# Arduino Gyroscope Driver

Generated by Doxygen 1.8.9.1

Tue Aug 18 2015 22:52:10

ii CONTENTS

## **Contents**

1	Hier	archica	I Index	2
	1.1	Class	Hierarchy	2
2	Clas	s Index	<b>K</b>	3
	2.1	Class	List	3
3	File	Index		5
	3.1	File Lis	st	5
4	Clas	s Docu	mentation	7
	4.1	Buffere	edInputStream Class Reference	7
		4.1.1	Detailed Description	9
		4.1.2	Constructor & Destructor Documentation	9
		4.1.3	Member Function Documentation	9
		4.1.4	Member Data Documentation	11
	4.2	Buffere	edOutputStream Class Reference	12
		4.2.1	Detailed Description	13
		4.2.2	Constructor & Destructor Documentation	13
		4.2.3	Member Function Documentation	13
		4.2.4	Member Data Documentation	14
	4.3	ByteA	rrayInputStream Class Reference	15
		4.3.1	Detailed Description	16
		4.3.2	Constructor & Destructor Documentation	16
		4.3.3	Member Function Documentation	17
		4.3.4	Member Data Documentation	17
	4.4	ByteAı	rrayOutputStream Class Reference	18
		4.4.1	Detailed Description	19
		4.4.2	Constructor & Destructor Documentation	19
		4.4.3	Member Function Documentation	20
		4.4.4	Member Data Documentation	20
	4.5	ByteA	rraySeekableInputStream Class Reference	21
		4.5.1	Detailed Description	22
		4.5.2	Constructor & Destructor Documentation	22
		4.5.3	Member Function Documentation	22
	4.6	Closea	able Class Reference	22
		4.6.1	Detailed Description	23
		4.6.2	Member Function Documentation	23
	4.7	DataIn	uput Class Reference	23
		4.7.1	Detailed Description	24

	4.7.2	Member Function Documentation	24
4.8	DataIn	outStream Class Reference	27
	4.8.1	Detailed Description	28
	4.8.2	Constructor & Destructor Documentation	28
	4.8.3	Member Function Documentation	28
	4.8.4	Member Data Documentation	31
4.9	DataOu	utput Class Reference	31
	4.9.1	Detailed Description	32
	4.9.2	Member Function Documentation	32
4.10	DataOu	utputStream Class Reference	36
	4.10.1	Detailed Description	37
	4.10.2	Constructor & Destructor Documentation	38
	4.10.3	Member Function Documentation	39
	4.10.4	Member Data Documentation	41
4.11	Externa	alEepromInputStream Class Reference	41
	4.11.1	Detailed Description	43
	4.11.2	Constructor & Destructor Documentation	43
	4.11.3	Member Function Documentation	43
	4.11.4	Member Data Documentation	44
4.12	Externa	alEepromOutputStream Class Reference	45
	4.12.1	Detailed Description	46
	4.12.2	Constructor & Destructor Documentation	46
	4.12.3	Member Function Documentation	46
	4.12.4	Member Data Documentation	46
4.13	Externa	alEepromSeekableInputStream Class Reference	47
	4.13.1	Detailed Description	48
	4.13.2	Constructor & Destructor Documentation	48
	4.13.3	Member Function Documentation	48
4.14	FilterIn	putStream Class Reference	50
	4.14.1	Detailed Description	51
	4.14.2	Constructor & Destructor Documentation	52
	4.14.3	Member Function Documentation	53
	4.14.4	Member Data Documentation	55
4.15	FilterO	utputStream Class Reference	55
	4.15.1	Detailed Description	57
	4.15.2	Constructor & Destructor Documentation	57
	4.15.3	Member Function Documentation	57
	4.15.4	Member Data Documentation	59
4.16	Hardwa	areSerialInputStream Class Reference	59
	4.16.1	Detailed Description	61

iv CONTENTS

	4.16.2	Constructor & Destructor Documentation	61
	4.16.3	Member Function Documentation	61
4.17	Hardwa	reSerialOutputStream Class Reference	61
	4.17.1	Detailed Description	63
	4.17.2	Constructor & Destructor Documentation	63
	4.17.3	Member Function Documentation	63
4.18	InputStr	ream Class Reference	63
	4.18.1	Detailed Description	64
	4.18.2	Member Function Documentation	64
4.19	OutputS	Stream Class Reference	66
	4.19.1	Detailed Description	66
	4.19.2	Member Function Documentation	67
4.20	Pgmspa	aceInputStream Class Reference	68
	4.20.1	Detailed Description	69
	4.20.2	Constructor & Destructor Documentation	69
	4.20.3	Member Function Documentation	69
	4.20.4	Member Data Documentation	70
4.21	Pgmspa	aceSeekableInputStream Class Reference	70
	4.21.1	Detailed Description	72
	4.21.2	Constructor & Destructor Documentation	72
	4.21.3	Member Function Documentation	72
4.22	Randon	nAccess Class Reference	72
	4.22.1	Detailed Description	73
4.23	Randon	nAccessByteArray Class Reference	73
	4.23.1	Detailed Description	75
	4.23.2	Constructor & Destructor Documentation	75
	4.23.3	Member Function Documentation	75
	4.23.4	Member Data Documentation	82
4.24	Randon	nAccessExternalEeprom Class Reference	83
	4.24.1	Detailed Description	84
	4.24.2	Constructor & Destructor Documentation	84
	4.24.3	Member Function Documentation	85
	4.24.4	Member Data Documentation	91
4.25	Seekab	le Class Reference	91
	4.25.1	Detailed Description	92
	4.25.2	Member Function Documentation	92
4.26	Seekab	leInputStream Class Reference	92
		Detailed Description	93
4.27		putStream Class Reference	93
	4.27.1	Detailed Description	94

	4.28	SerialOutputStream Class Reference	95
		4.28.1 Detailed Description	95
	4.29	SoftwareSerialInputStream Class Reference	96
		4.29.1 Detailed Description	97
		4.29.2 Constructor & Destructor Documentation	97
		4.29.3 Member Function Documentation	98
		4.29.4 Member Data Documentation	98
	4.30	SoftwareSerialOutputStream Class Reference	98
		4.30.1 Detailed Description	00
		4.30.2 Constructor & Destructor Documentation	00
		4.30.3 Member Function Documentation	00
		4.30.4 Member Data Documentation	00
	4.31	WireInputStream Class Reference	00
		4.31.1 Detailed Description	02
		4.31.2 Constructor & Destructor Documentation	02
		4.31.3 Member Function Documentation	02
		4.31.4 Member Data Documentation	02
5	Eilo I	Documentation 1	UJ
3	5.1	BufferedInputStream.cpp File Reference	
	0.1	5.1.1 Macro Definition Documentation	
	5.2	BufferedInputStream.cpp	
	5.3	BufferedInputStream.h File Reference	
	5.4	BufferedInputStream.h	
	5.5	BufferedOutputStream.cpp File Reference	
	0.0	5.5.1 Macro Definition Documentation	
	5.6	BufferedOutputStream.cpp	
	5.7	BufferedOutputStream.h File Reference	
	5.8	BufferedOutputStream.h	
	5.9	ByteArrayInputStream.cpp File Reference	
	0.0	5.9.1 Macro Definition Documentation	
	5.10	ByteArrayInputStream.cpp	
		ByteArrayInputStream.h	
		ByteArrayOutputStream.cpp File Reference	
		5.13.1 Macro Definition Documentation	
	5.14	ByteArrayOutputStream.cpp	15
		ByteArrayOutputStream.h File Reference	
		ByteArrayOutputStream.h	
	5.17	ByteArraySeekableInputStream.cpp File Reference	17

vi CONTENTS

	5.17.1 Macro Definition Documentation	117
5.18	ByteArraySeekableInputStream.cpp	118
5.19	ByteArraySeekableInputStream.h File Reference	118
5.20	ByteArraySeekableInputStream.h	119
5.21	Closeable.cpp File Reference	119
	5.21.1 Macro Definition Documentation	120
5.22	Closeable.cpp	120
5.23	Closeable.h File Reference	120
5.24	Closeable.h	120
5.25	DataInput.cpp File Reference	121
	5.25.1 Macro Definition Documentation	121
5.26	DataInput.cpp	121
5.27	DataInput.h File Reference	121
5.28	DataInput.h	122
5.29	DataInputStream.cpp File Reference	123
	5.29.1 Macro Definition Documentation	123
5.30	DataInputStream.cpp	123
5.31	DataInputStream.h File Reference	124
5.32	DataInputStream.h	125
5.33	DataOutput.cpp File Reference	126
	5.33.1 Macro Definition Documentation	126
	DataOutput.cpp	
5.35	DataOutput.h File Reference	127
5.36	DataOutput.h	127
5.37	DataOutputStream.cpp File Reference	128
	5.37.1 Macro Definition Documentation	129
5.38	DataOutputStream.cpp	129
5.39	DataOutputStream.h File Reference	130
5.40	DataOutputStream.h	130
5.41	ExternalEepromInputStream.cpp File Reference	131
	5.41.1 Macro Definition Documentation	132
	ExternalEepromInputStream.cpp	
5.43	ExternalEepromInputStream.h File Reference	133
5.44	ExternalEepromInputStream.h	134
5.45	ExternalEepromOutputStream.cpp File Reference	
	5.45.1 Macro Definition Documentation	135
	ExternalEepromOutputStream.cpp	
	ExternalEepromOutputStream.h File Reference	
	ExternalEepromOutputStream.h	
5.49	ExternalEepromSeekableInputStream.cpp File Reference	137

CONTENTS vii

	5.49.1 Macro Definition Documentation	138
5.50	ExternalEepromSeekableInputStream.cpp	138
5.51	ExternalEepromSeekableInputStream.h File Reference	138
5.52	ExternalEepromSeekableInputStream.h	139
5.53	FilterInputStream.cpp File Reference	139
	5.53.1 Macro Definition Documentation	140
5.54	FilterInputStream.cpp	140
5.55	FilterInputStream.h File Reference	141
5.56	FilterInputStream.h	142
5.57	FilterOutputStream.cpp File Reference	142
	5.57.1 Macro Definition Documentation	143
5.58	FilterOutputStream.cpp	143
5.59	FilterOutputStream.h File Reference	144
5.60	FilterOutputStream.h	145
5.61	HardwareSerialInputStream.cpp File Reference	145
	5.61.1 Macro Definition Documentation	146
5.62	HardwareSerialInputStream.cpp	146
5.63	HardwareSerialInputStream.h File Reference	147
5.64	HardwareSerialInputStream.h	148
5.65	HardwareSerialOutputStream.cpp File Reference	148
	5.65.1 Macro Definition Documentation	149
5.66	HardwareSerialOutputStream.cpp	149
5.67	HardwareSerialOutputStream.h File Reference	149
5.68	HardwareSerialOutputStream.h	150
5.69	InputStream.cpp File Reference	151
	5.69.1 Macro Definition Documentation	151
5.70	InputStream.cpp	151
5.71	InputStream.h File Reference	152
5.72	InputStream.h	153
5.73	OutputStream.cpp File Reference	153
	5.73.1 Macro Definition Documentation	154
5.74	OutputStream.cpp	154
5.75	OutputStream.h File Reference	155
5.76	OutputStream.h	155
5.77	PgmspaceInputStream.cpp File Reference	156
	5.77.1 Macro Definition Documentation	156
5.78	PgmspaceInputStream.cpp	156
	PgmspaceInputStream.h File Reference	
5.80	PgmspaceInputStream.h	158
5.81	PgmspaceSeekableInputStream.cpp File Reference	159

VIII CONTENTS

5.81.1 Macro Definition Documentation
5.82 PgmspaceSeekableInputStream.cpp
5.83 PgmspaceSeekableInputStream.h File Reference
5.84 PgmspaceSeekableInputStream.h
5.85 RandomAccess.cpp File Reference
5.85.1 Macro Definition Documentation
5.86 RandomAccess.cpp
5.87 RandomAccess.h File Reference
5.88 RandomAccess.h
5.89 RandomAccessByteArray.cpp File Reference
5.89.1 Macro Definition Documentation
5.90 RandomAccessByteArray.cpp
5.91 RandomAccessByteArray.h File Reference
5.92 RandomAccessByteArray.h
5.93 RandomAccessExternalEeprom.cpp File Reference
5.93.1 Macro Definition Documentation
5.94 RandomAccessExternalEeprom.cpp
5.95 RandomAccessExternalEeprom.h File Reference
5.96 RandomAccessExternalEeprom.h
5.97 RandomAccessResource.cpp File Reference
5.97.1 Macro Definition Documentation
5.98 RandomAccessResource.cpp
5.99 RandomAccessResource.h File Reference
5.100RandomAccessResource.h
5.101ResourceInputStream.cpp File Reference
5.101.1 Macro Definition Documentation
5.102ResourceInputStream.cpp
5.103ResourceInputStream.h File Reference
5.104ResourceInputStream.h
5.105ResourceOutputStream.cpp File Reference
5.105.1 Macro Definition Documentation
5.106ResourceOutputStream.cpp
5.107ResourceOutputStream.h File Reference
5.108ResourceOutputStream.h
5.109ResourceSeekableInputStream.cpp File Reference
5.109.1 Macro Definition Documentation
5.110ResourceSeekableInputStream.cpp
5.111 Resource Seekable Input Stream.h File Reference
5.112ResourceSeekableInputStream.h
5.113Seekable.cpp File Reference

	5.113.1 Macro Definition Documentation	180
	5.114Seekable.cpp	180
	5.115Seekable.h File Reference	180
	5.116Seekable.h	181
	5.117SeekableInputStream.cpp File Reference	181
	5.117.1 Macro Definition Documentation	181
	5.118SeekableInputStream.cpp	181
	5.119SeekableInputStream.h File Reference	182
	5.120SeekableInputStream.h	182
	5.121 SerialInputStream.cpp File Reference	183
	5.121.1 Macro Definition Documentation	183
	5.122SerialInputStream.cpp	183
	5.123SerialInputStream.h File Reference	184
	5.124SerialInputStream.h	184
	5.125SerialOutputStream.cpp File Reference	185
	5.125.1 Macro Definition Documentation	185
	5.126SerialOutputStream.cpp	185
	5.127SerialOutputStream.h File Reference	185
	5.128SerialOutputStream.h	186
	5.129SoftwareSerialInputStream.cpp File Reference	186
	5.129.1 Macro Definition Documentation	187
	5.130SoftwareSerialInputStream.cpp	187
	5.131 Software Serial Input Stream.h File Reference	188
	5.132SoftwareSerialInputStream.h	189
	5.133SoftwareSerialOutputStream.cpp File Reference	189
	5.133.1 Macro Definition Documentation	189
	5.134SoftwareSerialOutputStream.cpp	190
	5.135SoftwareSerialOutputStream.h File Reference	190
	5.136SoftwareSerialOutputStream.h	191
	5.137WireInputStream.cpp File Reference	191
	5.137.1 Macro Definition Documentation	192
	5.138WireInputStream.cpp	192
	5.139WireInputStream.h File Reference	193
	5.140WireInputStream.h	194
	5.141 WireOutputStream.cpp File Reference	194
	5.142WireOutputStream.cpp	194
	5.143WireOutputStream.h File Reference	
	5.144WireOutputStream.h	194
Ind	lex	195

## 1 Hierarchical Index

## 1.1 Class Hierarchy

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

Closeable	22
InputStream	63
ByteArrayInputStream	15
ByteArraySeekableInputStream	21
ExternalEepromInputStream	41
ExternalEepromSeekableInputStream	47
FilterInputStream	50
BufferedInputStream	7
PgmspaceInputStream	68
PgmspaceSeekableInputStream	70
SeekableInputStream	92
ByteArraySeekableInputStream	21
ExternalEepromSeekableInputStream	47
PgmspaceSeekableInputStream	70
SerialInputStream	93
HardwareSerialInputStream	59
SoftwareSerialInputStream	96
WireInputStream	100
OutputStream	66
ByteArrayOutputStream	18
ExternalEepromOutputStream	45
FilterOutputStream	55
BufferedOutputStream	12
SerialOutputStream	95
HardwareSerialOutputStream	61
SoftwareSerialOutputStream	98
RandomAccess	72
RandomAccessByteArray	73

2 Class Index

HandomAccessExternalEeprom	83
RandomAccessByteArray	73
RandomAccessExternalEeprom	83
DataInput	23
DataInputStream	27
RandomAccess	72
DataOutput	31
DataOutputStream	36
RandomAccess	72
Seekable	91
RandomAccess	72
SeekableInputStream	92
2 Class Index	
2.1 Class List	
Here are the classes, structs, unions and interfaces with brief descriptions:	
BufferedInputStream Arduino IO	7
BufferedOutputStream Arduino IO	12
ByteArrayInputStream Arduino IO	15
ByteArrayOutputStream Arduino IO	18
ByteArraySeekableInputStream Arduino IO	21
Closeable Arduino IO	22
DataInput Arduino IO	23
DataInputStream Arduino IO	27
DataOutput Arduino IO	31
DataOutputStream Arduino IO	36

Arduino IO	41
ExternalEepromOutputStream Arduino IO	45
ExternalEepromSeekableInputStream Arduino IO	47
FilterInputStream  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	50
FilterOutputStream Arduino IO	55
HardwareSerialInputStream Arduino IO	59
HardwareSerialOutputStream Arduino IO	61
InputStream Arduino IO	63
OutputStream Arduino IO	66
PgmspaceInputStream Arduino IO	68
PgmspaceSeekableInputStream Arduino IO	70
RandomAccess Araduino IO	72
RandomAccessByteArray Araduino IO	73
RandomAccessExternalEeprom Araduino IO	83
Seekable Arduino IO	91
SeekableInputStream Arduino IO	92
SerialInputStream Arduino IO	93
SerialOutputStream Arduino IO	95
SoftwareSerialInputStream Arduino IO	96
SoftwareSerialOutputStream Arduino IO	98

3 File Index 5

	Arduino IO	100
3	File Index	
3.1	File List	
He	re is a list of all files with brief descriptions:	
	BufferedInputStream.cpp	103
	BufferedInputStream.h	105
	BufferedOutputStream.cpp	107
	BufferedOutputStream.h	109
	ByteArrayInputStream.cpp	111
	ByteArrayInputStream.h	112
	ByteArrayOutputStream.cpp	114
	ByteArrayOutputStream.h	115
	ByteArraySeekableInputStream.cpp	117
	ByteArraySeekableInputStream.h	118
	Closeable.cpp	119
	Closeable.h	120
	DataInput.cpp	121
	DataInput.h	121
	DataInputStream.cpp	123
	DataInputStream.h	124
	DataOutput.cpp	126
	DataOutput.h	127
	DataOutputStream.cpp	128
	DataOutputStream.h	130
	ExternalEepromInputStream.cpp	131
	ExternalEepromInputStream.h	133
	ExternalEepromOutputStream.cpp	135
	ExternalEepromOutputStream.h	136
	ExternalEepromSeekableInputStream.cpp	137
	ExternalEepromSeekableInputStream.h	138

FilterInputStream.cpp	139
FilterInputStream.h	141
FilterOutputStream.cpp	142
FilterOutputStream.h	144
HardwareSerialInputStream.cpp	145
HardwareSerialInputStream.h	147
HardwareSerialOutputStream.cpp	148
HardwareSerialOutputStream.h	149
InputStream.cpp	151
InputStream.h	152
OutputStream.cpp	153
OutputStream.h	155
PgmspaceInputStream.cpp	156
PgmspaceInputStream.h	157
PgmspaceSeekableInputStream.cpp	159
PgmspaceSeekableInputStream.h	160
RandomAccess.cpp	161
RandomAccess.h	162
RandomAccessByteArray.cpp	163
RandomAccessByteArray.h	165
RandomAccessExternalEeprom.cpp	167
RandomAccessExternalEeprom.h	170
RandomAccessResource.cpp	172
RandomAccessResource.h	175
ResourceInputStream.cpp	176
ResourceInputStream.h	177
ResourceOutputStream.cpp	177
ResourceOutputStream.h	178
ResourceSeekableInputStream.cpp	178
ResourceSeekableInputStream.h	179
Seekable.cpp	179
Seekable.h	180

4 Class Documentation 7

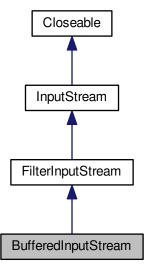
SeekableInputStream.cpp	181
SeekableInputStream.h	182
SerialInputStream.cpp	183
SerialInputStream.h	184
SerialOutputStream.cpp	185
SerialOutputStream.h	185
SoftwareSerialInputStream.cpp	186
SoftwareSerialInputStream.h	188
SoftwareSerialOutputStream.cpp	189
SoftwareSerialOutputStream.h	190
WireInputStream.cpp	191
WireInputStream.h	193
WireOutputStream.cpp	194
WireOutputStream.h	194

## 4 Class Documentation

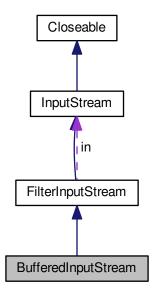
## 4.1 BufferedInputStream Class Reference

#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

Arduino IO.

### BufferedInputStream

A <u>BufferedInputStream</u> adds functionality to another input stream-namely, the ability to buffer the input and to support the mark and reset methods. When the <u>BufferedInputStream</u> 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 37 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 41 of file BufferedInputStream.cpp.

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

Fills the buffer.

#### **Parameters**

startPos

Definition at line 126 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 138 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 145 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 98 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 51 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 55 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 114 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 45 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 149 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  ${\tt pos}$  field at the time the last  ${\tt mark}$  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.

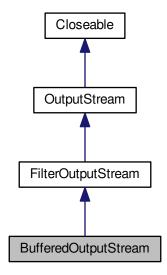
The documentation for this class was generated from the following files:

- · BufferedInputStream.h
- BufferedInputStream.cpp

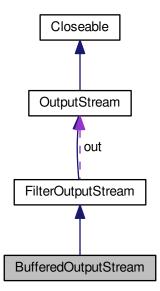
## 4.2 BufferedOutputStream Class Reference

#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

#### Arduino 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

## $\textbf{4.2.2.1} \quad \textbf{BufferedOutputStream::BufferedOutputStream ( \ \textbf{OutputStream}*\textit{out, unsigned char}*\textit{buf, int size} \ )$

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 <code>BufferedOutputStreams</code> 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.

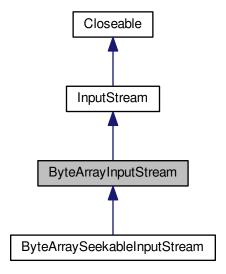
The documentation for this class was generated from the following files:

- · BufferedOutputStream.h
- BufferedOutputStream.cpp

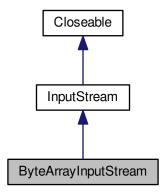
## 4.3 ByteArrayInputStream Class Reference

#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

#### Arduino 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.

NOTE: This implementation return 1 or 0. It is because the size of the array is unsigned int, and this method returns a signed int, which means there is no way to return the difference between the current position (can be 0) and the size of the array without possible overflow.

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 44 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.

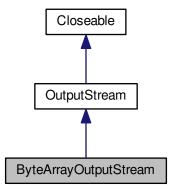
The documentation for this class was generated from the following files:

- ByteArrayInputStream.h
- ByteArrayInputStream.cpp

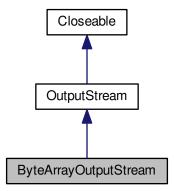
## 4.4 ByteArrayOutputStream Class Reference

#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

Arduino 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.

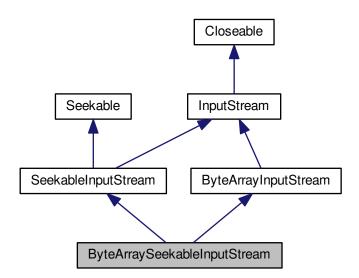
The documentation for this class was generated from the following files:

- ByteArrayOutputStream.h
- ByteArrayOutputStream.cpp

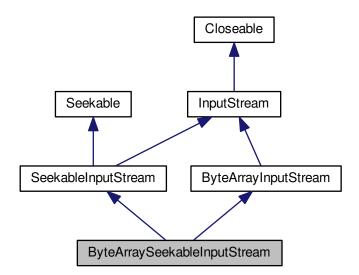
## 4.5 ByteArraySeekableInputStream Class Reference

#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

Arduino IO.

ByteArraySeekableInputStream

A ByteArraySeekableInputStream obtains input bytes from a resource in a file system that implements Seekable ← InputStream 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.

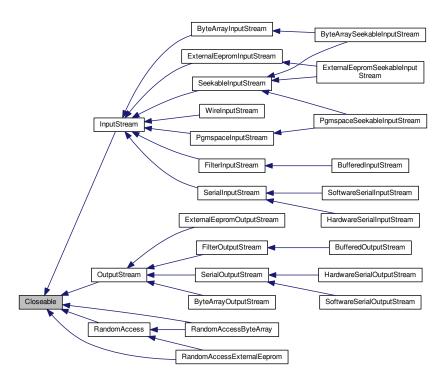
Definition at line 20 of file ByteArraySeekableInputStream.cpp.

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 void close ()=0

#### 4.6.1 Detailed Description

Arduino 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 Member Function Documentation

## 4.6.2.1 virtual void Closeable::close ( ) [pure virtual]

Implemented in BufferedInputStream, FilterInputStream, FilterOutputStream, BufferedOutputStream, Random← AccessByteArray, RandomAccessExternalEeprom, InputStream, and OutputStream.

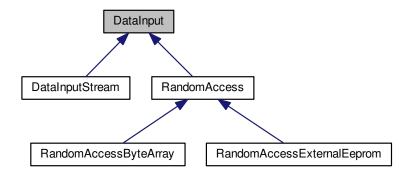
The documentation for this class was generated from the following file:

· Closeable.h

### 4.7 DataInput Class Reference

#include <DataInput.h>

Inheritance diagram for DataInput:



#### **Public Member Functions**

- virtual 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 word readWord ()=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

#### Arduino 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 16 of file DataInput.h.

## 4.7.2 Member Function Documentation

### **4.7.2.1** virtual bool DataInput::readBoolean() [pure virtual]

Reads a bool from the stream.

Returns

bool

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.

```
4.7.2.2 virtual unsigned char DataInput::readByte() [pure virtual]
Reads a unsigned char from the stream.
Returns
     unsigned char
Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.
4.7.2.3 virtual char DataInput::readChar() [pure virtual]
Reads a char from the stream.
Returns
     char
Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.
4.7.2.4 virtual double DataInput::readDouble ( ) [pure virtual]
Reads a double from the stream.
Returns
     double
Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.
4.7.2.5 virtual float DataInput::readFloat() [pure virtual]
Reads a float from the stream.
Returns
     float
Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.
4.7.2.6 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.
4.7.2.7 virtual int DataInput::readInt() [pure virtual]
Reads an int from the stream.
Returns
     int
```

Generated on Tue Aug 18 2015 22:52:10 for Arduino Gyroscope Driver by Doxygen

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.

```
virtual long DataInput::readLong( ) [pure virtual]
Reads a long from the stream.
Returns
     long
Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.
4.7.2.9 virtual unsigned char DataInput::readUnsignedChar() [pure virtual]
Reads an unsigned char from the stream.
Returns
     unsigned char
Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.
4.7.2.10 virtual unsigned int DataInput::readUnsignedInt() [pure virtual]
Reads an unsigned int from the stream.
Returns
     unsigned int
Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.
4.7.2.11 virtual unsigned long DataInput::readUnsignedLong() [pure virtual]
Reads a unsigned long from the stream.
Returns
     unsigned long
Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.
4.7.2.12 virtual word DataInput::readWord() [pure virtual]
Reads a word from the stream.
Returns
     word
Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.
4.7.2.13 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.

 $Implemented\ in\ Random Access Byte Array,\ Random Access External Eeprom,\ and\ DataInput Stream.$ 

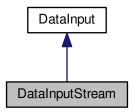
The documentation for this class was generated from the following file:

DataInput.h

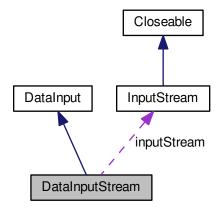
## 4.8 DataInputStream Class Reference

#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 word readWord ()
- 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

Arduino 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
Returns
     double
```

Implements DataInput.

Definition at line 26 of file DataInputStream.cpp.

4.8.3.4 double DataInputStream::readDouble( ) [virtual]

Reads a double from the stream.

Implements DataInput.

Definition at line 70 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 66 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 74 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 50 of file DataInputStream.cpp.

```
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 62 of file DataInputStream.cpp.
4.8.3.12 word DataInputStream::readWord() [virtual]
Reads a word from the stream.
Returns
     word
Implements DataInput.
Definition at line 46 of file DataInputStream.cpp.
4.8.3.13 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 80 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.

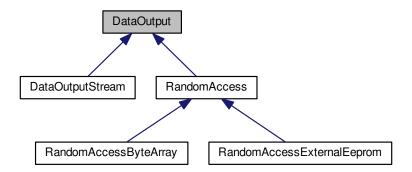
The documentation for this class was generated from the following files:

- · 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 writeWord (word 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

Arduino 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 15 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.

 $Implemented\ in\ Random Access Byte Array,\ Random Access External Eeprom,\ and\ Data Output Stream.$ 

**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.

 $Implemented\ in\ Random Access Byte Array,\ Random Access External Eeprom,\ and\ Data Output Stream.$ 

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

Writes an array of bytes into the stream.

#### **Parameters**

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

 $Implemented \ in \ Random Access Byte Array, \ Random Access External Eeprom, \ and \ Data Output Stream.$ 

**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** 

 $v \mid$  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** 

 $v \mid$  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.

 $Implemented\ in\ Random Access Byte Array,\ Random Access External Eeprom,\ and\ Data Output Stream.$ 

**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 ν )** [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  $\nu$  ) [pure virtual]

Writes a unsigned long into the stream.

#### **Parameters**

v The unsigned long to be written.

 $Implemented\ in\ Random Access Byte Array,\ Random Access External Eeprom,\ and\ Data Output Stream.$ 

**4.9.2.14** virtual void DataOutput::writeWord ( word v ) [pure virtual]

Writes a word into the stream.

**Parameters** 

V The word to be written.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

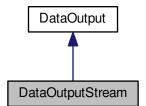
The documentation for this class was generated from the following file:

• DataOutput.h

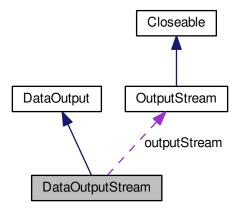
# 4.10 DataOutputStream Class Reference

#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 writeWord (word 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

# Arduino 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
- $4.10.2.1 \quad DataOutputStream::DataOutputStream \ ( \ OutputStream * \textit{outputStream} \ )$

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 76 of file DataOutputStream.cpp.

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

Writes a float into the stream.

**Parameters** 

ν The float to be written.

Implements DataOutput.

Definition at line 72 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** 

ν The long to be written.

Implements DataOutput.

Definition at line 61 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** 

ν The unsigned long to be written.

Implements DataOutput.

Definition at line 68 of file DataOutputStream.cpp.

**4.10.3.14 void DataOutputStream::writeWord ( word v )** [virtual]

Writes a word into the stream.

Parameters

ν The word to be written.

Implements DataOutput.

Definition at line 57 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.

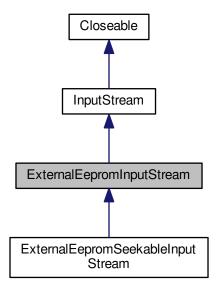
The documentation for this class was generated from the following files:

- · DataOutputStream.h
- DataOutputStream.cpp

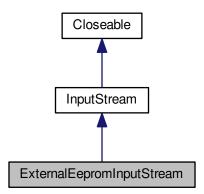
# 4.11 ExternalEepromInputStream Class Reference

#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

Arduino 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

```
\textbf{4.11.4.1} \quad \textbf{ExternalEeprom}* \quad \textbf{ExternalEeprominputStream} :: \textbf{externalEeprom} \quad \textbf{[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.

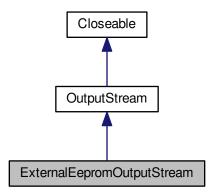
The documentation for this class was generated from the following files:

- ExternalEepromInputStream.h
- ExternalEepromInputStream.cpp

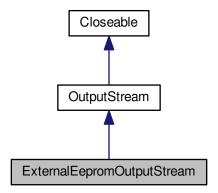
# 4.12 ExternalEepromOutputStream Class Reference

#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

Arduino 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.

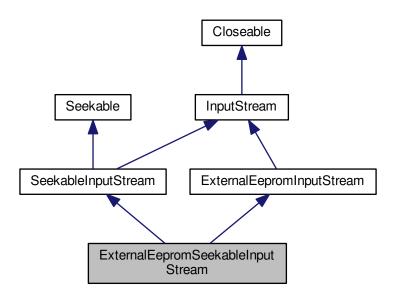
Definition at line 26 of file ExternalEepromOutputStream.h.

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

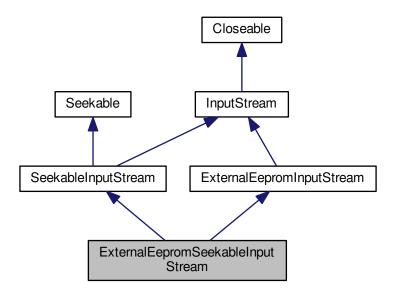
- ExternalEepromOutputStream.h
- ExternalEepromOutputStream.cpp

# 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

Arduino 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

Seeks this input stream to the position.

 $\textbf{4.13.3.1} \quad \textbf{void ExternalEepromSeekableInputStream::seek (unsigned int \textit{pos}\ )} \quad \texttt{[virtual]}$ 

### **Parameters**

pos	THe position.

Implements Seekable.

Definition at line 20 of file ExternalEepromSeekableInputStream.cpp.

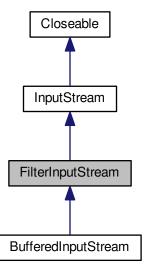
The documentation for this class was generated from the following files:

- ExternalEepromSeekableInputStream.h
- ExternalEepromSeekableInputStream.cpp

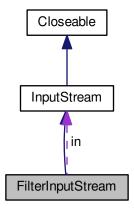
# 4.14 FilterInputStream Class Reference

#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.

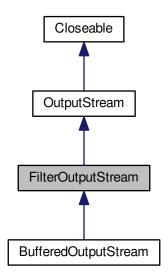
The documentation for this class was generated from the following files:

- · FilterInputStream.h
- FilterInputStream.cpp

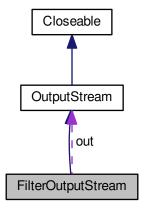
# 4.15 FilterOutputStream Class Reference

#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

Arduino 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).

58 **CONTENTS** 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**

,	the data.
Oi	the start offset in the data.
lei	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.

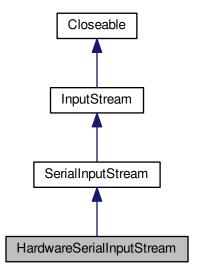
The documentation for this class was generated from the following files:

- · FilterOutputStream.h
- FilterOutputStream.cpp

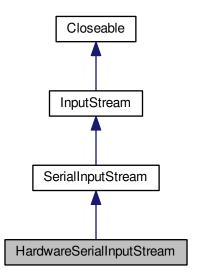
# 4.16 HardwareSerialInputStream Class Reference

#include <HardwareSerialInputStream.h>

Inheritance diagram for HardwareSerialInputStream:



Collaboration diagram for HardwareSerialInputStream:



### **Public Member Functions**

- HardwareSerialInputStream (unsigned int boudRate)
- virtual int available ()
- virtual int read ()

4.16.1 Detailed Description

Arduino IO.

HardwareSerialInputStream

A HardwareSerialInputStream obtains input bytes from a serial port.

Definition at line 16 of file HardwareSerialInputStream.h.

- 4.16.2 Constructor & Destructor Documentation
- 4.16.2.1 HardwareSerialInputStream::HardwareSerialInputStream ( unsigned int boudRate )

Public constructor.

**Parameters** 

boudRate

Definition at line 14 of file HardwareSerialInputStream.cpp.

- 4.16.3 Member Function Documentation
- **4.16.3.1** int HardwareSerialInputStream::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 18 of file HardwareSerialInputStream.cpp.

**4.16.3.2** int HardwareSerialInputStream::read() [virtual]

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

Implements InputStream.

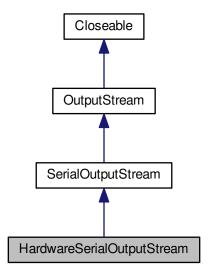
Definition at line 22 of file HardwareSerialInputStream.cpp.

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

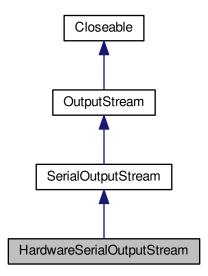
- · HardwareSerialInputStream.h
- · HardwareSerialInputStream.cpp
- 4.17 HardwareSerialOutputStream Class Reference

#include <HardwareSerialOutputStream.h>

Inheritance diagram for HardwareSerialOutputStream:



Collaboration diagram for HardwareSerialOutputStream:



# **Public Member Functions**

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

# 4.17.1 Detailed Description

Arduino IO.

### HardwareSerialOutputStream

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

Definition at line 16 of file HardwareSerialOutputStream.h.

### 4.17.2 Constructor & Destructor Documentation

# 4.17.2.1 HardwareSerialOutputStream::HardwareSerialOutputStream (unsigned int boudRate)

Public constructor.

**Parameters** 

boudRate

Definition at line 14 of file HardwareSerialOutputStream.cpp.

#### 4.17.3 Member Function Documentation

# **4.17.3.1 void HardwareSerialOutputStream::write (unsigned char b)** [virtual]

Writes the specified unsigned char to this output stream.

Implements OutputStream.

Definition at line 18 of file HardwareSerialOutputStream.cpp.

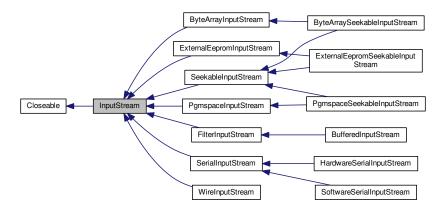
The documentation for this class was generated from the following files:

- HardwareSerialOutputStream.h
- HardwareSerialOutputStream.cpp

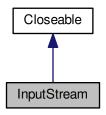
# 4.18 InputStream Class Reference

#include <InputStream.h>

Inheritance diagram for InputStream:



Collaboration diagram for InputStream:



#### **Public Member Functions**

- 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)

### 4.18.1 Detailed Description

# Arduino 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.

#### 4.18.2 Member Function Documentation

```
4.18.2.1 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, ExternalEepromInputStream, ByteArrayInputStream, PgmspaceInputStream, SoftwareSerialInputStream, WireInputStream, and HardwareSerialInputStream.

Definition at line 18 of file InputStream.cpp.

```
4.18.2.2 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.

```
4.18.2.3 void InputStream::mark() [virtual]
```

Marks the current position in this input stream.

Reimplemented in BufferedInputStream, FilterInputStream, ExternalEepromInputStream, ByteArrayInputStream, and PgmspaceInputStream.

Definition at line 25 of file InputStream.cpp.

```
4.18.2.4 bool InputStream::markSupported( ) [virtual]
```

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

Reimplemented in FilterInputStream, BufferedInputStream, ExternalEepromInputStream, ByteArrayInputStream, and PgmspaceInputStream.

Definition at line 28 of file InputStream.cpp.

```
4.18.2.5 virtual int InputStream::read ( ) [pure virtual]
```

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

Implemented in BufferedInputStream, ExternalEepromInputStream, ByteArrayInputStream, PgmspaceInputStream, FilterInputStream, SoftwareSerialInputStream, WireInputStream, and HardwareSerialInputStream.

```
4.18.2.6 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, and FilterInputStream.

Definition at line 32 of file InputStream.cpp.

```
4.18.2.7 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.

```
4.18.2.8 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, ByteArrayInputStream, and PgmspaceInputStream.

Definition at line 56 of file InputStream.cpp.

```
4.18.2.9 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.

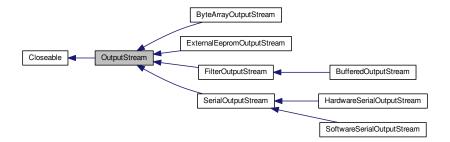
The documentation for this class was generated from the following files:

- · InputStream.h
- InputStream.cpp

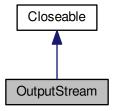
# 4.19 OutputStream Class Reference

#include <OutputStream.h>

Inheritance diagram for OutputStream:



Collaboration diagram for OutputStream:



## **Public Member Functions**

- 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)

### 4.19.1 Detailed Description

Arduino 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.

### 4.19.2 Member Function Documentation

```
4.19.2.1 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 34 of file OutputStream.cpp.

```
4.19.2.2 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 31 of file OutputStream.cpp.

```
4.19.2.3 virtual void OutputStream::write (unsigned char b) [pure virtual]
```

Writes the specified unsigned char to this output stream.

Implemented in ByteArrayOutputStream, BufferedOutputStream, FilterOutputStream, ExternalEepromOutput Stream, SoftwareSerialOutputStream, and HardwareSerialOutputStream.

```
4.19.2.4 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 18 of file OutputStream.cpp.

```
4.19.2.5 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 22 of file OutputStream.cpp.

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

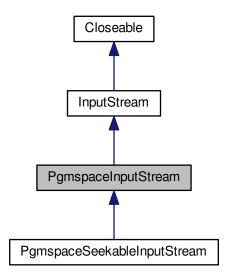
OutputStream.h

• OutputStream.cpp

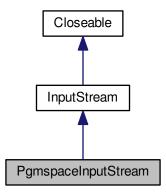
# 4.20 PgmspaceInputStream Class Reference

#include <PgmspaceInputStream.h>

Inheritance diagram for PgmspaceInputStream:



Collaboration diagram for PgmspaceInputStream:



**Public Member Functions** 

• PgmspaceInputStream (char PROGMEM \*buf, unsigned int count)

- virtual int available ()
- virtual void mark ()
- · virtual bool markSupported ()
- · virtual int read ()
- · virtual void reset ()

### **Protected Attributes**

- char PROGMEM \* buf
- · unsigned int count
- unsigned int pos
- unsigned int markpos

### 4.20.1 Detailed Description

Arduino IO.

### PgmspaceInputStream

A PgmspaceInputStream contains an internal buffer that contains bytes that may be read from the stream mapped to an part of the pgmspace.

Definition at line 16 of file PgmspaceInputStream.h.

```
4.20.2 Constructor & Destructor Documentation
```

```
4.20.2.1 PgmspaceInputStream::PgmspaceInputStream ( char PROGMEM * buf, unsigned int count ) [explicit]
```

Definition at line 15 of file PgmspaceInputStream.cpp.

4.20.3 Member Function Documentation

```
4.20.3.1 int PgmspaceInputStream::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 20 of file PgmspaceInputStream.cpp.

```
4.20.3.2 void PgmspaceInputStream::mark( ) [virtual]
```

Marks the current position in this input stream.

Reimplemented from InputStream.

Definition at line 27 of file PgmspaceInputStream.cpp.

```
4.20.3.3 bool PgmspaceInputStream::markSupported( ) [virtual]
```

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

Returns

```
Reimplemented from InputStream.
```

Definition at line 31 of file PgmspaceInputStream.cpp.

```
4.20.3.4 int PgmspaceInputStream::read() [virtual]
```

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

Returns

Implements InputStream.

Definition at line 35 of file PgmspaceInputStream.cpp.

```
4.20.3.5 void PgmspaceInputStream::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 39 of file PgmspaceInputStream.cpp.

4.20.4 Member Data Documentation

```
4.20.4.1 char PROGMEM* PgmspaceInputStream::buf [protected]
```

Definition at line 22 of file PgmspaceInputStream.h.

**4.20.4.2 unsigned int PgmspaceInputStream::count** [protected]

Definition at line 27 of file PgmspaceInputStream.h.

**4.20.4.3 unsigned int PgmspaceInputStream::markpos** [protected]

Definition at line 37 of file PgmspaceInputStream.h.

**4.20.4.4 unsigned int PgmspaceInputStream::pos** [protected]

Definition at line 32 of file PgmspaceInputStream.h.

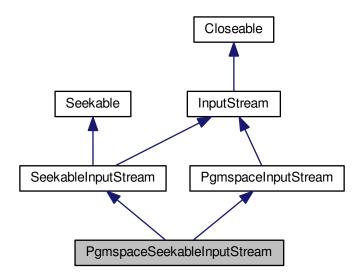
The documentation for this class was generated from the following files:

- · PgmspaceInputStream.h
- PgmspaceInputStream.cpp

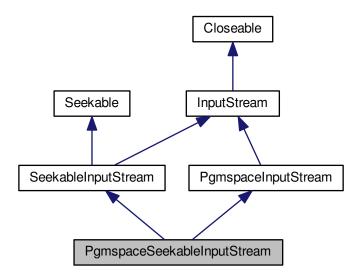
# 4.21 PgmspaceSeekableInputStream Class Reference

#include <PgmspaceSeekableInputStream.h>

Inheritance diagram for PgmspaceSeekableInputStream:



Collaboration diagram for PgmspaceSeekableInputStream:



# **Public Member Functions**

- PgmspaceSeekableInputStream (char PROGMEM \*buf, unsigned int count)
- virtual void seek (unsigned int pos)

**Additional Inherited Members** 

## 4.21.1 Detailed Description

Arduino IO.

### PgmspaceSeekableInputStream

A PgmspaceSeekableInputStream obtains input bytes from a resource in a file system that implements Seekable ← InputStream interface.

Definition at line 16 of file PgmspaceSeekableInputStream.h.

### 4.21.2 Constructor & Destructor Documentation

4.21.2.1 PgmspaceSeekableInputStream::PgmspaceSeekableInputStream ( char PROGMEM \* buf, unsigned int count )

Definition at line 15 of file PgmspaceSeekableInputStream.cpp.

### 4.21.3 Member Function Documentation

4.21.3.1 void PgmspaceSeekableInputStream::seek (unsigned int pos) [virtual]

Implements Seekable.

Definition at line 18 of file PgmspaceSeekableInputStream.cpp.

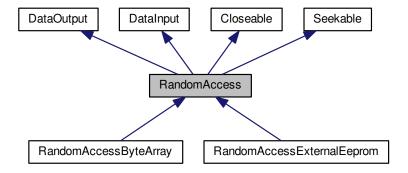
The documentation for this class was generated from the following files:

- · PgmspaceSeekableInputStream.h
- PgmspaceSeekableInputStream.cpp

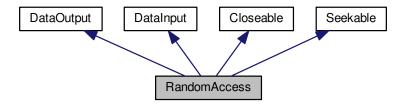
# 4.22 RandomAccess Class Reference

#include <RandomAccess.h>

Inheritance diagram for RandomAccess:



Collaboration diagram for RandomAccess:



**Additional Inherited Members** 

## 4.22.1 Detailed Description

Araduino IO.

### RandomAccess

Interface derived from DataInput, DataOutput, Closeable and Seekable.

Definition at line 17 of file RandomAccess.h.

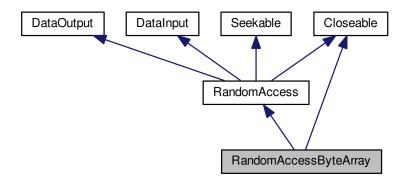
The documentation for this class was generated from the following file:

· RandomAccess.h

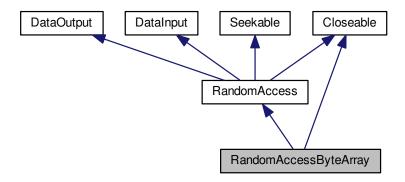
# 4.23 RandomAccessByteArray Class Reference

#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 writeWord (word 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 word readWord ()
- 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

### 4.23.1 Detailed Description

Araduino 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.

### 4.23.2 Constructor & Destructor Documentation

4.23.2.1 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.

4.23.3 Member Function Documentation

```
4.23.3.1 void RandomAccessByteArray::close() [virtual]
```

Closing a unsigned char array has no effect.

Implements Closeable.

Definition at line 29 of file RandomAccessByteArray.cpp.

4.23.3.2 unsigned int RandomAccessByteArray::length ( )

Returns the length of the stream.

Returns

The length.

Definition at line 21 of file RandomAccessByteArray.cpp.

4.23.3.3 bool RandomAccessByteArray::readBoolean() [virtual]

Reads a bool from the stream.

Returns

bool

Implements DataInput.

Definition at line 98 of file RandomAccessByteArray.cpp.

```
unsigned char RandomAccessByteArray::readByte( ) [virtual]
Reads a unsigned char from the stream.
Returns
     unsigned char
Implements DataInput.
Definition at line 94 of file RandomAccessByteArray.cpp.
4.23.3.5 char RandomAccessByteArray::readChar() [virtual]
Reads a char from the stream.
Returns
     char
Implements DataInput.
Definition at line 102 of file RandomAccessByteArray.cpp.
4.23.3.6 double RandomAccessByteArray::readDouble( ) [virtual]
Reads a double from the stream.
Returns
     double
Implements DataInput.
Definition at line 146 of file RandomAccessByteArray.cpp.
4.23.3.7 float RandomAccessByteArray::readFloat( ) [virtual]
Reads a float from the stream.
Returns
     float
Implements DataInput.
Definition at line 142 of file RandomAccessByteArray.cpp.
4.23.3.8 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 150 of file RandomAccessByteArray.cpp.
```

**4.23.3.9** int RandomAccessByteArray::readInt() [virtual]

Reads an int from the stream.

Generated on Tue Aug 18 2015 22:52:10 for Arduino Gyroscope Driver by Doxygen

```
Returns
     int
Implements DataInput.
Definition at line 110 of file RandomAccessByteArray.cpp.
4.23.3.10 long RandomAccessByteArray::readLong() [virtual]
Reads a long from the stream.
Returns
     long
Implements DataInput.
Definition at line 126 of file RandomAccessByteArray.cpp.
4.23.3.11 unsigned char RandomAccessByteArray::readUnsignedChar() [virtual]
Reads an unsigned char from the stream.
Returns
     unsigned char
Implements DataInput.
Definition at line 106 of file RandomAccessByteArray.cpp.
4.23.3.12 unsigned int RandomAccessByteArray::readUnsignedInt() [virtual]
Reads an unsigned int from the stream.
Returns
     unsigned int
Implements DataInput.
Definition at line 118 of file RandomAccessByteArray.cpp.
4.23.3.13 unsigned long RandomAccessByteArray::readUnsignedLong() [virtual]
Reads a unsigned long from the stream.
Returns
     unsigned long
Implements DataInput.
Definition at line 138 of file RandomAccessByteArray.cpp.
4.23.3.14 word RandomAccessByteArray::readWord() [virtual]
Reads a word from the stream.
Returns
     word
Implements DataInput.
Definition at line 122 of file RandomAccessByteArray.cpp.
```

**4.23.3.15** 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.

**4.23.3.16** 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 156 of file RandomAccessByteArray.cpp.

**4.23.3.17** 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.

**4.23.3.18 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.

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

Writes a bool into the stream.

Parameters

ν The bool to be written.

Implements DataOutput.

Definition at line 50 of file RandomAccessByteArray.cpp.

**4.23.3.20** 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.

**4.23.3.21** 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.

**4.23.3.22 void RandomAccessByteArray::writeChar(charc)** [virtual]

Writes a char into the stream.

**Parameters** 

С	The char to be written.
---	-------------------------

Implements DataOutput.

Definition at line 54 of file RandomAccessByteArray.cpp.

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

Writes a double into the stream.

**Parameters** 

V	The double to be written.

Implements DataOutput.

Definition at line 90 of file RandomAccessByteArray.cpp.

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

Writes a float into the stream.

**Parameters** 

V	The float to be written.

Implements DataOutput.

Definition at line 86 of file RandomAccessByteArray.cpp.

4.23.3.25 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.

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

Writes a long into the stream.

**Parameters** 

v The long to be written.

Implements DataOutput.

Definition at line 75 of file RandomAccessByteArray.cpp.

**4.23.3.27** 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.

**4.23.3.28** 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.

4.23.3.29 void RandomAccessByteArray::writeUnsignedLong (unsigned long  $\nu$  ) [virtual]

Writes a unsigned long into the stream.

**Parameters** 

v The unsigned long to be written.

Implements DataOutput.

Definition at line 82 of file RandomAccessByteArray.cpp.

**4.23.3.30 void RandomAccessByteArray::writeWord(word v)** [virtual]

Writes a word into the stream.

**Parameters** 

v The word to be written.

Implements DataOutput.

Definition at line 71 of file RandomAccessByteArray.cpp.

4.23.4 Member Data Documentation

**4.23.4.1** unsigned char\* RandomAccessByteArray::buf [private]

Buffer used to work.

Definition at line 21 of file RandomAccessByteArray.h.

**4.23.4.2 unsigned int RandomAccessByteArray::count** [private]

Buffer size.

Definition at line 26 of file RandomAccessByteArray.h.

**4.23.4.3** unsigned int RandomAccessByteArray::pos [private]

Current position.

Definition at line 31 of file RandomAccessByteArray.h.

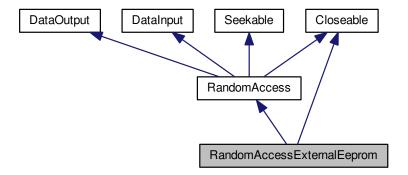
The documentation for this class was generated from the following files:

- · RandomAccessByteArray.h
- RandomAccessByteArray.cpp

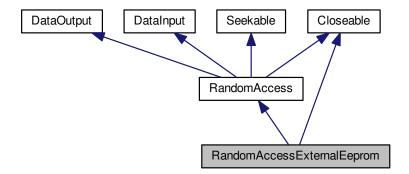
# 4.24 RandomAccessExternalEeprom Class Reference

#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 writeWord (word 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 word readWord ()
- 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

# 4.24.1 Detailed Description

### Araduino 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 19 of file RandomAccessExternalEeprom.h.

# 4.24.2 Constructor & Destructor Documentation

## 4.24.2.1 RandomAccessExternalEeprom::RandomAccessExternalEeprom ( ExternalEeprom \* externalEeprom )

Public constructor.

```
Parameters
```

```
externalEeprom
                     The external eeprom instance to be used.
Definition at line 17 of file RandomAccessExternalEeprom.cpp.
4.24.3 Member Function Documentation
4.24.3.1 void RandomAccessExternalEeprom::close() [virtual]
Closing a external eeprom has no effect.
Implements Closeable.
Definition at line 31 of file RandomAccessExternalEeprom.cpp.
4.24.3.2 unsigned int RandomAccessExternalEeprom::length ( )
Returns the length of the stream.
Returns
     The length.
Definition at line 23 of file RandomAccessExternalEeprom.cpp.
4.24.3.3 bool RandomAccessExternalEeprom::readBoolean() [virtual]
Reads a bool from the stream.
Returns
     bool
Implements DataInput.
Definition at line 100 of file RandomAccessExternalEeprom.cpp.
4.24.3.4 unsigned char RandomAccessExternalEeprom::readByte() [virtual]
Reads a unsigned char from the stream.
Returns
     unsigned char
Implements DataInput.
Definition at line 96 of file RandomAccessExternalEeprom.cpp.
4.24.3.5 char RandomAccessExternalEeprom::readChar() [virtual]
Reads a char from the stream.
Returns
     char
Implements DataInput.
```

Definition at line 104 of file RandomAccessExternalEeprom.cpp.

```
4.24.3.6 double RandomAccessExternalEeprom::readDouble( ) [virtual]
Reads a double from the stream.
Returns
     double
Implements DataInput.
Definition at line 148 of file RandomAccessExternalEeprom.cpp.
4.24.3.7 float RandomAccessExternalEeprom::readFloat() [virtual]
Reads a float from the stream.
Returns
     float
Implements DataInput.
Definition at line 144 of file RandomAccessExternalEeprom.cpp.
4.24.3.8 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 152 of file RandomAccessExternalEeprom.cpp.
4.24.3.9 int RandomAccessExternalEeprom::readInt() [virtual]
Reads an int from the stream.
Returns
     int
Implements DataInput.
Definition at line 112 of file RandomAccessExternalEeprom.cpp.
4.24.3.10 long RandomAccessExternalEeprom::readLong() [virtual]
Reads a long from the stream.
Returns
     long
Implements DataInput.
Definition at line 128 of file RandomAccessExternalEeprom.cpp.
4.24.3.11 unsigned char RandomAccessExternalEeprom::readUnsignedChar( ) [virtual]
Reads an unsigned char from the stream.
```

```
Returns
     unsigned char
Implements DataInput.
Definition at line 108 of file RandomAccessExternalEeprom.cpp.
4.24.3.12 unsigned int RandomAccessExternalEeprom::readUnsignedInt() [virtual]
Reads an unsigned int from the stream.
Returns
     unsigned int
Implements DataInput.
Definition at line 120 of file RandomAccessExternalEeprom.cpp.
4.24.3.13 unsigned long RandomAccessExternalEeprom::readUnsignedLong() [virtual]
Reads a unsigned long from the stream.
Returns
     unsigned long
Implements DataInput.
Definition at line 140 of file RandomAccessExternalEeprom.cpp.
4.24.3.14 word RandomAccessExternalEeprom::readWord() [virtual]
Reads a word from the stream.
Returns
     word
Implements DataInput.
Definition at line 124 of file RandomAccessExternalEeprom.cpp.
4.24.3.15 void RandomAccessExternalEeprom::seek (unsigned int pos ) [virtual]
Seeks the stream at the position.
Parameters
                     The position.
              pos
Implements Seekable.
Definition at line 27 of file RandomAccessExternalEeprom.cpp.
4.24.3.16 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 158 of file RandomAccessExternalEeprom.cpp.

**4.24.3.17** 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.

**4.24.3.18** 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.

 $\textbf{4.24.3.19} \quad \textbf{void} \; \textbf{RandomAccessExternalEeprom::writeBoolean(bool\, v)} \quad [\, \texttt{virtual} \,]$ 

Writes a bool into the stream.

Parameters

v The bool to be written.
---------------------------

Implements DataOutput.

Definition at line 52 of file RandomAccessExternalEeprom.cpp.

 $\textbf{4.24.3.20} \quad \textbf{void RandomAccessExternalEeprom::writeByte(unsigned char \textit{b})} \quad [\texttt{virtual}]$ 

Writes a unsigned char into the stream.

**Parameters** 



Implements DataOutput.

Definition at line 42 of file RandomAccessExternalEeprom.cpp.

4.24.3.21 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.

**4.24.3.22 void RandomAccessExternalEeprom::writeChar(charc)** [virtual]

Writes a char into the stream.

**Parameters** 

c The char to be written.

Implements DataOutput.

Definition at line 56 of file RandomAccessExternalEeprom.cpp.

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

Writes a double into the stream.

**Parameters** 

V	The double to be written.
---	---------------------------

Implements DataOutput.

Definition at line 92 of file RandomAccessExternalEeprom.cpp.

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

Writes a float into the stream.

**Parameters** 

V	The float to be written.

Implements DataOutput.

Definition at line 88 of file RandomAccessExternalEeprom.cpp.

**4.24.3.25 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.

4.24.3.26 void RandomAccessExternalEeprom::writeLong ( long v ) [virtual]

Writes a long into the stream.

**Parameters** 

V	The long to be written.

Implements DataOutput.

Definition at line 77 of file RandomAccessExternalEeprom.cpp.

 $\textbf{4.24.3.27} \quad \text{void RandomAccessExternalEeprom::writeUnsignedChar ( unsigned char } \textbf{\textit{c}} \text{ )} \quad [\texttt{virtual}]$ 

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.

4.24.3.28 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.

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

Writes a unsigned long into the stream.

**Parameters** 

ν The unsigned long to be written.

Implements DataOutput.

Definition at line 84 of file RandomAccessExternalEeprom.cpp.

**4.24.3.30** void RandomAccessExternalEeprom::writeWord ( word v ) [virtual]

Writes a word into the stream.

Parameters

ν The word to be written.

Implements DataOutput.

Definition at line 73 of file RandomAccessExternalEeprom.cpp.

4.24.4 Member Data Documentation

**4.24.4.1** ExternalEeprom\* RandomAccessExternalEeprom::externalEeprom [private]

The external eeprom to be used.

Definition at line 24 of file RandomAccessExternalEeprom.h.

**4.24.4.2 unsigned int RandomAccessExternalEeprom::pos** [private]

Current position.

Definition at line 29 of file RandomAccessExternalEeprom.h.

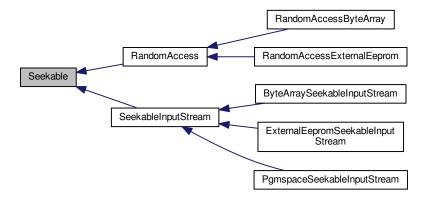
The documentation for this class was generated from the following files:

- · RandomAccessExternalEeprom.h
- · RandomAccessExternalEeprom.cpp

# 4.25 Seekable Class Reference

#include <Seekable.h>

Inheritance diagram for Seekable:



**Public Member Functions** 

virtual void seek (unsigned int pos)=0

# 4.25.1 Detailed Description

Arduino IO.

# Seekable

Definition at line 10 of file Seekable.h.

### 4.25.2 Member Function Documentation

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

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, ExternalEepromSeekableInput← Stream, ByteArraySeekableInputStream, and PgmspaceSeekableInputStream.

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

· Seekable.h

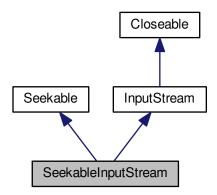
# 4.26 SeekableInputStream Class Reference

#include <SeekableInputStream.h>

Inheritance diagram for SeekableInputStream:



Collaboration diagram for SeekableInputStream:



**Additional Inherited Members** 

4.26.1 Detailed Description

Arduino IO.

SeekableInputStream

Definition at line 13 of file SeekableInputStream.h.

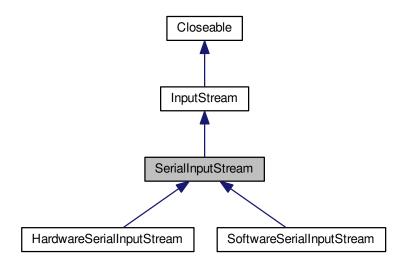
The documentation for this class was generated from the following file:

• SeekableInputStream.h

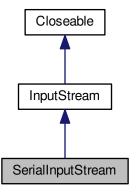
# 4.27 SerialInputStream Class Reference

#include <SerialInputStream.h>

Inheritance diagram for SerialInputStream:



Collaboration diagram for SerialInputStream:



**Additional Inherited Members** 

4.27.1 Detailed Description

Arduino IO.

SerialInputStream

A SerialInputStream obtains input bytes from a serial port.

Definition at line 14 of file SerialInputStream.h.

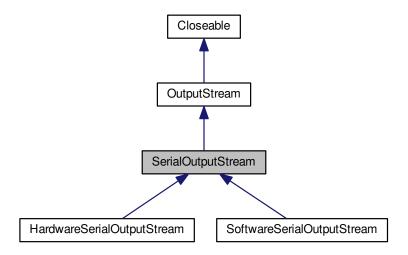
The documentation for this class was generated from the following file:

· SerialInputStream.h

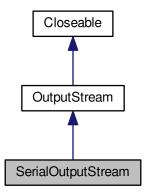
# 4.28 SerialOutputStream Class Reference

#include <SerialOutputStream.h>

Inheritance diagram for SerialOutputStream:



Collaboration diagram for SerialOutputStream:



**Additional Inherited Members** 

4.28.1 Detailed Description

Arduino IO.

# SerialOutputStream

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

Definition at line 14 of file SerialOutputStream.h.

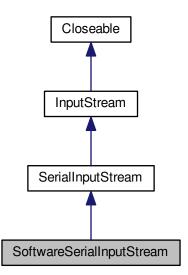
The documentation for this class was generated from the following file:

· SerialOutputStream.h

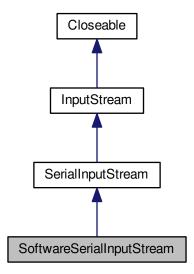
# 4.29 SoftwareSerialInputStream Class Reference

#include <SoftwareSerialInputStream.h>

 $Inheritance\ diagram\ for\ Software Serial Input Stream:$ 



Collaboration diagram for SoftwareSerialInputStream:



### **Public Member Functions**

- SoftwareSerialInputStream (SoftwareSerial \*softwareSerial, unsigned int boudRate)
- virtual int available ()
- virtual int read ()

### **Protected Attributes**

• SoftwareSerial \* softwareSerial

## 4.29.1 Detailed Description

Arduino IO.

SoftwareSerialInputStream

A SoftwareSerialInputStream obtains input bytes from a serial port.

Definition at line 17 of file SoftwareSerialInputStream.h.

# 4.29.2 Constructor & Destructor Documentation

4.29.2.1 SoftwareSerialInputStream::SoftwareSerialInputStream ( SoftwareSerial \* softwareSerial, unsigned int boudRate )

Public constructor.

**Parameters** 

serial	
boudRate	

Definition at line 14 of file SoftwareSerialInputStream.cpp.

4.29.3 Member Function Documentation

```
4.29.3.1 int SoftwareSerialInputStream::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 20 of file SoftwareSerialInputStream.cpp.

```
4.29.3.2 int SoftwareSerialInputStream::read() [virtual]
```

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

Implements InputStream.

Definition at line 24 of file SoftwareSerialInputStream.cpp.

4.29.4 Member Data Documentation

 $\textbf{4.29.4.1} \quad \textbf{SoftwareSerial} * \textbf{SoftwareSerialInputStream::softwareSerial} \quad \texttt{[protected]}$ 

The software serial where the data will be read.

Definition at line 23 of file SoftwareSerialInputStream.h.

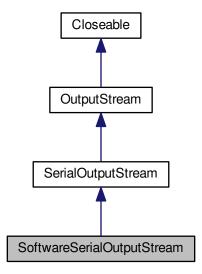
The documentation for this class was generated from the following files:

- · SoftwareSerialInputStream.h
- · SoftwareSerialInputStream.cpp

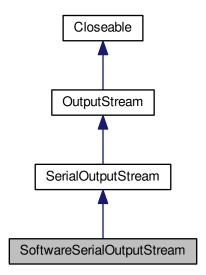
4.30 SoftwareSerialOutputStream Class Reference

#include <SoftwareSerialOutputStream.h>

Inheritance diagram for SoftwareSerialOutputStream:



Collaboration diagram for SoftwareSerialOutputStream:



# **Public Member Functions**

- SoftwareSerialOutputStream (SoftwareSerial \*serial, unsigned int boudRate)
- virtual void write (unsigned char b)

### **Protected Attributes**

• SoftwareSerial \* softwareSerial

## 4.30.1 Detailed Description

Arduino IO.

## SoftwareSerialOutputStream

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

Definition at line 17 of file SoftwareSerialOutputStream.h.

4.30.2 Constructor & Destructor Documentation

4.30.2.1 SoftwareSerialOutputStream::SoftwareSerialOutputStream ( SoftwareSerial \* serial, unsigned int boudRate )

Definition at line 14 of file SoftwareSerialOutputStream.cpp.

4.30.3 Member Function Documentation

**4.30.3.1 void SoftwareSerialOutputStream::write (unsigned char b)** [virtual]

Writes the specified unsigned char to this output stream.

Implements OutputStream.

Definition at line 20 of file SoftwareSerialOutputStream.cpp.

4.30.4 Member Data Documentation

**4.30.4.1 SoftwareSerial\* SoftwareSerialOutputStream::softwareSerial** [protected]

Definition at line 23 of file SoftwareSerialOutputStream.h.

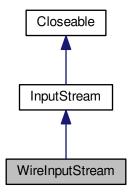
The documentation for this class was generated from the following files:

- · SoftwareSerialOutputStream.h
- SoftwareSerialOutputStream.cpp

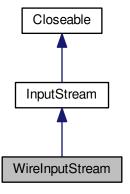
## 4.31 WireInputStream Class Reference

#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

### 4.31.1 Detailed Description

Arduino IO.

WireInputStream

A WireInputStream obtains input bytes from the wire bus.

Definition at line 16 of file WireInputStream.h.

#### 4.31.2 Constructor & Destructor Documentation

## 4.31.2.1 WireInputStream::WireInputStream ( unsigned char addredd )

Public constructor.

**Parameters** 

```
address
```

Definition at line 14 of file WireInputStream.cpp.

### 4.31.3 Member Function Documentation

```
4.31.3.1 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.

```
4.31.3.2 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.

```
4.31.3.3 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.

#### 4.31.4 Member Data Documentation

**4.31.4.1** unsigned char WireInputStream::address [protected]

The wire device address.

5 File Documentation 103

Definition at line 22 of file WireInputStream.h.

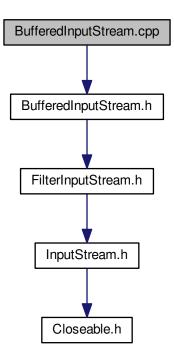
The documentation for this class was generated from the following files:

- · WireInputStream.h
- WireInputStream.cpp

## 5 File Documentation

## 5.1 BufferedInputStream.cpp File Reference

#include "BufferedInputStream.h"
Include dependency graph for BufferedInputStream.cpp:



### Macros

- #define \_\_ARDUINO\_IO\_BUFFERED\_INPUT\_STREAM\_CPP\_\_ 1
- 5.1.1 Macro Definition Documentation
- 5.1.1.1 #define \_\_ARDUINO\_IO\_BUFFERED\_INPUT\_STREAM\_CPP\_\_ 1

Arduino 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

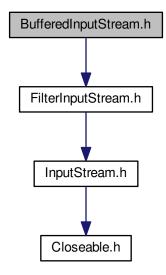
```
00001
00024 #ifndef __ARDUINO_IO_BUFFERED_INPUT_STREAM_CPP_
00025 #define __ARDUINO_IO_BUFFERED_INPUT_STREAM_CPP__ 1
00026
00027 #include "BufferedInputStream.h"
00029 BufferedInputStream::BufferedInputStream(
     InputStream* in,
00030
             unsigned char* buf, int size)
00031
              FilterInputStream(in), buf(buf) {
00032
         this->size = size;
00033
          count = 0;
         pos = 0;
00034
00035 }
00036
00037 int BufferedInputStream::available() {
00038
          return in->available() + (count - pos);
00039 }
00040
00041 void BufferedInputStream::close() {
00042
          in->close();
00043 }
00044
00045 void BufferedInputStream::reset() {
00046
       if (marked) {
00047
             pos = markpos;
00048
00049 }
00050
00051 int BufferedInputStream::read(unsigned char* b, int len) {
00052
          return read(b, 0, len);
00053 }
00054
00055 int BufferedInputStream::read(unsigned char* b, int off, int len) {
00056
          int cnt, available;
          available = count - pos;
00058
00059
00060
           * The needed data are already in the buffer?
00061
00062
          if (available >= len) {
              for (int i = 0; i < len; i++) {
    b[off + i] = buf[pos + i];</pre>
00063
00064
00065
              pos += len;
00066
00067
              return len;
00068
          }
00069
00070
00071
          * The buffer data is not enough, but is necessary.
00072
00073
          for (int i = 0; i < available; i++) {</pre>
00074
             b[off + i] = buf[pos + i];
00075
00076
          marked = false;
00077
          pos = 0;
00078
          count = 0;
00079
08000
00081
          * Reads the rest from the stream.
00082
00083
          cnt = in->read(b, off + available, len - available);
00084
00085
          * Tests if we had enough data.
00086
00087
88000
          if (cnt < 0) {</pre>
00089
              return available;
00090
          } else if (cnt < (len - available)) {</pre>
00091
             return available + cnt;
00092
          } else {
00093
              fill(0);
00094
00095
          return len;
```

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

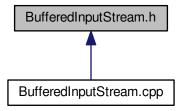
### 5.3 BufferedInputStream.h File Reference

#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

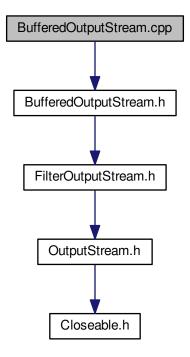
```
00001
00024 #ifndef __ARDUINO_IO_BUFFERED_INPUT_STREAM_H_
00025 #define __ARDUINO_IO_BUFFERED_INPUT_STREAM_H_ 1
00026
00027 #include <FilterInputStream.h>
00028
00029 class BufferedInputStream : public FilterInputStream {
00030
00034    unsigned int size;
00035
```

```
00036 protected:
00037
00041
          unsigned char* buf;
00042
00052
          int count;
00053
          int pos;
00068
00093
          int markpos;
00094
          bool marked;
00098
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
          virtual int read(unsigned char* b, int off, int len);
00153
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 /* __ARDUINO_IO_BUFFERED_INPUT_STREAM_H__ */
```

## 5.5 BufferedOutputStream.cpp File Reference

#include "BufferedOutputStream.h"

Include dependency graph for BufferedOutputStream.cpp:



### Macros

#define \_\_ARDUINO\_IO\_BUFFERED\_OUTPUT\_STREAM\_CPP\_\_ 1

#### 5.5.1 Macro Definition Documentation

```
5.5.1.1 #define __ARDUINO_IO_BUFFERED_OUTPUT_STREAM_CPP__1
```

Arduino 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.

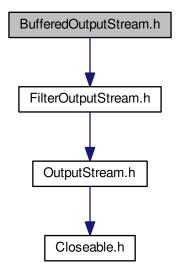
## 5.6 BufferedOutputStream.cpp

```
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
          \star If the request length exceeds the size of the output buffer,
00037
          * 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++) {</pre>
             buf[count + i] = b[off + i];
00050
00051
00052
          count += len;
00053 }
00054
00055 void BufferedOutputStream::flush() {
00056
         flushBuffer();
00057
         out->flush();
00058 }
00059
00060 void BufferedOutputStream::close() {
       flush();
00061
00062
         out->close();
00063 }
00065 void BufferedOutputStream::flushBuffer() {
00066 if (count > 0) {
             out->write(buf, 0, count);
00067
00068
              count = 0;
00069
         }
00070 }
00071
00072 #endif /* __ARDUINO_IO_BUFFERED_OUTPUT_STREAM_CPP__ */
```

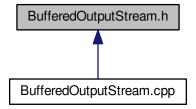
### 5.7 BufferedOutputStream.h File Reference

#include <FilterOutputStream.h>

Include dependency graph for BufferedOutputStream.h:



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



### Classes

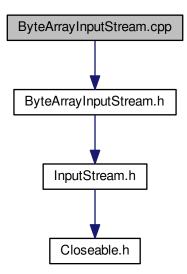
· class BufferedOutputStream

# 5.8 BufferedOutputStream.h

```
00024
          int size;
00029
00036
          int count;
00037
00038 public:
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 /* __ARDUINO_IO_BUFFERED_OUTPUT_STREAM_H__ */
```

# 5.9 ByteArrayInputStream.cpp File Reference

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



#### Macros

• #define \_\_ARDUINO\_IO\_BYTE\_ARRAY\_INPUT\_STREAM\_CPP\_\_ 1

### 5.9.1 Macro Definition Documentation

5.9.1.1 #define \_\_ARDUINO\_IO\_BYTE\_ARRAY\_INPUT\_STREAM\_CPP\_\_ 1

Arduino 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.

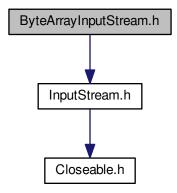
# 5.10 ByteArrayInputStream.cpp

```
00001
00010 #ifndef __ARDUINO_IO_BYTE_ARRAY_INPUT_STREAM_CPP_
00011 #define __ARDUINO_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
             buf(buf), count(count) {
00018
         markpos = 0;
         pos = 0;
00019
00020 }
00021
00022 int ByteArrayInputStream::available() {
       if ((count - pos) > 0) {
00023
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 }
00037 int ByteArrayInputStream::read() {
00039
              return -1;
00040
         return buf[pos++];
00041
00042 }
00043
00044 void ByteArrayInputStream::reset() {
00045
        pos = markpos;
00046 }
00047
00048 #endif /* __ARDUINO_IO_BYTE_ARRAY_INPUT_STREAM_CPP__ */
```

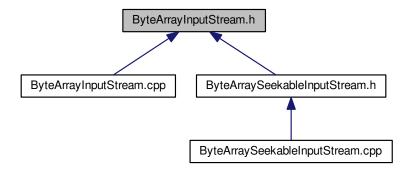
## 5.11 ByteArrayInputStream.h File Reference

#include <InputStream.h>

Include dependency graph for ByteArrayInputStream.h:



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



#### Classes

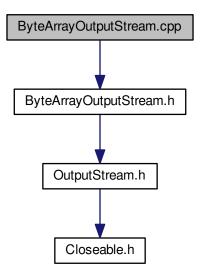
• class ByteArrayInputStream

# 5.12 ByteArrayInputStream.h

```
/\star \star The number of valid bytes in the buffer.
00024
00025
          unsigned int count;
00026
00027
00028
00029
           * Current position
00030
00031
          unsigned int pos;
00032
00033
00034
           \star 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
00054
          virtual int available();
00055
00059
          virtual void mark();
00060
00066
          virtual bool markSupported();
00067
00071
          using InputStream::read;
00072
00078
          virtual int read();
00079
00084
          virtual void reset();
00085 };
00086
00087 #endif /* __ARDUINO_IO_BYTE_ARRAY_INPUT_STREAM_H__ */
```

## 5.13 ByteArrayOutputStream.cpp File Reference

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



Macros

#define \_\_ARDUINO\_IO\_BYTE\_ARRAY\_OUTPUT\_STREAM\_CPP\_\_ 1

#### 5.13.1 Macro Definition Documentation

```
5.13.1.1 #define __ARDUINO_IO_BYTE_ARRAY_OUTPUT_STREAM_CPP__ 1
```

Arduino 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.

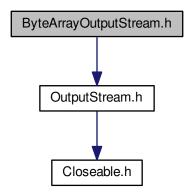
## 5.14 ByteArrayOutputStream.cpp

```
00001
00010 #ifndef __ARDUINO_IO_BYTE_ARRAY_OUTPUT_STREAM_CPP__
00011 #define __ARDUINO_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 }
00029 unsigned char* ByteArrayOutputStream::toByteArray() {
00030
00031 }
00032
00033 void ByteArrayOutputStream::write(unsigned char b) {
00034
         buf[pos++] = b;
00035 }
00036
00037 #endif /* __ARDUINO_IO_BYTE_ARRAY_OUTPUT_STREAM_CPP__ */
```

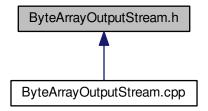
### 5.15 ByteArrayOutputStream.h File Reference

#include <OutputStream.h>

Include dependency graph for ByteArrayOutputStream.h:



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



### Classes

• class ByteArrayOutputStream

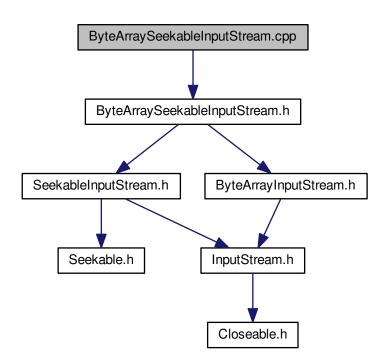
# 5.16 ByteArrayOutputStream.h

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

```
00026
          unsigned int count;
00027
00028
           * Current position
00029
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 /* __ARDUINO_IO_BYTE_ARRAY_OUTPUT_STREAM_H__ */
```

## 5.17 ByteArraySeekableInputStream.cpp File Reference

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



### Macros

• #define \_\_ARDUINO\_IO\_BYTE\_ARRAY\_SEEKABLE\_INPUT\_STREAM\_CPP\_\_ 1

### 5.17.1 Macro Definition Documentation

```
5.17.1.1 #define __ARDUINO_IO_BYTE_ARRAY_SEEKABLE_INPUT_STREAM_CPP__ 1
```

Arduino IO.

### ByteArraySeekableInputStream

A ByteArraySeekableInputStream obtains input bytes from a resource in a file system that implements Seekable← InputStream interface.

Definition at line 11 of file ByteArraySeekableInputStream.cpp.

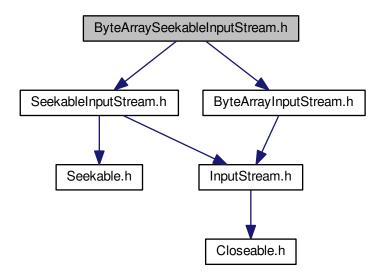
### 5.18 ByteArraySeekableInputStream.cpp

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

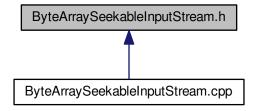
### 5.19 ByteArraySeekableInputStream.h File Reference

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

Include dependency graph for ByteArraySeekableInputStream.h:



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



#### Classes

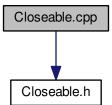
• class ByteArraySeekableInputStream

### 5.20 ByteArraySeekableInputStream.h

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

### 5.21 Closeable.cpp File Reference

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



**Macros** 

• #define \_\_ARDUINO\_IO\_CLOSEABLE\_CPP\_\_ 1

#### 5.21.1 Macro Definition Documentation

```
5.21.1.1 #define __ARDUINO_IO_CLOSEABLE_CPP__ 1
```

Arduino IO.

### Closeable

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

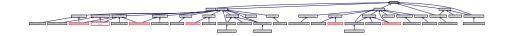
Definition at line 10 of file Closeable.cpp.

# 5.22 Closeable.cpp

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

00010 #define __ARDUINO_IO_CLOSEABLE_H__ 1

00011

00012 class Closeable {

00013 public:

00014

00015 virtual void close() = 0;

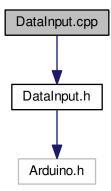
00016 };

00017

00018 #endif /* __ARDUINO_IO_CLOSEABLE_H__ */
```

### 5.25 DataInput.cpp File Reference

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



#### Macros

• #define \_\_ARDUINO\_IO\_DATA\_INPUT\_CPP\_\_ 1

### 5.25.1 Macro Definition Documentation

```
5.25.1.1 #define __ARDUINO_IO_DATA_INPUT_CPP__ 1
```

Arduino 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.

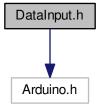
# 5.26 DataInput.cpp

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

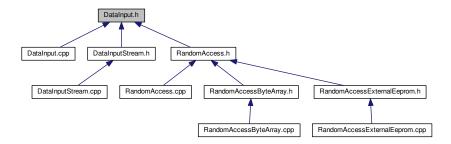
## 5.27 DataInput.h File Reference

#include <Arduino.h>

Include dependency graph for DataInput.h:



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



### Classes

class DataInput

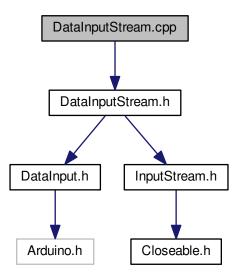
## 5.28 DataInput.h

```
00011 #ifndef __ARDUINO_IO_DATA_INPUT_H__
00012 #define __ARDUINO_IO_DATA_INPUT_H__
00013
00014 #include <Arduino.h>
00015
00016 class DataInput {
00017 public:
00018
00024
          virtual unsigned char readByte() = 0;
00025
00031
          virtual bool readBoolean() = 0;
00032
00038
          virtual char readChar() = 0;
00039
00045
          virtual unsigned char readUnsignedChar() = 0;
00046
00052
          virtual int readInt() = 0;
00053
00059
          virtual unsigned int readUnsignedInt() = 0;
00060
00066
          virtual word readWord() = 0;
00067
00073
          virtual long readLong() = 0;
00074
00080
          virtual unsigned long readUnsignedLong() = 0;
00081
```

```
00087     virtual float readFloat() = 0;
00088
00094     virtual double readDouble() = 0;
00095
00102     virtual void readFully(unsigned char* b, int len) = 0;
00103
00110     virtual unsigned int skipBytes(unsigned int n) = 0;
00111 };
00112
00113 #endif /* __ARDUINO_IO_DATA_INPUT_H__ */
```

## 5.29 DataInputStream.cpp File Reference

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



#### Macros

• #define \_\_ARDUINO\_IO\_DATA\_INPUT\_STREAM\_CPP\_\_ 1

### 5.29.1 Macro Definition Documentation

```
5.29.1.1 #define __ARDUINO_IO_DATA_INPUT_STREAM_CPP__1
```

Arduino 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

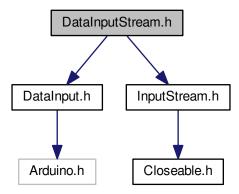
```
00001
00009 #ifndef __ARDUINO_IO_DATA_INPUT_STREAM_CPP__
```

```
00010 #define __ARDUINO_IO_DATA_INPUT_STREAM_CPP__ 1
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();
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 word DataInputStream::readWord() {
00047
         return (word) readInt();
00048 }
00049
00050 long DataInputStream::readLong() {
00051
         long v = 0;
          v = inputStream->read();
00052
00053
         v <<= 8;
00054
         v |= (inputStream->read() & 0xff);
00055
          v <<= 8;
00056
         v |= (inputStream->read() & 0xff);
00057
         v <<= 8;
         v |= (inputStream->read() & 0xff);
00058
00059
          return v:
00060 }
00061
00062 unsigned long DataInputStream::readUnsignedLong() {
00063
         return (unsigned long) readLong();
00064 }
00065
00066 float DataInputStream::readFloat() {
00067
         return (float) readLong();
00068 }
00069
00070 double DataInputStream::readDouble() {
00071
         return (double) readLong();
00072 }
00074 void DataInputStream::readFully(unsigned char* b, int len) {
00075
         for (int i = 0; i < len; i++) {</pre>
             b[i] = inputStream->read();
00076
00077
00078 }
00080 unsigned int DataInputStream::skipBytes(unsigned int n) {
00081
          return inputStream->skip(n);
00082 }
00083
00084 #endif /* __ARDUINO_IO_DATA_INPUT_STREAM_CPP__ */
```

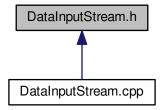
### 5.31 DataInputStream.h File Reference

```
#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

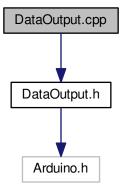
## 5.32 DataInputStream.h

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

```
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 word readWord();
00079
00085
          virtual long readLong();
00086
00092
          virtual unsigned long readUnsignedLong();
00093
00099
          virtual float readFloat();
00100
00106
          virtual double readDouble();
00107
00114
          virtual void readFully(unsigned char* b, int len);
00115
          virtual unsigned int skipBytes(unsigned int n);
00122
00123 };
00124
00125 #endif /* __ARDUINO_IO_DATA_INPUT_STREAM_H_ */
```

## 5.33 DataOutput.cpp File Reference

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



## Macros

• #define \_\_ARDUINO\_IO\_DATA\_OUTPUT\_CPP\_\_ 1

### 5.33.1 Macro Definition Documentation

5.33.1.1 #define \_\_ARDUINO\_IO\_DATA\_OUTPUT\_CPP\_\_ 1

Arduino 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.

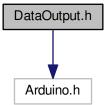
Definition at line 11 of file DataOutput.cpp.

# 5.34 DataOutput.cpp

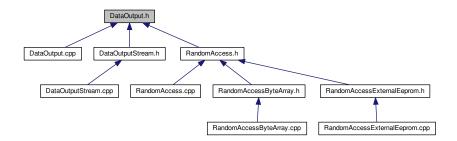
```
00001
00010 #ifndef _ARDUINO_IO_DATA_OUTPUT_CPP_
00011 #define _ARDUINO_IO_DATA_OUTPUT_CPP_ 1
00012
00013 #include "DataOutput.h"
00014
00015 #endif /* _ARDUINO_IO_DATA_OUTPUT_CPP_ */
00016
```

### 5.35 DataOutput.h File Reference

```
#include <Arduino.h>
Include dependency graph for DataOutput.h:
```



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



#### Classes

· class DataOutput

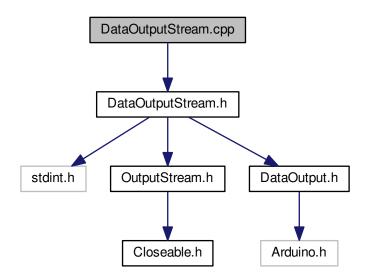
## 5.36 DataOutput.h

```
00001 #ifndef __ARDUINO_IO_DATA_OUTPUT_H_ 00011 #define __ARDUINO_IO_DATA_OUTPUT_H_ 1
```

```
00013 #include <Arduino.h>
00015 class DataOutput {
00016 public:
00017
00024
          virtual void write(unsigned char* b, int len) = 0;
00025
00031
          virtual void write(unsigned char b) = 0;
00032
00038
          virtual void writeByte(unsigned char b) = 0;
00039
00046
          virtual void writeBytes(unsigned char* b, int len) = 0;
00047
00053
          virtual void writeBoolean(bool v) = 0;
00054
00060
          virtual void writeChar(char c) = 0;
00061
00067
          virtual void writeUnsignedChar(unsigned char c) = 0;
00068
00074
          virtual void writeInt(int v) = 0;
00075
00081
          virtual void writeUnsignedInt(unsigned int v) = 0;
00082
00088
          virtual void writeWord(word v) = 0;
00089
00095
          virtual void writeLong(long v) = 0;
00096
00102
          virtual void writeUnsignedLong(unsigned long v) = 0;
00103
00109
          virtual void writeFloat(float v) = 0;
00110
00116
          virtual void writeDouble(double v) = 0;
00117 };
00118
00119 #endif /* __ARDUINO_IO_DATA_OUTPUT_H__ */
```

### 5.37 DataOutputStream.cpp File Reference

#include "DataOutputStream.h"
Include dependency graph for DataOutputStream.cpp:



### Macros

#define \_\_ARDUINO\_IO\_DATA\_OUTPUT\_STREAM\_CPP\_\_ 1

#### 5.37.1 Macro Definition Documentation

### 5.37.1.1 #define \_\_ARDUINO\_IO\_DATA\_OUTPUT\_STREAM\_CPP\_\_ 1

Arduino IO.

#### DataOutputStream

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

Definition at line 10 of file DataOutputStream.cpp.

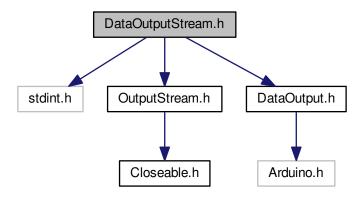
### 5.38 DataOutputStream.cpp

```
00001
00009 #ifndef __ARDUINO_IO_DATA_OUTPUT_STREAM_CPP_
00010 #define
               ARDUINO IO DATA OUTPUT STREAM CPP
00011
00012 #include "DataOutputStream.h"
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++) {</pre>
00032
             outputStream->write(b[i]);
00033
00034 }
00035
00036 void DataOutputStream::writeBoolean(bool v) {
         outputStream->write((unsigned char) v);
00037
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::writeWord(word v) {
00058
          writeInt((int) v);
00059 }
00060
00061 void DataOutputStream::writeLong(long v) {
00062
         outputStream->write((unsigned char) ((v >> 24) & 0xff));
          outputStream->write((unsigned char) ((v >> 16) & Oxff));
outputStream->write((unsigned char) ((v >> 8) & Oxff));
00063
00064
00065
          outputStream->write((unsigned char) (v & 0xff));
00066 }
00067
00068 void DataOutputStream::writeUnsignedLong(unsigned long v) {
00069
          writeLong((long) v);
00070 }
00071
00072 void DataOutputStream::writeFloat(float v) {
00073
         writeLong((long) v);
00074 }
```

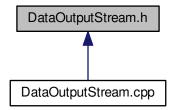
## 5.39 DataOutputStream.h File Reference

```
#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

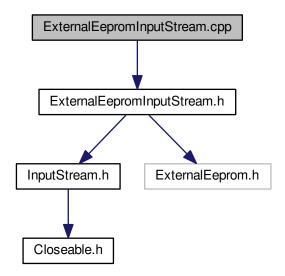
00001

```
00009 #ifndef __ARDUINO_IO_DATA_OUTPUT_STREAM_H_ 00010 #define __ARDUINO_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 writeWord(word v);
00103
          virtual void writeLong(long v);
00109
00110
00116
          virtual void writeUnsignedLong(unsigned long v);
00117
00123
          virtual void writeFloat(float v);
00124
00130
          virtual void writeDouble(double v);
00131 };
00132
00133 #endif /* __ARDUINO_IO_DATA_OUTPUT_STREAM_H__ */
```

## 5.41 ExternalEepromInputStream.cpp File Reference

#include "ExternalEepromInputStream.h"

Include dependency graph for ExternalEepromInputStream.cpp:



#### Macros

#define \_\_ARDUINO\_IO\_EXTERNAL\_EEPROM\_INPUT\_STREAM\_CPP\_\_ 1

### 5.41.1 Macro Definition Documentation

```
5.41.1.1 #define __ARDUINO_IO_EXTERNAL_EEPROM_INPUT_STREAM_CPP__ 1
```

Arduino IO.

## ExternalEepromInputStream

An ExternalEepromInputStream obtains input bytes from a externalEeprom.

Definition at line 11 of file ExternalEepromInputStream.cpp.

## 5.42 ExternalEepromInputStream.cpp

```
00010 #ifndef __ARDUINO_IO_EXTERNAL_EEPROM_INPUT_STREAM_CPP_
00011 #define __ARDUINO_IO_EXTERNAL_EEPROM_INPUT_STREAM_CPP__
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) {
   return -1;
00040
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);
          int cnt;
00048
          len = (int) ((unsigned int) len > available) ? available : len;
          cnt = externalEeprom->readBytes(pos, &b[off], len);
00049
00050
          pos += cnt;
00051
          return cnt;
00052 }
00053
00054 void ExternalEepromInputStream::reset() {
00055
          pos = markpos;
00056 }
00057
00058 #endif /* __ARDUINO_IO_EXTERNAL_EEPROM_INPUT_STREAM_CPP__ */
```

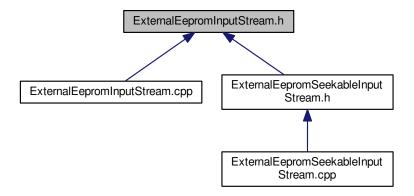
## 5.43 ExternalEepromInputStream.h File Reference

```
#include <InputStream.h>
#include <ExternalEeprom.h>
Include dependency graph for ExternalEepromInputStream.h:
```

InputStream.h ExternalEeprom.h

Closeable.h

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



#### Classes

· class ExternalEepromInputStream

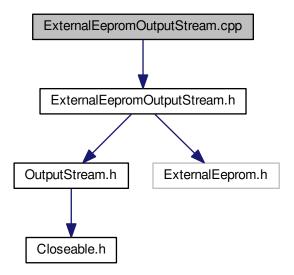
## 5.44 ExternalEepromInputStream.h

```
00010 #ifndef __ARDUINO_IO_EXTERNAL_EEPROM_INPUT_STREAM_H_
00011 #define __ARDUINO_IO_EXTERNAL_EEPROM_INPUT_STREAM_H__
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
           \star The currently marked position in the stream.
00030
00031
00032
          unsigned int markpos;
00033
00034
           * The size of the externalEeprom.
00035
00036
00037
          unsigned int externalEepromSize;
00038
00039 public:
00040
00046
          {\tt 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();
08000
```

```
00089    virtual int read(unsigned char* b, int off, int len);
00090
00095    virtual void reset();
00096 };
00097
00098 #endif /* __ARDUINO_IO_EXTERNAL_EEPROM_INPUT_STREAM_H__ */
```

## 5.45 ExternalEepromOutputStream.cpp File Reference

#include "ExternalEepromOutputStream.h"
Include dependency graph for ExternalEepromOutputStream.cpp:



#### Macros

#define \_\_ARDUINO\_IO\_EXTERNAL\_EEPROM\_OUTPUT\_STREAM\_CPP\_\_ 1

### 5.45.1 Macro Definition Documentation

5.45.1.1 #define \_\_ARDUINO\_IO\_EXTERNAL\_EEPROM\_OUTPUT\_STREAM\_CPP\_\_ 1

Arduino IO.

### ExternalEepromOutputStream

A external Eeprom output stream is an output stream for writing data to a External Eeprom.

Definition at line 10 of file ExternalEepromOutputStream.cpp.

### 5.46 ExternalEepromOutputStream.cpp

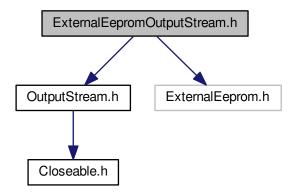
```
00001
00009 #ifndef __ARDUINO_IO_EXTERNAL_EEPROM_OUTPUT_STREAM_CPP__
00010 #define __ARDUINO_IO_EXTERNAL_EEPROM_OUTPUT_STREAM_CPP__ 1
00011
00012 #include "ExternalEepromOutputStream.h"
00013
```

```
00014 ExternalEepromOutputStream::ExternalEepromOutputStream
00015
             ExternalEeprom* externalEeprom) :
00016
             externalEeprom(externalEeprom) {
         pos = 0;
00017
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 /* __ARDUINO_IO_EXTERNAL_EEPROM_OUTPUT_STREAM_CPP__ */
```

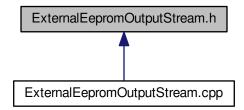
## 5.47 ExternalEepromOutputStream.h File Reference

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

Include dependency graph for ExternalEepromOutputStream.h:



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



#### Classes

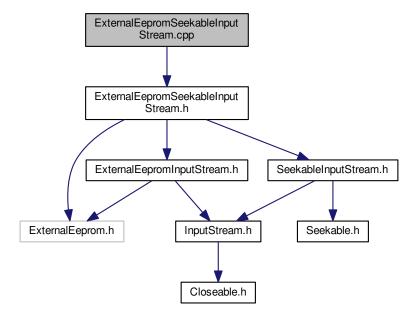
• class ExternalEepromOutputStream

# 5.48 ExternalEepromOutputStream.h

```
00001
00010 #ifndef __ARDUINO_IO_EXTERNAL_EEPROM_OUTPUT_STREAM_H_
00011 #define __ARDUINO_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
          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 };
00060 #endif /* __ARDUINO_IO_EXTERNAL_EEPROM_OUTPUT_STREAM_H__ */
```

### 5.49 ExternalEepromSeekableInputStream.cpp File Reference

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



#### **Macros**

#define \_\_ARDUINO\_IO\_EXTERNAL\_EEPROM\_SEEKABLE\_INPUT\_STREAM\_CPP\_\_1

#### 5.49.1 Macro Definition Documentation

```
5.49.1.1 #define __ARDUINO_IO_EXTERNAL_EEPROM_SEEKABLE_INPUT_STREAM_CPP__1
```

Arduino IO.

#### ExternalEepromSeekableInputStream

A ExternalEepromSeekableInputStream obtains input bytes from a external input stream.

Definition at line 11 of file ExternalEepromSeekableInputStream.cpp.

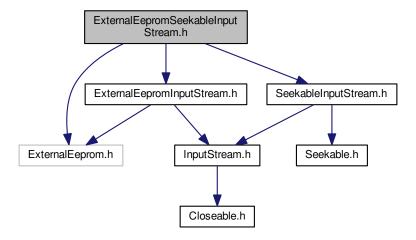
## 5.50 ExternalEepromSeekableInputStream.cpp

```
00010 #ifndef __ARDUINO_IO_EXTERNAL_EEPROM_SEEKABLE_INPUT_STREAM_CPP_
00011 #define __ARDUINO_IO_EXTERNAL_EEPROM_SEEKABLE_INPUT_STREAM_CPP_
00012
00013 #include "ExternalEepromSeekableInputStream.h"
00014
00016
            {\tt ExternalEeprom*\ externalEeprom})\ :
00017
            ExternalEepromInputStream(externalEeprom) {
00018 }
00019
00020 void ExternalEepromSeekableInputStream::seek(unsigned int pos) {
00021
00022 }
00023
00024 #endif /* __ARDUINO_IO_EXTERNAL_EEPROM_SEEKABLE_INPUT_STREAM_CPP__ */
```

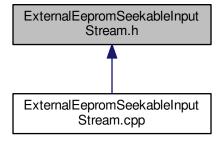
# 5.51 ExternalEepromSeekableInputStream.h File Reference

```
#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

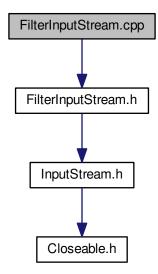
### 5.52 ExternalEepromSeekableInputStream.h

```
00001
00010 #ifndef __ARDUINO_IO_EXTERNAL_EEPROM_SEEKABLE_INPUT_STREAM_H_
00011 #define __ARDUINO_IO_EXTERNAL_EEPROM_SEEKABLE_INPUT_STREAM_H_ 1
00012
00013 #include <ExternalEeprom.h>
00014 #include <SeekableInputStream.h>
00015 #include <ExternalEepromInputStream.h>
00016
{\tt 00017 \ class \ External Eeprom See kable Input Stream \ : \ public}
UUU18 public SeekableInputStream {
00019 public:
00020
00026
          {\tt ExternalEepromSeekableInputStream\,(ExternalEeprom*}
externalEeprom);
00027
00033
          virtual void seek (unsigned int pos);
00034 };
00036 #endif /* __ARDUINO_IO_EXTERNAL_EEPROM_SEEKABLE_INPUT_STREAM_H__ */
```

## 5.53 FilterInputStream.cpp File Reference

#include "FilterInputStream.h"

Include dependency graph for FilterInputStream.cpp:



#### Macros

• #define \_\_ARDUINO\_IO\_FILTER\_INPUT\_STREAM\_CPP\_\_ 1

#### 5.53.1 Macro Definition Documentation

```
5.53.1.1 #define __ARDUINO_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

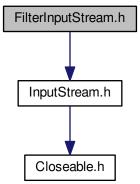
```
00001
00016 #ifndef __ARDUINO_IO_FILTER_INPUT_STREAM_CPP__ 00017 #define __ARDUINO_IO_FILTER_INPUT_STREAM_CPP__ 1
00018
00019 #include "FilterInputStream.h"
{\tt 00021\ FilterInputStream::FilterInputStream(}
      InputStream* in) :
00022
               in(in) {
00023 }
00024
00025 int FilterInputStream::read() {
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) {
         return in->skip(n);
00039 }
00040
00041 int FilterInputStream::available() {
00042    return in->available();
          return in->available();
00043 }
00044
00045 void FilterInputStream::close() {
00046
          in->close();
00047 }
00048
00049 void FilterInputStream::mark() {
00050
         in->mark();
00052
00053 void FilterInputStream::reset() {
00054
        in->reset();
00055 }
00056
00057 bool FilterInputStream::markSupported() {
00058
          return in->markSupported();
00059 }
00060
00061 #endif /* __ARDUINO_IO_FILTER_INPUT_STREAM_CPP__ */
```

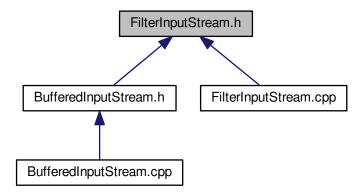
# 5.55 FilterInputStream.h File Reference

#include <InputStream.h>

Include dependency graph for FilterInputStream.h:



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



#### Classes

· class FilterInputStream

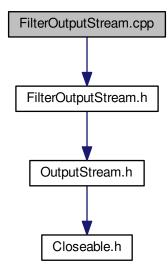
## 5.56 FilterInputStream.h

```
00016 #ifndef __ARDUINO_IO_FILTER_INPUT_STREAM_H_
00017 #define __ARDUINO_IO_FILTER_INPUT_STREAM_H_
00018
00019 #include <InputStream.h>
00020
00021 class FilterInputStream : public virtual InputStream {
00022
00023 protected:
00024
00028
          InputStream* in;
00029
          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 /* __ARDUINO_IO_FILTER_INPUT_STREAM_H_ */
```

## 5.57 FilterOutputStream.cpp File Reference

#include "FilterOutputStream.h"

Include dependency graph for FilterOutputStream.cpp:



#### Macros

#define \_\_ARDUINO\_IO\_FILTER\_OUTPUT\_STREAM\_CPP\_\_ 1

## 5.57.1 Macro Definition Documentation

```
5.57.1.1 #define __ARDUINO_IO_FILTER_OUTPUT_STREAM_CPP__ 1
```

Arduino 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.

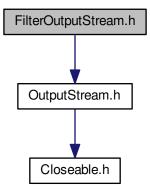
Definition at line 20 of file FilterOutputStream.cpp.

## 5.58 FilterOutputStream.cpp

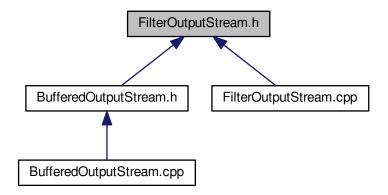
```
00029
          out->write(b);
00030 }
00031
00032 void FilterOutputStream::write(unsigned char* b, int len) {
00033
         out->write(b, len);
00034 }
00036 void FilterOutputStream::write(unsigned char* b, int off, int len) {
00037
         out->write(b, off, len);
00038 }
00039
00040 void FilterOutputStream::flush() {
00041
         out->flush();
00042 }
00043
00044 void FilterOutputStream::close() {
00045    out->flush();
00046
         out->close();
00048
00049 #endif /* __ARDUINO_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

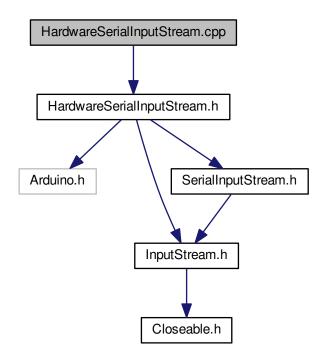
# 5.60 FilterOutputStream.h

```
00001
00019 #ifndef __ARDUINO_IO_FILTER_OUTPUT_STREAM_H_
00020 #define __ARDUINO_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 /* __ARDUINO_IO_FILTER_OUTPUT_STREAM_H__ */
```

# 5.61 HardwareSerialInputStream.cpp File Reference

#include "HardwareSerialInputStream.h"

Include dependency graph for HardwareSerialInputStream.cpp:



### Macros

#define \_\_ARDUINO\_IO\_HARDWARE\_SERIAL\_INPUT\_STREAM\_CPP\_\_ 1

#### 5.61.1 Macro Definition Documentation

5.61.1.1 #define \_\_ARDUINO\_IO\_HARDWARE\_SERIAL\_INPUT\_STREAM\_CPP\_\_1

Arduino IO.

## HardwareSerialInputStream

A HardwareSerialInputStream obtains input bytes from a serial port.

Definition at line 10 of file HardwareSerialInputStream.cpp.

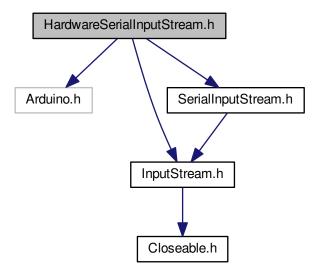
# 5.62 HardwareSerialInputStream.cpp

```
00020 }
00021
00022 int HardwareSerialInputStream::read() {
00023          if (available() > 0) {
00024              return Serial.read();
00025          }
00026          return -1;
00027 }
00028
00029 #endif /* _ARDUINO_IO_HARDWARE_SERIAL_INPUT_STREAM_CPP__ */
```

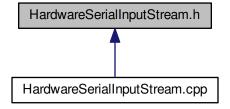
# 5.63 HardwareSerialInputStream.h File Reference

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

Include dependency graph for HardwareSerialInputStream.h:



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



### Classes

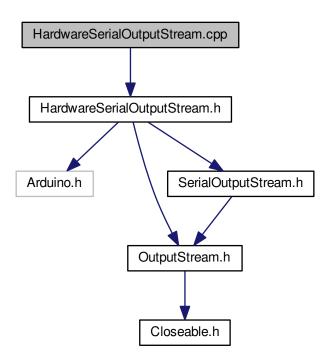
• class HardwareSerialInputStream

# 5.64 HardwareSerialInputStream.h

```
00001
00009 #ifndef __ARDUINO_IO_HARDWARE_SERIAL_INPUT_STREAM_H__ 00010 #define __ARDUINO_IO_HARDWARE_SERIAL_INPUT_STREAM_H__ 1
00011
00012 #include <Arduino.h>
00013 #include <InputStream.h>
00014 #include <SerialInputStream.h>
00015
00016 class HardwareSerialInputStream : public
      SerialInputStream {
00017 public:
00018
00024
           HardwareSerialInputStream(unsigned int boudRate);
00025
           virtual int available();
00030
00031
00035
           virtual int read();
00036 };
00037
00038 #endif /* __ARDUINO_IO_HARDWARE_SERIAL_INPUT_STREAM_H__ */
```

# 5.65 HardwareSerialOutputStream.cpp File Reference

#include "HardwareSerialOutputStream.h"
Include dependency graph for HardwareSerialOutputStream.cpp:



**Macros** 

#define \_\_ARDUINO\_IO\_HARDWARE\_SERIAL\_OUTPUT\_STREAM\_CPP\_\_ 1

#### 5.65.1 Macro Definition Documentation

```
5.65.1.1 #define __ARDUINO_IO_HARDWARE_SERIAL_OUTPUT_STREAM_CPP__ 1
```

Arduino IO.

### HardwareSerialOutputStream

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

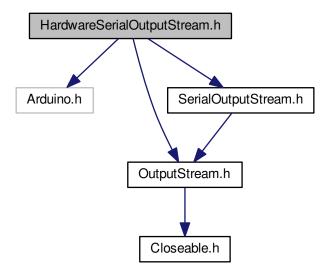
Definition at line 10 of file HardwareSerialOutputStream.cpp.

# 5.66 HardwareSerialOutputStream.cpp

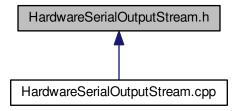
# 5.67 HardwareSerialOutputStream.h File Reference

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

Include dependency graph for HardwareSerialOutputStream.h:



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



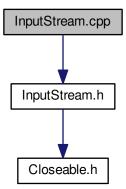
#### Classes

• class HardwareSerialOutputStream

# 5.68 HardwareSerialOutputStream.h

# 5.69 InputStream.cpp File Reference

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



#### Macros

```
• #define __ARDUINO_IO_INPUT_STREAM_CPP__ 1
```

#### 5.69.1 Macro Definition Documentation

```
5.69.1.1 #define __ARDUINO_IO_INPUT_STREAM_CPP__ 1
```

Arduino 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.

# 5.70 InputStream.cpp

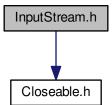
```
00001
00013 #ifndef __ARDUINO_IO_INPUT_STREAM_CPP__
00014 #define __ARDUINO_IO_INPUT_STREAM_CPP__ 1
00015
00016 #include "InputStream.h"
00017
00018 int InputStream::available() {
```

```
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) {
        int i, c;
if (b == (unsigned char*) 0) {
00038
             return 0;
00039
00040
          c = read();
00041
          if (c == -1)
00042
00043
              return -1;
00044
00045
          b[off] = (unsigned char) c;
          for (i = 1; i < len; i++) {
    c = read();
00046
00047
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) {
         unsigned int i;
for (i = 0; i < n && available() > 0; i++) {
00060
00061
00062
              read();
00063
00064
          return i;
00065 }
00066
00067 #endif /* __ARDUINO_IO_INPUT_STREAM_CPP__ */
```

### 5.71 InputStream.h File Reference

#include <Closeable.h>

Include dependency graph for InputStream.h:



5.72 InputStream.h

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



#### Classes

· class InputStream

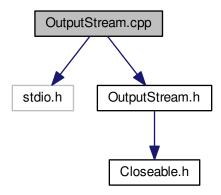
# 5.72 InputStream.h

```
00001
00013 #ifndef __ARDUINO_IO_INPUT_STREAM_H_
00014 #define __ARDUINO_IO_INPUT_STREAM_H_
00016 #include <Closeable.h>
00017
00018 class InputStream : public Closeable {
00019 public:
00020
00026
          virtual int available();
00027
00032
          virtual void close();
00033
00037
          virtual void mark();
00038
00042
          virtual bool markSupported();
00043
00047
          virtual int read() = 0;
00048
00053
          virtual int read(unsigned char* b, int len);
00054
00063
          virtual int read(unsigned char* b, int off, int len);
00064
00069
          virtual void reset();
00070
00074
          virtual unsigned int skip (unsigned int n);
00075 };
00076
00077 #endif /* __ARDUINO_IO_INPUT_STREAM_H__ */
```

## 5.73 OutputStream.cpp File Reference

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

Include dependency graph for OutputStream.cpp:



#### Macros

• #define \_\_ARDUINO\_IO\_OUTPUT\_STREAM\_CPP\_\_ 1

#### 5.73.1 Macro Definition Documentation

```
5.73.1.1 #define __ARDUINO_IO_OUTPUT_STREAM_CPP__1
```

Arduino 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.

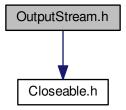
## 5.74 OutputStream.cpp

```
00001
00011 #ifndef __ARDUINO_IO_OUTPUT_STREAM_CPP__
00012 #define __ARDUINO_IO_OUTPUT_STREAM_CPP__ 1
00013
00014 #include <stdio.h>
00015
00016 #include "OutputStream.h"
00017
00018 void OutputStream::write(unsigned char* b, int len) {
00019
           write(b, 0, len);
00020 }
00021
00022 void OutputStream::write(unsigned char* b, int off, int len) {
        if (b == (unsigned char*) 0 || len == 0) {
00023
00024
               return;
00025
          for (int i = 0; i < len; i++) {
    write(b[off + i]);</pre>
00026
00027
00028
00029 }
00030
00031 void OutputStream::flush() {
00032 }
00033
00034 void OutputStream::close() {
00035 }
```

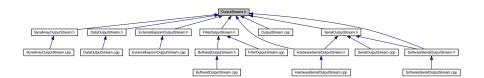
```
00036
00037 #endif /* __ARDUINO_IO_OUTPUT_STREAM_CPP__ */
```

# 5.75 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

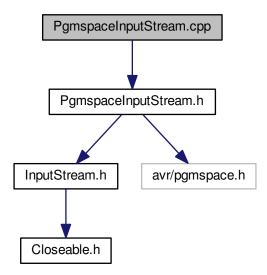
class OutputStream

### 5.76 OutputStream.h

```
00001
00015 #ifndef __ARDUINO_IO_OUTPUT_STREAM_H_
00016 #define __ARDUINO_IO_OUTPUT_STREAM_H__ 1
00017
00018 #include <Closeable.h>
00019
00020 class OutputStream : public Closeable {
00021 public:
00022
00027
           virtual void close();
00028
00033
          virtual void flush();
00034
00038
           virtual void write(unsigned char b) = 0;
00039
00047
           virtual void write(unsigned char* b, int len);
00048
00057
           virtual void write(unsigned char* b, int off, int len);
00058 };
00059
00060 #endif /* __ARDUINO_IO_OUTPUT_STREAM_H__ */
```

# 5.77 PgmspaceInputStream.cpp File Reference

#include "PgmspaceInputStream.h"
Include dependency graph for PgmspaceInputStream.cpp:



#### Macros

#define \_\_ARDUINO\_IO\_PGMSPACE\_INPUT\_STREAM\_CPP\_\_ 1

### 5.77.1 Macro Definition Documentation

5.77.1.1 #define \_\_ARDUINO\_IO\_PGMSPACE\_INPUT\_STREAM\_CPP\_\_ 1

Arduino IO.

# PgmspaceInputStream

A PgmspaceInputStream contains an internal buffer that contains bytes that may be read from the stream mapped to an part of the pgmspace.

Definition at line 11 of file PgmspaceInputStream.cpp.

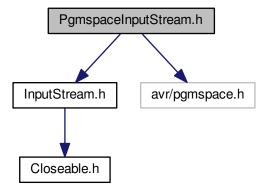
# 5.78 PgmspaceInputStream.cpp

```
00010 #ifndef __ARDUINO_IO_PGMSPACE_INPUT_STREAM_CPP_
00011 #define __ARDUINO_IO_PGMSPACE_INPUT_STREAM_CPP__ 1
00012
00013 #include "PgmspaceInputStream.h"
00014
00015 PgmspaceInputStream::PgmspaceInputStream(char PROGMEM* buf,
     unsigned int count) : buf(buf), count(count) {
00016
         markpos = 0;
00017
         pos = 0;
00018 }
00019
00020 int PgmspaceInputStream::available() {
00021
         if ((count - pos) > 0) {
```

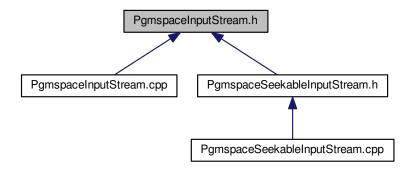
```
return 1;
00023
00024
           return 0;
00025 }
00026
00027 void PgmspaceInputStream::mark() {
          markpos = pos;
00029 }
00030
00031 bool PgmspaceInputStream::markSupported() {
00032 return true;
00033 }
00034
00035 int PgmspaceInputStream::read() {
00036
           return pgm_read_byte(buf + pos++);
00037 }
00038
00039 void PgmspaceInputStream::reset() {
00040    pos = markpos;
00041 }
00042
00043 #endif /* __ARDUINO_IO_PGMSPACE_INPUT_STREAM_CPP__ */
```

# 5.79 PgmspaceInputStream.h File Reference

```
#include <InputStream.h>
#include <avr/pgmspace.h>
Include dependency graph for PgmspaceInputStream.h:
```



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



#### Classes

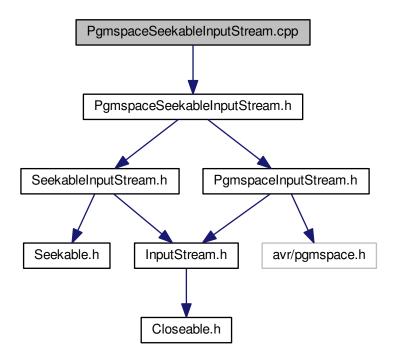
class PgmspaceInputStream

### 5.80 PgmspaceInputStream.h

```
00001
00010 #ifndef __ARDUINO_IO_PGMSPACE_INPUT_STREAM_H__
00011 #define __ARDUINO_IO_PGMSPACE_INPUT_STREAM_H_
00013 #include <InputStream.h>
00014 #include <avr/pgmspace.h>
00015
00016 class PgmspaceInputStream : public virtual InputStream {
00017 protected:
00018
00019
00020
           \star The buffer where data is stored.
00021
          char PROGMEM* buf;
00022
00023
00024
00025
           * The number of valid bytes in the buffer.
00026
00027
          unsigned int count;
00028
00029
00030
          * Current position
00031
00032
          unsigned int pos;
00033
00034
           ^{'} * The currently marked position in the stream.
00035
00036
          unsigned int markpos;
00038
00039 public:
00040
00041
          explicit PgmspaceInputStream(char PROGMEM* buf, unsigned int count);
00042
00049
          virtual int available();
00050
00054
          virtual void mark();
00055
00061
          virtual bool markSupported();
00062
00066
          using InputStream::read;
00067
00073
          virtual int read();
00074
00079
          virtual void reset();
00080 };
00082 #endif /* __ARDUINO_IO_PGMSPACE_INPUT_STREAM_H__ */
```

## 5.81 PgmspaceSeekableInputStream.cpp File Reference

#include "PgmspaceSeekableInputStream.h"
Include dependency graph for PgmspaceSeekableInputStream.cpp:



### Macros

• #define \_\_ARDUINO\_IO\_PGMSPACE\_SEEKABLE\_INPUT\_STREAM\_CPP\_\_ 1

#### 5.81.1 Macro Definition Documentation

5.81.1.1 #define \_\_ARDUINO\_IO\_PGMSPACE\_SEEKABLE\_INPUT\_STREAM\_CPP\_\_ 1

Arduino IO.

## PgmspaceSeekableInputStream

A PgmspaceSeekableInputStream obtains input bytes from a resource in a file system that implements Seekable ← InputStream interface.

Definition at line 11 of file PgmspaceSeekableInputStream.cpp.

# 5.82 PgmspaceSeekableInputStream.cpp

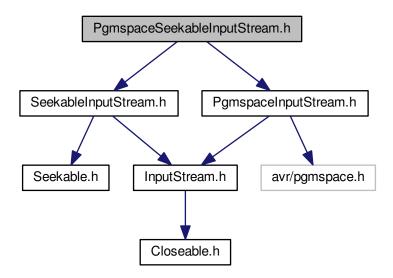
```
00001
00010 #ifndef __ARDUINO_IO_PGMSPACE_SEEKABLE_INPUT_STREAM_CPP_
00011 #define __ARDUINO_IO_PGMSPACE_SEEKABLE_INPUT_STREAM_CPP__ 1
00012
00013 #include "PgmspaceSeekableInputStream.h"
00014
00015 PgmspaceSeekableInputStream::PgmspaceSeekableInputStream
```

```
(char PROGMEM* buf, unsigned int count) : PgmspaceInputStream(buf, count) {
00016 }
00017
00018 void PgmspaceSeekableInputStream::seek(unsigned int pos) {
00019     this->pos = pos;
00020 }
00020 }
00021
00022 #endif /* __ARDUINO_IO_PGMSPACE_SEEKABLE_INPUT_STREAM_CPP__ */
```

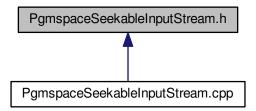
# 5.83 PgmspaceSeekableInputStream.h File Reference

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

Include dependency graph for PgmspaceSeekableInputStream.h:



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



### Classes

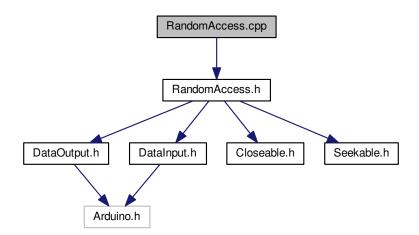
• class PgmspaceSeekableInputStream

### 5.84 PgmspaceSeekableInputStream.h

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

### 5.85 RandomAccess.cpp File Reference

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



#### Macros

• #define ARDUINO IO RANDOM ACCESS CPP 1

#### 5.85.1 Macro Definition Documentation

5.85.1.1 #define \_\_ARDUINO\_IO\_RANDOM\_ACCESS\_CPP\_\_1

Araduino IO.

# RandomAccess

Interface derived from DataInput, DataOutput, Closeable and Seekable.

Definition at line 10 of file RandomAccess.cpp.

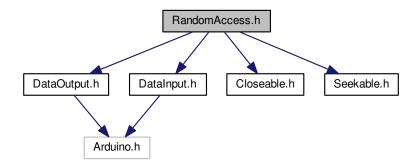
### 5.86 RandomAccess.cpp

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

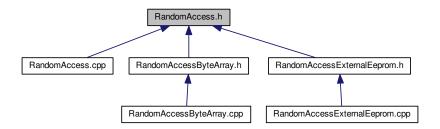
### 5.87 RandomAccess.h File Reference

```
#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

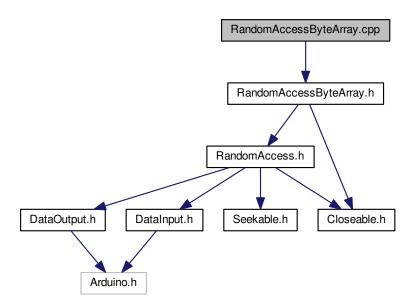
• class RandomAccess

# 5.88 RandomAccess.h

```
00001
00009 #ifndef __ARDUINO_IO_RANDOM_ACCESS_H_
```

## 5.89 RandomAccessByteArray.cpp File Reference

#include "RandomAccessByteArray.h"
Include dependency graph for RandomAccessByteArray.cpp:



#### **Macros**

#define \_\_ARDUINO\_IO\_RANDOM\_ACCESS\_BYTE\_ARRAY\_CPP\_\_ 1

## 5.89.1 Macro Definition Documentation

5.89.1.1 #define \_\_ARDUINO\_IO\_RANDOM\_ACCESS\_BYTE\_ARRAY\_CPP\_\_1

#### Araduino 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.

## 5.90 RandomAccessByteArray.cpp

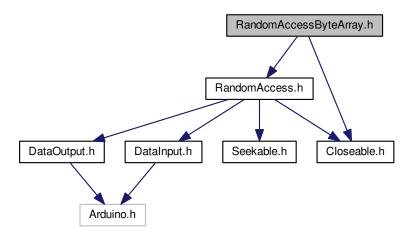
```
00001
00010 #ifndef __ARDUINO_IO_RANDOM_ACCESS_BYTE_ARRAY_CPP_
00011 #define __ARDUINO_IO_RANDOM_ACCESS_BYTE_ARRAY_CPP_ 1
00013 #include "RandomAccessByteArray.h"
00014
00015 RandomAccessByteArray::RandomAccessByteArray(unsigned char* buf
00016
              unsigned int count) :
00017
             buf(buf), count(count) {
00018
00019 }
00020
00021 unsigned int RandomAccessByteArray::length() {
00022
         return count;
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) {
         for (int i = 0; i < len; i++) {
   buf[pos++] = b[i];</pre>
00045
00046
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 }
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::writeWord(word v) {
         writeInt((int) v);
00074
00075 void RandomAccessByteArray::writeLong(long v) {
00076
         buf[pos++] = (unsigned char) ((v >> 24) & 0xff);
          buf[pos++] = (unsigned char) ((v >> 16) & 0xff);
00077
00078
          buf[pos++] = (unsigned char) ((v >> 8) & 0xff);
          buf[pos++] = (unsigned char) (v & 0xff);
00080 }
00081
00082 void RandomAccessByteArray::writeUnsignedLong(unsigned long v) {
00083
         writeLong((long) v);
00084 }
00086 void RandomAccessByteArray::writeFloat(float v) {
00087
          writeLong((long) v);
00088 }
00089
00090 void RandomAccessByteArray::writeDouble(double v) {
00091
         writeLong((long) v);
00092 }
```

```
00093
00094 unsigned char RandomAccessByteArray::readByte() {
00095
          return buf[pos++];
00096 }
00097
00098 bool RandomAccessByteArray::readBoolean() {
00099
         return (bool) buf[pos++];
00100 }
00101
00102 char RandomAccessByteArray::readChar() {
00103
          return (char) buf[pos++];
00104 }
00105
00106 unsigned char RandomAccessByteArray::readUnsignedChar() {
00107
          return (unsigned char) buf[pos++];
00108 }
00109
00110 int RandomAccessByteArray::readInt() {
00111
         int v = 0;
          v = buf[pos++];
00112
00113
          v <<= 8;
00114
          v |= buf[pos++];
00115
          return v;
00116 }
00117
00118 unsigned int RandomAccessByteArray::readUnsignedInt() {
00119
          return (unsigned int) readInt();
00120 }
00121
00122 word RandomAccessByteArray::readWord() {
00123
         return (word) readInt();
00124 }
00125
00126 long RandomAccessByteArray::readLong() {
00127
         long v = 0;
          v = (buf[pos++] & 0xff);
00128
00129
          v <<= 8;
         v |= (buf[pos++] & 0xff);
00130
00131
          v <<= 8;
00132
          v |= (buf[pos++] & 0xff);
00133
          v <<= 8;
          v |= (buf[pos++] & 0xff);
00134
00135
          return v;
00136 }
00137
00138 unsigned long RandomAccessByteArray::readUnsignedLong() {
00139
         return (unsigned long) readLong();
00140 }
00141
00142 float RandomAccessByteArray::readFloat() {
00143
         return (float) readLong();
00144 }
00145
00146 double RandomAccessByteArray::readDouble() {
00147
          return (double) readLong();
00148 }
00150 void RandomAccessByteArray::readFully(unsigned char* b, int len) {
00151
       for (int i = 0; i < len; i++) {
00152
             b[i] = buf[pos++];
00153
00154 }
00155
00156 unsigned int RandomAccessByteArray::skipBytes(unsigned int n) {
00157
         unsigned int skipped;
00158
          unsigned int newpos;
          newpos = pos + n;
if (newpos > count) {
00159
00160
             newpos = count;
00161
00162
00163
          skipped = newpos - pos;
          pos = newpos;
return skipped;
00164
00165
00166 }
00167
00168 #endif /* __ARDUINO_IO_RANDOM_ACCESS_BYTE_ARRAY_CPP__ */
```

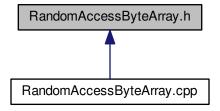
### 5.91 RandomAccessByteArray.h File Reference

```
#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

# 5.92 RandomAccessByteArray.h

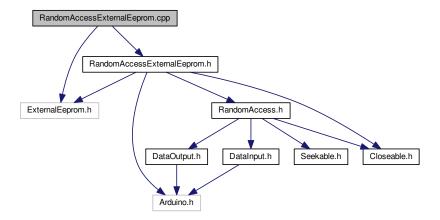
```
00001
00010 #ifndef __ARDUINO_IO_RANDOM_ACCESS_BYTE_ARRAY_H_
00011 #define __ARDUINO_IO_RANDOM_ACCESS_BYTE_ARRAY_H_ 1
00012
00013 #include <RandomAccess.h>
00014 #include <Closeable.h>
00015
00016 class RandomAccessByteArray : public RandomAccess, public virtual
      Closeable {
00017
00021
          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 writeWord(word v);
00133
00139
          virtual void writeLong(long v);
00140
00146
          virtual void writeUnsignedLong(unsigned long v);
00147
00153
          virtual void writeFloat(float v);
00154
00160
          virtual void writeDouble(double v);
00161
00167
          virtual unsigned char readByte();
00168
00174
          virtual bool readBoolean();
00175
00181
          virtual char readChar();
00182
00188
          virtual unsigned char readUnsignedChar();
00189
00195
          virtual int readInt();
00196
          virtual unsigned int readUnsignedInt();
00202
00203
00209
          virtual word readWord();
00210
00216
          virtual long readLong();
00217
00223
          virtual unsigned long readUnsignedLong();
00224
00230
          virtual float readFloat();
00231
00237
          virtual double readDouble();
00238
00245
          virtual void readFully(unsigned char* b, int len);
00246
00253
          virtual unsigned int skipBytes(unsigned int n);
00254 };
00255 #endif /* __ARDUINO_IO_RANDOM_ACCESS_BYTE_ARRAY_H__ */
```

### 5.93 RandomAccessExternalEeprom.cpp File Reference

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

Include dependency graph for RandomAccessExternalEeprom.cpp:



#### Macros

• #define \_\_ARDUINO\_IO\_RANDOM\_ACCESS\_EXTERNAL\_EEPROM\_CPP\_\_ 1

#### 5.93.1 Macro Definition Documentation

```
5.93.1.1 #define __ARDUINO_IO_RANDOM_ACCESS_EXTERNAL_EEPROM_CPP__ 1
```

Araduino IO.

#### RandomAccessExternalEeprom

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

Definition at line 12 of file RandomAccessExternalEeprom.cpp.

### 5.94 RandomAccessExternalEeprom.cpp

```
00011 #ifndef __ARDUINO_IO_RANDOM_ACCESS_EXTERNAL_EEPROM_CPP_
00012 #define __ARDUINO_IO_RANDOM_ACCESS_EXTERNAL_EEPROM_CPP__
00013
00014 #include <ExternalEeprom.h>
00015 #include "RandomAccessExternalEeprom.h"
00016
00017 RandomAccessExternalEeprom::RandomAccessExternalEeprom
00018
               ExternalEeprom* externalEeprom) :
00019
               \verb| 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);
```

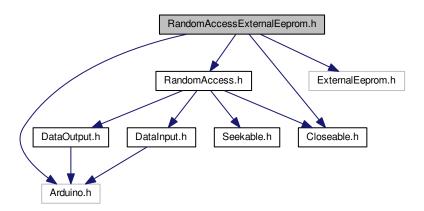
```
00036 }
00037
00038 void RandomAccessExternalEeprom::write(unsigned char b) {
00039
          writeByte(b);
00040 }
00041
00042 void RandomAccessExternalEeprom::writeByte(unsigned char b) {
00043
          externalEeprom->write(pos++, b);
00044 }
00045
00046 void RandomAccessExternalEeprom::writeBytes(unsigned char* b, int len
00047
          for (int i = 0; i < len; i++) {</pre>
00048
             externalEeprom->write(pos++, b[i]);
00049
00050 }
00051
00052 void RandomAccessExternalEeprom::writeBoolean(bool v) {
         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
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::writeWord(word v) {
00074
         writeInt((int) v);
00075 }
00076
00077 void RandomAccessExternalEeprom::writeLong(long v) {
00078
          externalEeprom->write(pos++, (unsigned char) ((v >> 24) & 0xff));
00079
          externalEeprom->write(pos++, (unsigned char) ((v >> 16) & 0xff));
          externalEeprom->write(pos++, (unsigned char) ((v >> 8) & 0xff));
00080
00081
          externalEeprom->write(pos++, (unsigned char) (v & 0xff));
00082 }
00083
00084 void RandomAccessExternalEeprom::writeUnsignedLong(unsigned
     long v) {
00085
         writeLong((long) v);
00086 }
00087
00088 void RandomAccessExternalEeprom::writeFloat(float v) {
00089
         writeLong((long) v);
00090 }
00091
00092 void RandomAccessExternalEeprom::writeDouble(double v) {
00093
         writeLong((long) v);
00094 }
00095
00096 unsigned char RandomAccessExternalEeprom::readByte() {
00097
          return (unsigned char) externalEeprom->read(pos++);
00098 }
00099
00100 bool RandomAccessExternalEeprom::readBoolean() {
         return (bool) externalEeprom->read(pos++);
00102 }
00103
00104 char RandomAccessExternalEeprom::readChar() {
00105
         return (char) externalEeprom->read(pos++);
00106 }
00108 unsigned char RandomAccessExternalEeprom::readUnsignedChar() {
00109
         return (unsigned char) externalEeprom->read(pos++);
00110 }
00111
00112 int RandomAccessExternalEeprom::readInt() {
00113
         int v = 0;
          v = externalEeprom->read(pos++);
00114
00115
          v <<= 8;
00116
          v |= (externalEeprom->read(pos++) & 0xff);
00117
          return v;
00118 }
```

```
00120 unsigned int RandomAccessExternalEeprom::readUnsignedInt() {
00121
         return (unsigned int) readInt();
00122 }
00123
00124 word RandomAccessExternalEeprom::readWord() {
00125
         return (word) readInt();
00126 }
00127
00128 long RandomAccessExternalEeprom::readLong() {
00129
         long v = 0;
00130
          v = externalEeprom->read(pos++);
00131
         v <<= 8;
00132
         v |= (externalEeprom->read(pos++) & 0xff);
00133
         v <<= 8;
00134
         v |= (externalEeprom->read(pos++) & 0xff);
00135
         v <<= 8;
00136
         v |= (externalEeprom->read(pos++) & 0xff);
00137
          return v;
00138 }
00139
00140 unsigned long RandomAccessExternalEeprom::readUnsignedLong() {
00141
         return (unsigned long) readLong();
00142 }
00143
00144 float RandomAccessExternalEeprom::readFloat() {
00145
         return (float) readLong();
00146 }
00147
00148 double RandomAccessExternalEeprom::readDouble() {
00149
         return (double) readLong();
00150 }
00151
00152 void RandomAccessExternalEeprom::readFully(unsigned char* b, int len)
00153
          for (int i = 0; i < len; i++) {</pre>
00154
             b[i] = externalEeprom->read(pos++);
00155
00156 }
00157
00158 unsigned int RandomAccessExternalEeprom::skipBytes(unsigned int n) {
00159
         unsigned int skipped;
00160
         unsigned int newpos;
00161
         newpos = pos + n;
00162
         if (newpos > length()) {
00163
              newpos = length();
00164
         skipped = newpos - pos;
00165
00166
         pos = newpos;
return skipped;
00167
00168 }
00169
00170 #endif /* __ARDUINO_IO_RANDOM_ACCESS_EXTERNAL_EEPROM_CPP__ */
```

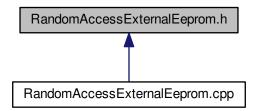
## 5.95 RandomAccessExternalEeprom.h File Reference

```
#include <Arduino.h>
#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

# 5.96 RandomAccessExternalEeprom.h

```
00001
00011 #ifndef __ARDUINO_IO_RANDOM_ACCESS_EXTERNAL_EEPROM_H_
00012 #define __ARDUINO_IO_RANDOM_ACCESS_EXTERNAL_EEPROM_H_ 1
00013
00014 #include <Arduino.h>
00015 #include <RandomAccess.h>
00016 #include <Closeable.h>
00017 #include <ExternalEeprom.h>
00018
00019 class RandomAccessExternalEeprom : public RandomAccess, public
     virtual Closeable {
00020
00024
          ExternalEeprom* externalEeprom;
00025
00029
00030
         unsigned int pos;
00031 public:
00032
00038
          RandomAccessExternalEeprom(ExternalEeprom* externalEeprom);
```

```
00039
00045
          virtual void seek (unsigned int pos);
00046
00052
          unsigned int length();
00053
00057
          virtual void close();
00058
00065
          virtual void write(unsigned char* b, int len);
00066
00072
          virtual void write (unsigned char b);
00073
00079
          virtual void writeBvte(unsigned char b);
00080
00087
          virtual void writeBytes (unsigned char* b, int len);
00088
00094
          virtual void writeBoolean(bool v);
00095
00101
          virtual void writeChar(char c);
00102
00108
          virtual void writeUnsignedChar(unsigned char c);
00109
00115
          virtual void writeInt(int v);
00116
00122
          virtual void writeUnsignedInt(unsigned int v);
00123
00129
          virtual void writeWord(word v);
00130
00136
          virtual void writeLong(long v);
00137
00143
          virtual void writeUnsignedLong(unsigned long v);
00144
00150
          virtual void writeFloat(float v);
00151
00157
          virtual void writeDouble(double v);
00158
00164
          virtual unsigned char readByte();
00165
00171
          virtual bool readBoolean();
00172
00178
          virtual char readChar();
00179
00185
          virtual unsigned char readUnsignedChar();
00186
00192
          virtual int readInt();
00193
00199
          virtual unsigned int readUnsignedInt();
00200
00206
          virtual word readWord();
00207
00213
          virtual long readLong();
00214
00220
          virtual unsigned long readUnsignedLong();
00221
00227
          virtual float readFloat();
00228
00234
          virtual double readDouble();
00235
00242
          virtual void readFully(unsigned char* b, int len);
00243
00250
          virtual unsigned int skipBytes(unsigned int n);
00251 }:
00252 #endif /* ARDUINO IO RANDOM ACCESS EXTERNAL EEPROM H */
```

## 5.97 RandomAccessResource.cpp File Reference

Macros

```
#define __ARDUINO_IO_RANDOM_ACCESS_RESOURCE_CPP__ 1
```

5.97.1 Macro Definition Documentation

```
5.97.1.1 #define __ARDUINO_IO_RANDOM_ACCESS_RESOURCE_CPP__ 1
```

Araduino 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.

### 5.98 RandomAccessResource.cpp

```
00001
00011 #ifndef __ARDUINO_IO_RANDOM_ACCESS_RESOURCE_CPP_
00012 #define __ARDUINO_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) {
         for (int i = 0; i < len; i++) {</pre>
00046
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));
         resource->write((unsigned char) ((v >> 16) & 0xff));
00078
          resource->write((unsigned char) ((v >> 8) & 0xff));
00079
08000
          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) {
```

```
88000
         writeLong((long) v);
00089 }
00090
00091 void RandomAccessResource::writeDouble(double v) {
00092
         writeLong((long) v);
00093 }
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 }
00107 unsigned char RandomAccessResource::readUnsignedChar() {
00108
         return (unsigned char) resource->read();
00109 }
00110
00111 int RandomAccessResource::readInt() {
00112
         int v = 0;
          v = resource->read();
00113
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();
00126
00127 long RandomAccessResource::readLong() {
00128
          long v = 0;
         v = resource->read();
00129
00130
         v <<= 8:
00131
         v |= (resource->read() & 0xff);
         v <<= 8;
00132
00133
          v |= (resource->read() & 0xff);
00134
         v <<= 8;
00135
          v \mid = (resource -> read() & 0xff);
          return v;
00136
00137 }
00138
00139 unsigned long RandomAccessResource::readUnsignedLong() {
00140
         return (unsigned long) readLong();
00141 }
00142
00143 float RandomAccessResource::readFloat() {
         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++) {</pre>
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;
          pos = (unsigned int) resource->tell();
00161
          len = resource->size();
00162
00163
          newpos = pos + n;
00164
          if (newpos > len)
00165
              newpos = len;
00166
          seek (newpos);
00167
          return (unsigned int) (newpos - pos);
00168
00169 }
00170
00171 #endif /* USING_RESOURCE_LIBRARIES */
00172
00173 #endif /* __ARDUINO_IO_RANDOM_ACCESS_RESOURCE_CPP__ */
```

#### 5.99 RandomAccessResource.h File Reference

#### 5.100 RandomAccessResource.h

```
00001
00011 #ifndef __ARDUINO_IO_RANDOM_ACCESS_RESOURCE_H_
00012 #define __ARDUINO_IO_RANDOM_ACCESS_RESOURCE_H_
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
          virtual void writeUnsignedLong(unsigned long v);
00139
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 /* __ARDUINO_IO_RANDOM_ACCESS_RESOURCE_H__ */
```

### 5.101 ResourceInputStream.cpp File Reference

**Macros** 

#define \_\_ARDUINO\_IO\_RESOURCE\_INPUT\_STREAM\_CPP\_\_ 1

#### 5.101.1 Macro Definition Documentation

```
5.101.1.1 #define __ARDUINO_IO_RESOURCE_INPUT_STREAM_CPP__ 1
```

Arduino IO.

ResourceInputStream

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

Definition at line 10 of file ResourceInputStream.cpp.

## 5.102 ResourceInputStream.cpp

```
00001
00009 #ifndef __ARDUINO_IO_RESOURCE_INPUT_STREAM_CPP_
00010 #define __ARDUINO_IO_RESOURCE_INPUT_STREAM_CPP_
00011
00012 #if USING_RESOURCE_LIBRARIES
00014 #include "ResourceInputStream.h"
00015
00016 ResourceInputStream::ResourceInputStream(Resource* resource) : resource(resource) {
00017
         markpos = 0;
00018
         pos = 0;
          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 */
00057 #endif /* __ARDUINO_IO_RESOURCE_INPUT_STREAM_CPP__ */
```

### 5.103 ResourceInputStream.h File Reference

#### 5.104 ResourceInputStream.h

```
00001
00009 #ifndef __ARDUINO_IO_RESOURCE_INPUT_STREAM_H_
00010 #define __ARDUINO_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
          * The resource where data is stored.
00022
00023
          Resource* resource;
00024
00025
          * Current position
*/
00026
00027
00028
          unsigned int pos;
00029
00030
00031
          \star The currently marked position in the stream.
00032
00033
          unsigned int markpos;
00034
00035
          \star The size of the resource.
00036
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 /* __ARDUINO_IO_RESOURCE_INPUT_STREAM_H__ */
```

## 5.105 ResourceOutputStream.cpp File Reference

Macros

```
    #define __ARDUINO_IO_RESOURCE_OUTPUT_STREAM_CPP__ 1
```

5.105.1 Macro Definition Documentation

```
5.105.1.1 #define __ARDUINO_IO_RESOURCE_OUTPUT_STREAM_CPP__ 1
```

Arduino IO.

ResourceOutputStream

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

Definition at line 10 of file ResourceOutputStream.cpp.

### 5.106 ResourceOutputStream.cpp

```
00001
00009 #ifndef __ARDUINO_IO_RESOURCE_OUTPUT_STREAM_CPP_
00010 #define __ARDUINO_IO_RESOURCE_OUTPUT_STREAM_CPP__ 1
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
00025 }
00026
00027 #endif /* USING RESOURCE LIBRARIES */
00028
00029 #endif /* __ARDUINO_IO_RESOURCE_OUTPUT_STREAM_CPP__ */
```

## 5.107 ResourceOutputStream.h File Reference

## 5.108 ResourceOutputStream.h

```
00001
00009 #ifndef __ARDUINO_IO_RESOURCE_OUTPUT_STREAM_H_
00010 #define __ARDUINO_IO_RESOURCE_OUTPUT_STREAM_H_
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 /* __ARDUINO_IO_RESOURCE_OUTPUT_STREAM_H__ */
```

#### 5.109 ResourceSeekableInputStream.cpp File Reference

Macros

```
    #define __ARDUINO_IO_RESOURCE_SEEKABLE_INPUT_STREAM_CPP__ 1
```

5.109.1 Macro Definition Documentation

```
5.109.1.1 #define __ARDUINO_IO_RESOURCE_SEEKABLE_INPUT_STREAM_CPP__ 1
```

Arduino IO.

ResourceSeekableInputStream

A ResourceSeekableInputStream obtains input bytes from a resource in a file system that implements Seekable ← InputStream interface.

Definition at line 11 of file ResourceSeekableInputStream.cpp.

#### 5.110 ResourceSeekableInputStream.cpp

```
00001
00010 #ifndef __ARDUINO_IO_RESOURCE_SEEKABLE_INPUT_STREAM_CPP_
00011 #define __ARDUINO_IO_RESOURCE_SEEKABLE_INPUT_STREAM_CPP__ 1
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 /* __ARDUINO_IO_RESOURCE_SEEKABLE_INPUT_STREAM_CPP__ */
```

### 5.111 ResourceSeekableInputStream.h File Reference

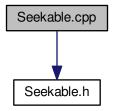
## 5.112 ResourceSeekableInputStream.h

```
00010 #ifndef __ARDUINO_IO_RESOURCE_SEEKABLE_INPUT_STREAM_H_
00011 #define __ARDUINO_IO_RESOURCE_SEEKABLE_INPUT_STREAM_H_
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 /* __ARDUINO_IO_RESOURCE_SEEKABLE_INPUT_STREAM_H__ */
```

### 5.113 Seekable.cpp File Reference

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

Include dependency graph for Seekable.cpp:



#### Macros

#define \_\_ARDUINO\_IO\_SEEKABLE\_CPP\_\_ 1

#### 5.113.1 Macro Definition Documentation

5.113.1.1 #define \_\_ARDUINO\_IO\_SEEKABLE\_CPP\_\_ 1

Arduino IO.

#### Seekable

Definition at line 8 of file Seekable.cpp.

#### 5.114 Seekable.cpp

```
00001

00007 #ifndef __ARDUINO_IO_SEEKABLE_CPP__

00008 #define __ARDUINO_IO_SEEKABLE_CPP__ 1

00009

00010 #include "Seekable.h"

00011

00012 #endif /* __ARDUINO_IO_SEEKABLE_CPP__ */
```

### 5.115 Seekable.h File Reference

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



#### Classes

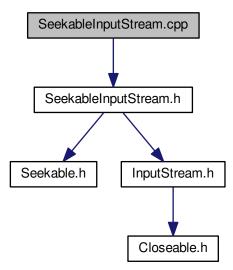
· class Seekable

5.116 Seekable.h 181

#### 5.116 Seekable.h

## 5.117 SeekableInputStream.cpp File Reference

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



### Macros

• #define \_\_ARDUINO\_IO\_SEEKABLE\_INPUT\_STREAM\_CPP\_\_ 1

### 5.117.1 Macro Definition Documentation

5.117.1.1 #define \_\_ARDUINO\_IO\_SEEKABLE\_INPUT\_STREAM\_CPP\_\_ 1

Arduino IO.

## SeekableInputStream

Definition at line 8 of file SeekableInputStream.cpp.

## 5.118 SeekableInputStream.cpp

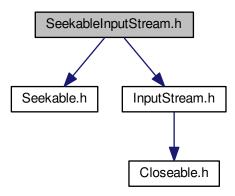
00001

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

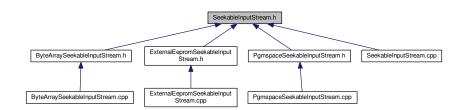
## 5.119 SeekableInputStream.h File Reference

```
#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

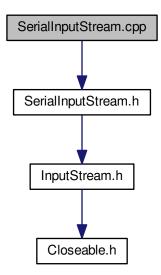
## 5.120 SeekableInputStream.h

```
00001
00007 #ifndef __ARDUINO_IO_SEEKABLE_INPUT_STREAM_H_
00008 #define __ARDUINO_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 /* __ARDUINO_IO_SEEKABLE_INPUT_STREAM_H__ */
```

#### 5.121 SerialInputStream.cpp File Reference

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



#### Macros

```
    #define __ARDUINO_IO_SERIAL_INPUT_STREAM_CPP__ 1
```

## 5.121.1 Macro Definition Documentation

```
5.121.1.1 #define __ARDUINO_IO_SERIAL_INPUT_STREAM_CPP__ 1
```

Arduino IO.

## SerialInputStream

A SerialInputStream obtains input bytes from a serial port.

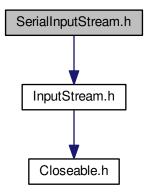
Definition at line 10 of file SerialInputStream.cpp.

## 5.122 SerialInputStream.cpp

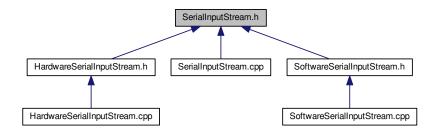
```
00001
00009 #ifndef __ARDUINO_IO_SERIAL_INPUT_STREAM_CPP__
00010 #define __ARDUINO_IO_SERIAL_INPUT_STREAM_CPP__ 1
00011
00012 #include "SerialInputStream.h"
00013
00014 #endif /* __ARDUINO_IO_SERIAL_INPUT_STREAM_CPP__ */
```

## 5.123 SerialInputStream.h File Reference

#include <InputStream.h>
Include dependency graph for SerialInputStream.h:



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



#### Classes

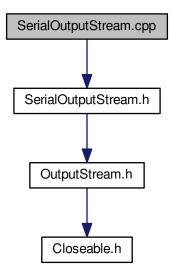
• class SerialInputStream

## 5.124 SerialInputStream.h

```
00001
00009 #ifndef __ARDUINO_IO_SERIAL_INPUT_STREAM_H_
00010 #define __ARDUINO_IO_SERIAL_INPUT_STREAM_H__ 1
00011
00012 #include <InputStream.h>
00013
00014 class SerialInputStream : public InputStream {
00015 };
00016
00017 #endif /* __ARDUINO_IO_SERIAL_INPUT_STREAM_H__ */
```

### 5.125 SerialOutputStream.cpp File Reference

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



#### Macros

• #define \_\_ARDUINO\_IO\_SERIAL\_OUTPUT\_STREAM\_CPP\_\_ 1

#### 5.125.1 Macro Definition Documentation

5.125.1.1 #define \_\_ARDUINO\_IO\_SERIAL\_OUTPUT\_STREAM\_CPP\_\_ 1

Arduino IO.

#### SerialOutputStream

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

Definition at line 10 of file SerialOutputStream.cpp.

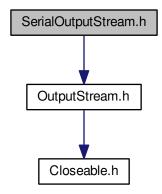
### 5.126 SerialOutputStream.cpp

```
00001
00009 #ifndef __ARDUINO_IO_SERIAL_OUTPUT_STREAM_CPP__
00010 #define __ARDUINO_IO_SERIAL_OUTPUT_STREAM_CPP__ 1
00011
00012 #include <SerialOutputStream.h>
00013
00013
00014 #endif /* __ARDUINO_IO_SERIAL_OUTPUT_STREAM_CPP__ */
```

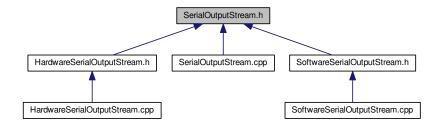
## 5.127 SerialOutputStream.h File Reference

#include <OutputStream.h>

Include dependency graph for SerialOutputStream.h:



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



### Classes

• class SerialOutputStream

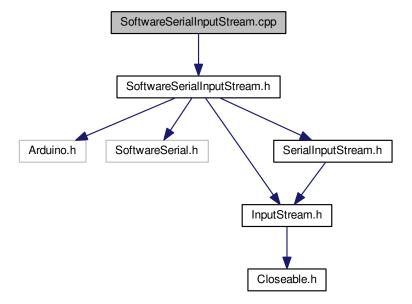
## 5.128 SerialOutputStream.h

```
00001
00009 #ifndef __ARDUINO_IO_SERIAL_OUTPUT_STREAM_H_
00010 #define __ARDUINO_IO_SERIAL_OUTPUT_STREAM_H_ 1
00011
00012 #include <OutputStream.h>
00013
00014 class SerialOutputStream : public OutputStream {
00015 };
00016
00017 #endif /* __ARDUINO_IO_SERIAL_OUTPUT_STREAM_H__ */
```

## 5.129 SoftwareSerialInputStream.cpp File Reference

#include "SoftwareSerialInputStream.h"

Include dependency graph for SoftwareSerialInputStream.cpp:



#### Macros

#define \_\_ARDUINO\_IO\_SOFTWARE\_SERIAL\_INPUT\_STREAM\_CPP\_\_ 1

#### 5.129.1 Macro Definition Documentation

5.129.1.1 #define \_\_ARDUINO\_IO\_SOFTWARE\_SERIAL\_INPUT\_STREAM\_CPP\_\_ 1

Arduino IO.

### SoftwareSerialInputStream

A SoftwareSerialInputStream obtains input bytes from a serial port.

Definition at line 10 of file SoftwareSerialInputStream.cpp.

## 5.130 SoftwareSerialInputStream.cpp

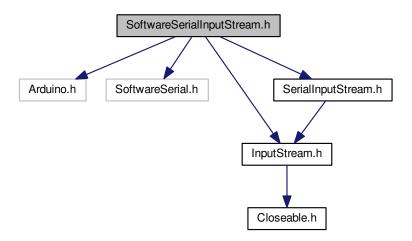
```
00001
00009 #ifndef __ARDUINO_IO_SOFTWARE_SERIAL_INPUT_STREAM_CPP_
00010 #define __ARDUINO_IO_SOFTWARE_SERIAL_INPUT_STREAM_CPP__ 1
00011
00012 #include "SoftwareSerialInputStream.h"
00013
00014 SoftwareSerialInputStream::SoftwareSerialInputStream(
     SoftwareSerial *softwareSerial,
00015
              unsigned int boudRate) :
00016
              softwareSerial(softwareSerial) {
00017
         softwareSerial->begin(boudRate);
00018 }
00019
00020 int SoftwareSerialInputStream::available() {
00021
         softwareSerial->available();
00022 }
00023
00024 int SoftwareSerialInputStream::read() {
00025
         return softwareSerial->read();
```

```
00026 }
00027
00028 #endif /* __ARDUINO_IO_SOFTWARE_SERIAL_INPUT_STREAM_CPP__ */
```

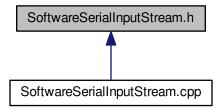
## 5.131 SoftwareSerialInputStream.h File Reference

```
#include <Arduino.h>
#include <SoftwareSerial.h>
#include <InputStream.h>
#include <SerialInputStream.h>
```

Include dependency graph for SoftwareSerialInputStream.h:



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



#### Classes

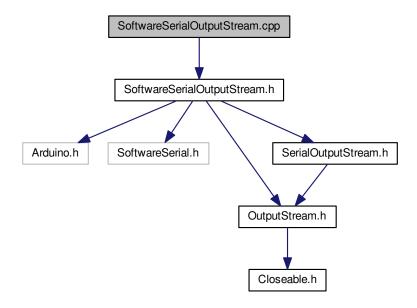
• class SoftwareSerialInputStream

## 5.132 SoftwareSerialInputStream.h

```
00001
00009 #ifndef __ARDUINO_IO_SOFTWARE_SERIAL_INPUT_STREAM_H_
00010 #define __ARDUINO_IO_SOFTWARE_SERIAL_INPUT_STREAM_H_ 1
00012 #include <Arduino.h>
00013 #include <SoftwareSerial.h>
00014 #include <InputStream.h>
00015 #include <SerialInputStream.h>
00016
00017 class SoftwareSerialInputStream : public
      SerialInputStream {
00018 protected:
00019
00023
          SoftwareSerial *softwareSerial;
00024
00025 public:
00026
00033
           {\tt SoftwareSerialInputStream} ({\tt SoftwareSerial} \ \star {\tt softwareSerial}, \ {\tt unsigned} \ {\tt int}
      boudRate);
00034
00039
          virtual int available();
00040
00044
          virtual int read();
00045 };
00046
00047 #endif /* __ARDUINO_IO_SOFTWARE_SERIAL_INPUT_STREAM_H__ */
```

#### 5.133 SoftwareSerialOutputStream.cpp File Reference

#include "SoftwareSerialOutputStream.h"
Include dependency graph for SoftwareSerialOutputStream.cpp:



#### **Macros**

• #define \_\_ARDUINO\_IO\_SOFTWARE\_SERIAL\_OUTPUT\_STREAM\_CPP\_\_ 1

#### 5.133.1 Macro Definition Documentation

5.133.1.1 #define \_\_ARDUINO\_IO\_SOFTWARE\_SERIAL\_OUTPUT\_STREAM\_CPP\_\_ 1

Arduino IO.

#### SoftwareSerialOutputStream

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

Definition at line 10 of file SoftwareSerialOutputStream.cpp.

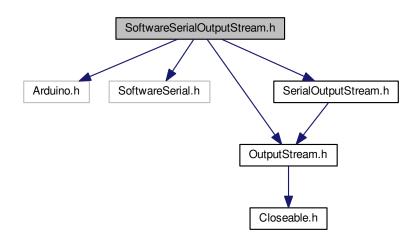
#### 5.134 SoftwareSerialOutputStream.cpp

```
00001
00009 #ifndef _ARDUINO_IO_SOFTWARE_SERIAL_OUTPUT_STREAM_CPP_
00010 #define _ARDUINO_IO_SOFTWARE_SERIAL_OUTPUT_STREAM_CPP_
00012 #include "SoftwareSerialOutputStream.h"
00013
{\tt 00014~SoftwareSerialOutputStream::SoftwareSerialOutputStream}
(SoftwareSerial *serial, 00015 unsigned int boudRate):
                softwareSerial(serial) {
00017
          serial->begin(boudRate);
00018 }
00019
00020 void SoftwareSerialOutputStream::write(unsigned char b) {
00021
           softwareSerial->write(b);
00022 }
00023
00024 #endif /* __ARDUINO_IO_SOFTWARE_SERIAL_OUTPUT_STREAM_CPP__ */
```

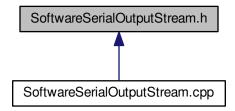
#### 5.135 SoftwareSerialOutputStream.h File Reference

```
#include <Arduino.h>
#include <SoftwareSerial.h>
#include <OutputStream.h>
#include <SerialOutputStream.h>
```

Include dependency graph for SoftwareSerialOutputStream.h:



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



#### Classes

· class SoftwareSerialOutputStream

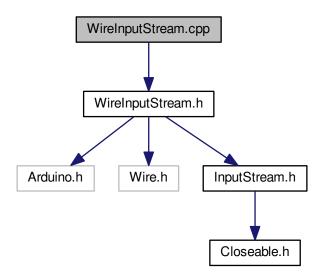
## 5.136 SoftwareSerialOutputStream.h

```
00001
00009 #ifndef __ARDUINO_IO_SOFTWARE_SERIAL_OUTPUT_STREAM_H_
00010 #define __ARDUINO_IO_SOFTWARE_SERIAL_OUTPUT_STREAM_H_
00011
00012 #include <Arduino.h>
00013 #include <SoftwareSerial.h>
00014 #include <OutputStream.h>
00015 #include <SerialOutputStream.h>
00017 class SoftwareSerialOutputStream : public
      SerialOutputStream {
00018 protected:
00019
00020
          \star The software serial where data is written.
00021
00023
          SoftwareSerial *softwareSerial;
00024
00025 public:
00026
00027
          SoftwareSerialOutputStream(SoftwareSerial *serial, unsigned int boudRate);
00028
00032
           virtual void write(unsigned char b);
00033 };
00034
00035 #endif /* __ARDUINO_IO_SOFTWARE_SERIAL_OUTPUT_STREAM_H__ */
```

#### 5.137 WireInputStream.cpp File Reference

#include "WireInputStream.h"

Include dependency graph for WireInputStream.cpp:



#### Macros

#define \_\_ARDUINO\_IO\_WIRE\_INPUT\_STREAM\_CPP\_\_ 1

#### 5.137.1 Macro Definition Documentation

5.137.1.1 #define \_\_ARDUINO\_IO\_WIRE\_INPUT\_STREAM\_CPP\_\_1

Arduino IO.

## WireInputStream

A WireInputStream obtains input bytes from the wire bus.

Definition at line 10 of file WireInputStream.cpp.

## 5.138 WireInputStream.cpp

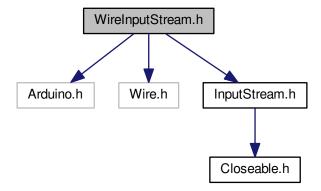
```
00009 #ifndef __ARDUINO_IO_WIRE_INPUT_STREAM_CPP_
00010 #define __ARDUINO_IO_WIRE_INPUT_STREAM_CPP__
00011
00012 #include "WireInputStream.h"
00013
00014 WireInputStream::WireInputStream(unsigned char address) {
00015
         this->address = address;
00016
         Wire.begin();
00017 }
00018
00019 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
00035
           Wire.beginTransmission(address);
00036
           Wire.write((unsigned char) (address & 0xff));
00037
           Wire.endTransmission();
          Wire.requestFrom(address, (int) len);
for (i = 0; i < len; i++) {</pre>
00038
00039
00040
               while (!Wire.available())
00041
00042
               b[off + i] = (unsigned char) Wire.read();
00043
00044
           return i:
00045 }
00047 #endif /* __ARDUINO_IO_WIRE_INPUT_STREAM_CPP__ */
```

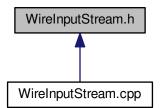
#### 5.139 WireInputStream.h File Reference

```
#include <Arduino.h>
#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

## 5.140 WireInputStream.h

```
00009 #ifndef __ARDUINO_IO_WIRE_INPUT_STREAM_H_
00010 #define __ARDUINO_IO_WIRE_INPUT_STREAM_H_ 1
00011
00011 #include <Arduino.h>
00013 #include <Wire.h>
00014 #include <InputStream.h>
00015
00016 class WireInputStream : public InputStream {
00017 protected:
00018
00022
          unsigned char address;
00023
00024 public:
00025
00031
00032
          WireInputStream(unsigned char addredd);
00037
          virtual int available();
00038
00042
          virtual int read();
00043
           virtual int read(unsigned char* b, int off, int len);
00052
00053 };
00054
00055 #endif /* __ARDUINO_IO_WIRE_INPUT_STREAM_H__ */
```

- 5.141 WireOutputStream.cpp File Reference
- 5.142 WireOutputStream.cpp
- 5.143 WireOutputStream.h File Reference
- 5.144 WireOutputStream.h

# Index

ARDUINO_IO_BUFFERED_INPUT_STREAM_CP↔ P	ARDUINO_IO_RANDOM_ACCESS_BYTE_ARRA↔ Y CPP
<del></del>	<del>-</del>
BufferedInputStream.cpp, 103	RandomAccessByteArray.cpp, 163
ARDUINO_IO_BUFFERED_OUTPUT_STREAM_	ARDUINO_IO_RANDOM_ACCESS_CPP
CPP	RandomAccess.cpp, 161
BufferedOutputStream.cpp, 108	ARDUINO_IO_RANDOM_ACCESS_EXTERNAL_
ARDUINO_IO_BYTE_ARRAY_INPUT_STREAM_	EEPROM_CPP
CPP	RandomAccessExternalEeprom.cpp, 168
ByteArrayInputStream.cpp, 111	ARDUINO_IO_RANDOM_ACCESS_RESOURCE↔
ARDUINO IO BYTE ARRAY OUTPUT STREA↔	_CPP
M CPP	RandomAccessResource.cpp, 172
ByteArrayOutputStream.cpp, 115	ARDUINO_IO_RESOURCE_INPUT_STREAM_C
_ARDUINO_IO_BYTE_ARRAY_SEEKABLE_INPU↔	PP
T_STREAM_CPP	ResourceInputStream.cpp, 176
	_ARDUINO_IO_RESOURCE_OUTPUT_STREAM_
ByteArraySeekableInputStream.cpp, 117	CPP CPP
ARDUINO_IO_CLOSEABLE_CPP	<del></del>
Closeable.cpp, 120	ResourceOutputStream.cpp, 177
ARDUINO_IO_DATA_INPUT_CPP	ARDUINO_IO_RESOURCE_SEEKABLE_INPUT_
DataInput.cpp, 121	STREAM_CPP
ARDUINO_IO_DATA_INPUT_STREAM_CPP	ResourceSeekableInputStream.cpp, 178
DataInputStream.cpp, 123	ARDUINO_IO_SEEKABLE_CPP
ARDUINO_IO_DATA_OUTPUT_CPP	Seekable.cpp, 180
DataOutput.cpp, 126	ARDUINO_IO_SEEKABLE_INPUT_STREAM_CP
ARDUINO_IO_DATA_OUTPUT_STREAM_CPP	P
DataOutputStream.cpp, 129	SeekableInputStream.cpp, 181
ARDUINO_IO_EXTERNAL_EEPROM_INPUT_ST	ARDUINO_IO_SERIAL_INPUT_STREAM_CPP
	SerialInputStream.cpp, 183
REAM_CPP	ARDUINO_IO_SERIAL_OUTPUT_STREAM_CPP
ExternalEepromInputStream.cpp, 132	
ARDUINO_IO_EXTERNAL_EEPROM_OUTPUT_	SerialOutputStream.cpp, 185
STREAM_CPP	_ARDUINO_IO_SOFTWARE_SERIAL_INPUT_ST
ExternalEepromOutputStream.cpp, 135	REAM CPP
ARDUINO_IO_EXTERNAL_EEPROM_SEEKABL	SoftwareSerialInputStream.cpp, 187
E_INPUT_STREAM_CPP	
ExternalEepromSeekableInputStream.cpp, 138	ARDUINO_IO_SOFTWARE_SERIAL_OUTPUT_S
ARDUINO IO FILTER INPUT STREAM CPP	TREAM_CPP
FilterInputStream.cpp, 140	SoftwareSerialOutputStream.cpp, 189
ARDUINO_IO_FILTER_OUTPUT_STREAM_CPP	ARDUINO_IO_WIRE_INPUT_STREAM_CPP
	WireInputStream.cpp, 192
FilterOutputStream.cpp, 143	
_ARDUINO_IO_HARDWARE_SERIAL_INPUT_ST	address
REAM_CPP	WireInputStream, 102
	available
HardwareSerialInputStream.cpp, 146	BufferedInputStream, 9
ARDUINO_IO_HARDWARE_SERIAL_OUTPUT_S	ByteArrayInputStream, 17
TREAM_CPP	ExternalEepromInputStream, 43
HardwareSerialOutputStream.cpp, 149	FilterInputStream, 53
ARDUINO_IO_INPUT_STREAM_CPP	HardwareSerialInputStream, 61
InputStream.cpp, 151	InputStream, 64
ARDUINO_IO_OUTPUT_STREAM_CPP	PgmspaceInputStream, 69
OutputStream.cpp, 154	SoftwareSerialInputStream, 98
ARDUINO_IO_PGMSPACE_INPUT_STREAM_C	WireInputStream, 102
PP	
PgmspaceInputStream.cpp, 156	buf
_ARDUINO_IO_PGMSPACE_SEEKABLE_INPUT_	BufferedInputStream, 11
STREAM_CPP	BufferedOutputStream, 14
PgmspaceSeekableInputStream.cpp, 159	ByteArrayInputStream, 17
i girispaceseekabieiriputstieairi.cpp, 133	bytemnayinputotream, 17

ByteArrayOutputStream, 20	size, 20
PgmspaceInputStream, 70	toByteArray, 20
RandomAccessByteArray, 82	write, 20
BufferedInputStream, 7	ByteArrayOutputStream.cpp, 114, 115
available, 9	ARDUINO_IO_BYTE_ARRAY_OUTPUT_ST←
buf, 11	REAM_CPP, 115
BufferedInputStream, 9	ByteArrayOutputStream.h, 115, 116
close, 9	ByteArraySeekableInputStream, 21
count, 11	ByteArraySeekableInputStream, 22
fill, 9	seek, 22
mark, 10	ByteArraySeekableInputStream.cpp, 117, 118
markSupported, 10	ARDUINO_IO_BYTE_ARRAY_SEEKABLE_I←
marked, 11	NPUT_STREAM_CPP, 117
markpos, 11	ByteArraySeekableInputStream.h, 118, 119
pos, 11	<b>,,</b> , ,
read, 10	close
realineBufferContent, 10	BufferedInputStream, 9
	BufferedOutputStream, 13
reset, 10	Closeable, 23
size, 11	FilterInputStream, 53
skip, 10	FilterOutputStream, 57
BufferedInputStream.cpp, 103, 104	InputStream, 64
ARDUINO_IO_BUFFERED_INPUT_STREA↔	OutputStream, 67
M_CPP, 103	RandomAccessByteArray, 75
BufferedInputStream.h, 105, 106	RandomAccessExternalEeprom, 85
BufferedOutputStream, 12	•
buf, 14	Closeable, 22
BufferedOutputStream, 13	close, 23
close, 13	Closeable.cpp, 119, 120
count, 15	ARDUINO_IO_CLOSEABLE_CPP, 120
flush, 13	Closeable.h, 120
flushBuffer, 14	count
size, 15	BufferedInputStream, 11
write, 14	BufferedOutputStream, 15
BufferedOutputStream.cpp, 107, 108	ByteArrayInputStream, 17
ARDUINO IO BUFFERED OUTPUT STRE←	ByteArrayOutputStream, 20
AM_CPP, 108	PgmspaceInputStream, 70
	RandomAccessByteArray, 82
BufferedOutputStream.h, 109, 110	
ByteArrayInputStream, 15	DataInput, 23
available, 17	readBoolean, 24
buf, 17	readByte, 24
ByteArrayInputStream, 16	readChar, 25
count, 17	readDouble, 25
mark, 17	readFloat, 25
markSupported, 17	readFully, 25
markpos, 18	readInt, 25
pos, 18	readLong, 25
read, 17	readUnsignedChar, 26
reset, 17	readUnsignedInt, 26
ByteArrayInputStream.cpp, 111, 112	readUnsignedLong, 26
ARDUINO_IO_BYTE_ARRAY_INPUT_STRE	readWord, 26
AM_CPP, 111	skipBytes, 26
ByteArrayInputStream.h, 112, 113	DataInput.cpp, 121
ByteArrayOutputStream, 18	ARDUINO_IO_DATA_INPUT_CPP, 121
buf, 20	
ByteArrayOutputStream, 19	DataInputStream, 27
count, 20	DataInputStream, 28
	inputStream, 31
pos, 20	•
reset, 20	readBoolean, 28

readByte, 28	available, 43
readChar, 28	externalEeprom, 44
readDouble, 29	ExternalEepromInputStream, 43
readFloat, 29	externalEepromSize, 44
readFully, 29	mark, 43
readInt, 29	markSupported, 43
readLong, 29	markpos, 44
readUnsignedChar, 29	pos, 44
readUnsignedInt, 30	read, 43, 44
readUnsignedLong, 30	reset, 44
readWord, 30	ExternalEepromInputStream.cpp, 131, 132
skipBytes, 30	ARDUINO_IO_EXTERNAL_EEPROM_INPU
DataInputStream.cpp, 123	T_STREAM_CPP, 132
ARDUINO_IO_DATA_INPUT_STREAM_CPP	ExternalEepromInputStream.h, 133, 134
ANDONO_10_DATA_NTOT_STREAM_OFF ← , 123	ExternalEepromOutputStream, 45
DataInputStream.h, 124, 125	externalEeprom, 46
•	•
DataOutput, 31	ExternalEepromOutputStream, 46
write, 32	pos, 46
writeBoolean, 32	write, 46
writeByte, 32	ExternalEepromOutputStream.cpp, 135
writeBytes, 32	ARDUINO_IO_EXTERNAL_EEPROM_OUTP
writeChar, 32	UT_STREAM_CPP, 135
writeDouble, 34	ExternalEepromOutputStream.h, 136, 137
writeFloat, 34	ExternalEepromSeekableInputStream, 47
writeInt, 34	ExternalEepromSeekableInputStream, 48
writeLong, 34	seek, 48
writeUnsignedChar, 34	ExternalEepromSeekableInputStream.cpp, 137, 138
writeUnsignedInt, 34	ARDUINO_IO_EXTERNAL_EEPROM_SEEK
writeUnsignedLong, 34	ABLE_INPUT_STREAM_CPP, 138
writeWord, 36	ExternalEepromSeekableInputStream.h, 138, 139
DataOutput.cpp, 126, 127	externalEepromSize
ARDUINO_IO_DATA_OUTPUT_CPP, 126	ExternalEepromInputStream, 44
DataOutput.h, 127	fill
DataOutputStream, 36	BufferedInputStream, 9
DataOutputStream, 38	FilterInputStream, 50
outputStream, 41	available, 53
write, 39	close, 53
writeBoolean, 39	FilterInputStream, 52
writeByte, 39	in, 55
writeBytes, 39	mark, 53
writeChar, 40	markSupported, 53
writeDouble, 40	read, 53, 54
writeFloat, 40	reset, 54
writeInt, 40	skip, 55
writeLong, 40	FilterInputStream.cpp, 139, 140
writeUnsignedChar, 40	_ARDUINO_IO_FILTER_INPUT_STREAM_C
writeUnsignedInt, 41	PP , 140
writeUnsignedLong, 41	FilterInputStream.h, 141, 142
writeWord, 41	FilterOutputStream, 55
DataOutputStream.cpp, 128, 129	close, 57
ARDUINO_IO_DATA_OUTPUT_STREAM_C	FilterOutputStream, 57
PP, 129	flush, 57
DataOutputStream.h, 130	out, 59
externalEeprom	write, 57, 59
ExternalEepromInputStream, 44	FilterOutputStream.cpp, 142, 143
ExternalEepromOutputStream, 46	ARDUINO_IO_FILTER_OUTPUT_STREAM_←
RandomAccessExternalEeprom, 91	CPP , 143
ExternalEepromInputStream, 41	FilterOutputStream.h, 144, 145
	•

flush	markpos
BufferedOutputStream, 13	BufferedInputStream, 11
FilterOutputStream, 57	ByteArrayInputStream, 18
OutputStream, 67	ExternalEepromInputStream, 44
flushBuffer	PgmspaceInputStream, 70
BufferedOutputStream, 14	· gopacopatou ca, / c
Bancica Garpatoti Garri, 14	out
HardwareSerialInputStream, 59	FilterOutputStream, 59
available, 61	OutputStream, 66
HardwareSerialInputStream, 61	close, 67
	flush, 67
read, 61	
HardwareSerialInputStream.cpp, 145, 146	write, 67
ARDUINO_IO_HARDWARE_SERIAL_INPUT	outputStream
_STREAM_CPP, 146	DataOutputStream, 41
HardwareSerialInputStream.h, 147, 148	OutputStream.cpp, 153, 154
HardwareSerialOutputStream, 61	ARDUINO_IO_OUTPUT_STREAM_CPP
HardwareSerialOutputStream, 63	, 154
write, 63	OutputStream.h, 155
HardwareSerialOutputStream.cpp, 148, 149	
ARDUINO_IO_HARDWARE_SERIAL_OUTP↔	PgmspaceInputStream, 68
UT_STREAM_CPP, 149	available, 69
HardwareSerialOutputStream.h, 149, 150	buf, 70
, ,	count, 70
in	mark, 69
FilterInputStream, 55	markSupported, 69
InputStream, 63	markpos, 70
available, 64	PgmspaceInputStream, 69
close, 64	pos, 70
mark, 65	read, 70
	reset, 70
markSupported, 65	
read, 65	PgmspaceInputStream.cpp, 156
reset, 65	ARDUINO_IO_PGMSPACE_INPUT_STREA
skip, 65	M_CPP, 156
inputStream	PgmspaceInputStream.h, 157, 158
DataInputStream, 31	PgmspaceSeekableInputStream, 70
InputStream.cpp, 151	PgmspaceSeekableInputStream, 72
ARDUINO_IO_INPUT_STREAM_CPP, 151	seek, 72
InputStream.h, 152, 153	PgmspaceSeekableInputStream.cpp, 159
	ARDUINO_IO_PGMSPACE_SEEKABLE_IN←
length	PUT_STREAM_CPP, 159
RandomAccessByteArray, 75	PgmspaceSeekableInputStream.h, 160, 161
RandomAccessExternalEeprom, 85	pos
' '	BufferedInputStream, 11
mark	ByteArrayInputStream, 18
BufferedInputStream, 10	ByteArrayOutputStream, 20
ByteArrayInputStream, 17	ExternalEepromInputStream, 44
ExternalEepromInputStream, 43	·
FilterInputStream, 53	ExternalEepromOutputStream, 46
InputStream, 65	PgmspaceInputStream, 70
•	RandomAccessByteArray, 83
PgmspaceInputStream, 69	RandomAccessExternalEeprom, 91
markSupported	D   A 70
BufferedInputStream, 10	RandomAccess, 72
ByteArrayInputStream, 17	RandomAccess.cpp, 161, 162
ExternalEepromInputStream, 43	ARDUINO_IO_RANDOM_ACCESS_CPP,
FilterInputStream, 53	161
InputStream, 65	RandomAccess.h, 162
PgmspaceInputStream, 69	RandomAccessByteArray, 73
marked	buf, 82
BufferedInputStream, 11	close, 75
•	

count, 82	writeBytes, 88
length, 75	writeChar, 89
pos, 83	writeDouble, 89
RandomAccessByteArray, 75	writeFloat, 89
readBoolean, 75	writeInt, 89
readByte, 75	writeLong, 89
readChar, 76	writeUnsignedChar, 89
readDouble, 76	writeUnsignedInt, 91
readFloat, 76	writeUnsignedLong, 91
readFully, 76	writeWord, 91
readInt, 76	RandomAccessExternalEeprom.cpp, 167, 168
readLong, 77	ARDUINO_IO_RANDOM_ACCESS_EXTER
readUnsignedChar, 77	NAL_EEPROM_CPP, 168
readUnsignedInt, 77	RandomAccessExternalEeprom.h, 170, 171
readUnsignedLong, 77	RandomAccessResource.cpp, 172, 173
readWord, 77	_ARDUINO_IO_RANDOM_ACCESS_RESOU
seek, 77	RCE_CPP, 172
skipBytes, 79	RandomAccessResource.h, 175
write, 79	read
writeBoolean, 79	BufferedInputStream, 10
writeByte, 79	ByteArrayInputStream, 17
writeBytes, 80	ExternalEepromInputStream, 43, 44
writeChar, 80	FilterInputStream, 53, 54
writeDouble, 80	HardwareSerialInputStream, 61
writeFloat, 80	InputStream, 65
writer loat, 80	PgmspaceInputStream, 70
writeInt, 80 writeLong, 80	SoftwareSerialInputStream, 98
writeUnsignedChar, 82	WireInputStream, 102
writeUnsignedInt, 82	readBoolean
writeUnsignedLong, 82	DataInputStroom 38
writeWord, 82	DataInputStream, 28
RandomAccessByteArray.cpp, 163, 164	RandomAccessByteArray, 75
ARDUINO_IO_RANDOM_ACCESS_BYTE_A↔	RandomAccessExternalEeprom, 85
RRAY_CPP, 163	readByte
RandomAccessByteArray.h, 165, 166	DataInput, 24
RandomAccessExternalEeprom, 83	DataInputStream, 28
close, 85	RandomAccessByteArray, 75
externalEeprom, 91	RandomAccessExternalEeprom, 85
length, 85	readChar
pos, 91	DataInput, 25
RandomAccessExternalEeprom, 84	DataInputStream, 28
readBoolean, 85	RandomAccessByteArray, 76
readByte, 85	RandomAccessExternalEeprom, 85
readChar, 85	readDouble
readDouble, 85	DataInput, 25
readFloat, 86	DataInputStream, 29
readFully, 86	RandomAccessByteArray, 76
readInt, 86	RandomAccessExternalEeprom, 85
readLong, 86	readFloat
readUnsignedChar, 86	DataInput, 25
readUnsignedInt, 87	DataInputStream, 29
readUnsignedLong, 87	RandomAccessByteArray, 76
readWord, 87	RandomAccessExternalEeprom, 86
seek, 87	readFully
skipBytes, 87	DataInput, 25
write, 88	DataInputStream, 29
writeBoolean, 88	RandomAccessByteArray, 76
writeByte, 88	RandomAccessExternalEeprom, 86

readInt	Seekable, 92
DataInput, 25	Seekable, 91
DataInputStream, 29	seek, 92
RandomAccessByteArray, 76	Seekable.cpp, 179, 180
RandomAccessExternalEeprom, 86	ARDUINO_IO_SEEKABLE_CPP, 180
readLong	Seekable.h, 180, 181
DataInput, 25	SeekableInputStream, 92
DataInputStream, 29	SeekableInputStream.cpp, 181
RandomAccessByteArray, 77	ARDUINO_IO_SEEKABLE_INPUT_STREAM
RandomAccessExternalEeprom, 86	_CPP, 181
readUnsignedChar	SeekableInputStream.h, 182
DataInput, 26	SerialInputStream, 93
DataInputStream, 29	SerialInputStream.cpp, 183
RandomAccessByteArray, 77	ARDUINO_IO_SERIAL_INPUT_STREAM_C
RandomAccessExternalEeprom, 86	PP, 183
readUnsignedInt	SerialInputStream.h, 184
DataInput, 26	SerialOutputStream, 95
DataInputStream, 30	SerialOutputStream.cpp, 185
RandomAccessByteArray, 77	ARDUINO_IO_SERIAL_OUTPUT_STREAM_
RandomAccessExternalEeprom, 87	CPP, 185
readUnsignedLong	SerialOutputStream.h, 185, 186
DataInput, 26	size
DataInputStream, 30	BufferedInputStream, 11
RandomAccessByteArray, 77	BufferedOutputStream, 15
RandomAccessExternalEeprom, 87	ByteArrayOutputStream, 20
readWord	skip
DataInput, 26	BufferedInputStream, 10
DataInputStream, 30	FilterInputStream, 55
RandomAccessByteArray, 77	InputStream, 65
RandomAccessExternalEeprom, 87 realineBufferContent	skipBytes
BufferedInputStream, 10	DataInput, 26 DataInputStream, 30
reset	RandomAccessByteArray, 79
BufferedInputStream, 10	RandomAccessExternalEeprom, 87
ByteArrayInputStream, 17	softwareSerial
ByteArrayOutputStream, 20	SoftwareSerialInputStream, 98
ExternalEepromInputStream, 44	SoftwareSerialOutputStream, 100
FilterInputStream, 54	SoftwareSerialInputStream, 96
InputStream, 65	available, 98
PgmspaceInputStream, 70	read, 98
ResourceInputStream.cpp, 176	softwareSerial, 98
ARDUINO_IO_RESOURCE_INPUT_STREA↔	SoftwareSerialInputStream, 97
M_CPP, 176	SoftwareSerialInputStream.cpp, 186, 187
ResourceInputStream.h, 177	ARDUINO_IO_SOFTWARE_SERIAL_INPUT↔
ResourceOutputStream.cpp, 177, 178	STREAM_CPP, 187
ARDUINO_IO_RESOURCE_OUTPUT_STRE	SoftwareSerialInputStream.h, 188, 189
AM_CPP, 177	SoftwareSerialOutputStream, 98
ResourceOutputStream.h, 178	softwareSerial, 100
ResourceSeekableInputStream.cpp, 178, 179	SoftwareSerialOutputStream, 100
ARDUINO_IO_RESOURCE_SEEKABLE_IN←	write, 100
PUT_STREAM_CPP, 178	SoftwareSerialOutputStream.cpp, 189, 190
ResourceSeekableInputStream.h, 179	ARDUINO_IO_SOFTWARE_SERIAL_OUTP
	UT_STREAM_CPP, 189
seek	SoftwareSerialOutputStream.h, 190, 191
ByteArraySeekableInputStream, 22	
ExternalEepromSeekableInputStream, 48	toByteArray
PgmspaceSeekableInputStream, 72	ByteArrayOutputStream, 20
RandomAccessByteArray, 77	WirelenutStroom 100
RandomAccessExternalEeprom, 87	WireInputStream, 100

address, 102	DataOutput, 34
available, 102	DataOutputStream, 40
read, 102	RandomAccessByteArray, 80
WireInputStream, 102	RandomAccessExternalEeprom, 89
WireInputStream.cpp, 191, 192	writeUnsignedChar
ARDUINO_IO_WIRE_INPUT_STREAM_CPP←	DataOutput, 34
, 192	DataOutputStream, 40
WireInputStream.h, 193, 194	RandomAccessByteArray, 82
WireOutputStream.cpp, 194	RandomAccessExternalEeprom, 89
WireOutputStream.h, 194	writeUnsignedInt
write	DataOutput, 34
BufferedOutputStream, 14	DataOutputStream, 41
ByteArrayOutputStream, 20	RandomAccessByteArray, 82
DataOutput, 32	RandomAccessExternalEeprom, 91
DataOutputStream, 39	writeUnsignedLong
ExternalEepromOutputStream, 46	DataOutput, 34
FilterOutputStream, 57, 59	DataOutputStream, 41
HardwareSerialOutputStream, 63	RandomAccessByteArray, 82
OutputStream, 67	RandomAccessExternalEeprom, 91
RandomAccessByteArray, 79	writeWord
RandomAccessExternalEeprom, 88	DataOutput, 36
SoftwareSerialOutputStream, 100	DataOutputStream, 41
writeBoolean	RandomAccessByteArray, 82
DataOutput, 32	RandomAccessExternalEeprom, 91
DataOutputStream, 39	
RandomAccessByteArray, 79	
RandomAccessExternalEeprom, 88	
writeByte	
DataOutput, 32	
DataOutputStream, 39	
RandomAccessByteArray, 79	
RandomAccessExternalEeprom, 88	
writeBytes	
DataOutput, 32	
DataOutputStream, 39	
RandomAccessByteArray, 80	
RandomAccessExternalEeprom, 88	
writeChar	
DataOutput, 32	
DataOutputStream, 40	
RandomAccessByteArray, 80	
RandomAccessExternalEeprom, 89	
writeDouble	
DataOutput, 34	
DataOutputStream, 40	
RandomAccessByteArray, 80	
RandomAccessExternalEeprom, 89	
writeFloat	
DataOutput, 34	
DataOutputStream, 40	
RandomAccessByteArray, 80	
RandomAccessExternalEeprom, 89	
writeInt	
DataOutput, 34	
DataOutputStream, 40	
RandomAccessByteArray, 80	
RandomAccessExternalEeprom, 89	
writeLong	