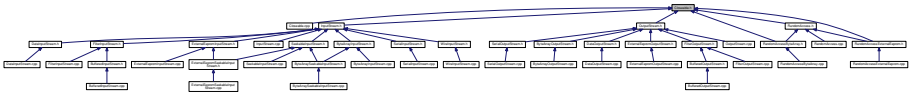
Definition at line 10 of file Closeable.cpp.

5.22 Closeable.cpp

```
00001
00009 #ifndef __RASPBERRY_IO_CLOSEABLE_CPP__
00010 #define __RASPBERRY_IO_CLOSEABLE_CPP__ 1
00011
00012 #include "Closeable.h"
00013
00014 #endif /* __RASPBERRY_IO_CLOSEABLE_CPP__ */
```

5.23 Closeable.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

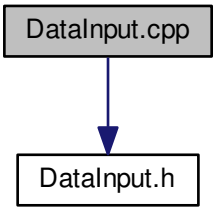
- class Closeable

5.24 Closeable.h

```
00001
00009 #ifndef __RASPBERRY_IO_CLOSEABLE_H__
00010 #define __RASPBERRY_IO_CLOSEABLE_H__ 1
00011
00012 class Closeable {
00013 public:
00014
00015     virtual ~Closeable() {
00016     }
00017
00018     virtual void close() = 0;
00019 };
00020
00021 #endif /* __RASPBERRY_IO_CLOSEABLE_H__ */
```

5.25 DataInput.cpp File Reference

`#include "DataInput.h"`
Include dependency graph for DataInput.cpp:



Macros

- #define __RASPBERRY_IO_DATA_INPUT_CPP__ 1

5.25.1 Macro Definition Documentation

5.25.1.1 #define __RASPBERRY_IO_DATA_INPUT_CPP__ 1

Raspberry IO.

DataInput

The DataInput interface provides for reading bytes from a binary stream and reconstructing from them data in any of the primitive types.

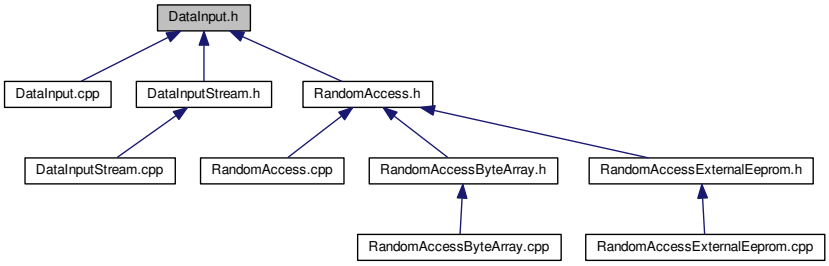
Definition at line 11 of file DataInput.cpp.

5.26 DataInput.cpp

```
00001
00010 #ifndef __RASPBERRY_IO_DATA_INPUT_CPP__
00011 #define __RASPBERRY_IO_DATA_INPUT_CPP__ 1
00012
00013 #include "DataInput.h"
00014
00015 #endif /* __RASPBERRY_IO_DATA_INPUT_CPP__ */
00016
```

5.27 DataInput.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

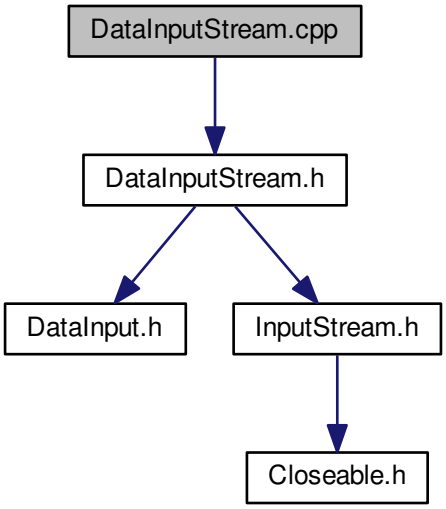
- class DataInput

5.28 DataInput.h

```
00001
00011 #ifndef __RASPBERRY_IO_DATA_INPUT_H__
00012 #define __RASPBERRY_IO_DATA_INPUT_H__ 1
00013
00014 class DataInput {
00015 public:
00016
00017     virtual ~DataInput() {
00018     }
00019
00025     virtual unsigned char readByte() = 0;
00026
00032     virtual bool readBoolean() = 0;
00033
00039     virtual char readChar() = 0;
00040
00046     virtual unsigned char readUnsignedChar() = 0;
00047
00053     virtual int readInt() = 0;
00054
00060     virtual unsigned int readUnsignedInt() = 0;
00061
00067     virtual long readLong() = 0;
00068
00074     virtual unsigned long readUnsignedLong() = 0;
00075
00081     virtual float readFloat() = 0;
00082
00088     virtual double readDouble() = 0;
00089
00096     virtual void readFully(unsigned char* b, int len) = 0;
00097
00104     virtual unsigned int skipBytes(unsigned int n) = 0;
00105 };
00106
00107 #endif /* __RASPBERRY_IO_DATA_INPUT_H__ */
```

5.29 DataInputStream.cpp File Reference

`#include "DataInputStream.h"`
Include dependency graph for DataInputStream.cpp:



Macros

- #define __RASPBERRY_IO_DATA_INPUT_STREAM_CPP__ 1

5.29.1 Macro Definition Documentation

5.29.1.1 #define __RASPBERRY_IO_DATA_INPUT_STREAM_CPP__ 1

Raspberry IO.

DataInputStream

A data input stream lets an application read data from a InputStream.

Definition at line 10 of file DataInputStream.cpp.

5.30 DataInputStream.cpp

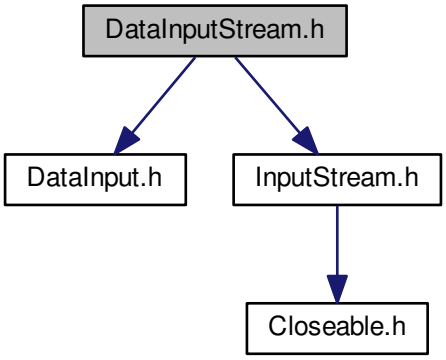
```
00001
00009 #ifndef __RASPBERRY_IO_DATA_INPUT_STREAM_CPP__
00010 #define __RASPBERRY_IO_DATA_INPUT_STREAM_CPP__ 1
00011
00012 #include "DataInputStream.h"
00013
00014 DataInputStream::DataInputStream(InputStream* inputStream) :
00015         inputStream(inputStream) {
00016 }
00017
00018 unsigned char DataInputStream::readByte() {
00019     return (unsigned char) inputStream->read();
00020 }
00021
00022 bool DataInputStream::readBoolean() {
```

```
00023     return (bool) inputStream->read();
00024 }
00025
00026 char DataInputStream::readChar() {
00027     return (char) inputStream->read();
00028 }
00029
00030 unsigned char DataInputStream::readUnsignedChar() {
00031     return (unsigned char) inputStream->read();
00032 }
00033
00034 int DataInputStream::readInt() {
00035     int v = 0;
00036     v = inputStream->read();
00037     v <<= 8;
00038     v |= (inputStream->read() & 0xff);
00039     return v;
00040 }
00041
00042 unsigned int DataInputStream::readUnsignedInt() {
00043     return (unsigned int) readInt();
00044 }
00045
00046 long DataInputStream::readLong() {
00047     long v = 0;
00048     v = inputStream->read();
00049     v <<= 8;
00050     v |= (inputStream->read() & 0xff);
00051     v <<= 8;
00052     v |= (inputStream->read() & 0xff);
00053     v <<= 8;
00054     v |= (inputStream->read() & 0xff);
00055     return v;
00056 }
00057
00058 unsigned long DataInputStream::readUnsignedLong() {
00059     return (unsigned long) readLong();
00060 }
00061
00062 float DataInputStream::readFloat() {
00063     return (float) readLong();
00064 }
00065
00066 double DataInputStream::readDouble() {
00067     return (double) readLong();
00068 }
00069
00070 void DataInputStream::readFully(unsigned char* b, int len) {
00071     for (int i = 0; i < len; i++) {
00072         b[i] = inputStream->read();
00073     }
00074 }
00075
00076 unsigned int DataInputStream::skipBytes(unsigned int n) {
00077     return inputStream->skip(n);
00078 }
00079
00080 #endif /* __RASPBERRY_IO_DATA_INPUT_STREAM_CPP__ */
```

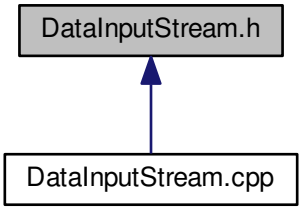
5.31 DataInputStream.h File Reference

```
#include <DataInput.h>
#include <InputStream.h>
```

Include dependency graph for DataInputStream.h:



This graph shows which files directly or indirectly include this file:



Classes

- class DataInputStream

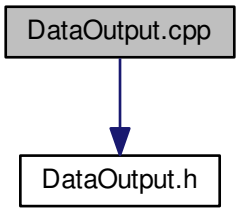
5.32 DataInputStream.h

```
00001
00009 #ifndef __RASPBERRY_IO_DATA_INPUT_STREAM_H__
00010 #define __RASPBERRY_IO_DATA_INPUT_STREAM_H__ 1
00011
00012 #include <DataInput.h>
00013 #include <InputStream.h>
00014
00015 class DataInputStream: public DataInput {
00016
00020     InputStream* inputStream;
00021
00022 public:
00023
00029     DataInputStream(InputStream* inputStream);
00030
00036     virtual unsigned char readByte();
00037
```

```
00043     virtual bool readBoolean();
00044
00050     virtual char readChar();
00051
00057     virtual unsigned char readUnsignedChar();
00058
00064     virtual int readInt();
00065
00071     virtual unsigned int readUnsignedInt();
00072
00078     virtual long readLong();
00079
00085     virtual unsigned long readUnsignedLong();
00086
00092     virtual float readFloat();
00093
00099     virtual double readDouble();
00100
00107     virtual void readFully(unsigned char* b, int len);
00108
00115     virtual unsigned int skipBytes(unsigned int n);
00116 };
00117
00118 #endif /* __RASPBERRY_IO_DATA_INPUT_STREAM_H__ */
```

5.33 DataOutput.cpp File Reference

`#include "DataOutput.h"`
Include dependency graph for DataOutput.cpp:



Macros

- #define __RASPBERRY_IO_DATA_OUTPUT_CPP__ 1

5.33.1 Macro Definition Documentation

5.33.1.1 #define __RASPBERRY_IO_DATA_OUTPUT_CPP__ 1

Raspberry IO.

DataOutput

The DataOutput interface provides for converting data from any of the primitive types to a series of bytes and writing these bytes to a binary stream.

Definition at line 11 of file DataOutput.cpp.

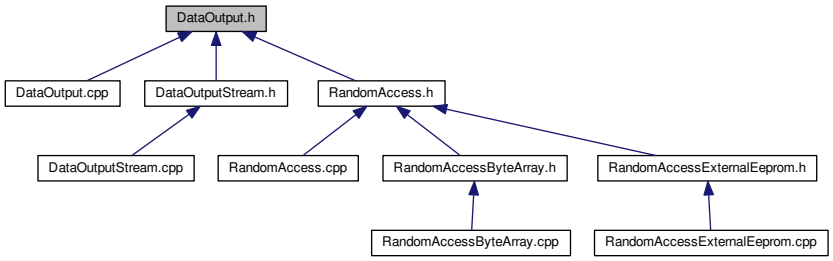
5.34 DataOutput.cpp

```
00001
```

```
00010 #ifndef __RASPBERRY_IO_DATA_OUTPUT_CPP__
00011 #define __RASPBERRY_IO_DATA_OUTPUT_CPP__ 1
00012
00013 #include "DataOutput.h"
00014
00015 #endif /* __RASPBERRY_IO_DATA_OUTPUT_CPP__ */
00016
```

5.35 DataOutput.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class DataOutput

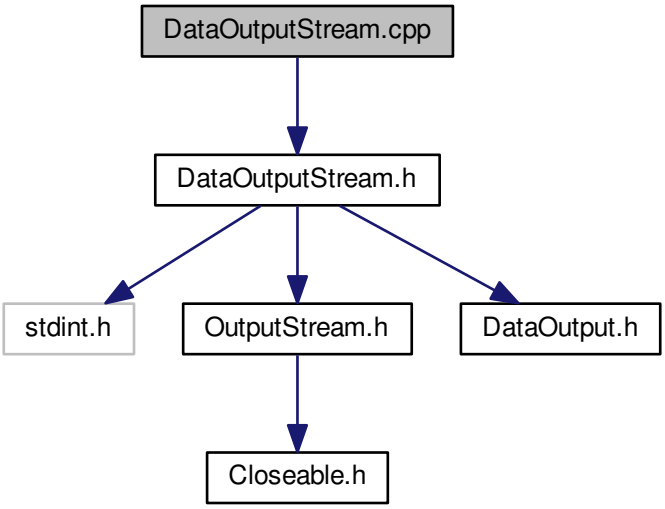
5.36 DataOutput.h

```
00001
00010 #ifndef __RASPBERRY_IO_DATA_OUTPUT_H__
00011 #define __RASPBERRY_IO_DATA_OUTPUT_H__ 1
00012
00013 class DataOutput {
00014 public:
00015
00022     virtual void write(unsigned char* b, int len) = 0;
00023
00029     virtual void write(unsigned char b) = 0;
00030
00036     virtual void writeByte(unsigned char b) = 0;
00037
00044     virtual void writeBytes(unsigned char* b, int len) = 0;
00045
00051     virtual void writeBoolean(bool v) = 0;
00052
00058     virtual void writeChar(char c) = 0;
00059
00065     virtual void writeUnsignedChar(unsigned char c) = 0;
00066
00072     virtual void writeInt(int v) = 0;
00073
00079     virtual void writeUnsignedInt(unsigned int v) = 0;
00080
00086     virtual void writeLong(long v) = 0;
00087
00093     virtual void writeUnsignedLong(unsigned long v) = 0;
00094
00100     virtual void writeFloat(float v) = 0;
00101
00107     virtual void writeDouble(double v) = 0;
00108 };
00109
00110 #endif /* __RASPBERRY_IO_DATA_OUTPUT_H__ */
```

5.37 DataOutputStream.cpp File Reference

```
#include "DataOutputStream.h"
```

Include dependency graph for DataOutputStream.cpp:



Macros

- #define __RASPBERRY_IO_DATA_OUTPUT_STREAM_CPP__ 1

5.37.1 Macro Definition Documentation

5.37.1.1 #define __RASPBERRY_IO_DATA_OUTPUT_STREAM_CPP__ 1

Raspberry IO.

DataOutputStream

A data output stream lets an application write types to an OutputStream.

Definition at line 10 of file DataOutputStream.cpp.

5.38 DataOutputStream.cpp

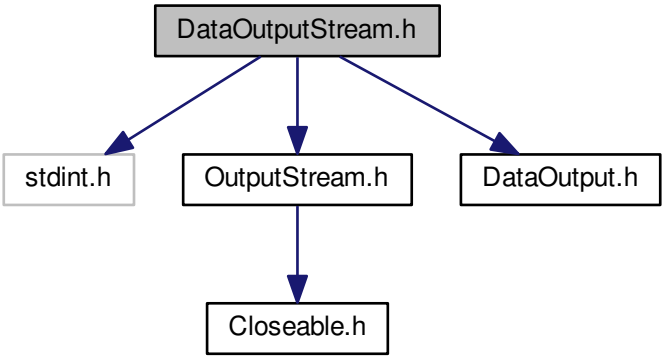
```
00001
00009 #ifndef __RASPBERRY_IO_DATA_OUTPUT_STREAM_CPP__
00010 #define __RASPBERRY_IO_DATA_OUTPUT_STREAM_CPP__ 1
00011
00012 #include "DataOutputStream.h"
00013
00014 DataOutputStream::DataOutputStream(
      OutputStream* outputStream) :
00015         outputStream(outputStream) {
00016 }
00017
00018 void DataOutputStream::write(unsigned char* b, int len) {
00019     writeBytes(b, len);
00020 }
00021
```

```
00022 void DataOutputStream::write(unsigned char b) {
00023     writeByte(b);
00024 }
00025
00026 void DataOutputStream::writeByte(unsigned char b) {
00027     outputStream->write(b);
00028 }
00029
00030 void DataOutputStream::writeBytes(unsigned char* b, int len) {
00031     for (int i = 0; i < len; i++) {
00032         outputStream->write(b[i]);
00033     }
00034 }
00035
00036 void DataOutputStream::writeBoolean(bool v) {
00037     outputStream->write((unsigned char) v);
00038 }
00039
00040 void DataOutputStream::writeChar(char c) {
00041     outputStream->write((unsigned char) c);
00042 }
00043
00044 void DataOutputStream::writeUnsignedChar(unsigned char c) {
00045     outputStream->write((unsigned char) c);
00046 }
00047
00048 void DataOutputStream::writeInt(int v) {
00049     outputStream->write((unsigned char) ((v >> 8) & 0xff));
00050     outputStream->write((unsigned char) (v & 0xff));
00051 }
00052
00053 void DataOutputStream::writeUnsignedInt(unsigned int v) {
00054     writeInt((int) v);
00055 }
00056
00057 void DataOutputStream::writeLong(long v) {
00058     outputStream->write((unsigned char) ((v >> 24) & 0xff));
00059     outputStream->write((unsigned char) ((v >> 16) & 0xff));
00060     outputStream->write((unsigned char) ((v >> 8) & 0xff));
00061     outputStream->write((unsigned char) (v & 0xff));
00062 }
00063
00064 void DataOutputStream::writeUnsignedLong(unsigned long v) {
00065     writeLong((long) v);
00066 }
00067
00068 void DataOutputStream::writeFloat(float v) {
00069     writeLong((long) v);
00070 }
00071
00072 void DataOutputStream::writeDouble(double v) {
00073     writeLong((long) v);
00074 }
00075
00076 #endif /* __RASPBERRY_IO_DATA_OUTPUT_STREAM_CPP__ */
```

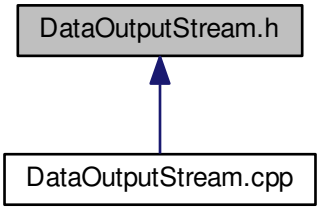
5.39 DataOutputStream.h File Reference

```
#include <stdint.h>
#include <OutputStream.h>
#include <DataOutput.h>
```

Include dependency graph for DataOutputStream.h:



This graph shows which files directly or indirectly include this file:



Classes

- class DataOutputStream

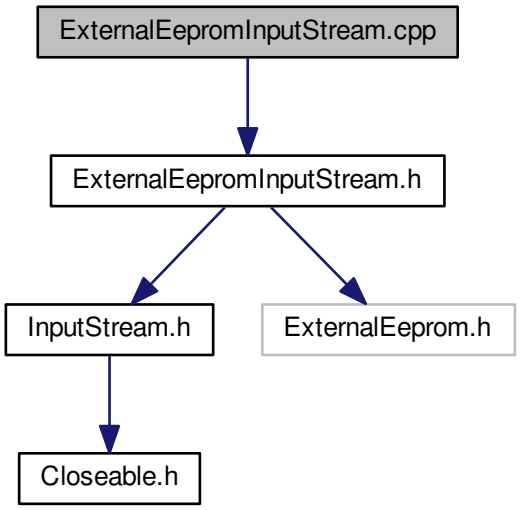
5.40 DataOutputStream.h

```
00001
00009 #ifndef __RASPBERRY_IO_DATA_OUTPUT_STREAM_H__
00010 #define __RASPBERRY_IO_DATA_OUTPUT_STREAM_H__ 1
00011
00012 #include <stdint.h>
00013 #include <OutputStream.h>
00014 #include <DataOutput.h>
00015
00016 class DataOutputStream: public DataOutput {
00017
00021     OutputStream* outputStream;
00022
00023 public:
00024
00030     DataOutputStream(OutputStream* outputStream);
00031
00038     virtual void write(unsigned char* b, int len);
```

```
00039
00045     virtual void write(unsigned char b);
00046
00052     virtual void writeByte(unsigned char b);
00053
00060     virtual void writeBytes(unsigned char* b, int len);
00061
00067     virtual void writeBoolean(bool v);
00068
00074     virtual void writeChar(char c);
00075
00081     virtual void writeUnsignedChar(unsigned char c);
00082
00088     virtual void writeInt(int v);
00089
00095     virtual void writeUnsignedInt(unsigned int v);
00096
00102     virtual void writeLong(long v);
00103
00109     virtual void writeUnsignedLong(unsigned long v);
00110
00116     virtual void writeFloat(float v);
00117
00123     virtual void writeDouble(double v);
00124 };
00125
00126 #endif /* __RASPBERRY_IO_DATA_OUTPUT_STREAM_H__ */
```

5.41 ExternalEepromInputStream.cpp File Reference

`#include "ExternalEepromInputStream.h"`
Include dependency graph for ExternalEepromInputStream.cpp:



Macros

- #define __RASPBERRY_IO_EXTERNAL_EEPROM_INPUT_STREAM_CPP__ 1

5.41.1 Macro Definition Documentation

5.41.1.1 #define __RASPBERRY_IO_EXTERNAL_EEPROM_INPUT_STREAM_CPP__ 1

Raspberry IO.

ExternalEepromInputStream

An ExternalEepromInputStream obtains input bytes from a externalEeprom.

Definition at line 11 of file ExternalEepromInputStream.cpp.

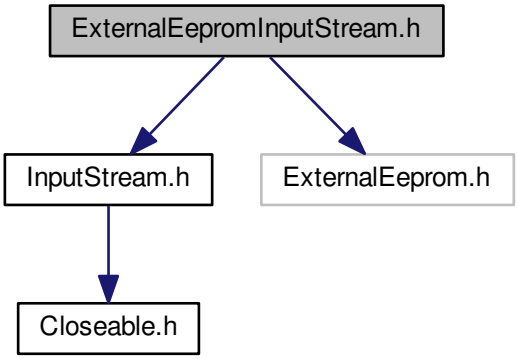
5.42 ExternalEepromInputStream.cpp

```
00001
00010 #ifndef __RASPBERRY_IO_EXTERNAL_EEPROM_INPUT_STREAM_CPP__
00011 #define __RASPBERRY_IO_EXTERNAL_EEPROM_INPUT_STREAM_CPP__ 1
00012
00013 #include "ExternalEepromInputStream.h"
00014
00015 ExternalEepromInputStream::ExternalEepromInputStream(
00016         ExternalEeprom* externalEeprom) :
00017         externalEeprom(externalEeprom) {
00018     markpos = 0;
00019     pos = 0;
00020     externalEepromSize = externalEeprom->getDeviceSize();
00021 }
00022
00023 int ExternalEepromInputStream::available() {
00024     if (externalEepromSize > pos) {
00025         return 1;
00026     }
00027     return 0;
00028 }
00029
00030 void ExternalEepromInputStream::mark() {
00031     markpos = pos;
00032 }
00033
00034 bool ExternalEepromInputStream::markSupported() {
00035     return true;
00036 }
00037
00038 int ExternalEepromInputStream::read() {
00039     if (pos >= externalEepromSize) {
00040         return -1;
00041     }
00042     return (int) externalEeprom->read(pos++);
00043 }
00044
00045 int ExternalEepromInputStream::read(unsigned char* b, int off, int len) {
00046     unsigned int available = (externalEepromSize -
      pos);
00047     int cnt;
00048     len = (int) ((unsigned int) len > available) ? available : len;
00049     cnt = externalEeprom->readBytes(pos, &b[off], len);
00050     pos += cnt;
00051     return cnt;
00052 }
00053
00054 void ExternalEepromInputStream::reset() {
00055     pos = markpos;
00056 }
00057
00058 #endif /* __RASPBERRY_IO_EXTERNAL_EEPROM_INPUT_STREAM_CPP__ */
```

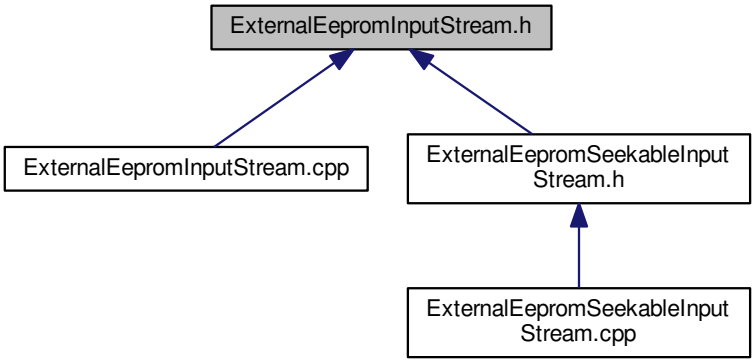
5.43 ExternalEepromInputStream.h File Reference

```
#include <InputStream.h>
#include <ExternalEeprom.h>
```

Include dependency graph for ExternalEepromInputStream.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ExternalEepromInputStream

5.44 ExternalEepromInputStream.h

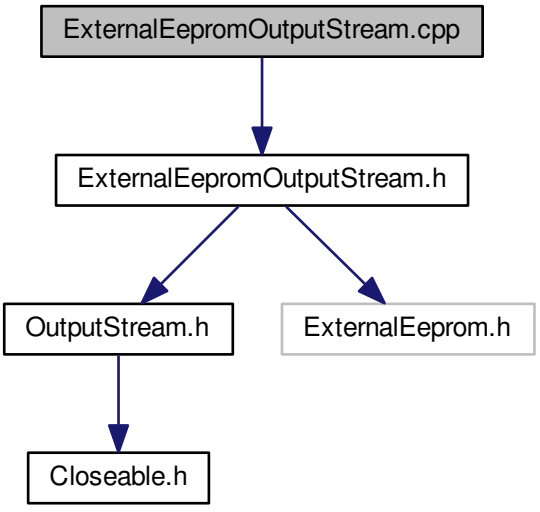
```
00001
00010 #ifndef __RASPBERRY_IO_EXTERNAL_EEPROM_INPUT_STREAM_H__
00011 #define __RASPBERRY_IO_EXTERNAL_EEPROM_INPUT_STREAM_H__ 1
00012
00013 #include <InputStream.h>
00014 #include <ExternalEeprom.h>
00015
00016 class ExternalEepromInputStream: public virtual
      InputStream {
00017 protected:
00018
00019     /*
```

```
00020      * The externalEeprom where data is stored.
00021      */
00022     ExternalEeprom* externalEeprom;
00023
00024     /*
00025      * Current position
00026      */
00027     unsigned int pos;
00028
00029     /*
00030      * The currently marked position in the stream.
00031      */
00032     unsigned int markpos;
00033
00034     /*
00035      * The size of the externalEeprom.
00036      */
00037     unsigned int externalEepromSize;
00038
00039 public:
00040
00046     ExternalEepromInputStream(ExternalEeprom* externalEeprom);
00047
00055     virtual int available();
00056
00060     virtual void mark();
00061
00067     virtual bool markSupported();
00068
00072     using InputStream::read;
00073
00079     virtual int read();
00080
00089     virtual int read(unsigned char* b, int off, int len);
00090
00095     virtual void reset();
00096 };
00097
00098 #endif /* __RASPBERRY_IO_EXTERNAL_EEPROM_INPUT_STREAM_H__ */
```

5.45 ExternalEepromOutputStream.cpp File Reference

`#include "ExternalEepromOutputStream.h"`

Include dependency graph for ExternalEepromOutputStream.cpp:



Macros

- #define __RASPBERRY_IO_EXTERNAL_EEPROM_OUTPUT_STREAM_CPP__ 1

5.45.1 Macro Definition Documentation

5.45.1.1 #define __RASPBERRY_IO_EXTERNAL_EEPROM_OUTPUT_STREAM_CPP__ 1

Raspberry IO.

ExternalEepromOutputStream

A externalEeprom output stream is an output stream for writing data to a ExternalEeprom.

Definition at line 10 of file ExternalEepromOutputStream.cpp.

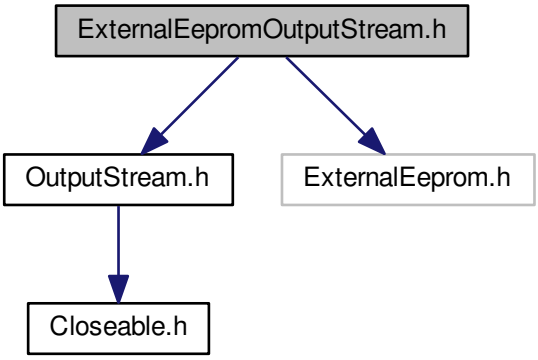
5.46 ExternalEepromOutputStream.cpp

```
00001
00009 #ifndef __RASPBERRY_IO_EXTERNAL_EEPROM_OUTPUT_STREAM_CPP__
00010 #define __RASPBERRY_IO_EXTERNAL_EEPROM_OUTPUT_STREAM_CPP__ 1
00011
00012 #include "ExternalEepromOutputStream.h"
00013
00014 ExternalEepromOutputStream::ExternalEepromOutputStream
      (
00015         ExternalEeprom* externalEeprom) :
00016         externalEeprom(externalEeprom) {
00017     pos = 0;
00018 }
00019
00020 void ExternalEepromOutputStream::write(unsigned char b) {
00021     externalEeprom->write(pos++, b);
00022 }
00023
00024 void ExternalEepromOutputStream::write(unsigned char* b, int off, int len)
       {
00025     externalEeprom->writeBytes(pos, &b[off], len);
00026     pos += len;
00027 }
00028
00029 #endif /* __RASPBERRY_IO_EXTERNAL_EEPROM_OUTPUT_STREAM_CPP__ */
```

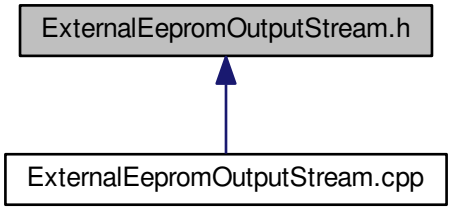
5.47 ExternalEepromOutputStream.h File Reference

```
#include <OutputStream.h>
#include <ExternalEeprom.h>
```

Include dependency graph for ExternalEepromOutputStream.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ExternalEepromOutputStream

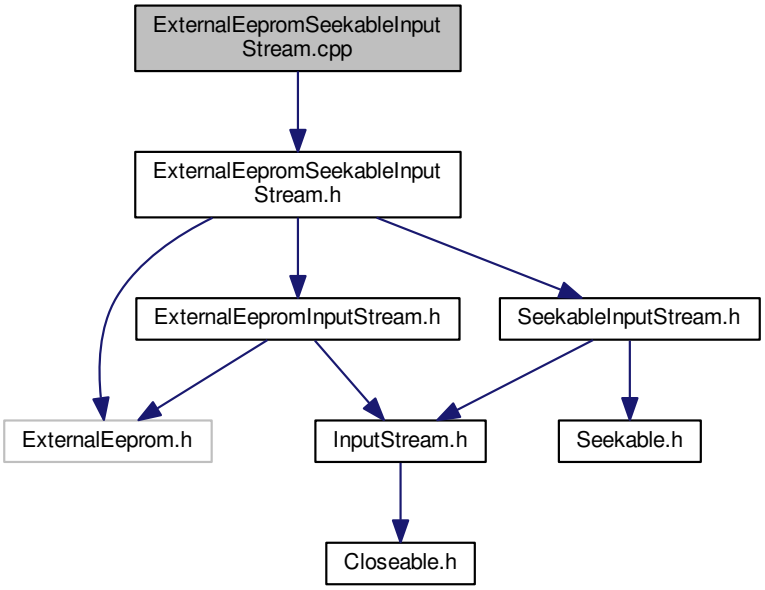
5.48 ExternalEepromOutputStream.h

```
00001
00010 #ifndef __RASPBERRY_IO_EXTERNAL_EEPROM_OUTPUT_STREAM_H__
00011 #define __RASPBERRY_IO_EXTERNAL_EEPROM_OUTPUT_STREAM_H__ 1
00012
00013 #include <OutputStream.h>
00014 #include <ExternalEeprom.h>
00015
00016 class ExternalEepromOutputStream: public OutputStream {
00017
00021     ExternalEeprom* externalEeprom;
00022
00026     unsigned int pos;
00027
00028 public:
00029
00035     ExternalEepromOutputStream(ExternalEeprom* externalEeprom);
00036
```

```
00040     using OutputStream::write;
00041
00047     virtual void write(unsigned char b);
00048
00057     virtual void write(unsigned char* b, int off, int len);
00058 };
00059
00060 #endif /* __RASPBERRY_IO_EXTERNAL_EEPROM_OUTPUT_STREAM_H__ */
```

5.49 ExternalEepromSeekableInputStream.cpp File Reference

`#include "ExternalEepromSeekableInputStream.h"`
Include dependency graph for ExternalEepromSeekableInputStream.cpp:



Macros

- #define __RASPBERRY_IO_EXTERNAL_EEPROM_SEEKABLE_INPUT_STREAM_CPP__ 1

5.49.1 Macro Definition Documentation

5.49.1.1 #define __RASPBERRY_IO_EXTERNAL_EEPROM_SEEKABLE_INPUT_STREAM_CPP__ 1

Raspberry IO.

ExternalEepromSeekableInputStream

A ExternalEepromSeekableInputStream obtains input bytes from a external input stream.

Definition at line 11 of file ExternalEepromSeekableInputStream.cpp.

5.50 ExternalEepromSeekableInputStream.cpp

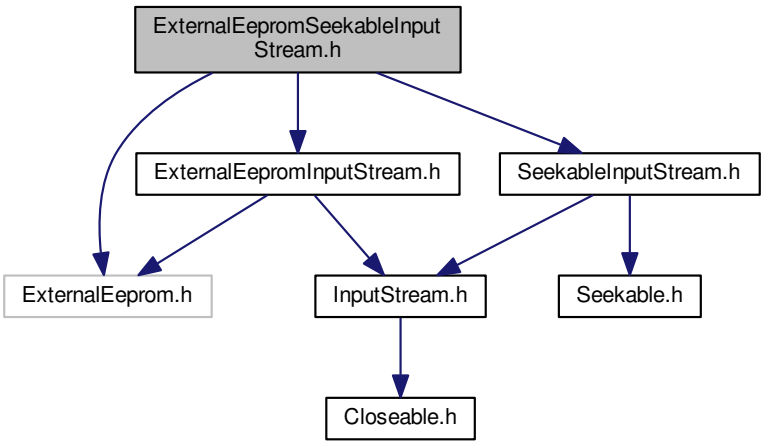
```
00001
00010 #ifndef __RASPBERRY_IO_EXTERNAL_EEPROM_SEEKABLE_INPUT_STREAM_CPP__
```

```
00011 #define __RASPBERRY_IO_EXTERNAL_EEPROM_SEEKABLE_INPUT_STREAM_CPP__ 1
00012
00013 #include "ExternalEepromSeekableInputStream.h"
00014
00015 ExternalEepromSeekableInputStream::ExternalEepromSeekableInputStream
      (
00016         ExternalEeprom* externalEeprom) :
00017         ExternalEepromInputStream(externalEeprom) {
00018 }
00019
00020 void ExternalEepromSeekableInputStream::seek(unsigned int pos) {
00021     this->pos = pos;
00022 }
00023
00024 #endif /* __RASPBERRY_IO_EXTERNAL_EEPROM_SEEKABLE_INPUT_STREAM_CPP__ */
```

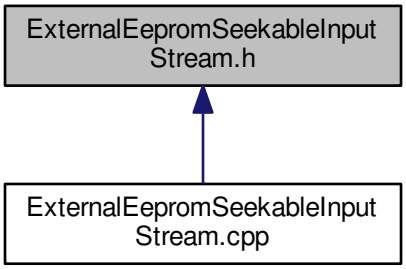
5.51 ExternalEepromSeekableInputStream.h File Reference

```
#include <ExternalEeprom.h>
#include <SeekableInputStream.h>
#include <ExternalEepromInputStream.h>
```

Include dependency graph for ExternalEepromSeekableInputStream.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ExternalEepromSeekableInputStream

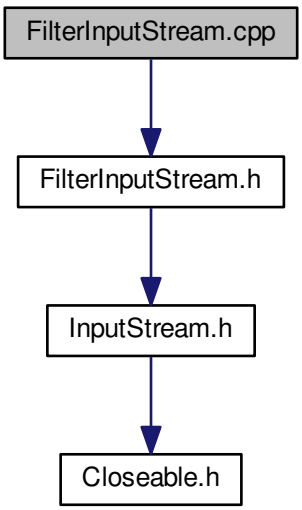
5.52 ExternalEepromSeekableInputStream.h

```
00001
00010 #ifndef __RASPBERRY_IO_EXTERNAL_EEPROM_SEEKABLE_INPUT_STREAM_H__
00011 #define __RASPBERRY_IO_EXTERNAL_EEPROM_SEEKABLE_INPUT_STREAM_H__ 1
00012
00013 #include <ExternalEeprom.h>
00014 #include <SeekableInputStream.h>
00015 #include <ExternalEepromInputStream.h>
00016
00017 class ExternalEepromSeekableInputStream: public
      ExternalEepromInputStream,
00018         public SeekableInputStream {
00019 public:
00020
00026     ExternalEepromSeekableInputStream(ExternalEeprom*
      externalEeprom);
00027
00033     virtual void seek(unsigned int pos);
00034 };
00035
00036 #endif /* __RASPBERRY_IO_EXTERNAL_EEPROM_SEEKABLE_INPUT_STREAM_H__ */
```

5.53 FilterInputStream.cpp File Reference

```
#include "FilterInputStream.h"
```

Include dependency graph for FilterInputStream.cpp:



Macros

- #define __RASPBERRY_IO_FILTER_INPUT_STREAM_CPP__ 1

5.53.1 Macro Definition Documentation

5.53.1.1 #define __RASPBERRY_IO_FILTER_INPUT_STREAM_CPP__ 1

A `FilterInputStream` contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.

The class `FilterInputStream` itself simply overrides all methods of `InputStream` with versions that pass all requests to the contained input stream. Subclasses of `FilterInputStream` may further override some of these methods and may also provide additional methods and fields.

Definition at line 17 of file FilterInputStream.cpp.

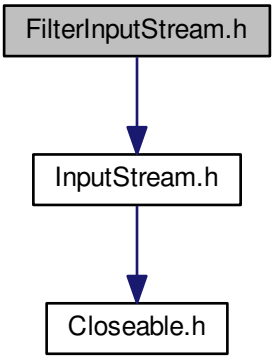
5.54 FilterInputStream.cpp

```
00001
00016 #ifndef __RASPBERRY_IO_FILTER_INPUT_STREAM_CPP__
00017 #define __RASPBERRY_IO_FILTER_INPUT_STREAM_CPP__ 1
00018
00019 #include "FilterInputStream.h"
00020
00021 FilterInputStream::FilterInputStream(
      InputStream* in) :
00022         in(in) {
00023 }
00024
00025 int FilterInputStream::read() {
00026     return in->read();
00027 }
00028
00029 int FilterInputStream::read(unsigned char* b, int len) {
00030     return in->read(b, len);
00031 }
00032
00033 int FilterInputStream::read(unsigned char* b, int off, int len) {
00034     return in->read(b, off, len);
00035 }
00036
00037 unsigned int FilterInputStream::skip(unsigned int n) {
00038     return in->skip(n);
00039 }
00040
00041 int FilterInputStream::available() {
00042     return in->available();
00043 }
00044
00045 void FilterInputStream::close() {
00046     in->close();
00047 }
00048
00049 void FilterInputStream::mark() {
00050     in->mark();
00051 }
00052
00053 void FilterInputStream::reset() {
00054     in->reset();
00055 }
00056
00057 bool FilterInputStream::markSupported() {
00058     return in->markSupported();
00059 }
00060
00061 #endif /* __RASPBERRY_IO_FILTER_INPUT_STREAM_CPP__ */
```

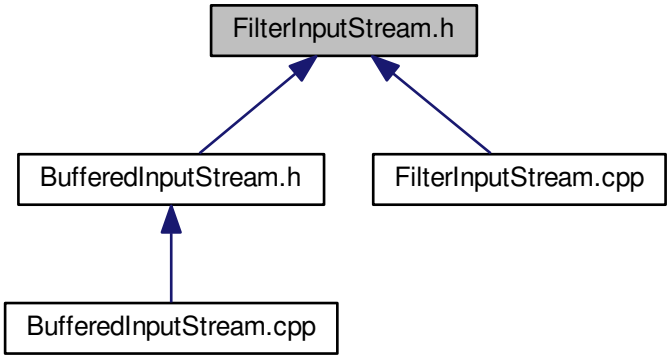
5.55 FilterInputStream.h File Reference

```
#include <InputStream.h>
```

Include dependency graph for FilterInputStream.h:



This graph shows which files directly or indirectly include this file:



Classes

- class FilterInputStream

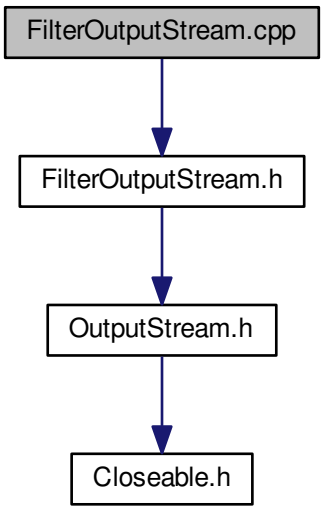
5.56 FilterInputStream.h

```
00001
00016 #ifndef __RASPBERRY_IO_FILTER_INPUT_STREAM_H__
00017 #define __RASPBERRY_IO_FILTER_INPUT_STREAM_H__ 1
00018
00019 #include <InputStream.h>
00020
00021 class FilterInputStream: public virtual InputStream {
00022
00023 protected:
00024
```

```
00028     InputStream* in;
00029
00038     FilterInputStream(InputStream* in);
00039
00040 public:
00041
00055     virtual int read();
00056
00074     virtual int read(unsigned char* b, int len);
00075
00090     virtual int read(unsigned char* b, int off, int len);
00091
00097     virtual unsigned int skip(unsigned int n);
00098
00107     virtual int available();
00108
00113     virtual void close();
00114
00122     virtual void mark();
00123
00137     virtual void reset();
00138
00149     virtual bool markSupported();
00150 };
00151
00152 #endif /* __RASPBERRY_IO_FILTER_INPUT_STREAM_H__ */
```

5.57 FilterOutputStream.cpp File Reference

`#include "FilterOutputStream.h"`
Include dependency graph for FilterOutputStream.cpp:



Macros

- #define __RASPBERRY_IO_FILTER_OUTPUT_STREAM_CPP__ 1

5.57.1 Macro Definition Documentation

5.57.1.1 #define __RASPBERRY_IO_FILTER_OUTPUT_STREAM_CPP__ 1

Raspberry IO.

FilterOutputStream

This class is the superclass of all classes that filter output streams. These streams sit on top of an already existing output stream (the *underlying* output stream) which it uses as its basic sink of data, but possibly transforming the data along the way or providing additional functionality.

The class `FilterOutputStream` itself simply overrides all methods of `OutputStream` with versions that pass all requests to the underlying output stream. Subclasses of `FilterOutputStream` may further override some of these methods as well as provide additional methods and fields.

Definition at line 20 of file FilterOutputStream.cpp.

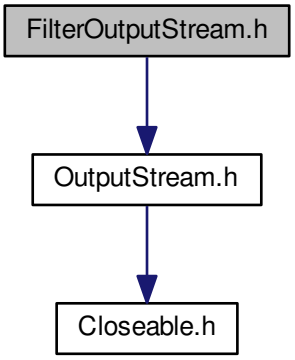
5.58 FilterOutputStream.cpp

```
00001
00019 #ifndef __RASPBERRY_IO_FILTER_OUTPUT_STREAM_CPP__
00020 #define __RASPBERRY_IO_FILTER_OUTPUT_STREAM_CPP__ 1
00021
00022 #include "FilterOutputStream.h"
00023
00024 FilterOutputStream::FilterOutputStream(
      OutputStream* out) :
00025         out(out) {
00026 }
00027
00028 void FilterOutputStream::write(unsigned char b) {
00029     out->write(b);
00030 }
00031
00032 void FilterOutputStream::write(unsigned char* b, int len) {
00033     out->write(b, len);
00034 }
00035
00036 void FilterOutputStream::write(unsigned char* b, int off, int len) {
00037     out->write(b, off, len);
00038 }
00039
00040 void FilterOutputStream::flush() {
00041     out->flush();
00042 }
00043
00044 void FilterOutputStream::close() {
00045     out->flush();
00046     out->close();
00047 }
00048
00049 #endif /* __RASPBERRY_IO_FILTER_OUTPUT_STREAM_CPP__ 1 */
00050
```

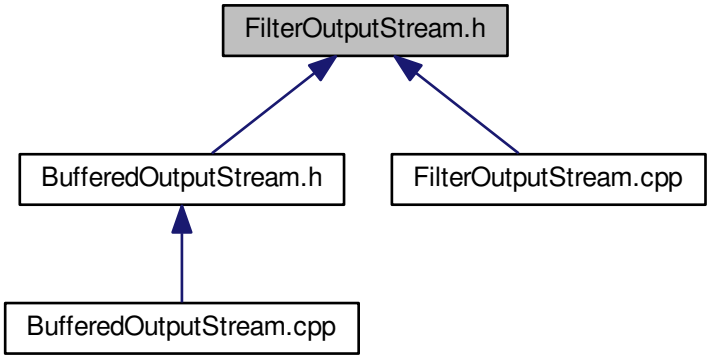
5.59 FilterOutputStream.h File Reference

```
#include <OutputStream.h>
```

Include dependency graph for FilterOutputStream.h:



This graph shows which files directly or indirectly include this file:



Classes

- class FilterOutputStream

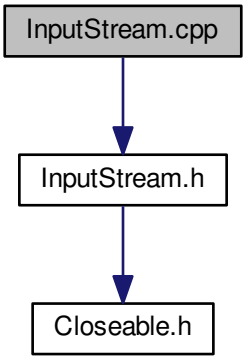
5.60 FilterOutputStream.h

```
00001
00019 #ifndef __RASPBERRY_IO_FILTER_OUTPUT_STREAM_H__
00020 #define __RASPBERRY_IO_FILTER_OUTPUT_STREAM_H__ 1
00021
00022 #include <OutputStream.h>
00023
00024 class FilterOutputStream: public OutputStream {
00025 protected:
00026
00030     OutputStream* out;
```

```
00031 public:
00032
00040     FilterOutputStream(OutputStream* out);
00041
00053     virtual void write(unsigned char b);
00054
00066     virtual void write(unsigned char* b, int len);
00067
00077     virtual void write(unsigned char* b, int off, int len);
00078
00086     virtual void flush();
00087
00096     virtual void close();
00097 };
00098
00099 #endif /* __RASPBERRY_IO_FILTER_OUTPUT_STREAM_H__ */
```

5.61 InputStream.cpp File Reference

`#include "InputStream.h"`
Include dependency graph for InputStream.cpp:



Macros

- #define __RASPBERRY_IO_INPUT_STREAM_CPP__ 1

5.61.1 Macro Definition Documentation

5.61.1.1 #define __RASPBERRY_IO_INPUT_STREAM_CPP__ 1

Raspberry IO.

InputStream

This abstract class is the superclass of all classes representing an input stream of bytes.

Applications that need to define a subclass of InputStream must always provide a method that returns the next unsigned char of input.

Definition at line 14 of file InputStream.cpp.

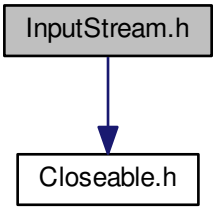
5.62 InputStream.cpp

```
00001
00013 #ifndef __RASPBERRY_IO_INPUT_STREAM_CPP__
00014 #define __RASPBERRY_IO_INPUT_STREAM_CPP__ 1
00015
00016 #include "InputStream.h"
00017
00018 int InputStream::available() {
00019     return 0;
00020 }
00021
00022 void InputStream::close() {
00023 }
00024
00025 void InputStream::mark() {
00026 }
00027
00028 bool InputStream::markSupported() {
00029     return false;
00030 }
00031
00032 int InputStream::read(unsigned char* b, int len) {
00033     return read(b, 0, len);
00034 }
00035
00036 int InputStream::read(unsigned char* b, int off, int len) {
00037     int i, c;
00038     if (b == (unsigned char*) 0) {
00039         return 0;
00040     }
00041     c = read();
00042     if (c == -1) {
00043         return -1;
00044     }
00045     b[off] = (unsigned char) c;
00046     for (i = 1; i < len; i++) {
00047         c = read();
00048         if (c == -1) {
00049             break;
00050         }
00051         b[off + i] = (unsigned char) c;
00052     }
00053     return i;
00054 }
00055
00056 void InputStream::reset() {
00057 }
00058
00059 unsigned int InputStream::skip(unsigned int n) {
00060     unsigned int i;
00061     for (i = 0; i < n && available() > 0; i++) {
00062         read();
00063     }
00064     return i;
00065 }
00066
00067 #endif /* __RASPBERRY_IO_INPUT_STREAM_CPP__ */
```

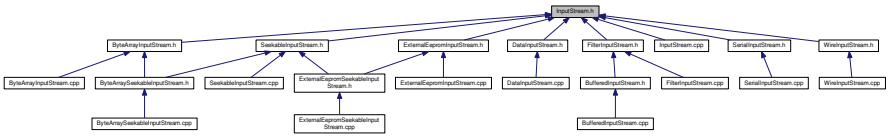
5.63 InputStream.h File Reference

```
#include <Closeable.h>
```

Include dependency graph for InputStream.h:



This graph shows which files directly or indirectly include this file:



Classes

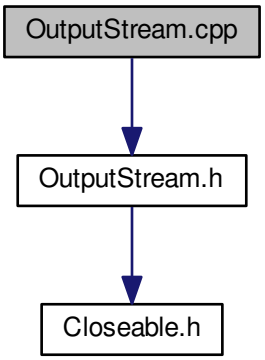
- class InputStream

5.64 InputStream.h

```
00001
00013 #ifndef __RASPBERRY_IO_INPUT_STREAM_H__
00014 #define __RASPBERRY_IO_INPUT_STREAM_H__ 1
00015
00016 #include <Closeable.h>
00017
00018 class InputStream: public Closeable {
00019 public:
00020
00021     virtual ~InputStream() {
00022     }
00023
00029     virtual int available();
00030
00035     virtual void close();
00036
00040     virtual void mark();
00041
00045     virtual bool markSupported();
00046
00050     virtual int read() = 0;
00051
00056     virtual int read(unsigned char* b, int len);
00057
00066     virtual int read(unsigned char* b, int off, int len);
00067
00072     virtual void reset();
00073
00077     virtual unsigned int skip(unsigned int n);
00078 };
00079
00080 #endif /* __RASPBERRY_IO_INPUT_STREAM_H__ */
```

5.65 OutputStream.cpp File Reference

`#include "OutputStream.h"`
Include dependency graph for OutputStream.cpp:



Macros

- #define __RASPBERRY_IO_OUTPUT_STREAM_CPP__ 1

5.65.1 Macro Definition Documentation

5.65.1.1 #define __RASPBERRY_IO_OUTPUT_STREAM_CPP__ 1

Raspberry IO.

OutputStream

This abstract class is the superclass of all classes representing an output stream of bytes. An output stream accepts output bytes and sends them to some sink.

Definition at line 12 of file OutputStream.cpp.

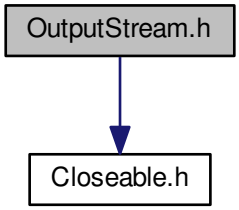
5.66 OutputStream.cpp

```
00001
00011 #ifndef __RASPBERRY_IO_OUTPUT_STREAM_CPP__
00012 #define __RASPBERRY_IO_OUTPUT_STREAM_CPP__ 1
00013
00014 #include "OutputStream.h"
00015
00016 void OutputStream::write(unsigned char* b, int len) {
00017     write(b, 0, len);
00018 }
00019
00020 void OutputStream::write(unsigned char* b, int off, int len) {
00021     if (b == (unsigned char*) 0 || len == 0) {
00022         return;
00023     }
00024     for (int i = 0; i < len; i++) {
00025         write(b[off + i]);
00026     }
00027 }
00028
00029 void OutputStream::flush() {
00030 }
```

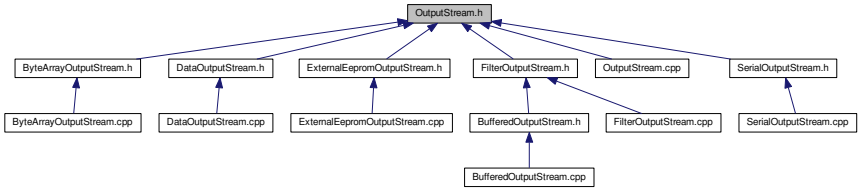
```
00031
00032 void OutputStream::close() {
00033 }
00034
00035 #endif /* __RASPBERRY_IO_OUTPUT_STREAM_CPP__ */
```

5.67 OutputStream.h File Reference

`#include <Closeable.h>`
Include dependency graph for OutputStream.h:



This graph shows which files directly or indirectly include this file:



Classes

- class OutputStream

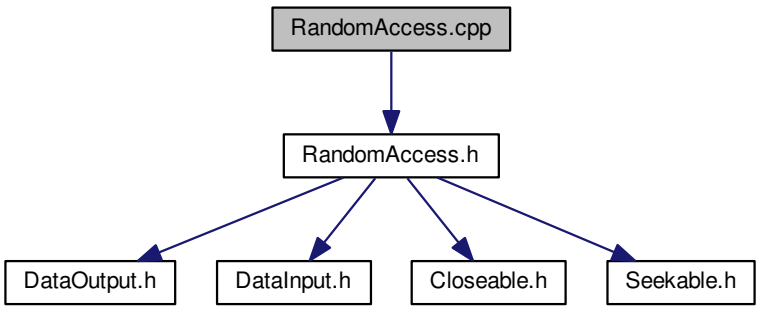
5.68 OutputStream.h

```
00001
00015 #ifndef __RASPBERRY_IO_OUTPUT_STREAM_H__
00016 #define __RASPBERRY_IO_OUTPUT_STREAM_H__ 1
00017
00018 #include <Closeable.h>
00019
00020 class OutputStream: public Closeable {
00021 public:
00022
00023     virtual ~OutputStream() {
00024     }
00025
00030     virtual void close();
00031
00036     virtual void flush();
00037
00041     virtual void write(unsigned char b) = 0;
00042
00050     virtual void write(unsigned char* b, int len);
```

```
00051
00060     virtual void write(unsigned char* b, int off, int len);
00061 };
00062
00063 #endif /* __RASPBERRY_IO_OUTPUT_STREAM_H__ */
```

5.69 RandomAccess.cpp File Reference

`#include "RandomAccess.h"`
Include dependency graph for RandomAccess.cpp:



Macros

- #define __RASPBERRY_IO_RANDOM_ACCESS_CPP__ 1

5.69.1 Macro Definition Documentation

5.69.1.1 #define __RASPBERRY_IO_RANDOM_ACCESS_CPP__ 1

Raspberry IO.

RandomAccess

Interface derived from DataInput, DataOutput, Closeable and Seekable.

Definition at line 10 of file RandomAccess.cpp.

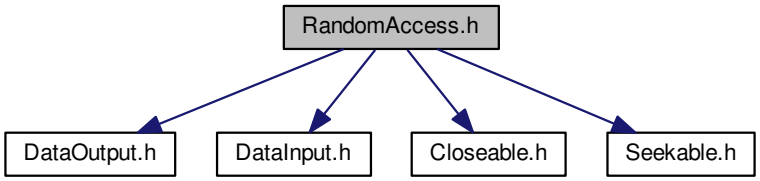
5.70 RandomAccess.cpp

```
00001
00009 #ifndef __RASPBERRY_IO_RANDOM_ACCESS_CPP__
00010 #define __RASPBERRY_IO_RANDOM_ACCESS_CPP__ 1
00011
00012 #include "RandomAccess.h"
00013
00014 #endif /* __RASPBERRY_IO_RANDOM_ACCESS_CPP__ */
```

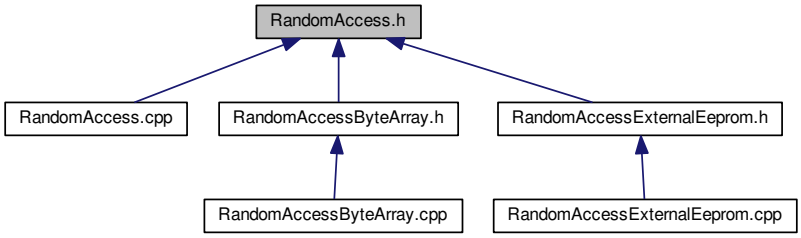
5.71 RandomAccess.h File Reference

```
#include <DataOutput.h>
#include <DataInput.h>
#include <Closeable.h>
#include <Seekable.h>
```

Include dependency graph for RandomAccess.h:



This graph shows which files directly or indirectly include this file:



Classes

- class RandomAccess

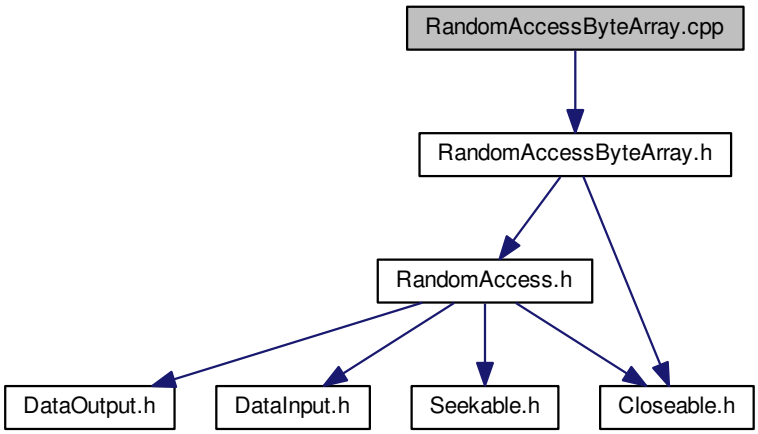
5.72 RandomAccess.h

```
00001
00009 #ifndef __RASPBERRY_IO_RANDOM_ACCESS_H__
00010 #define __RASPBERRY_IO_RANDOM_ACCESS_H__ 1
00011
00012 #include <DataOutput.h>
00013 #include <DataInput.h>
00014 #include <Closeable.h>
00015 #include <Seekable.h>
00016
00017 class RandomAccess: public virtual DataOutput,
00018         public virtual DataInput,
00019         public virtual Closeable,
00020         public virtual Seekable {
00021 };
00022
00023 #endif /* __RASPBERRY_IO_RANDOM_ACCESS_H__ */
```

5.73 RandomAccessByteArray.cpp File Reference

```
#include "RandomAccessByteArray.h"
```

Include dependency graph for RandomAccessByteArray.cpp:



Macros

- #define __RASPBERRY_IO_RANDOM_ACCESS_BYTE_ARRAY_CPP__ 1

5.73.1 Macro Definition Documentation

5.73.1.1 #define __RASPBERRY_IO_RANDOM_ACCESS_BYTE_ARRAY_CPP__ 1

Raspberry IO.

RandomAccessByteArray

Instances of this class support both reading and writing to a random access unsigned char array.

Definition at line 11 of file RandomAccessByteArray.cpp.

5.74 RandomAccessByteArray.cpp

```
00001
00010 #ifndef __RASPBERRY_IO_RANDOM_ACCESS_BYTE_ARRAY_CPP__
00011 #define __RASPBERRY_IO_RANDOM_ACCESS_BYTE_ARRAY_CPP__ 1
00012
00013 #include "RandomAccessByteArray.h"
00014
00015 RandomAccessByteArray::RandomAccessByteArray(unsigned char* buf
      ,
00016         unsigned int count) :
00017         buf(buf), count(count) {
00018     pos = 0;
00019 }
00020
00021 unsigned int RandomAccessByteArray::length() {
00022     return count;
00023 }
00024
00025 void RandomAccessByteArray::seek(unsigned int pos) {
00026     this->pos = pos;
00027 }
00028
00029 void RandomAccessByteArray::close() {
00030 }
00031
00032 void RandomAccessByteArray::write(unsigned char* b, int len) {
```

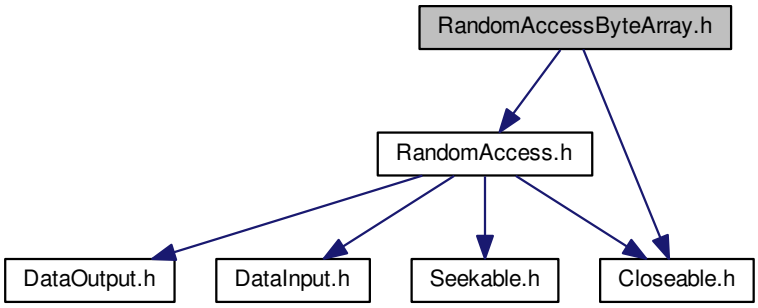
```
00033     writeBytes(b, len);
00034 }
00035
00036 void RandomAccessByteArray::write(unsigned char b) {
00037     buf[pos++] = b;
00038 }
00039
00040 void RandomAccessByteArray::writeByte(unsigned char b) {
00041     buf[pos++] = b;
00042 }
00043
00044 void RandomAccessByteArray::writeBytes(unsigned char* b, int len) {
00045     for (int i = 0; i < len; i++) {
00046         buf[pos++] = b[i];
00047     }
00048 }
00049
00050 void RandomAccessByteArray::writeBoolean(bool v) {
00051     buf[pos++] = (unsigned char) v;
00052 }
00053
00054 void RandomAccessByteArray::writeChar(char c) {
00055     buf[pos++] = (unsigned char) c;
00056 }
00057
00058 void RandomAccessByteArray::writeUnsignedChar(unsigned char c) {
00059     buf[pos++] = (unsigned char) c;
00060 }
00061
00062 void RandomAccessByteArray::writeInt(int v) {
00063     buf[pos++] = (unsigned char) ((v >> 8) & 0xff);
00064     buf[pos++] = (unsigned char) (v & 0xff);
00065 }
00066
00067 void RandomAccessByteArray::writeUnsignedInt(unsigned int v) {
00068     writeInt((int) v);
00069 }
00070
00071 void RandomAccessByteArray::writeLong(long v) {
00072     buf[pos++] = (unsigned char) ((v >> 24) & 0xff);
00073     buf[pos++] = (unsigned char) ((v >> 16) & 0xff);
00074     buf[pos++] = (unsigned char) ((v >> 8) & 0xff);
00075     buf[pos++] = (unsigned char) (v & 0xff);
00076 }
00077
00078 void RandomAccessByteArray::writeUnsignedLong(unsigned long v) {
00079     writeLong((long) v);
00080 }
00081
00082 void RandomAccessByteArray::writeFloat(float v) {
00083     writeLong((long) v);
00084 }
00085
00086 void RandomAccessByteArray::writeDouble(double v) {
00087     writeLong((long) v);
00088 }
00089
00090 unsigned char RandomAccessByteArray::readByte() {
00091     return buf[pos++];
00092 }
00093
00094 bool RandomAccessByteArray::readBoolean() {
00095     return (bool) buf[pos++];
00096 }
00097
00098 char RandomAccessByteArray::readChar() {
00099     return (char) buf[pos++];
00100 }
00101
00102 unsigned char RandomAccessByteArray::readUnsignedChar() {
00103     return (unsigned char) buf[pos++];
00104 }
00105
00106 int RandomAccessByteArray::readInt() {
00107     int v = 0;
00108     v = buf[pos++];
00109     v <<= 8;
00110     v |= buf[pos++];
00111     return v;
00112 }
00113
00114 unsigned int RandomAccessByteArray::readUnsignedInt() {
00115     return (unsigned int) readInt();
00116 }
00117
00118 long RandomAccessByteArray::readLong() {
00119     long v = 0;
```

```
00120     v = (buf[pos++] & 0xff);
00121     v <<= 8;
00122     v |= (buf[pos++] & 0xff);
00123     v <<= 8;
00124     v |= (buf[pos++] & 0xff);
00125     v <<= 8;
00126     v |= (buf[pos++] & 0xff);
00127     return v;
00128 }
00129
00130 unsigned long RandomAccessByteArray::readUnsignedLong() {
00131     return (unsigned long) readLong();
00132 }
00133
00134 float RandomAccessByteArray::readFloat() {
00135     return (float) readLong();
00136 }
00137
00138 double RandomAccessByteArray::readDouble() {
00139     return (double) readLong();
00140 }
00141
00142 void RandomAccessByteArray::readFully(unsigned char* b, int len) {
00143     for (int i = 0; i < len; i++) {
00144         b[i] = buf[pos++];
00145     }
00146 }
00147
00148 unsigned int RandomAccessByteArray::skipBytes(unsigned int n) {
00149     unsigned int skipped;
00150     unsigned int newpos;
00151     newpos = pos + n;
00152     if (newpos > count) {
00153         newpos = count;
00154     }
00155     skipped = newpos - pos;
00156     pos = newpos;
00157     return skipped;
00158 }
00159
00160 #endif /* __RASPBERRY_IO_RANDOM_ACCESS_BYTE_ARRAY_CPP__ */
```

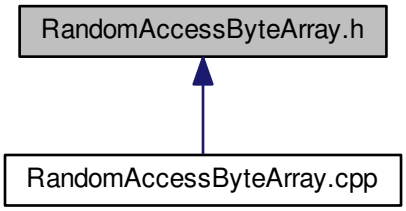
5.75 RandomAccessByteArray.h File Reference

```
#include <RandomAccess.h>
#include <Closeable.h>
```

Include dependency graph for RandomAccessByteArray.h:



This graph shows which files directly or indirectly include this file:



Classes

- class RandomAccessByteArray

5.76 RandomAccessByteArray.h

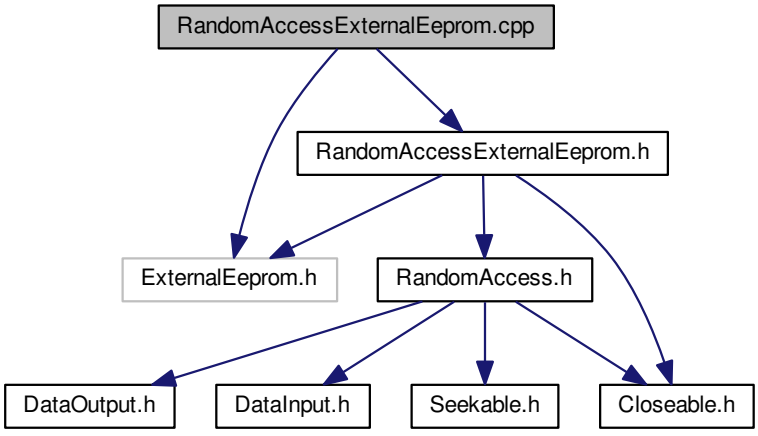
```
00001
00010 #ifndef __RASPBERRY_IO_RANDOM_ACCESS_BYTE_ARRAY_H__
00011 #define __RASPBERRY_IO_RANDOM_ACCESS_BYTE_ARRAY_H__ 1
00012
00013 #include <RandomAccess.h>
00014 #include <Closeable.h>
00015
00016 class RandomAccessByteArray: public RandomAccess, public virtual
      Closeable {
00017
00021     unsigned char* buf;
00022
00026     unsigned int count;
00027
00031     unsigned int pos;
00032
00033 public:
00034
00041     RandomAccessByteArray(unsigned char* buf, unsigned int count);
00042
00048     virtual void seek(unsigned int pos);
00049
00055     unsigned int length();
00056
00060     virtual void close();
00061
00068     virtual void write(unsigned char* b, int len);
00069
00075     virtual void write(unsigned char b);
00076
00082     virtual void writeByte(unsigned char b);
00083
00090     virtual void writeBytes(unsigned char* b, int len);
00091
00097     virtual void writeBoolean(bool v);
00098
00104     virtual void writeChar(char c);
00105
00111     virtual void writeUnsignedChar(unsigned char c);
00112
00118     virtual void writeInt(int v);
00119
00125     virtual void writeUnsignedInt(unsigned int v);
00126
00132     virtual void writeLong(long v);
00133
00139     virtual void writeUnsignedLong(unsigned long v);
00140
00146     virtual void writeFloat(float v);
00147
00153     virtual void writeDouble(double v);
```

```
00154
00160     virtual unsigned char readByte();
00161
00167     virtual bool readBoolean();
00168
00174     virtual char readChar();
00175
00181     virtual unsigned char readUnsignedChar();
00182
00188     virtual int readInt();
00189
00195     virtual unsigned int readUnsignedInt();
00196
00202     virtual long readLong();
00203
00209     virtual unsigned long readUnsignedLong();
00210
00216     virtual float readFloat();
00217
00223     virtual double readDouble();
00224
00231     virtual void readFully(unsigned char* b, int len);
00232
00239     virtual unsigned int skipBytes(unsigned int n);
00240 };
00241 #endif /* __RASPBERRY_IO_RANDOM_ACCESS_BYTE_ARRAY_H__ */
```

5.77 RandomAccessExternalEeprom.cpp File Reference

```
#include <ExternalEeprom.h>
#include "RandomAccessExternalEeprom.h"
```

Include dependency graph for RandomAccessExternalEeprom.cpp:



Macros

- #define __RASPBERRY_IO_RANDOM_ACCESS_EXTERNAL_EEPROM_CPP__ 1

5.77.1 Macro Definition Documentation

5.77.1.1 #define __RASPBERRY_IO_RANDOM_ACCESS_EXTERNAL_EEPROM_CPP__ 1

Raspberry IO.

RandomAccessExternalEeprom

Instances of this class support both reading and writing to a random access externalEeprom. A random access externalEeprom behaves like a large array of bytes stored in the externalEeprom system.

Definition at line 12 of file RandomAccessExternalEeprom.cpp.

5.78 RandomAccessExternalEeprom.cpp

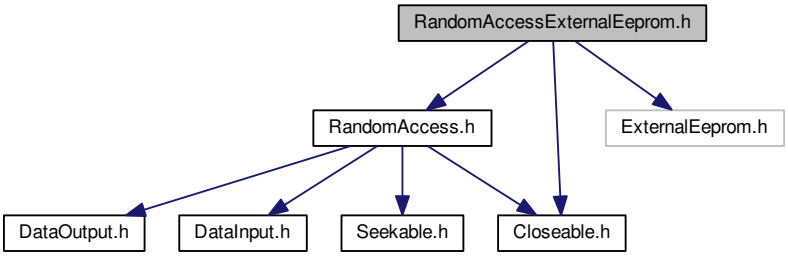
```
00001
00011 #ifndef __RASPBERRY_IO_RANDOM_ACCESS_EXTERNAL_EEPROM_CPP__
00012 #define __RASPBERRY_IO_RANDOM_ACCESS_EXTERNAL_EEPROM_CPP__ 1
00013
00014 #include <ExternalEeprom.h>
00015 #include "RandomAccessExternalEeprom.h"
00016
00017 RandomAccessExternalEeprom::RandomAccessExternalEeprom
      (
00018         ExternalEeprom* externalEeprom) :
00019         externalEeprom(externalEeprom) {
00020     pos = 0;
00021 }
00022
00023 unsigned int RandomAccessExternalEeprom::length() {
00024     return (unsigned int) externalEeprom->getDeviceSize();
00025 }
00026
00027 void RandomAccessExternalEeprom::seek(unsigned int pos) {
00028     this->pos = pos;
00029 }
00030
00031 void RandomAccessExternalEeprom::close() {
00032 }
00033
00034 void RandomAccessExternalEeprom::write(unsigned char* b, int len) {
00035     writeBytes(b, len);
00036 }
00037
00038 void RandomAccessExternalEeprom::write(unsigned char b) {
00039     writeByte(b);
00040 }
00041
00042 void RandomAccessExternalEeprom::writeByte(unsigned char b) {
00043     externalEeprom->write(pos++, b);
00044 }
00045
00046 void RandomAccessExternalEeprom::writeBytes(unsigned char* b, int len
      ) {
00047     for (int i = 0; i < len; i++) {
00048         externalEeprom->write(pos++, b[i]);
00049     }
00050 }
00051
00052 void RandomAccessExternalEeprom::writeBoolean(bool v) {
00053     externalEeprom->write(pos++, (unsigned char) v);
00054 }
00055
00056 void RandomAccessExternalEeprom::writeChar(char c) {
00057     externalEeprom->write(pos++, (unsigned char) c);
00058 }
00059
00060 void RandomAccessExternalEeprom::writeUnsignedChar(unsigned
      char c) {
00061     externalEeprom->write(pos++, (unsigned char) c);
00062 }
00063
00064 void RandomAccessExternalEeprom::writeInt(int v) {
00065     externalEeprom->write(pos++, (unsigned char) ((v >> 8) & 0xff));
00066     externalEeprom->write(pos++, (unsigned char) (v & 0xff));
00067 }
00068
00069 void RandomAccessExternalEeprom::writeUnsignedInt(unsigned int
      v) {
00070     writeInt((int) v);
00071 }
00072
00073 void RandomAccessExternalEeprom::writeLong(long v) {
00074     externalEeprom->write(pos++, (unsigned char) ((v >> 24) & 0xff));
00075     externalEeprom->write(pos++, (unsigned char) ((v >> 16) & 0xff));
00076     externalEeprom->write(pos++, (unsigned char) ((v >> 8) & 0xff));
00077     externalEeprom->write(pos++, (unsigned char) (v & 0xff));
00078 }
00079
00080 void RandomAccessExternalEeprom::writeUnsignedLong(unsigned
      long v) {
00081     writeLong((long) v);
```

```
00082 }
00083
00084 void RandomAccessExternalEeprom::writeFloat(float v) {
00085     writeLong((long) v);
00086 }
00087
00088 void RandomAccessExternalEeprom::writeDouble(double v) {
00089     writeLong((long) v);
00090 }
00091
00092 unsigned char RandomAccessExternalEeprom::readByte() {
00093     return (unsigned char) externalEeprom->read(pos++);
00094 }
00095
00096 bool RandomAccessExternalEeprom::readBoolean() {
00097     return (bool) externalEeprom->read(pos++);
00098 }
00099
00100 char RandomAccessExternalEeprom::readChar() {
00101     return (char) externalEeprom->read(pos++);
00102 }
00103
00104 unsigned char RandomAccessExternalEeprom::readUnsignedChar() {
00105     return (unsigned char) externalEeprom->read(pos++);
00106 }
00107
00108 int RandomAccessExternalEeprom::readInt() {
00109     int v = 0;
00110     v = externalEeprom->read(pos++);
00111     v <<= 8;
00112     v |= (externalEeprom->read(pos++) & 0xff);
00113     return v;
00114 }
00115
00116 unsigned int RandomAccessExternalEeprom::readUnsignedInt() {
00117     return (unsigned int) readInt();
00118 }
00119
00120 long RandomAccessExternalEeprom::readLong() {
00121     long v = 0;
00122     v = externalEeprom->read(pos++);
00123     v <<= 8;
00124     v |= (externalEeprom->read(pos++) & 0xff);
00125     v <<= 8;
00126     v |= (externalEeprom->read(pos++) & 0xff);
00127     v <<= 8;
00128     v |= (externalEeprom->read(pos++) & 0xff);
00129     return v;
00130 }
00131
00132 unsigned long RandomAccessExternalEeprom::readUnsignedLong() {
00133     return (unsigned long) readLong();
00134 }
00135
00136 float RandomAccessExternalEeprom::readFloat() {
00137     return (float) readLong();
00138 }
00139
00140 double RandomAccessExternalEeprom::readDouble() {
00141     return (double) readLong();
00142 }
00143
00144 void RandomAccessExternalEeprom::readFully(unsigned char* b, int len)
      {
00145     for (int i = 0; i < len; i++) {
00146         b[i] = externalEeprom->read(pos++);
00147     }
00148 }
00149
00150 unsigned int RandomAccessExternalEeprom::skipBytes(unsigned int n) {
00151     unsigned int skipped;
00152     unsigned int newpos;
00153     newpos = pos + n;
00154     if (newpos > length()) {
00155         newpos = length();
00156     }
00157     skipped = newpos - pos;
00158     pos = newpos;
00159     return skipped;
00160 }
00161
00162 #endif /* __RASPBERRY_IO_RANDOM_ACCESS_EXTERNAL_EEPROM_CPP__ */
```

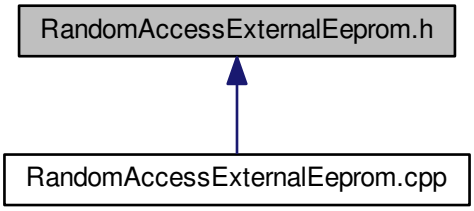
5.79 RandomAccessExternalEeprom.h File Reference

```
#include <RandomAccess.h>
#include <Closeable.h>
#include <ExternalEeprom.h>
```

Include dependency graph for RandomAccessExternalEeprom.h:



This graph shows which files directly or indirectly include this file:



Classes

- class RandomAccessExternalEeprom

5.80 RandomAccessExternalEeprom.h

```
00001
00011 #ifndef __RASPBERRY_IO_RANDOM_ACCESS_EXTERNAL_EEPROM_H__
00012 #define __RASPBERRY_IO_RANDOM_ACCESS_EXTERNAL_EEPROM_H__ 1
00013
00014 #include <RandomAccess.h>
00015 #include <Closeable.h>
00016 #include <ExternalEeprom.h>
00017
00018 class RandomAccessExternalEeprom: public RandomAccess, public virtual
       Closeable {
00019
00023     ExternalEeprom* externalEeprom;
00024
00028     unsigned int pos;
00029
00030 public:
00031
00037     RandomAccessExternalEeprom(ExternalEeprom* externalEeprom);
```

```
00038
00044     virtual void seek(unsigned int pos);
00045
00051     unsigned int length();
00052
00056     virtual void close();
00057
00064     virtual void write(unsigned char* b, int len);
00065
00071     virtual void write(unsigned char b);
00072
00078     virtual void writeByte(unsigned char b);
00079
00086     virtual void writeBytes(unsigned char* b, int len);
00087
00093     virtual void writeBoolean(bool v);
00094
00100     virtual void writeChar(char c);
00101
00107     virtual void writeUnsignedChar(unsigned char c);
00108
00114     virtual void writeInt(int v);
00115
00121     virtual void writeUnsignedInt(unsigned int v);
00122
00128     virtual void writeLong(long v);
00129
00135     virtual void writeUnsignedLong(unsigned long v);
00136
00142     virtual void writeFloat(float v);
00143
00149     virtual void writeDouble(double v);
00150
00156     virtual unsigned char readByte();
00157
00163     virtual bool readBoolean();
00164
00170     virtual char readChar();
00171
00177     virtual unsigned char readUnsignedChar();
00178
00184     virtual int readInt();
00185
00191     virtual unsigned int readUnsignedInt();
00192
00198     virtual long readLong();
00199
00205     virtual unsigned long readUnsignedLong();
00206
00212     virtual float readFloat();
00213
00219     virtual double readDouble();
00220
00227     virtual void readFully(unsigned char* b, int len);
00228
00235     virtual unsigned int skipBytes(unsigned int n);
00236 };
00237 #endif /* __RASPBERRY_IO_RANDOM_ACCESS_EXTERNAL_EEPROM_H__ */
```

5.81 RandomAccessResource.cpp File Reference

Macros

- #define __RASPBERRY_IO_RANDOM_ACCESS_RESOURCE_CPP__ 1

5.81.1 Macro Definition Documentation

5.81.1.1 #define __RASPBERRY_IO_RANDOM_ACCESS_RESOURCE_CPP__ 1

Raspberry IO.

RandomAccessResource

Instances of this class support both reading and writing to a random access resource. A random access resource behaves like a large array of bytes stored in the resource system.

Definition at line 12 of file RandomAccessResource.cpp.

5.82 RandomAccessResource.cpp

```
00001
00011 #ifndef __RASPBERRY_IO_RANDOM_ACCESS_RESOURCE_CPP__
00012 #define __RASPBERRY_IO_RANDOM_ACCESS_RESOURCE_CPP__ 1
00013
00014 #if USING_RESOURCE_LIBRARIES
00015
00016 #include "RandomAccessResource.h"
00017
00018 RandomAccessResource::RandomAccessResource(Resource* resource) : resource(resource) {
00019 }
00020
00021 unsigned int RandomAccessResource::length() {
00022     return (unsigned int) resource->size();
00023 }
00024
00025 void RandomAccessResource::seek(unsigned int pos) {
00026     resource->seek(0, pos);
00027 }
00028
00029 void RandomAccessResource::close() {
00030     resource->close();
00031 }
00032
00033 void RandomAccessResource::write(unsigned char* b, int len) {
00034     writeBytes(b, len);
00035 }
00036
00037 void RandomAccessResource::write(unsigned char b) {
00038     writeByte(b);
00039 }
00040
00041 void RandomAccessResource::writeByte(unsigned char b) {
00042     resource->write(b);
00043 }
00044
00045 void RandomAccessResource::writeBytes(unsigned char* b, int len) {
00046     for (int i = 0; i < len; i++) {
00047         resource->write(b[i]);
00048     }
00049 }
00050
00051 void RandomAccessResource::writeBoolean(bool v) {
00052     resource->write((unsigned char) v);
00053 }
00054
00055 void RandomAccessResource::writeChar(char c) {
00056     resource->write((unsigned char) c);
00057 }
00058
00059 void RandomAccessResource::writeUnsignedChar(unsigned char c) {
00060     resource->write((unsigned char) c);
00061 }
00062
00063 void RandomAccessResource::writeInt(int v) {
00064     resource->write((unsigned char) ((v >> 8) & 0xff));
00065     resource->write((unsigned char) (v & 0xff));
00066 }
00067
00068 void RandomAccessResource::writeUnsignedInt(unsigned int v) {
00069     writeInt((int) v);
00070 }
00071
00072 void RandomAccessResource::writeWord(word v) {
00073     writeInt((int) v);
00074 }
00075
00076 void RandomAccessResource::writeLong(long v) {
00077     resource->write((unsigned char) ((v >> 24) & 0xff));
00078     resource->write((unsigned char) ((v >> 16) & 0xff));
00079     resource->write((unsigned char) ((v >> 8) & 0xff));
00080     resource->write((unsigned char) (v & 0xff));
00081 }
00082
00083 void RandomAccessResource::writeUnsignedLong(unsigned long v) {
00084     writeLong((long) v);
00085 }
00086
00087 void RandomAccessResource::writeFloat(float v) {
00088     writeLong((long) v);
00089 }
00090
00091 void RandomAccessResource::writeDouble(double v) {
00092     writeLong((long) v);
00093 }
00094
```

```
00095 unsigned char RandomAccessResource::readByte() {
00096     return (unsigned char) resource->read();
00097 }
00098
00099 bool RandomAccessResource::readBoolean() {
00100     return (bool) resource->read();
00101 }
00102
00103 char RandomAccessResource::readChar() {
00104     return (char) resource->read();
00105 }
00106
00107 unsigned char RandomAccessResource::readUnsignedChar() {
00108     return (unsigned char) resource->read();
00109 }
00110
00111 int RandomAccessResource::readInt() {
00112     int v = 0;
00113     v = resource->read();
00114     v <<= 8;
00115     v |= (resource->read() & 0xff);
00116     return v;
00117 }
00118
00119 unsigned int RandomAccessResource::readUnsignedInt() {
00120     return (unsigned int) readInt();
00121 }
00122
00123 word RandomAccessResource::readWord() {
00124     return (word) readInt();
00125 }
00126
00127 long RandomAccessResource::readLong() {
00128     long v = 0;
00129     v = resource->read();
00130     v <<= 8;
00131     v |= (resource->read() & 0xff);
00132     v <<= 8;
00133     v |= (resource->read() & 0xff);
00134     v <<= 8;
00135     v |= (resource->read() & 0xff);
00136     return v;
00137 }
00138
00139 unsigned long RandomAccessResource::readUnsignedLong() {
00140     return (unsigned long) readLong();
00141 }
00142
00143 float RandomAccessResource::readFloat() {
00144     return (float) readLong();
00145 }
00146
00147 double RandomAccessResource::readDouble() {
00148     return (double) readLong();
00149 }
00150
00151 void RandomAccessResource::readFully(unsigned char* b, int len) {
00152     for (int i = 0; i < len; i++) {
00153         b[i] = resource->read();
00154     }
00155 }
00156
00157 unsigned int RandomAccessResource::skipBytes(unsigned int n) {
00158     unsigned int pos;
00159     unsigned int len;
00160     unsigned int newpos;
00161     pos = (unsigned int) resource->tell();
00162     len = resource->size();
00163     newpos = pos + n;
00164     if (newpos > len) {
00165         newpos = len;
00166     }
00167     seek(newpos);
00168     return (unsigned int) (newpos - pos);
00169 }
00170
00171 #endif /* USING_RESOURCE_LIBRARIES */
00172
00173 #endif /* __RASPBERRY_IO_RANDOM_ACCESS_RESOURCE_CPP__ */
```

5.83 RandomAccessResource.h File Reference

5.84 RandomAccessResource.h

```
00001
00011 #ifndef __RASPBERRY_IO_RANDOM_ACCESS_RESOURCE_H__
00012 #define __RASPBERRY_IO_RANDOM_ACCESS_RESOURCE_H__ 1
00013
00014 #if USING_RESOURCE_LIBRARIES
00015
00016 #include <RandomAccess.h>
00017 #include <Closeable.h>
00018 #include <Resource.h>
00019
00020 class RandomAccessResource : public RandomAccess {
00021
00025     Resource* resource;
00026
00027 public:
00028
00034     RandomAccessResource(Resource* resource);
00035
00041     virtual void seek(unsigned int pos);
00042
00048     unsigned int length();
00049
00053     virtual void close();
00054
00061     virtual void write(unsigned char* b, int len);
00062
00068     virtual void write(unsigned char b);
00069
00075     virtual void writeByte(unsigned char b);
00076
00083     virtual void writeBytes(unsigned char* b, int len);
00084
00090     virtual void writeBoolean(bool v);
00091
00097     virtual void writeChar(char c);
00098
00104     virtual void writeUnsignedChar(unsigned char c);
00105
00111     virtual void writeInt(int v);
00112
00118     virtual void writeUnsignedInt(unsigned int v);
00119
00125     virtual void writeWord(word v);
00126
00132     virtual void writeLong(long v);
00133
00139     virtual void writeUnsignedLong(unsigned long v);
00140
00146     virtual void writeFloat(float v);
00147
00153     virtual void writeDouble(double v);
00154
00160     virtual unsigned char readByte();
00161
00167     virtual bool readBoolean();
00168
00174     virtual char readChar();
00175
00181     virtual unsigned char readUnsignedChar();
00182
00188     virtual int readInt();
00189
00195     virtual unsigned int readUnsignedInt();
00196
00202     virtual word readWord();
00203
00209     virtual long readLong();
00210
00216     virtual unsigned long readUnsignedLong();
00217
00223     virtual float readFloat();
00224
00230     virtual double readDouble();
00231
00238     virtual void readFully(unsigned char* b, int len);
00239
00246     virtual unsigned int skipBytes(unsigned int n);
00247 };
00248
00249 #endif /* USING_RESOURCE_LIBRARIES */
00250
00251 #endif /* __RASPBERRY_IO_RANDOM_ACCESS_RESOURCE_H__ */
```

5.85 Raspberry.h File Reference

5.86 Raspberry.h

5.87 ResourceInputStream.cpp File Reference

Macros

- #define __RASPBERRY_IO_RESOURCE_INPUT_STREAM_CPP__ 1

5.87.1 Macro Definition Documentation

5.87.1.1 #define __RASPBERRY_IO_RESOURCE_INPUT_STREAM_CPP__ 1

Raspberry IO.

ResourceInputStream

A ResourceInputStream obtains input bytes from a resource in a file system.

Definition at line 10 of file ResourceInputStream.cpp.

5.88 ResourceInputStream.cpp

```
00001
00009 #ifndef __RASPBERRY_IO_RESOURCE_INPUT_STREAM_CPP__
00010 #define __RASPBERRY_IO_RESOURCE_INPUT_STREAM_CPP__ 1
00011
00012 #if USING_RESOURCE_LIBRARIES
00013
00014 #include "ResourceInputStream.h"
00015
00016 ResourceInputStream::ResourceInputStream(Resource* resource) : resource(resource) {
00017     markpos = 0;
00018     pos = 0;
00019     resourceSize = resource->size();
00020     resource->rewind();
00021 }
00022
00023 int ResourceInputStream::available() {
00024     if ((resourceSize - pos) > 0) {
00025         return 1;
00026     }
00027     return 0;
00028 }
00029
00030 void ResourceInputStream::close() {
00031     resource->close();
00032 }
00033
00034 void ResourceInputStream::mark() {
00035     markpos = pos;
00036 }
00037
00038 bool ResourceInputStream::markSupported() {
00039     return true;
00040 }
00041
00042 int ResourceInputStream::read() {
00043     if (resource->eor()) {
00044         pos = resourceSize;
00045         return -1;
00046     }
00047     pos++;
00048     return (int) resource->read();
00049 }
00050
00051 void ResourceInputStream::reset() {
00052     resource->seek((Resource::ResourceSeekOrigin)0, markpos);
00053 }
00054
```

```
00055 #endif /* USING_RESOURCE_LIBRARIES */
00056
00057 #endif /* __RASPBERRY_IO_RESOURCE_INPUT_STREAM_CPP__ */
```

5.89 ResourceInputStream.h File Reference

5.90 ResourceInputStream.h

```
00001
00009 #ifndef __RASPBERRY_IO_RESOURCE_INPUT_STREAM_H__
00010 #define __RASPBERRY_IO_RESOURCE_INPUT_STREAM_H__ 1
00011
00012 #if USING_RESOURCE_LIBRARIES
00013
00014 #include <InputStream.h>
00015 #include <Resource.h>
00016
00017 class ResourceInputStream : public virtual InputStream {
00018 protected:
00019
00020     /*
00021      * The resource where data is stored.
00022      */
00023     Resource* resource;
00024
00025     /*
00026      * Current position
00027      */
00028     unsigned int pos;
00029
00030     /*
00031      * The currently marked position in the stream.
00032      */
00033     unsigned int markpos;
00034
00035     /*
00036      * The size of the resource.
00037      */
00038     unsigned int resourceSize;
00039
00040 public:
00041
00042     ResourceInputStream(Resource* resource);
00043
00049     virtual int available();
00050
00055     virtual void close();
00056
00060     virtual void mark();
00061
00065     virtual bool markSupported();
00066
00070     using InputStream::read;
00071
00075     virtual int read();
00076
00081     virtual void reset();
00082 };
00083
00084 #endif /* USING_RESOURCE_LIBRARIES */
00085
00086 #endif /* __RASPBERRY_IO_RESOURCE_INPUT_STREAM_H__ */
```

5.91 ResourceOutputStream.cpp File Reference

Macros

- #define __RASPBERRY_IO_RESOURCE_OUTPUT_STREAM_CPP__ 1

5.91.1 Macro Definition Documentation

5.91.1.1 #define __RASPBERRY_IO_RESOURCE_OUTPUT_STREAM_CPP__ 1

Raspberry IO.

ResourceOutputStream

A resource output stream is an output stream for writing data to a Resource.

Definition at line 10 of file ResourceOutputStream.cpp.

5.92 ResourceOutputStream.cpp

```
00001
00009 #ifndef __RASPBERRY_IO_RESOURCE_OUTPUT_STREAM_CPP__
00010 #define __RASPBERRY_IO_RESOURCE_OUTPUT_STREAM_CPP__ 1
00011
00012 #if USING_RESOURCE_LIBRARIES
00013
00014 #include "ResourceOutputStream.h"
00015
00016 ResourceOutputStream::ResourceOutputStream(Resource* resource) : resource(resource) {
00017 }
00018
00019 void ResourceOutputStream::close() {
00020     resource->close();
00021 }
00022
00023 void ResourceOutputStream::write(unsigned char b) {
00024     resource->write(b);
00025 }
00026
00027 #endif /* USING_RESOURCE_LIBRARIES */
00028
00029 #endif /* __RASPBERRY_IO_RESOURCE_OUTPUT_STREAM_CPP__ */
```

5.93 ResourceOutputStream.h File Reference

5.94 ResourceOutputStream.h

```
00001
00009 #ifndef __RASPBERRY_IO_RESOURCE_OUTPUT_STREAM_H__
00010 #define __RASPBERRY_IO_RESOURCE_OUTPUT_STREAM_H__ 1
00011
00012 #if USING_RESOURCE_LIBRARIES
00013
00014 #include <OutputStream.h>
00015 #include <Resource.h>
00016
00017 class ResourceOutputStream : public OutputStream {
00018 protected:
00019
00020     /*
00021      * The resource where data is stored.
00022      */
00023     Resource* resource;
00024
00025 public:
00026
00027     ResourceOutputStream(Resource* resource);
00028
00032     virtual void close();
00033
00037     using OutputStream::write;
00038
00042     virtual void write(unsigned char b);
00043 };
00044
00045 #endif /* USING_RESOURCE_LIBRARIES */
00046
00047 #endif /* __RASPBERRY_IO_RESOURCE_OUTPUT_STREAM_H__ */
```

5.95 ResourceSeekableInputStream.cpp File Reference

Macros

- #define __RASPBERRY_IO_RESOURCE_SEEKABLE_INPUT_STREAM_CPP__ 1

5.95.1 Macro Definition Documentation

5.95.1.1 #define __RASPBERRY_IO_RESOURCE_SEEKABLE_INPUT_STREAM_CPP__ 1

Raspberry IO.

ResourceSeekableInputStream

A ResourceSeekableInputStream obtains input bytes from a resource in a file system that implements SeekableInputStream interface.

Definition at line 11 of file ResourceSeekableInputStream.cpp.

5.96 ResourceSeekableInputStream.cpp

```
00001
00010 #ifndef __RASPBERRY_IO_RESOURCE_SEEKABLE_INPUT_STREAM_CPP__
00011 #define __RASPBERRY_IO_RESOURCE_SEEKABLE_INPUT_STREAM_CPP__ 1
00012
00013 #if USING_RESOURCE_LIBRARIES
00014
00015 #include "ResourceSeekableInputStream.h"
00016
00017 ResourceSeekableInputStream::ResourceSeekableInputStream(Resource* resource) : ResourceInputStream(resource
      ) {
00018 }
00019
00020 void ResourceSeekableInputStream::seek(unsigned int pos) {
00021     resource->seek((Resource::ResourceSeekOrigin)0, pos);
00022 }
00023
00024 #endif /* USING_RESOURCE_LIBRARIES */
00025
00026 #endif /* __RASPBERRY_IO_RESOURCE_SEEKABLE_INPUT_STREAM_CPP__ */
```

5.97 ResourceSeekableInputStream.h File Reference

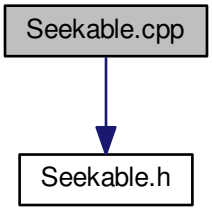
5.98 ResourceSeekableInputStream.h

```
00001
00010 #ifndef __RASPBERRY_IO_RESOURCE_SEEKABLE_INPUT_STREAM_H__
00011 #define __RASPBERRY_IO_RESOURCE_SEEKABLE_INPUT_STREAM_H__ 1
00012
00013 #if USING_RESOURCE_LIBRARIES
00014
00015 #include <SeekableInputStream.h>
00016 #include <ResourceInputStream.h>
00017 #include <Resource.h>
00018
00019 class ResourceSeekableInputStream : public ResourceInputStream, public
      SeekableInputStream {
00020 public:
00021
00027     ResourceSeekableInputStream(Resource* resource);
00028
00034     virtual void seek(unsigned int pos);
00035 };
00036
00037 #endif /* USING_RESOURCE_LIBRARIES */
00038
00039 #endif /* __RASPBERRY_IO_RESOURCE_SEEKABLE_INPUT_STREAM_H__ */
```

5.99 Seekable.cpp File Reference

```
#include "Seekable.h"
```

Include dependency graph for Seekable.cpp:



Macros

- #define __RASPBERRY_IO_SEEKABLE_CPP__ 1

5.99.1 Macro Definition Documentation

5.99.1.1 #define __RASPBERRY_IO_SEEKABLE_CPP__ 1

Raspberry IO.

Seekable

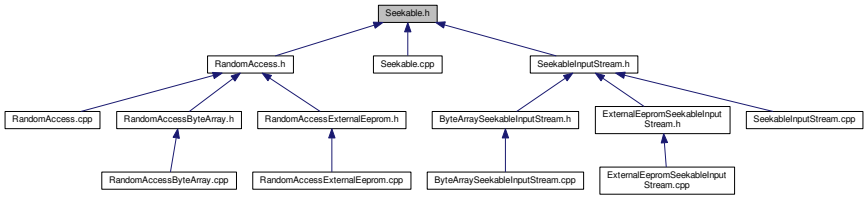
Definition at line 8 of file Seekable.cpp.

5.100 Seekable.cpp

```
00001
00007 #ifndef __RASPBERRY_IO_SEEKABLE_CPP__
00008 #define __RASPBERRY_IO_SEEKABLE_CPP__ 1
00009
00010 #include "Seekable.h"
00011
00012 #endif /* __RASPBERRY_IO_SEEKABLE_CPP__ */
```

5.101 Seekable.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

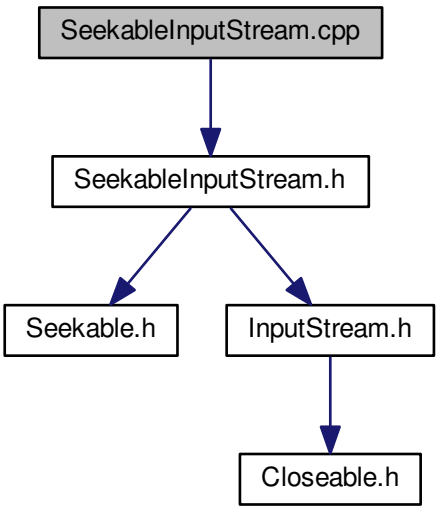
- class Seekable

5.102 Seekable.h

```
00001
00007 #ifndef __RASPBERRY_IO_SEEKABLE_H__
00008 #define __RASPBERRY_IO_SEEKABLE_H__ 1
00009
00010 class Seekable {
00011 public:
00012
00013     virtual ~Seekable() {
00014     }
00015
00016     virtual void seek(unsigned int pos) = 0;
00017 };
00018
00019 #endif /* __RASPBERRY_IO_SEEKABLE_H__ */
```

5.103 SeekableInputStream.cpp File Reference

`#include "SeekableInputStream.h"`
Include dependency graph for SeekableInputStream.cpp:



Macros

- #define __RASPBERRY_IO_SEEKABLE_INPUT_STREAM_CPP__ 1

5.103.1 Macro Definition Documentation

5.103.1.1 #define __RASPBERRY_IO_SEEKABLE_INPUT_STREAM_CPP__ 1

Raspberry IO.

SeekableInputStream

Definition at line 8 of file SeekableInputStream.cpp.

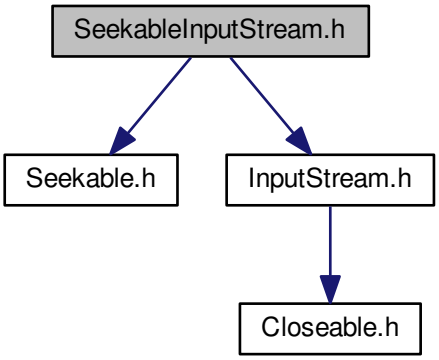
5.104 SeekableInputStream.cpp

```
00001
00007 #ifndef __RASPBERRY_IO_SEEKABLE_INPUT_STREAM_CPP__
00008 #define __RASPBERRY_IO_SEEKABLE_INPUT_STREAM_CPP__ 1
00009
00010 #include "SeekableInputStream.h"
00011
00012 #endif /* __RASPBERRY_IO_SEEKABLE_INPUT_STREAM_CPP__ */
```

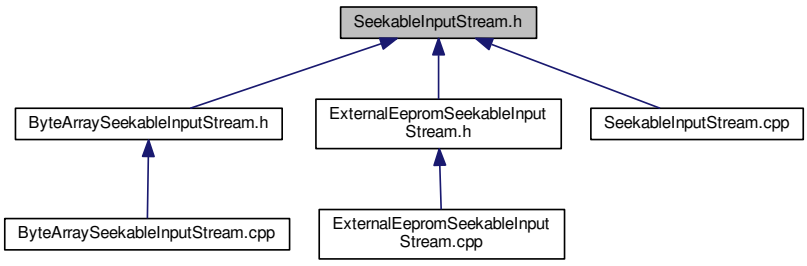
5.105 SeekableInputStream.h File Reference

```
#include <Seekable.h>
#include <InputStream.h>
```

Include dependency graph for SeekableInputStream.h:



This graph shows which files directly or indirectly include this file:



Classes

- class SeekableInputStream

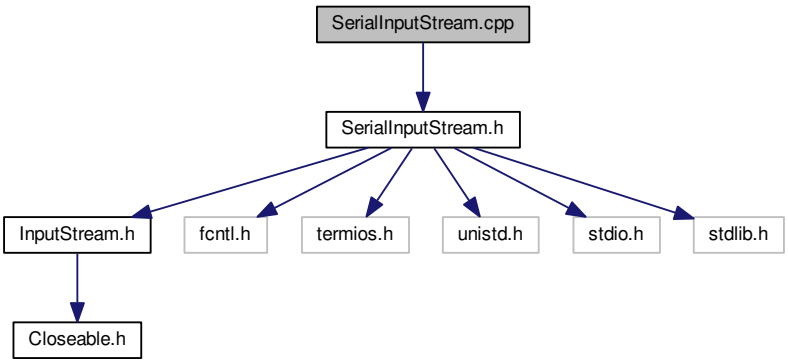
5.106 SeekableInputStream.h

```
00001
```

```
00007 #ifndef __RASPBERRY_IO_SEEKABLE_INPUT_STREAM_H__
00008 #define __RASPBERRY_IO_SEEKABLE_INPUT_STREAM_H__ 1
00009
00010 #include <Seekable.h>
00011 #include <InputStream.h>
00012
00013 class SeekableInputStream: public virtual Seekable, public virtual
      InputStream {
00014 public:
00015
00016 };
00017
00018 #endif /* __RASPBERRY_IO_SEEKABLE_INPUT_STREAM_H__ */
```

5.107 SerialInputStream.cpp File Reference

`#include "SerialInputStream.h"`
Include dependency graph for SerialInputStream.cpp:



Macros

- #define __RASPBERRY_IO_SERIAL_INPUT_STREAM_CPP__ 1

5.107.1 Macro Definition Documentation

5.107.1.1 #define __RASPBERRY_IO_SERIAL_INPUT_STREAM_CPP__ 1

Raspberry IO.

SerialInputStream

A SerialInputStream obtains input bytes from a serial port.

Definition at line 10 of file SerialInputStream.cpp.

5.108 SerialInputStream.cpp

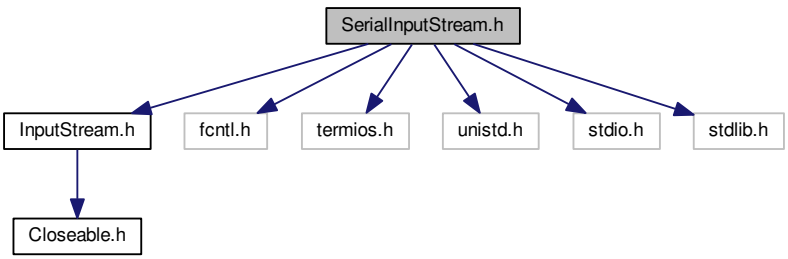
```
00001
00009 #ifndef __RASPBERRY_IO_SERIAL_INPUT_STREAM_CPP__
00010 #define __RASPBERRY_IO_SERIAL_INPUT_STREAM_CPP__ 1
00011
00012 #include "SerialInputStream.h"
00013
00014 SerialInputStream::SerialInputStream(const char *dev,
      BoudRate boudRate) {
00015     struct termios options;
```

```
00016     tmp = -1;
00017     fd = open(dev, O_RDONLY | O_NOCTTY | O_NONBLOCK);
00018     if (fd == -1) {
00019         perror("Unable to open port.");
00020         exit(1);
00021     }
00022     tcgetattr(fd, &options);
00023     cfsetispeed(&options, (speed_t)boudRate);
00024     cfsetospeed(&options, (speed_t)boudRate);
00025     options.c_cflag |= (CLOCAL | CREAD);
00026     options.c_cflag |= PARENB;
00027     options.c_cflag |= PARODD;
00028     options.c_cflag &= ~CSTOPB;
00029     options.c_cflag &= ~CSIZE;
00030     options.c_cflag |= CS8;
00031     options.c_iflag |= (INPCK | ISTRIP);
00032     tcsetattr(fd, TCSANOW, &options);
00033     fcntl(fd, F_SETFL, FNDELAY);
00034 }
00035
00036 int SerialInputStream::available() {
00037     int b;
00038     int n = ::read(fd, &b, 1);
00039     if (n <= 0) {
00040         b = -1;
00041         return 0;
00042     }
00043     return 1;
00044 }
00045
00046 int SerialInputStream::read() {
00047     int b;
00048     if (tmp != -1) {
00049         b = tmp;
00050         tmp = -1;
00051     } else {
00052         int n = ::read(fd, &b, 1);
00053         if (n <= 0) {
00054             return -1;
00055         }
00056     }
00057     return b;
00058 }
00059
00060 int SerialInputStream::read(unsigned char* b, int len) {
00061     return (int) ::read (fd, b, len);
00062 }
00063
00064 #endif /* __RASPBERRY_IO_SERIAL_INPUT_STREAM_CPP__ */
```

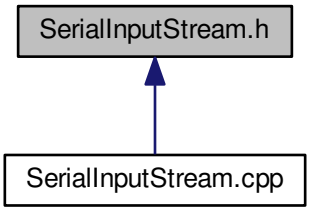
5.109 SerialInputStream.h File Reference

```
#include <InputStream.h>
#include <fcntl.h>
#include <termios.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
```

Include dependency graph for SerialInputStream.h:



This graph shows which files directly or indirectly include this file:



Classes

- class SerialInputStream

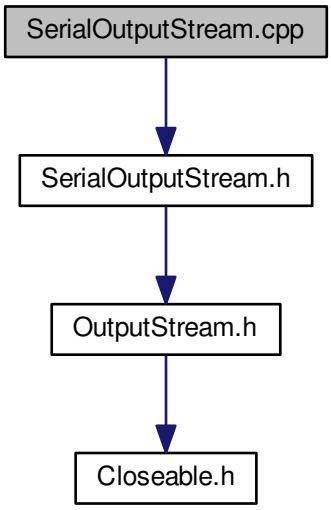
5.110 SerialInputStream.h

```
00001
00009 #ifndef __RASPBERRY_IO_SERIAL_INPUT_STREAM_H__
00010 #define __RASPBERRY_IO_SERIAL_INPUT_STREAM_H__ 1
00011
00012 #include <InputStream.h>
00013 //#include <Raspberry.h>
00014 #include <InputStream.h>
00015 #include <fcntl.h>
00016 #include <termios.h>
00017 #include <unistd.h>
00018 #include <stdio.h>
00019 #include <stdlib.h>
00020
00021
00022 class SerialInputStream: public InputStream {
00023
00024
00028     int fd;
00029
00033     int tmp;
00034
00035 public:
00036
00037     enum BoudRate {
00038         BR_50 = B50,
00039         BR_75 = B75,
00040         BR_110 = B110,
00041         BR_134 = B134,
00042         BR_150 = B150,
00043         BR_200 = B200,
00044         BR_300 = B300,
00045         BR_600 = B600,
00046         BR_1200 = B1200,
00047         BR_1800 = B1800,
00048         BR_2400 = B2400,
00049         BR_4800 = B4800,
00050         BR_9600 = B9600,
00051         BR_19200 = B19200,
00052         BR_38400 = B38400,
00053         BR_57600 = B57600,
00054         BR_115200 = B115200,
00055         BR_230400 = B230400,
00056         BR_460800 = B460800,
00057         BR_500000 = B500000,
00058         BR_576000 = B576000,
00059         BR_921600 = B921600,
00060         BR_1000000 = B1000000,
00061         BR_1152000 = B1152000,
00062         BR_1500000 = B1500000,
00063         BR_2000000 = B2000000,
00064         BR_2500000 = B2500000,
```

```
00065         BR_3000000 = B3000000,
00066         BR_3500000 = B3500000,
00067         BR_4000000 = B4000000
00068     };
00069
00075     SerialInputStream(const char *dev, BoudRate boundRate);
00076
00081     virtual int available();
00082
00086     virtual int read();
00087
00092     virtual int read(unsigned char* b, int len);
00093 };
00094
00095 #endif /* __RASPBERRY_IO_SERIAL_INPUT_STREAM_H__ */
```

5.111 SerialOutputStream.cpp File Reference

`#include <SerialOutputStream.h>`
Include dependency graph for SerialOutputStream.cpp:



Macros

- #define __RASPBERRY_IO_SERIAL_OUTPUT_STREAM_CPP__ 1

5.111.1 Macro Definition Documentation

5.111.1.1 #define __RASPBERRY_IO_SERIAL_OUTPUT_STREAM_CPP__ 1

Raspberry IO.

SerialOutputStream

A serial output stream is a output stream to write in a serial port.

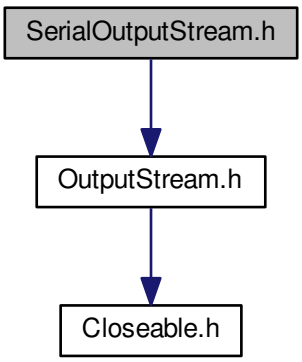
Definition at line 10 of file SerialOutputStream.cpp.

5.112 SerialOutputStream.cpp

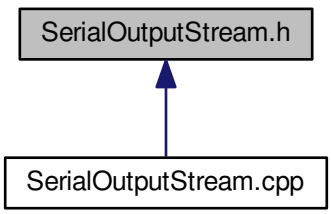
```
00001
00009 #ifndef __RASPBERRY_IO_SERIAL_OUTPUT_STREAM_CPP__
00010 #define __RASPBERRY_IO_SERIAL_OUTPUT_STREAM_CPP__ 1
00011
00012 #include <SerialOutputStream.h>
00013
00014 SerialOutputStream::SerialOutputStream(unsigned int boudRate) {
00015 }
00016
00017 void SerialOutputStream::write(unsigned char b) {
00018 }
00019
00020 #endif /* __RASPBERRY_IO_SERIAL_OUTPUT_STREAM_CPP__ */
```

5.113 SerialOutputStream.h File Reference

`#include <OutputStream.h>`
Include dependency graph for SerialOutputStream.h:



This graph shows which files directly or indirectly include this file:



Classes

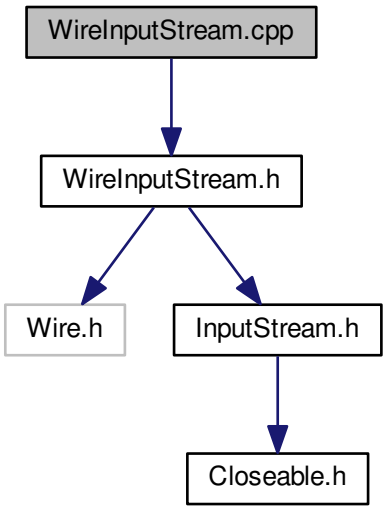
- class SerialOutputStream

5.114 SerialOutputStream.h

```
00001
00009 #ifndef __RASPBERRY_IO_SERIAL_OUTPUT_STREAM_H__
00010 #define __RASPBERRY_IO_SERIAL_OUTPUT_STREAM_H__ 1
00011
00012 #include <OutputStream.h>
00013
00014 class SerialOutputStream: public OutputStream {
00015
00021     SerialOutputStream(unsigned int boudRate);
00022
00026     virtual void write(unsigned char b);
00027 };
00028
00029 #endif /* __RASPBERRY_IO_SERIAL_OUTPUT_STREAM_H__ */
```

5.115 WireInputStream.cpp File Reference

`#include "WireInputStream.h"`
Include dependency graph for WireInputStream.cpp:



Macros

- #define __RASPBERRY_IO_WIRE_INPUT_STREAM_CPP__ 1

5.115.1 Macro Definition Documentation

5.115.1.1 #define __RASPBERRY_IO_WIRE_INPUT_STREAM_CPP__ 1

Raspberry IO.

WireInputStream

A WireInputStream obtains input bytes from the wire bus.

Definition at line 10 of file WireInputStream.cpp.

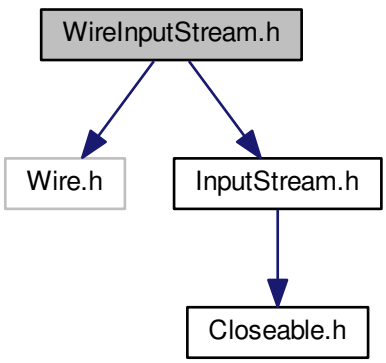
5.116 WireInputStream.cpp

```
00001
00009 #ifndef __RASPBERRY_IO_WIRE_INPUT_STREAM_CPP__
00010 #define __RASPBERRY_IO_WIRE_INPUT_STREAM_CPP__ 1
00011
00012 #include "WireInputStream.h"
00013
00014 WireInputStream::WireInputStream(unsigned char address) {
00015     this->address = address;
00016     Wire.begin();
00017 }
00018
00019 int WireInputStream::available() {
00020     return Wire.available();
00021 }
00022
00023 int WireInputStream::read() {
00024     Wire.beginTransmission(address);
00025     Wire.write((unsigned char) (address & 0xff));
00026     Wire.endTransmission();
00027     Wire.requestFrom(address, (unsigned char) 1);
00028     while (!Wire.available())
00029         ;
00030     return Wire.read();
00031 }
00032
00033 int WireInputStream::read(unsigned char* b, int off, int len) {
00034     int i;
00035     Wire.beginTransmission(address);
00036     Wire.write((unsigned char) (address & 0xff));
00037     Wire.endTransmission();
00038     Wire.requestFrom(address, (int) len);
00039     for (i = 0; i < len; i++) {
00040         while (!Wire.available())
00041             ;
00042         b[off + i] = (unsigned char) Wire.read();
00043     }
00044     return i;
00045 }
00046
00047 #endif /* __RASPBERRY_IO_WIRE_INPUT_STREAM_CPP__ */
```

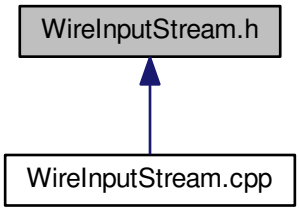
5.117 WireInputStream.h File Reference

```
#include <Wire.h>
#include <InputStream.h>
```

Include dependency graph for WireInputStream.h:



This graph shows which files directly or indirectly include this file:



Classes

- class WireInputStream

5.118 WireInputStream.h

```
00001
00009 #ifndef __RASPBERRY_IO_WIRE_INPUT_STREAM_H__
00010 #define __RASPBERRY_IO_WIRE_INPUT_STREAM_H__ 1
00011
00012 #include <Wire.h>
00013 #include <InputStream.h>
00014
00015 class WireInputStream: public InputStream {
00016 protected:
00017
00021     unsigned char address;
00022
00023 public:
00024
00030     WireInputStream(unsigned char addredd);
00031
00036     virtual int available();
00037
00041     virtual int read();
00042
00051     virtual int read(unsigned char* b, int off, int len);
00052 };
00053
00054 #endif /* __RASPBERRY_IO_WIRE_INPUT_STREAM_H__ */
```

5.119 WireOutputStream.cpp File Reference

5.120 WireOutputStream.cpp

5.121 WireOutputStream.h File Reference

5.122 WireOutputStream.h

Index