Arduino IO Library Driver

Generated by Doxygen 1.8.9.1

Tue Dec 29 2015 15:18:31

ii CONTENTS

Contents

1	Hier	archica	I Index	1
	1.1	Class	Hierarchy	1
2	Clas	s Index	τ	2
	2.1	Class	List	2
3	File	Index		4
•	3.1		st	4
	0.1	THE EN		
4	Clas		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	14
		4.2.3	Member Function Documentation	14
		4.2.4	Member Data Documentation	15
	4.3	ByteA	rrayInputStream Class Reference	16
		4.3.1	Detailed Description	17
		4.3.2	Constructor & Destructor Documentation	17
		4.3.3	Member Function Documentation	17
		4.3.4	Member Data Documentation	18
	4.4	ByteA	rrayOutputStream Class Reference	19
		-	Detailed Description	20
		4.4.2	Constructor & Destructor Documentation	20
		4.4.3	Member Function Documentation	20
		4.4.4	Member Data Documentation	22
	4.5	ByteA	rraySeekableInputStream Class Reference	22
		4.5.1	Detailed Description	23
		4.5.2	Constructor & Destructor Documentation	23
		4.5.3	Member Function Documentation	23
	4.6		able Class Reference	24
	1 .∪	4.6.1	Detailed Description	24
		4.6.2	Member Function Documentation	25
	47	_	nput Class Reference	
	4.7		•	25
		4.7.1	Detailed Description	26

	4.7.2	Member Function Documentation	26
4.8	DataIn	putStream Class Reference	29
	4.8.1	Detailed Description	30
	4.8.2	Constructor & Destructor Documentation	30
	4.8.3	Member Function Documentation	30
	4.8.4	Member Data Documentation	34
4.9	DataOu	utput Class Reference	34
	4.9.1	Detailed Description	35
	4.9.2	Member Function Documentation	35
4.10	DataOu	utputStream Class Reference	39
	4.10.1	Detailed Description	40
	4.10.2	Constructor & Destructor Documentation	41
	4.10.3	Member Function Documentation	42
	4.10.4	Member Data Documentation	44
4.11	Externa	alEepromInputStream Class Reference	44
	4.11.1	Detailed Description	46
	4.11.2	Constructor & Destructor Documentation	46
	4.11.3	Member Function Documentation	46
	4.11.4	Member Data Documentation	47
4.12	Externa	alEepromOutputStream Class Reference	48
	4.12.1	Detailed Description	49
	4.12.2	Constructor & Destructor Documentation	49
	4.12.3	Member Function Documentation	50
	4.12.4	Member Data Documentation	50
4.13	Externa	alEepromSeekableInputStream Class Reference	50
	4.13.1	Detailed Description	52
	4.13.2	Constructor & Destructor Documentation	52
	4.13.3	Member Function Documentation	52
4.14	FilterIn	putStream Class Reference	52
	4.14.1	Detailed Description	54
	4.14.2	Constructor & Destructor Documentation	54
	4.14.3	Member Function Documentation	54
	4.14.4	Member Data Documentation	57
4.15	FilterO	utputStream Class Reference	57
	4.15.1	Detailed Description	58
	4.15.2	Constructor & Destructor Documentation	58
	4.15.3	Member Function Documentation	59
		Member Data Documentation	61
4.16	Hardwa	areSerialInputStream Class Reference	61
	4.16.1	Detailed Description	62

iv CONTENTS

	4.16.2	Constructor & Destructor Documentation	62
	4.16.3	Member Function Documentation	62
4.17	Hardwa	areSerialOutputStream Class Reference	63
	4.17.1	Detailed Description	64
	4.17.2	Constructor & Destructor Documentation	64
	4.17.3	Member Function Documentation	64
4.18	InputSt	tream Class Reference	65
	4.18.1	Detailed Description	66
	4.18.2	Member Function Documentation	66
4.19	Interna	IEepromInputStream Class Reference	67
	4.19.1	Detailed Description	69
	4.19.2	Constructor & Destructor Documentation	69
	4.19.3	Member Function Documentation	69
	4.19.4	Member Data Documentation	70
4.20	Interna	IEepromOutputStream Class Reference	70
	4.20.1	Detailed Description	71
	4.20.2	Constructor & Destructor Documentation	72
	4.20.3	Member Function Documentation	72
	4.20.4	Member Data Documentation	72
4.21	Output	Stream Class Reference	72
	4.21.1	Detailed Description	73
		Member Function Documentation	73
4.22		aceInputStream Class Reference	74
		Detailed Description	76
		Constructor & Destructor Documentation	76
	4.22.3	Member Function Documentation	76
		Member Data Documentation	77
4.23	Pgmsp	aceSeekableInputStream Class Reference	77
	4.23.1	Detailed Description	79
	4.23.2	Constructor & Destructor Documentation	79
	4.23.3	Member Function Documentation	79
4.24		mAccess Class Reference	79
		Detailed Description	80
4.25		mAccessByteArray Class Reference	80
		Detailed Description	82
		Constructor & Destructor Documentation	82
		Member Function Documentation	82
		Member Data Documentation	89
4.26		mAccessExternalEeprom Class Reference	90
	4.26.1	Detailed Description	91

	4.26.2 Constructor & Destructor Documentation	91
	4.26.3 Member Function Documentation	92
	4.26.4 Member Data Documentation	98
4.27	Seekable Class Reference	98
	4.27.1 Detailed Description	99
	4.27.2 Member Function Documentation	99
4.28	SeekableInputStream Class Reference	99
	4.28.1 Detailed Description	00
4.29	SerialInputStream Class Reference	00
	4.29.1 Detailed Description	01
4.30	SerialOutputStream Class Reference	02
	4.30.1 Detailed Description	02
4.31	SoftwareSerialInputStream Class Reference	03
	4.31.1 Detailed Description	04
	4.31.2 Constructor & Destructor Documentation	04
	4.31.3 Member Function Documentation	05
	4.31.4 Member Data Documentation	05
4.32	SoftwareSerialOutputStream Class Reference	05
	4.32.1 Detailed Description	07
	4.32.2 Constructor & Destructor Documentation	07
	4.32.3 Member Function Documentation	07
	4.32.4 Member Data Documentation	07
4.33	WireInputStream Class Reference	07
	4.33.1 Detailed Description	09
	4.33.2 Constructor & Destructor Documentation	09
	4.33.3 Member Function Documentation	09
	4.33.4 Member Data Documentation	09
Eilo I	Documentation 1 ⁻	10
5.1	BufferedInputStream.cpp File Reference	
5.2	BufferedInputStream.cpp	
5.3	BufferedInputStream.h File Reference	
5.4	BufferedInputStream.h	
5.5	BufferedOutputStream.cpp File Reference	
5.6	BufferedOutputStream.cpp	
5.7	BufferedOutputStream.h File Reference	
5.7	BufferedOutputStream.h	
5.9	ByteArrayInputStream.cpp File Reference	
J.8	5.9.1 Macro Definition Documentation	
E 10	ByteArrayInputStream.cpp	
5.10	byteAmayinputotream.cpp	1/

5

vi CONTENTS

5.11	ByteArrayInputStream.h File Reference	18
5.12	ByteArrayInputStream.h	19
5.13	ByteArrayOutputStream.cpp File Reference	20
	5.13.1 Macro Definition Documentation	20
5.14	ByteArrayOutputStream.cpp	20
5.15	ByteArrayOutputStream.h File Reference	21
5.16	ByteArrayOutputStream.h	22
5.17	ByteArraySeekableInputStream.cpp File Reference	22
5.18	ByteArraySeekableInputStream.cpp	23
5.19	ByteArraySeekableInputStream.h File Reference	23
5.20	ByteArraySeekableInputStream.h	24
5.21	Closeable.cpp File Reference	25
	5.21.1 Macro Definition Documentation	25
5.22	Closeable.cpp	25
5.23	Closeable.h File Reference	26
5.24	Closeable.h	26
5.25	DataInput.cpp File Reference	26
	5.25.1 Macro Definition Documentation	26
5.26	DataInput.cpp	27
5.27	DataInput.h File Reference	27
5.28	DataInput.h	28
5.29	DataInputStream.cpp File Reference	28
	5.29.1 Macro Definition Documentation	29
5.30	DataInputStream.cpp	29
5.31	DataInputStream.h File Reference	30
5.32	DataInputStream.h	31
5.33	DataOutput.cpp File Reference	31
	5.33.1 Macro Definition Documentation	32
5.34	DataOutput.cpp	32
5.35	DataOutput.h File Reference	32
5.36	DataOutput.h	33
5.37	DataOutputStream.cpp File Reference	34
	5.37.1 Macro Definition Documentation	34
5.38	DataOutputStream.cpp	34
5.39	DataOutputStream.h File Reference	35
5.40	DataOutputStream.h	36
5.41	ExternalEepromInputStream.cpp File Reference	37
5.42	ExternalEepromInputStream.cpp	37
5.43	ExternalEepromInputStream.h File Reference	38
5.44	ExternalEepromInputStream.h	39

CONTENTS vii

5.45	ExternalEepromOutputStream.cpp File Reference	140
	5.45.1 Macro Definition Documentation	140
5.46	ExternalEepromOutputStream.cpp	140
5.47	ExternalEepromOutputStream.h File Reference	141
5.48	ExternalEepromOutputStream.h	142
5.49	ExternalEepromSeekableInputStream.cpp File Reference	142
	5.49.1 Macro Definition Documentation	143
5.50	ExternalEepromSeekableInputStream.cpp	143
5.51	ExternalEepromSeekableInputStream.h File Reference	144
5.52	ExternalEepromSeekableInputStream.h	144
5.53	FilterInputStream.cpp File Reference	145
	5.53.1 Macro Definition Documentation	145
5.54	FilterInputStream.cpp	146
5.55	FilterInputStream.h File Reference	146
5.56	FilterInputStream.h	147
5.57	FilterOutputStream.cpp File Reference	148
	5.57.1 Macro Definition Documentation	148
	FilterOutputStream.cpp	
5.59	FilterOutputStream.h File Reference	149
5.60	FilterOutputStream.h	150
5.61	HardwareSerialInputStream.cpp File Reference	151
	5.61.1 Macro Definition Documentation	151
5.62	HardwareSerialInputStream.cpp	152
5.63	HardwareSerialInputStream.h File Reference	152
5.64	HardwareSerialInputStream.h	153
5.65	HardwareSerialOutputStream.cpp File Reference	153
	5.65.1 Macro Definition Documentation	154
5.66	HardwareSerialOutputStream.cpp	154
5.67	HardwareSerialOutputStream.h File Reference	155
5.68	HardwareSerialOutputStream.h	156
5.69	InputStream.cpp File Reference	156
	5.69.1 Macro Definition Documentation	156
5.70	InputStream.cpp	157
5.71	InputStream.h File Reference	157
5.72	InputStream.h	158
5.73	InternalEepromInputStream.h File Reference	158
5.74	InternalEepromInputStream.h	159
5.75	InternalEepromOutputStream.cpp File Reference	160
	5.75.1 Macro Definition Documentation	160
5.76	InternalEepromOutputStream.cpp	160

viii CONTENTS

5.77 InternalEepromOutputStream.h File Reference
5.78 InternalEepromOutputStream.h
5.79 OutputStream.cpp File Reference
5.79.1 Macro Definition Documentation
5.80 OutputStream.cpp
5.81 OutputStream.h File Reference
5.82 OutputStream.h
5.83 PgmspaceInputStream.cpp File Reference
5.83.1 Macro Definition Documentation
5.84 PgmspaceInputStream.cpp
5.85 PgmspaceInputStream.h File Reference
5.86 PgmspaceInputStream.h
5.87 PgmspaceSeekableInputStream.cpp File Reference
5.87.1 Macro Definition Documentation
5.88 PgmspaceSeekableInputStream.cpp
5.89 PgmspaceSeekableInputStream.h File Reference
5.90 PgmspaceSeekableInputStream.h
5.91 RandomAccess.cpp File Reference
5.91.1 Macro Definition Documentation
5.92 RandomAccess.cpp
5.93 RandomAccess.h File Reference
5.94 RandomAccess.h
5.95 RandomAccessByteArray.cpp File Reference
5.95.1 Macro Definition Documentation
5.96 RandomAccessByteArray.cpp
5.97 RandomAccessByteArray.h File Reference
5.98 RandomAccessByteArray.h
5.99 RandomAccessExternalEeprom.cpp File Reference
5.99.1 Macro Definition Documentation
5.100RandomAccessExternalEeprom.cpp
5.101RandomAccessExternalEeprom.h File Reference
5.102RandomAccessExternalEeprom.h
5.103RandomAccessResource.cpp File Reference
5.103.1 Macro Definition Documentation
5.104RandomAccessResource.cpp
5.105RandomAccessResource.h File Reference
5.106RandomAccessResource.h
5.107ResourceInputStream.cpp File Reference
5.107.1 Macro Definition Documentation
5.108ResourceInputStream.cpp

5.109ResourceInputStream.h File Reference
5.110ResourceInputStream.h
5.111 ResourceOutputStream.cpp File Reference
5.111.1 Macro Definition Documentation
5.112ResourceOutputStream.cpp
5.113ResourceOutputStream.h File Reference
5.114ResourceOutputStream.h
5.115ResourceSeekableInputStream.cpp File Reference
5.115.1 Macro Definition Documentation
5.116ResourceSeekableInputStream.cpp
5.117ResourceSeekableInputStream.h File Reference
5.118ResourceSeekableInputStream.h
5.119Seekable.cpp File Reference
5.119.1 Macro Definition Documentation
5.120Seekable.cpp
5.121 Seekable.h File Reference
5.122Seekable.h
5.123SeekableInputStream.cpp File Reference
5.123.1 Macro Definition Documentation
5.124SeekableInputStream.cpp
5.125SeekableInputStream.h File Reference
5.126SeekableInputStream.h
5.127SerialInputStream.cpp File Reference
5.127.1 Macro Definition Documentation
5.128SerialInputStream.cpp
5.129SerialInputStream.h File Reference
5.130SerialInputStream.h
5.131 SerialOutputStream.cpp File Reference
5.131.1 Macro Definition Documentation
5.132SerialOutputStream.cpp
5.133 SerialOutputStream.h File Reference
5.134SerialOutputStream.h
5.135SoftwareSerialInputStream.cpp File Reference
5.135.1 Macro Definition Documentation
5.136SoftwareSerialInputStream.cpp
5.137SoftwareSerialInputStream.h File Reference
5.138SoftwareSerialInputStream.h
5.139SoftwareSerialOutputStream.cpp File Reference
5.139.1 Macro Definition Documentation
5.140SoftwareSerialOutputStream.cpp

1 Hierarchical Index 1

	5.141 Software Serial Output Stream.h File Reference	100
	5.142SoftwareSerialOutputStream.h	
	5.143WireInputStream.cpp File Reference	
	5.143.1 Macro Definition Documentation	
	5.144WireInputStream.cpp	
	5.145WireInputStream.h File Reference	202
	5.146WireInputStream.h	203
	5.147WireOutputStream.cpp File Reference	203
	5.148WireOutputStream.cpp	203
	5.149WireOutputStream.h File Reference	203
	$5.150 Wire Output Stream.h. \dots $	203
Ind	lex	205
1	Hierarchical Index	
1.1	Class Hierarchy	
	s inheritance list is sorted roughly, but not completely, alphabetically:	
		0.4
	Closeable	24
	InputStream	65
	ByteArrayInputStream	16
	ByteArraySeekableInputStream	22
	ExternalEepromInputStream	44
	ExternalEepromSeekableInputStream	50
	FilterInputStream	52
	BufferedInputStream	7
	InternalEepromInputStream	67
	PgmspaceInputStream	74
	PgmspaceSeekableInputStream	77
	SeekableInputStream	99
	ByteArraySeekableInputStream	22
	ExternalEepromSeekableInputStream	50
	PgmspaceSeekableInputStream	77
	SerialInputStream	100
	HardwareSerialInputStream	61

SoftwareSerialInputStream	103
WireInputStream	107
OutputStream	72
ByteArrayOutputStream	19
ExternalEepromOutputStream	48
FilterOutputStream	57
BufferedOutputStream	12
InternalEepromOutputStream	70
SerialOutputStream	102
HardwareSerialOutputStream	63
SoftwareSerialOutputStream	105
RandomAccess	79
RandomAccessByteArray	80
RandomAccessExternalEeprom	90
RandomAccessByteArray	80
RandomAccessExternalEeprom	90
DataInput	25
DataInputStream	29
RandomAccess	79
DataOutput	34
DataOutputStream	39
RandomAccess	79
Seekable	98
RandomAccess	79
SeekableInputStream	99
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

2

2.1 Class List

ByteArrayInputStream Arduino IO	16
ByteArrayOutputStream Arduino IO	19
ByteArraySeekableInputStream Arduino IO	22
Closeable Arduino IO	24
DataInput Arduino IO	25
DataInputStream Arduino IO	29
DataOutput Arduino IO	34
DataOutputStream Arduino IO	39
ExternalEepromInputStream Arduino IO	44
ExternalEepromOutputStream Arduino IO	48
ExternalEepromSeekableInputStream Arduino IO	50
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	52
FilterOutputStream Arduino IO	57
HardwareSerialInputStream Arduino IO	61
HardwareSerialOutputStream Arduino IO	63
InputStream Arduino IO	65
InternalEepromInputStream Arduino IO	67
InternalEepromOutputStream Arduino IO	70
OutputStream Arduino IO	72
PgmspaceInputStream Arduino IO	74

	PgmspaceSeekableInputStream Arduino IO	77
	RandomAccess Araduino IO	79
	RandomAccessByteArray Araduino IO	80
	RandomAccessExternalEeprom Araduino IO	90
	Seekable Arduino IO	98
	SeekableInputStream Arduino IO	99
	SerialInputStream Arduino IO	100
	SerialOutputStream Arduino IO	102
	SoftwareSerialInputStream Arduino IO	103
	SoftwareSerialOutputStream Arduino IO	105
	WireInputStream Arduino IO	107
3	File Index	
3.1	File List	
Her	re is a list of all files with brief descriptions:	
	BufferedInputStream.cpp	110
	BufferedInputStream.h	112
	BufferedOutputStream.cpp	114
	BufferedOutputStream.h	115
	ByteArrayInputStream.cpp	116
	ByteArrayInputStream.h	118
	ByteArrayOutputStream.cpp	120
	ByteArrayOutputStream.h	121
	ByteArraySeekableInputStream.cpp	122
	ByteArraySeekableInputStream.h	123
	Closeable.cpp	125

3

3.1 File List 5

Closeable.h	126
DataInput.cpp	126
DataInput.h	127
DataInputStream.cpp	128
DataInputStream.h	130
DataOutput.cpp	131
DataOutput.h	132
DataOutputStream.cpp	134
DataOutputStream.h	135
ExternalEepromInputStream.cpp	137
ExternalEepromInputStream.h	138
ExternalEepromOutputStream.cpp	140
ExternalEepromOutputStream.h	141
ExternalEepromSeekableInputStream.cpp	142
ExternalEepromSeekableInputStream.h	144
FilterInputStream.cpp	145
FilterInputStream.h	146
FilterOutputStream.cpp	148
FilterOutputStream.h	149
HardwareSerialInputStream.cpp	151
HardwareSerialInputStream.h	152
HardwareSerialOutputStream.cpp	153
HardwareSerialOutputStream.h	155
InputStream.cpp	156
InputStream.h	157
InternalEepromInputStream.h	158
InternalEepromOutputStream.cpp	160
InternalEepromOutputStream.h	161
OutputStream.cpp	162
OutputStream.h	163
PgmspaceInputStream.cpp	164
PgmspaceInputStream.h	166

PgmspaceSeekableInputStream.cpp	167
PgmspaceSeekableInputStream.h	169
RandomAccess.cpp	170
RandomAccess.h	171
RandomAccessByteArray.cpp	172
RandomAccessByteArray.h	174
RandomAccessExternalEeprom.cpp	176
RandomAccessExternalEeprom.h	179
RandomAccessResource.cpp	181
RandomAccessResource.h	184
ResourceInputStream.cpp	185
ResourceInputStream.h	186
ResourceOutputStream.cpp	186
ResourceOutputStream.h	187
ResourceSeekableInputStream.cpp	187
ResourceSeekableInputStream.h	188
Seekable.cpp	188
Seekable.h	189
SeekableInputStream.cpp	190
SeekableInputStream.h	191
SerialInputStream.cpp	192
SerialInputStream.h	193
SerialOutputStream.cpp	194
SerialOutputStream.h	194
SoftwareSerialInputStream.cpp	195
SoftwareSerialInputStream.h	197
SoftwareSerialOutputStream.cpp	198
SoftwareSerialOutputStream.h	199
WireInputStream.cpp	200
WireInputStream.h	202
WireOutputStream.cpp	203
WireOutputStream.h	203

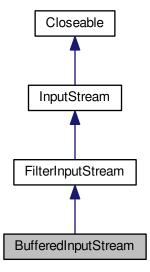
4 Class Documentation 7

4 Class Documentation

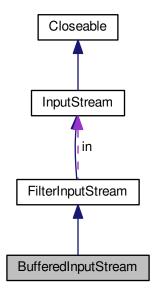
4.1 BufferedInputStream Class Reference

#include <BufferedInputStream.h>

Inheritance diagram for BufferedInputStream:



Collaboration diagram for BufferedInputStream:



Public Member Functions

- BufferedInputStream (InputStream *in, unsigned char *buf, int size)
- virtual ~BufferedInputStream ()
- 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
- · bool marked
- · int markpos

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.

Arduino IO.

Parameters

in	
buf	
size	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 26 of file BufferedInputStream.cpp.

```
4.1.2.2 virtual BufferedInputStream::~BufferedInputStream( ) [inline], [virtual]
```

Virtual destructor.

Definition at line 114 of file BufferedInputStream.h.

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 30 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 34 of file BufferedInputStream.cpp.

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

Fills the buffer.

Parameters

```
startPos
```

Definition at line 119 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 131 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 138 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 91 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	
ĺ	len	

Returns

Reimplemented from FilterInputStream.

Definition at line 44 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 48 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 107 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 38 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 142 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 72 of file BufferedInputStream.h.

```
4.1.4.4 int BufferedInputStream::markpos [protected]
```

The value of the $\verb"pos"$ field at the time the last $\verb"mark"$ 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 98 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 \mathtt{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.

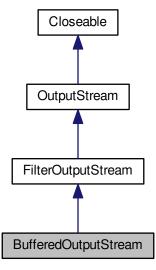
The documentation for this class was generated from the following files:

- · BufferedInputStream.h
- BufferedInputStream.cpp

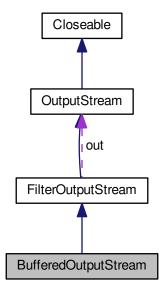
4.2 BufferedOutputStream Class Reference

#include <BufferedOutputStream.h>

Inheritance diagram for BufferedOutputStream:



Collaboration diagram for BufferedOutputStream:



Public Member Functions

- BufferedOutputStream (OutputStream *out, unsigned char *buf, int size)
- virtual ~BufferedOutputStream ()
- 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

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

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

Arduino IO.

Parameters

out	the underlying output stream.
size	the buffer size.

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 14 of file BufferedOutputStream.cpp.

4.2.2.2 virtual BufferedOutputStream::~BufferedOutputStream() [virtual]

Virtual destructor.

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 55 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 50 of file BufferedOutputStream.cpp.

```
4.2.3.3 void BufferedOutputStream::flushBuffer( ) [private]
```

Flush the internal buffer.

Definition at line 60 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 18 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 25 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 29 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 24 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 37 of file BufferedOutputStream.h.

4.2.4.3 int BufferedOutputStream::size [protected]

The size of the buffer where data is stored.

Definition at line 29 of file BufferedOutputStream.h.

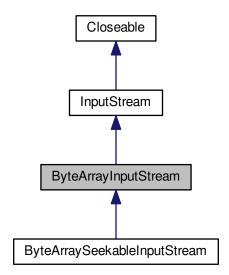
The documentation for this class was generated from the following files:

- BufferedOutputStream.h
- BufferedOutputStream.cpp

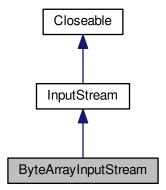
4.3 ByteArrayInputStream Class Reference

#include <ByteArrayInputStream.h>

Inheritance diagram for ByteArrayInputStream:



Collaboration diagram for ByteArrayInputStream:



Public Member Functions

- ByteArrayInputStream (unsigned char *buf, unsigned int count)
- virtual ~ByteArrayInputStream ()
- 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.

```
4.3.2.2 virtual ByteArrayInputStream:: >ByteArrayInputStream() [inline], [virtual]
```

Virtual destructor.

Definition at line 46 of file ByteArrayInputStream.h.

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.

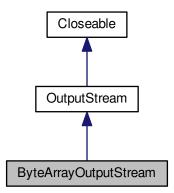
```
void ByteArrayInputStream::mark( ) [virtual]
Marks the current position in this input stream.
Reimplemented from InputStream.
Definition at line 29 of file ByteArrayInputStream.cpp.
4.3.3.3 bool ByteArrayInputStream::markSupported( ) [virtual]
Tests if this input stream supports the mark and reset 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 22 of file ByteArrayInputStream.h.
4.3.4.2 unsigned int ByteArrayInputStream::count [protected]
Definition at line 27 of file ByteArrayInputStream.h.
4.3.4.3 unsigned int ByteArrayInputStream::markpos [protected]
Definition at line 37 of file ByteArrayInputStream.h.
4.3.4.4 unsigned int ByteArrayInputStream::pos [protected]
Definition at line 32 of file ByteArrayInputStream.h.
The documentation for this class was generated from the following files:
```

- · ByteArrayInputStream.h
- ByteArrayInputStream.cpp

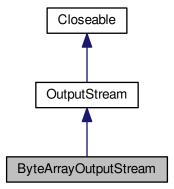
4.4 ByteArrayOutputStream Class Reference

#include <ByteArrayOutputStream.h>

Inheritance diagram for ByteArrayOutputStream:



Collaboration diagram for ByteArrayOutputStream:



Public Member Functions

- ByteArrayOutputStream (unsigned char *buf, unsigned int count)
- virtual ~ByteArrayOutputStream ()
- 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.

4.4.2.2 virtual ByteArrayOutputStream::∼ByteArrayOutputStream() [inline], [virtual]

Virtual destructor.

Definition at line 47 of file ByteArrayOutputStream.h.

- 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 22 of file ByteArrayOutputStream.h.

4.4.4.2 unsigned int ByteArrayOutputStream::count [protected]

Definition at line 27 of file ByteArrayOutputStream.h.

4.4.4.3 unsigned int ByteArrayOutputStream::pos [protected]

Definition at line 32 of file ByteArrayOutputStream.h.

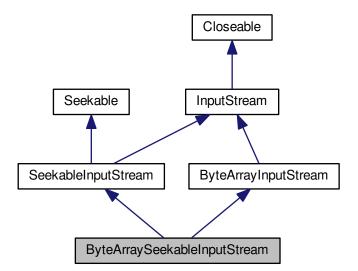
The documentation for this class was generated from the following files:

- · ByteArrayOutputStream.h
- ByteArrayOutputStream.cpp

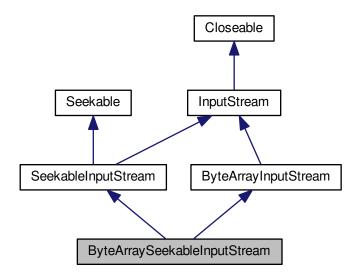
4.5 ByteArraySeekableInputStream Class Reference

#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)

Arduino IO.

ByteArraySeekableInputStream

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

Definition at line 12 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 16 of file ByteArraySeekableInputStream.cpp.

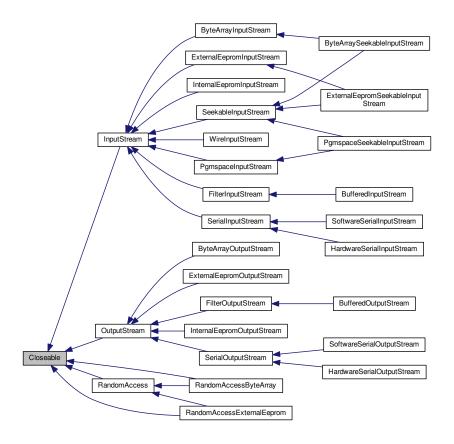
The documentation for this class was generated from the following files:

- 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, BufferedOutputStream, FilterOutputStream, Random← AccessByteArray, RandomAccessExternalEeprom, InputStream, and OutputStream.

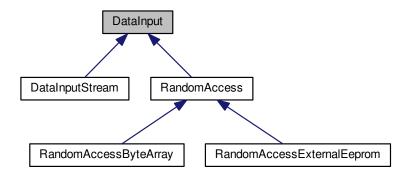
The documentation for this class was generated from the following file:

· 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

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataInputStream.

```
4.7.2.8 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

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

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.

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

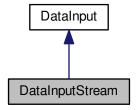
The documentation for this class was generated from the following file:

• DataInput.h

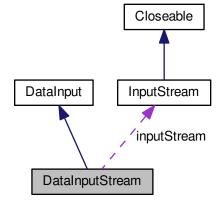
4.8 DataInputStream Class Reference

#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
- $\textbf{4.8.2.1} \quad \textbf{DataInputStream}: \textbf{DataInputStream} \ (\ \textbf{InputStream} * \textit{inputStream} \)$

Public constructor.

Parameters

inputStream

Definition at line 14 of file DataInputStream.cpp.

4.8.3 Member Function Documentation

4.8.3.1 bool DataInputStream::readBoolean() [virtual]

Reads a bool from the stream.

Returns

bool

Implements DataInput.

Definition at line 22 of file DataInputStream.cpp.

```
4.8.3.2 unsigned char DataInputStream::readByte() [virtual]
Reads a unsigned char from the stream.
Returns
     unsigned char
Implements DataInput.
Definition at line 18 of file DataInputStream.cpp.
4.8.3.3 char DataInputStream::readChar( ) [virtual]
Reads a char from the stream.
Returns
     char
Implements DataInput.
Definition at line 26 of file DataInputStream.cpp.
4.8.3.4 double DataInputStream::readDouble( ) [virtual]
Reads a double from the stream.
Returns
     double
Implements DataInput.
Definition at line 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.
```

Generated on Tue Dec 29 2015 15:18:31 for Arduino IO Library Driver by Doxygen

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.
4.8.3.9 unsigned char DataInputStream::readUnsignedChar() [virtual]
Reads an unsigned char from the stream.
Returns
     unsigned char
Implements DataInput.
Definition at line 30 of file DataInputStream.cpp.
4.8.3.10 unsigned int DataInputStream::readUnsignedInt() [virtual]
Reads an unsigned int from the stream.
Returns
     unsigned int
Implements DataInput.
Definition at line 42 of file DataInputStream.cpp.
4.8.3.11 unsigned long DataInputStream::readUnsignedLong() [virtual]
Reads a unsigned long from the stream.
Returns
     unsigned long
Implements DataInput.
Definition at line 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.

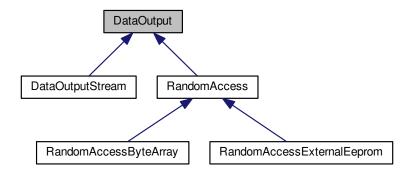
The documentation for this class was generated from the following files:

- · 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

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

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

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

Writes a unsigned char into the stream.

Parameters

b The unsigned char to be written.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

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

Writes a bool into the stream.

Parameters

V	The bool to be written.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

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

Writes a unsigned char into the stream.

Parameters

b	The unsigned char to be written.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

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

Writes an array of bytes into the stream.

Parameters

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

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

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

Writes a char into the stream.

Parameters

c The char to be written.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

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

Writes a double into the stream.

Parameters

v The double to be written.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

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

Writes a float into the stream.

Parameters

v The float to be written.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

4.9.2.9 virtual void DataOutput::writeInt (int \nu) [pure virtual]

Writes an int into the stream.

Parameters

v The int to be written.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

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

Writes a long into the stream.

Parameters

ν 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 ν) [pure virtual]

Writes a unsigned long into the stream.

Parameters

v The unsigned long to be written.

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, and DataOutputStream.

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.

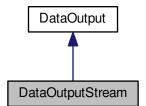
The documentation for this class was generated from the following file:

• DataOutput.h

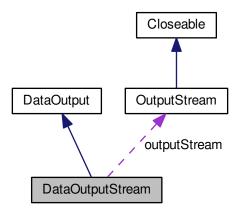
4.10 DataOutputStream Class Reference

#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
- $\textbf{4.10.2.1} \quad \textbf{DataOutputStream::DataOutputStream (\ \textbf{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

|--|

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

v 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

v 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

v 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

v 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.

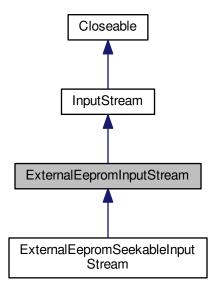
The documentation for this class was generated from the following files:

- · DataOutputStream.h
- DataOutputStream.cpp

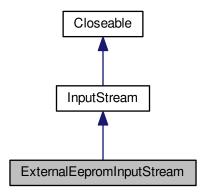
4.11 ExternalEepromInputStream Class Reference

#include <ExternalEepromInputStream.h>

 $Inheritance\ diagram\ for\ External Eeprom Input Stream:$



Collaboration diagram for ExternalEepromInputStream:



Public Member Functions

- ExternalEepromInputStream (ExternalEeprom *externalEeprom)
- virtual \sim ExternalEepromInputStream ()
- 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

- · const int maxAvailableChunk
- 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.

Arduino IO.

Parameters

externalEeprom The externalEeprom where data is stored.

ExternalEepromInputStream

An ExternalEepromInputStream obtains input bytes from a externalEeprom.

Definition at line 12 of file ExternalEepromInputStream.cpp.

4.11.2.2 virtual ExternalEepromInputStream: ~ExternalEepromInputStream() [inline], [virtual]

Virtual destructor.

Definition at line 57 of file ExternalEepromInputStream.h.

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 16 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 24 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 28 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 32 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 39 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 48 of file ExternalEepromInputStream.cpp.

4.11.4 Member Data Documentation

4.11.4.1 ExternalEeprom* ExternalEepromInputStream::externalEeprom [protected]

Definition at line 28 of file ExternalEepromInputStream.h.

4.11.4.2 unsigned int ExternalEepromInputStream::externalEepromSize [protected]

Definition at line 43 of file ExternalEepromInputStream.h.

4.11.4.3 unsigned int ExternalEepromInputStream::markpos [protected]

Definition at line 38 of file ExternalEepromInputStream.h.

4.11.4.4 const int ExternalEepromInputStream::maxAvailableChunk [protected]

When asking for available, this is the max number to return.

Definition at line 23 of file ExternalEepromInputStream.h.

4.11.4.5 unsigned int ExternalEepromInputStream::pos [protected]

Definition at line 33 of file ExternalEepromInputStream.h.

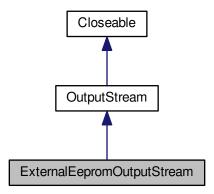
The documentation for this class was generated from the following files:

- ExternalEepromInputStream.h
- ExternalEepromInputStream.cpp

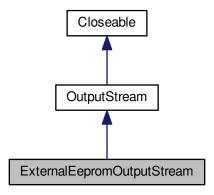
4.12 ExternalEepromOutputStream Class Reference

#include <ExternalEepromOutputStream.h>

Inheritance diagram for ExternalEepromOutputStream:



Collaboration diagram for ExternalEepromOutputStream:



Public Member Functions

- ExternalEepromOutputStream (ExternalEeprom *externalEeprom)
- virtual ~ExternalEepromOutputStream ()
- 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.

4.12.2.2 virtual ExternalEepromOutputStream::~ExternalEepromOutputStream() [inline], [virtual]

Definition at line 37 of file ExternalEepromOutputStream.h.

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.

 $\textbf{4.12.4.2} \quad unsigned \ int \ \textbf{ExternalEepromOutputStream::pos} \quad \texttt{[private]}$

Current eeprom position.

Definition at line 26 of file ExternalEepromOutputStream.h.

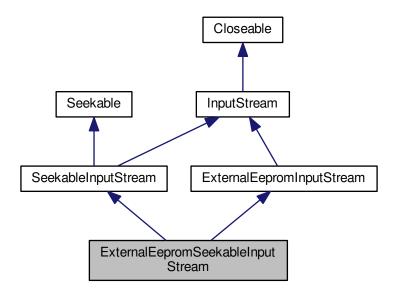
The documentation for this class was generated from the following files:

- ExternalEepromOutputStream.h
- ExternalEepromOutputStream.cpp

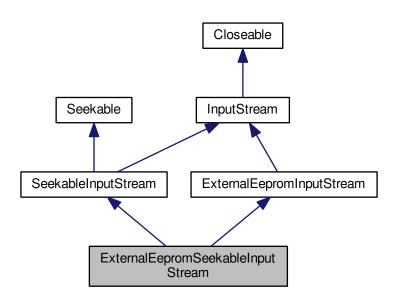
4.13 ExternalEepromSeekableInputStream Class Reference

#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.

 ${\sf External Eeprom Seekable Input Stream}$

A ExternalEepromSeekableInputStream obtains input bytes from a external input stream.

Definition at line 17 of file ExternalEepromSeekableInputStream.h.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 ExternalEepromSeekableInputStream: ExternalEepromSeekableInputStream (ExternalEeprom * externalEeprom)

Public constructor.

Parameters

resource	The external eeprom to be used.
----------	---------------------------------

Definition at line 15 of file ExternalEepromSeekableInputStream.cpp.

4.13.3 Member Function Documentation

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

Seeks this input stream to the position.

Parameters

pos	THe position
Poo	The position

Implements Seekable.

Definition at line 20 of file ExternalEepromSeekableInputStream.cpp.

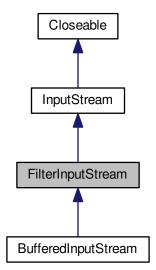
The documentation for this class was generated from the following files:

- ExternalEepromSeekableInputStream.h
- ExternalEepromSeekableInputStream.cpp

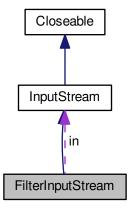
4.14 FilterInputStream Class Reference

#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.

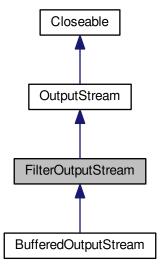
The documentation for this class was generated from the following files:

- FilterInputStream.h
- FilterInputStream.cpp

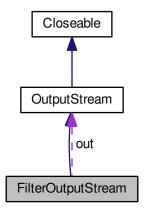
4.15 FilterOutputStream Class Reference

#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 <code>FilterOutputStream</code> itself simply overrides all methods of <code>OutputStream</code> with versions that pass all requests to the underlying output stream. Subclasses of <code>FilterOutputStream</code> 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.

```
\textbf{4.15.3.3} \quad \textbf{void FilterOutputStream::write ( unsigned char \textit{b} )} \quad \texttt{[virtual]}
```

Writes the specified unsigned char to this output stream.

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

Implements the abstract write method of OutputStream.

Parameters

```
b | the unsigned char.
```

Implements OutputStream.

Reimplemented in BufferedOutputStream.

Definition at line 28 of file FilterOutputStream.cpp.

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

Writes len bytes to this output stream.

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

Parameters

Ь	the data to be written.
len	the length

Reimplemented from OutputStream.

Reimplemented in BufferedOutputStream.

Definition at line 32 of file FilterOutputStream.cpp.

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

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

Parameters

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

Reimplemented from OutputStream.

Reimplemented in BufferedOutputStream.

Definition at line 36 of file FilterOutputStream.cpp.

4.15.4 Member Data Documentation

4.15.4.1 OutputStream* FilterOutputStream::out [protected]

The underlying output stream to be filtered.

Definition at line 30 of file FilterOutputStream.h.

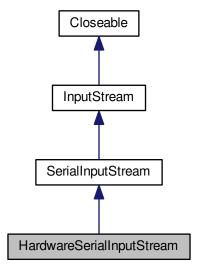
The documentation for this class was generated from the following files:

- · FilterOutputStream.h
- FilterOutputStream.cpp

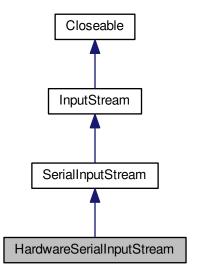
4.16 HardwareSerialInputStream Class Reference

#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.

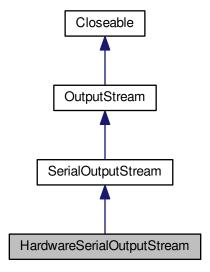
The documentation for this class was generated from the following files:

- · HardwareSerialInputStream.h
- · HardwareSerialInputStream.cpp

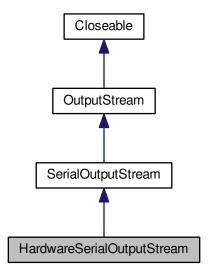
4.17 HardwareSerialOutputStream Class Reference

#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.

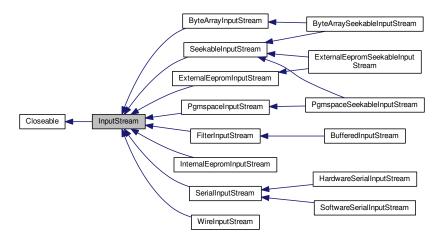
The documentation for this class was generated from the following files:

- HardwareSerialOutputStream.h
- HardwareSerialOutputStream.cpp

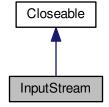
4.18 InputStream Class Reference

#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, ByteArrayInput

Stream, InternalEepromInputStream, 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, InternalEepromInputStream, 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, InternalEepromInputStream, 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, InternalEepromInput

Stream, PgmspaceInputStream, FilterInputStream, SoftwareSerialInputStream, WireInputStream, and Hardware

SerialInputStream.

```
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, ExternalEepromInputStream, FilterInputStream, InternalEepromInput

Stream, 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, InternalEepromInput

Stream, 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.

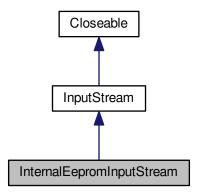
The documentation for this class was generated from the following files:

- · InputStream.h
- InputStream.cpp

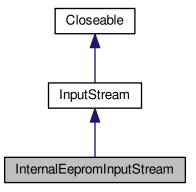
4.19 InternalEepromInputStream Class Reference

#include <InternalEepromInputStream.h>

Inheritance diagram for InternalEepromInputStream:



Collaboration diagram for InternalEepromInputStream:



Public Member Functions

- InternalEepromInputStream ()
- 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

· const int maxAvailableChunk

- · unsigned int pos
- unsigned int markpos
- unsigned int eepromSize

4.19.1 Detailed Description

Arduino IO.

ExternalEepromInputStream

An ExternalEepromInputStream obtains input bytes from a externalEeprom.

Definition at line 15 of file InternalEepromInputStream.h.

```
4.19.2 Constructor & Destructor Documentation
```

4.19.2.1 InternalEepromInputStream::InternalEepromInputStream()

Public constructor.

4.19.3 Member Function Documentation

```
4.19.3.1 virtual int InternalEepromInputStream::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.

```
4.19.3.2 virtual void InternalEepromInputStream::mark( ) [virtual]
```

Marks the current position in this input stream.

Reimplemented from InputStream.

```
4.19.3.3 virtual bool InternalEepromInputStream::markSupported() [virtual]
```

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

Returns

bool

Reimplemented from InputStream.

```
4.19.3.4 virtual int InternalEepromInputStream::read( ) [virtual]
```

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

Returns

int The read unsigned char as an int.

Implements InputStream.

```
4.19.3.5 virtual int Internal EepromInputStream::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.

4.19.3.6 virtual void InternalEepromInputStream::reset() [virtual]

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

Reimplemented from InputStream.

4.19.4 Member Data Documentation

4.19.4.1 unsigned int InternalEepromInputStream::eepromSize [protected]

Definition at line 37 of file InternalEepromInputStream.h.

4.19.4.2 unsigned int InternalEepromInputStream::markpos [protected]

Definition at line 32 of file InternalEepromInputStream.h.

4.19.4.3 const int InternalEepromInputStream::maxAvailableChunk [protected]

When asking for available, this is the max number to return.

Definition at line 22 of file InternalEepromInputStream.h.

4.19.4.4 unsigned int InternalEepromInputStream::pos [protected]

Definition at line 27 of file InternalEepromInputStream.h.

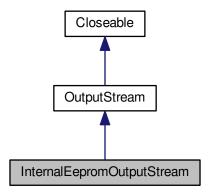
The documentation for this class was generated from the following file:

· InternalEepromInputStream.h

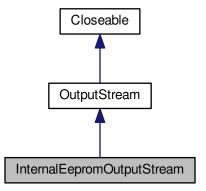
4.20 InternalEepromOutputStream Class Reference

#include <InternalEepromOutputStream.h>

Inheritance diagram for InternalEepromOutputStream:



Collaboration diagram for InternalEepromOutputStream:



Public Member Functions

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

Private Attributes

unsigned int pos

4.20.1 Detailed Description

Arduino IO.

InternalEepromOutputStream

A resource output stream is an output stream for writing data to an internal EEPROM.

Definition at line 16 of file InternalEepromOutputStream.h.

4.20.2 Constructor & Destructor Documentation

4.20.2.1 InternalEepromOutputStream::InternalEepromOutputStream()

Public constructor.

Definition at line 15 of file InternalEepromOutputStream.cpp.

4.20.3 Member Function Documentation

4.20.3.1 void InternalEepromOutputStream::write (unsigned char b) [virtual]

Writes the specified unsigned char to this output stream.

Parameters

b	
---	--

Implements OutputStream.

Definition at line 19 of file InternalEepromOutputStream.cpp.

4.20.3.2 void InternalEepromOutputStream::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 23 of file InternalEepromOutputStream.cpp.

4.20.4 Member Data Documentation

4.20.4.1 unsigned int InternalEepromOutputStream::pos [private]

Current eeprom position.

Definition at line 21 of file InternalEepromOutputStream.h.

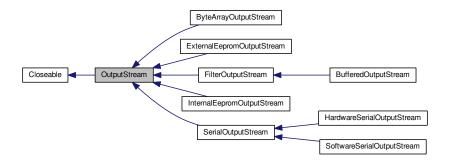
The documentation for this class was generated from the following files:

- InternalEepromOutputStream.h
- InternalEepromOutputStream.cpp

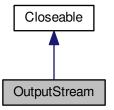
4.21 OutputStream Class Reference

#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)

4.21.1 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.

4.21.2 Member Function Documentation

```
4.21.2.1 void OutputStream::close() [virtual]
```

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

Implements Closeable.

Reimplemented in BufferedOutputStream, and FilterOutputStream.

Definition at line 34 of file OutputStream.cpp.

```
4.21.2.2 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.

```
4.21.2.3 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, InternalEepromOutputStream, SoftwareSerialOutputStream, and HardwareSerialOutputStream.

```
4.21.2.4 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.

```
4.21.2.5 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, ExternalEepromOutputStream, and Internal EepromOutputStream.

Definition at line 22 of file OutputStream.cpp.

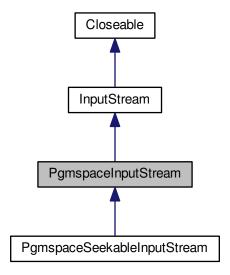
The documentation for this class was generated from the following files:

- · OutputStream.h
- OutputStream.cpp

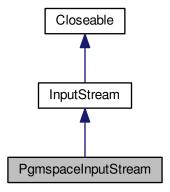
4.22 PgmspaceInputStream Class Reference

#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

4.22.1 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.

4.22.2 Constructor & Destructor Documentation

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

Definition at line 15 of file PgmspaceInputStream.cpp.

4.22.3 Member Function Documentation

```
4.22.3.1 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.

4.22.3.2 void PgmspaceInputStream::mark() [virtual]

Marks the current position in this input stream.

Reimplemented from InputStream.

Definition at line 27 of file PgmspaceInputStream.cpp.

4.22.3.3 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.

```
4.22.3.4 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.

```
4.22.3.5 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.

4.22.4 Member Data Documentation

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

Definition at line 22 of file PgmspaceInputStream.h.

4.22.4.2 unsigned int PgmspaceInputStream::count [protected]

Definition at line 27 of file PgmspaceInputStream.h.

4.22.4.3 unsigned int PgmspaceInputStream::markpos [protected]

Definition at line 37 of file PgmspaceInputStream.h.

4.22.4.4 unsigned int PgmspaceInputStream::pos [protected]

Definition at line 32 of file PgmspaceInputStream.h.

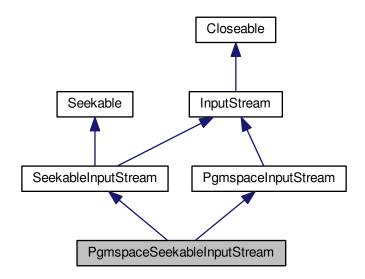
The documentation for this class was generated from the following files:

- PgmspaceInputStream.h
- PgmspaceInputStream.cpp

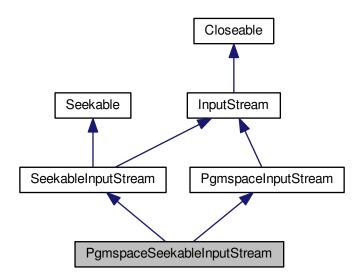
4.23 PgmspaceSeekableInputStream Class Reference

#include <PgmspaceSeekableInputStream.h>

Inheritance diagram for PgmspaceSeekableInputStream:



Collaboration diagram for PgmspaceSeekableInputStream:



Public Member Functions

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

Additional Inherited Members

4.23.1 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 17 of file PgmspaceSeekableInputStream.h.

4.23.2 Constructor & Destructor Documentation

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

Definition at line 15 of file PgmspaceSeekableInputStream.cpp.

4.23.2.2 virtual PgmspaceSeekableInputStream::~PgmspaceSeekableInputStream() [inline], [virtual]

Definition at line 23 of file PgmspaceSeekableInputStream.h.

4.23.3 Member Function Documentation

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

Implements Seekable.

Definition at line 18 of file PgmspaceSeekableInputStream.cpp.

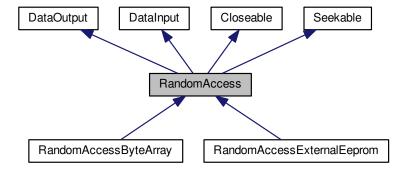
The documentation for this class was generated from the following files:

- PgmspaceSeekableInputStream.h
- PgmspaceSeekableInputStream.cpp

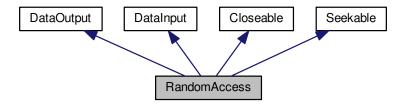
4.24 RandomAccess Class Reference

#include <RandomAccess.h>

Inheritance diagram for RandomAccess:



Collaboration diagram for RandomAccess:



Additional Inherited Members

4.24.1 Detailed Description

Araduino IO.

RandomAccess

Interface derived from DataInput, DataOutput, Closeable and Seekable.

Definition at line 17 of file RandomAccess.h.

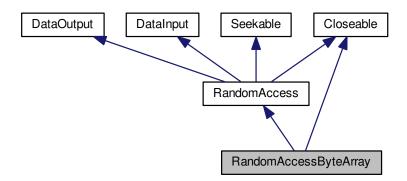
The documentation for this class was generated from the following file:

· RandomAccess.h

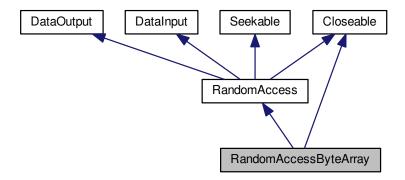
4.25 RandomAccessByteArray Class Reference

#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

4.25.1 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.

4.25.2 Constructor & Destructor Documentation

4.25.2.1 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.

4.25.3 Member Function Documentation

4.25.3.1 void RandomAccessByteArray::close() [virtual]

Closing a unsigned char array has no effect.

Implements Closeable.

Definition at line 29 of file RandomAccessByteArray.cpp.

4.25.3.2 unsigned int RandomAccessByteArray::length ()

Returns the length of the stream.

Returns

The length.

Definition at line 21 of file RandomAccessByteArray.cpp.

4.25.3.3 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.
4.25.3.5 char RandomAccessByteArray::readChar() [virtual]
Reads a char from the stream.
Returns
     char
Implements DataInput.
Definition at line 102 of file RandomAccessByteArray.cpp.
4.25.3.6 double RandomAccessByteArray::readDouble( ) [virtual]
Reads a double from the stream.
Returns
     double
Implements DataInput.
Definition at line 146 of file RandomAccessByteArray.cpp.
4.25.3.7 float RandomAccessByteArray::readFloat( ) [virtual]
Reads a float from the stream.
Returns
     float
Implements DataInput.
Definition at line 142 of file RandomAccessByteArray.cpp.
4.25.3.8 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.
```

4.25.3.9 int RandomAccessByteArray::readInt() [virtual]

Reads an int from the stream.

```
Returns
     int
Implements DataInput.
Definition at line 110 of file RandomAccessByteArray.cpp.
4.25.3.10 long RandomAccessByteArray::readLong() [virtual]
Reads a long from the stream.
Returns
     long
Implements DataInput.
Definition at line 126 of file RandomAccessByteArray.cpp.
4.25.3.11 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.
4.25.3.12 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.
4.25.3.13 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.
4.25.3.14 word RandomAccessByteArray::readWord() [virtual]
Reads a word from the stream.
Returns
     word
Implements DataInput.
Definition at line 122 of file RandomAccessByteArray.cpp.
```

Seeks the stream at the position.

 $\textbf{4.25.3.15} \quad \textbf{void RandomAccessByteArray::seek (unsigned int } \textit{pos} \text{)} \quad \texttt{[virtual]}$

Parameters

pos The position.

Implements Seekable.

Definition at line 25 of file RandomAccessByteArray.cpp.

4.25.3.16 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.

4.25.3.17 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.

4.25.3.18 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.

4.25.3.19 void RandomAccessByteArray::writeBoolean (bool v) [virtual]

Writes a bool into the stream.

Parameters

V	The bool to be written.

Implements DataOutput.

Definition at line 50 of file RandomAccessByteArray.cpp.

4.25.3.20 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.

4.25.3.21 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.

4.25.3.22 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.

4.25.3.23 void RandomAccessByteArray::writeDouble (double v) [virtual]

Writes a double into the stream.

Parameters

ν The double to be written.

Implements DataOutput.

Definition at line 90 of file RandomAccessByteArray.cpp.

4.25.3.24 void RandomAccessByteArray::writeFloat (float v) [virtual]

Writes a float into the stream.

Parameters

 ν The float to be written.

Implements DataOutput.

Definition at line 86 of file RandomAccessByteArray.cpp.

4.25.3.25 void RandomAccessByteArray::writeInt(int v) [virtual]

Writes an int into the stream.

Parameters

ν The int to be written.

Implements DataOutput.

Definition at line 62 of file RandomAccessByteArray.cpp.

 $\textbf{4.25.3.26} \quad \text{void RandomAccessByteArray::writeLong(long ν)} \quad [\texttt{virtual}]$

Writes a long into the stream.

Parameters

v The long to be written.

Implements DataOutput.

Definition at line 75 of file RandomAccessByteArray.cpp.

4.25.3.27 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.

 $\textbf{4.25.3.28} \quad \textbf{void RandomAccessByteArray::writeUnsignedInt (unsigned int ν)} \quad \texttt{[virtual]}$

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.

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

Writes a unsigned long into the stream.

Parameters

v The unsigned long to be written.

Implements DataOutput.

Definition at line 82 of file RandomAccessByteArray.cpp.

4.25.3.30 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.

4.25.4 Member Data Documentation

4.25.4.1 unsigned char* RandomAccessByteArray::buf [private]

Buffer used to work.

Definition at line 21 of file RandomAccessByteArray.h.

4.25.4.2 unsigned int RandomAccessByteArray::count [private]

Buffer size.

Definition at line 26 of file RandomAccessByteArray.h.

4.25.4.3 unsigned int RandomAccessByteArray::pos [private]

Current position.

Definition at line 31 of file RandomAccessByteArray.h.

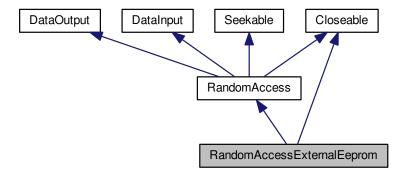
The documentation for this class was generated from the following files:

- · RandomAccessByteArray.h
- RandomAccessByteArray.cpp

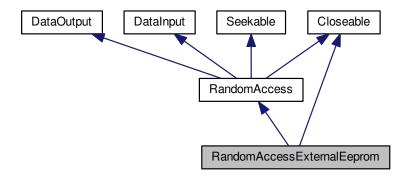
4.26 RandomAccessExternalEeprom Class Reference

#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

4.26.1 Detailed Description

Araduino IO.

Random Access External Eeprom

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.

4.26.2 Constructor & Destructor Documentation

4.26.2.1 RandomAccessExternalEeprom::RandomAccessExternalEeprom (ExternalEeprom * externalEeprom)

Public constructor.

Parameters

char

Implements DataInput.

Definition at line 104 of file RandomAccessExternalEeprom.cpp.

```
externalEeprom
                     The external eeprom instance to be used.
Definition at line 17 of file RandomAccessExternalEeprom.cpp.
4.26.3 Member Function Documentation
4.26.3.1 void RandomAccessExternalEeprom::close() [virtual]
Closing a external eeprom has no effect.
Implements Closeable.
Definition at line 31 of file RandomAccessExternalEeprom.cpp.
4.26.3.2 unsigned int RandomAccessExternalEeprom::length ( )
Returns the length of the stream.
Returns
     The length.
Definition at line 23 of file RandomAccessExternalEeprom.cpp.
4.26.3.3 bool RandomAccessExternalEeprom::readBoolean() [virtual]
Reads a bool from the stream.
Returns
     bool
Implements DataInput.
Definition at line 100 of file RandomAccessExternalEeprom.cpp.
4.26.3.4 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.
4.26.3.5 char RandomAccessExternalEeprom::readChar() [virtual]
Reads a char from the stream.
Returns
```

Generated on Tue Dec 29 2015 15:18:31 for Arduino IO Library Driver by Doxygen

```
4.26.3.6 double RandomAccessExternalEeprom::readDouble( ) [virtual]
Reads a double from the stream.
Returns
     double
Implements DataInput.
Definition at line 148 of file RandomAccessExternalEeprom.cpp.
4.26.3.7 float RandomAccessExternalEeprom::readFloat() [virtual]
Reads a float from the stream.
Returns
     float
Implements DataInput.
Definition at line 144 of file RandomAccessExternalEeprom.cpp.
4.26.3.8 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.
4.26.3.9 int RandomAccessExternalEeprom::readInt() [virtual]
Reads an int from the stream.
Returns
     int
Implements DataInput.
Definition at line 112 of file RandomAccessExternalEeprom.cpp.
4.26.3.10 long RandomAccessExternalEeprom::readLong() [virtual]
Reads a long from the stream.
Returns
     long
Implements DataInput.
Definition at line 128 of file RandomAccessExternalEeprom.cpp.
4.26.3.11 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.
4.26.3.12 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.
4.26.3.13 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.
4.26.3.14 word RandomAccessExternalEeprom::readWord() [virtual]
Reads a word from the stream.
Returns
     word
Implements DataInput.
Definition at line 124 of file RandomAccessExternalEeprom.cpp.
4.26.3.15 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.
4.26.3.16 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.

4.26.3.17 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.

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

Writes a unsigned char into the stream.

Parameters

b	The unsigned char to be written.
	9

Implements DataOutput.

Definition at line 38 of file RandomAccessExternalEeprom.cpp.

4.26.3.19 void RandomAccessExternalEeprom::writeBoolean (bool v) [virtual]

Writes a bool into the stream.

Parameters

v The bool to be written.

Implements DataOutput.

Definition at line 52 of file RandomAccessExternalEeprom.cpp.

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

Writes a unsigned char into the stream.

Parameters

```
b The unsigned char to be written.
```

Implements DataOutput.

Definition at line 42 of file RandomAccessExternalEeprom.cpp.

4.26.3.21 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.

4.26.3.22 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.

4.26.3.23 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.

4.26.3.24 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.

4.26.3.25 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.

 $\textbf{4.26.3.26} \quad \textbf{void} \; \textbf{RandomAccessExternalEeprom::writeLong(long v)} \quad [\texttt{virtual}]$

Writes a long into the stream.

Parameters

V	The long to be written.

Implements DataOutput.

Definition at line 77 of file RandomAccessExternalEeprom.cpp.

Writes an unsigned char into the stream.

 $\textbf{4.26.3.27} \quad \textbf{void RandomAccessExternalEeprom::writeUnsignedChar(unsigned char c)} \quad [\texttt{virtual}]$

Parameters

c The unsigned char to be written.

Implements DataOutput.

Definition at line 60 of file RandomAccessExternalEeprom.cpp.

4.26.3.28 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.

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

Writes a unsigned long into the stream.

Parameters

v The unsigned long to be written.

Implements DataOutput.

Definition at line 84 of file RandomAccessExternalEeprom.cpp.

4.26.3.30 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.

4.26.4 Member Data Documentation

4.26.4.1 ExternalEeprom* RandomAccessExternalEeprom::externalEeprom [private]

The external eeprom to be used.

Definition at line 24 of file RandomAccessExternalEeprom.h.

4.26.4.2 unsigned int RandomAccessExternalEeprom::pos [private]

Current position.

Definition at line 29 of file RandomAccessExternalEeprom.h.

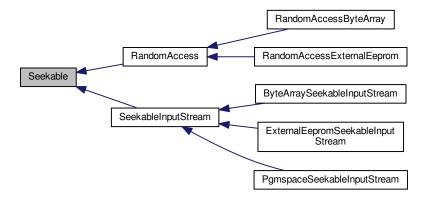
The documentation for this class was generated from the following files:

- · RandomAccessExternalEeprom.h
- RandomAccessExternalEeprom.cpp

4.27 Seekable Class Reference

#include <Seekable.h>

Inheritance diagram for Seekable:



Public Member Functions

virtual void seek (unsigned int pos)=0

4.27.1 Detailed Description

Arduino IO.

Seekable

Definition at line 10 of file Seekable.h.

4.27.2 Member Function Documentation

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

Implemented in RandomAccessByteArray, RandomAccessExternalEeprom, ExternalEepromSeekableInput

Stream, PgmspaceSeekableInputStream, and ByteArraySeekableInputStream.

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

· Seekable.h

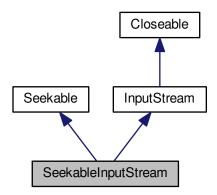
4.28 SeekableInputStream Class Reference

#include <SeekableInputStream.h>

Inheritance diagram for SeekableInputStream:



Collaboration diagram for SeekableInputStream:



Additional Inherited Members

4.28.1 Detailed Description

Arduino IO.

SeekableInputStream

Definition at line 13 of file SeekableInputStream.h.

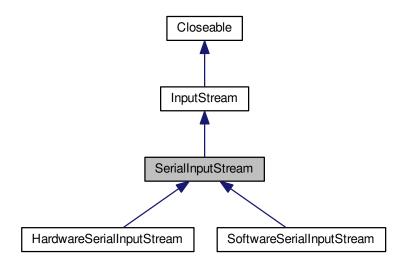
The documentation for this class was generated from the following file:

• SeekableInputStream.h

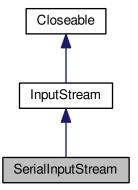
4.29 SerialInputStream Class Reference

#include <SerialInputStream.h>

Inheritance diagram for SerialInputStream:



Collaboration diagram for SerialInputStream:



Additional Inherited Members

4.29.1 Detailed Description

Arduino IO.

SerialInputStream

A SerialInputStream obtains input bytes from a serial port.

Definition at line 14 of file SerialInputStream.h.

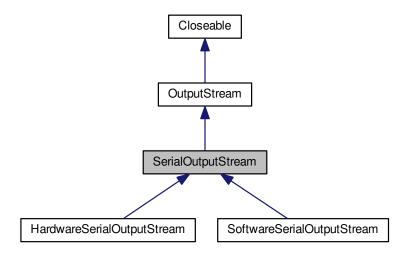
The documentation for this class was generated from the following file:

· SerialInputStream.h

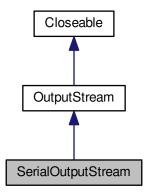
4.30 SerialOutputStream Class Reference

#include <SerialOutputStream.h>

Inheritance diagram for SerialOutputStream:



 $Collaboration\ diagram\ for\ Serial Output Stream:$



Additional Inherited Members

4.30.1 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.

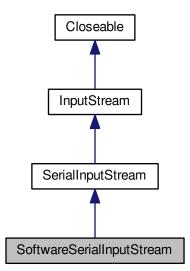
The documentation for this class was generated from the following file:

· SerialOutputStream.h

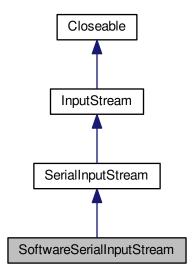
4.31 SoftwareSerialInputStream Class Reference

#include <SoftwareSerialInputStream.h>

Inheritance diagram for SoftwareSerialInputStream:



Collaboration diagram for SoftwareSerialInputStream:



Public Member Functions

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

Protected Attributes

• SoftwareSerial * softwareSerial

4.31.1 Detailed Description

Arduino IO.

SoftwareSerialInputStream

A SoftwareSerialInputStream obtains input bytes from a serial port.

Definition at line 17 of file SoftwareSerialInputStream.h.

- 4.31.2 Constructor & Destructor Documentation
- 4.31.2.1 SoftwareSerialInputStream::SoftwareSerialInputStream (SoftwareSerial * softwareSerial, unsigned int boudRate)

Public constructor.

Parameters

serial	
boudRate	

Definition at line 14 of file SoftwareSerialInputStream.cpp.

4.31.3 Member Function Documentation

```
4.31.3.1 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.

```
4.31.3.2 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.

4.31.4 Member Data Documentation

 $\textbf{4.31.4.1} \quad \textbf{SoftwareSerial} * \textbf{SoftwareSerialInputStream} \\ :: \textbf{softwareSerial} \quad \texttt{[protected]}$

The software serial where the data will be read.

Definition at line 23 of file SoftwareSerialInputStream.h.

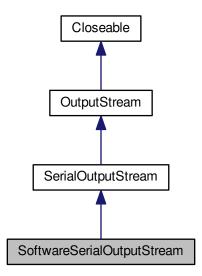
The documentation for this class was generated from the following files:

- · SoftwareSerialInputStream.h
- · SoftwareSerialInputStream.cpp

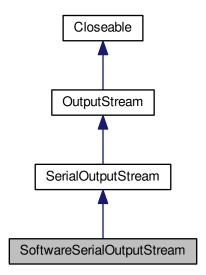
4.32 SoftwareSerialOutputStream Class Reference

#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

4.32.1 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.

4.32.2 Constructor & Destructor Documentation

4.32.2.1 SoftwareSerialOutputStream::SoftwareSerialOutputStream (SoftwareSerial * serial, unsigned int boudRate)

Definition at line 14 of file SoftwareSerialOutputStream.cpp.

4.32.3 Member Function Documentation

4.32.3.1 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.

4.32.4 Member Data Documentation

4.32.4.1 SoftwareSerial* SoftwareSerialOutputStream::softwareSerial [protected]

Definition at line 23 of file SoftwareSerialOutputStream.h.

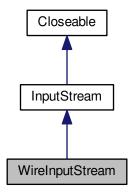
The documentation for this class was generated from the following files:

- · SoftwareSerialOutputStream.h
- · SoftwareSerialOutputStream.cpp

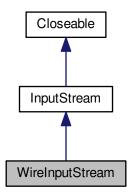
4.33 WireInputStream Class Reference

#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

4.33.1 Detailed Description

Arduino IO.

WireInputStream

A WireInputStream obtains input bytes from the wire bus.

Definition at line 16 of file WireInputStream.h.

4.33.2 Constructor & Destructor Documentation

4.33.2.1 WireInputStream::WireInputStream (unsigned char addredd)

Public constructor.

Parameters

```
address
```

Definition at line 14 of file WireInputStream.cpp.

4.33.3 Member Function Documentation

```
4.33.3.1 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.

```
4.33.3.2 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.

```
4.33.3.3 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.

4.33.4 Member Data Documentation

4.33.4.1 unsigned char WireInputStream::address [protected]

The wire device address.

Definition at line 22 of file WireInputStream.h.

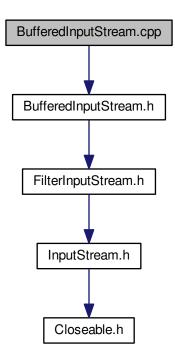
The documentation for this class was generated from the following files:

- · WireInputStream.h
- WireInputStream.cpp

5 File Documentation

5.1 BufferedInputStream.cpp File Reference

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



5.2 BufferedInputStream.cpp

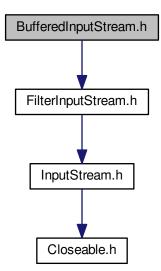
```
00001
00024 #include "BufferedInputStream.h"
00025
00026 BufferedInputStream::BufferedInputStream(
      InputStream* in, unsigned char* buf, int size)
00027
              : FilterInputStream(in), size(size), buf(buf), count(0), pos(0), marked(false),
      markpos(0) {
00028 }
00029
00030 int BufferedInputStream::available() {
00031
         return in->available() + (count - pos);
00032 }
00033
00034 void BufferedInputStream::close() {
00035
         in->close();
00036 }
```

```
00037
00038 void BufferedInputStream::reset() {
        if (marked) {
00039
00040
            pos = markpos;
00041
00042 }
00043
00044 int BufferedInputStream::read(unsigned char* b, int len) {
00045
         return read(b, 0, len);
00046 }
00047
00048 int BufferedInputStream::read(unsigned char* b, int off, int len) {
00049
          int cnt, available;
          available = count - pos;
00050
00051
00052
          * The needed data are already in the buffer?
00053
00054
          if (available >= len) {
00055
00056
              for (int i = 0; i < len; i++) {</pre>
00057
                  b[off + i] = buf[pos + i];
00058
00059
              pos += len;
00060
              return len;
00061
          }
00062
00063
          \star The buffer data is not enough, but is necessary.
00064
00065
          for (int i = 0; i < available; i++) {</pre>
00066
00067
              b[off + i] = buf[pos + i];
00068
00069
          marked = false;
00070
          pos = 0;
          count = 0;
00071
00072
00073
00074
          \star Reads the rest from the stream.
00075
00076
          cnt = in->read(b, off + available, len - available);
00077
00078
00079
          * Tests if we had enough data.
           */
08000
00081
          if (cnt < 0) {</pre>
00082
              return available;
00083
          } else if (cnt < (len - available)) {</pre>
00084
             return available + cnt;
00085
          } else {
00086
             fill(0);
00087
00088
          return len;
00089 }
00090
00091 int BufferedInputStream::read() {
00092
00093
00094
           * Tests if the buffer is completely used.
00095
          if (pos >= count) {
00096
              marked = false;
00097
00098
              fill(0);
00099
              if (count == 0) {
00100
                  return -1;
00101
              }
00102
              pos = 0;
00103
00104
          return (int) buf[pos++];
00105 }
00106
00107 void BufferedInputStream::realineBufferContent() {
00108
         int n;
00109
          if (pos > 0) {
              n = count - pos;
for (int i = 0; i < n; i++) {
  buf[i] = buf[pos + i];</pre>
00110
00111
00112
00113
00114
              count -= pos;
00115
              pos = 0;
          }
00116
00117 }
00118
00119 void BufferedInputStream::fill(int startPos) {
00120
         int n, needed;
00121
          needed = size - startPos;
          if (needed <= 0) {
00122
00123
              return:
```

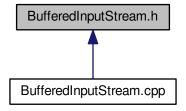
```
00124
00125
          n = in->read(buf, startPos, needed);
00126
          if (n > 0) {
             count = startPos + n;
00127
00128
00129 }
00130
00131 void BufferedInputStream::mark() {
00132
        realineBufferContent();
00133
          fill(count);
         markpos = 0;
marked = true;
00134
00135
00136 }
00137
00138 bool BufferedInputStream::markSupported() {
00139
         return true;
00140 }
00141
00142 unsigned int BufferedInputStream::skip(unsigned int n) {
       unsigned int buffered, skiped;
00143
00144
          buffered = count - pos;
00145
          if (buffered >= n) {
00146
           pos += n;
              return n;
00147
00148
00149
         pos = 0;
00150
          count = 0;
          marked = false;
skiped = buffered + in->skip(n - buffered);
00151
00152
          return skiped;
00153
00154 }
```

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

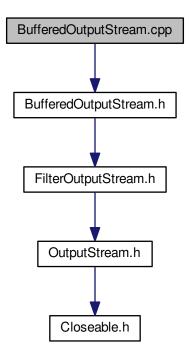
5.4 BufferedInputStream.h

```
00001
00024 #ifndef __ARDUINO_IO_BUFFERED_INPUT_STREAM_H_
00025 #define __ARDUINO_IO_BUFFERED_INPUT_STREAM_H__
00026
00027 #include <FilterInputStream.h>
00028
00029 class BufferedInputStream: public FilterInputStream {
00030
00034
          unsigned int size;
00035
00036 protected:
00037
         unsigned char* buf;
00042
00052
         int count;
00053
         int pos;
00067
00068
00072
         bool marked;
00073
00098
          int markpos;
00099
00100 public:
00101
00109
          BufferedInputStream(InputStream* in, unsigned char* buf, int size);
00110
00114
          virtual ~BufferedInputStream() {
00115
00116
00122
          virtual int available();
00123
00128
          virtual void close();
00129
00133
          virtual void mark();
00134
00138
         virtual bool markSupported();
00139
00143
          virtual int read();
00144
00153
          virtual int read(unsigned char* b, int len);
00154
00159
         virtual int read(unsigned char* b, int off, int len);
00160
00165
          virtual void reset();
00166
00170
          virtual unsigned int skip(unsigned int n);
00171
00172 private:
00173
00177
          void realineBufferContent();
00178
```

```
00184     void fill(int startPos);
00185 };
00186
00187 #endif /* __ARDUINO_IO_BUFFERED_INPUT_STREAM_H__ */
```

5.5 BufferedOutputStream.cpp File Reference

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



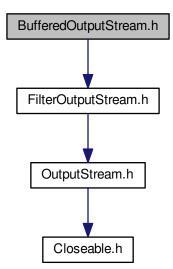
5.6 BufferedOutputStream.cpp

```
00001
00012 #include "BufferedOutputStream.h"
00013
00014 BufferedOutputStream::BufferedOutputStream(
     OutputStream* out, unsigned char* buf, int size)
00015
             : FilterOutputStream(out), buf(buf), size(size), count(0) {
00016 }
00017
00018 void BufferedOutputStream::write(unsigned char b) {
00019
        if (count >= size) {
              flushBuffer();
00020
00021
00022
00023 }
00024
00025 void BufferedOutputStream::write(unsigned char* b, int len) {
00026
         write(b, 0, len);
00027 }
00029 void BufferedOutputStream::write(unsigned char* b, int off, int len) {
00030
00031
00032
          * If the request length exceeds the size of the output buffer,
00033
          * flush the output buffer and then write the data directly.
00034
           * In this way buffered streams will cascade harmlessly.
```

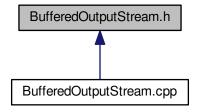
```
00036
          if (len >= size) {
00037
               flushBuffer();
00038
               out->write(b, off, len);
00039
              return;
00040
00041
         if (len > size - count) {
00042
              flushBuffer();
00043
          for (int i = 0; i < len; i++) {
   buf[count + i] = b[off + i];
}</pre>
00044
00045
00046
00047
          count += len;
00048 }
00049
00050 void BufferedOutputStream::flush() {
00051 flushBuffer();
          flushBuffer();
00052
          out->flush();
00053 }
00054
00055 void BufferedOutputStream::close() {
00056 flush();
00057
          out->close();
00058 }
00059
00060 void BufferedOutputStream::flushBuffer() {
        if (count > 0) {
00061
00062
              out->write(buf, 0, count);
00063
               count = 0;
00064
          }
00065 }
```

5.7 BufferedOutputStream.h File Reference

#include <FilterOutputStream.h>
Include dependency graph for BufferedOutputStream.h:



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



Classes

· class BufferedOutputStream

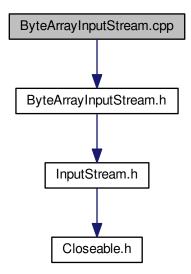
5.8 BufferedOutputStream.h

```
00001
00012 #ifndef __ARDUINO_IO_BUFFERED_OUTPUT_STREAM_H_
00013 #define __ARDUINO_IO_BUFFERED_OUTPUT_STREAM_H_ 1
00014
00015 #include <FilterOutputStream.h>
00016
00017 class BufferedOutputStream: public FilterOutputStream {
00018
00019 protected:
00020
00024
          unsigned char* buf;
00025
00029
          int size;
00030
          int count;
00038
00039 public:
00040
00049
          {\tt BufferedOutputStream (OutputStream \star out, unsigned char \star buf, int size}
00050
00054
          virtual ~BufferedOutputStream();
00055
00062
          void write(unsigned char b);
00063
00072
          virtual void write (unsigned char* b, int len);
00073
00089
          virtual void write(unsigned char* b, int off, int len);
00090
00095
          virtual void flush();
00096
00097
          virtual void close();
00098
00099 private:
00100
00104
          void flushBuffer();
00105 };
00106
00107 #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.

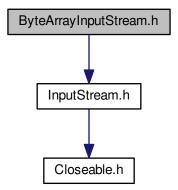
5.10 ByteArrayInputStream.cpp

```
00010 #ifndef __ARDUINO_IO_BYTE_ARRAY_INPUT_STREAM_CPP_
00011 #define __ARDUINO_IO_BYTE_ARRAY_INPUT_STREAM_CPP__
00012
00013 #include "ByteArrayInputStream.h"
00014
00015 ByteArrayInputStream::ByteArrayInputStream(unsigned char* buf,
00016
       unsigned int count) :
00017
             buf(buf), count(count) {
         markpos = 0;
pos = 0;
00018
00019
00020 }
00021
00022 int ByteArrayInputStream::available() {
       if ((count - pos) > 0) {
    return 1;
00023
00024
00025
00026
          return 0;
00027 }
00028
```

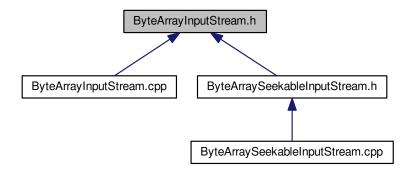
5.11 ByteArrayInputStream.h File Reference

#include <InputStream.h>

Include dependency graph for ByteArrayInputStream.h:



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



Classes

· class ByteArrayInputStream

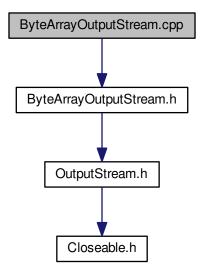
5.12 ByteArrayInputStream.h

```
00001
00010 #ifndef __ARDUINO_IO_BYTE_ARRAY_INPUT_STREAM_H_
00011 #define __ARDUINO_IO_BYTE_ARRAY_INPUT_STREAM_H_ 1
00013 #include <InputStream.h>
00014
00015 class ByteArrayInputStream : public virtual InputStream {
00016
00017 protected:
00018
00019
00020
           \star The buffer where data is stored.
00021
00022
          unsigned char* buf;
00023
00024
          \star The number of valid bytes in the buffer.
00025
00026
00027
          unsigned int count;
00028
00029
00030
           * Current position
00031
00032
00033
          unsigned int pos;
00034
00035
          * The currently marked position in the stream.
00036
00037
          unsigned int markpos;
00038
00039 public:
00040
00041
          ByteArrayInputStream(unsigned char* buf, unsigned int count);
00042
00046
          virtual ~ByteArrayInputStream() {
00047
00048
          virtual int available();
00061
00062
00066
          virtual void mark();
00067
00073
          virtual bool markSupported();
00074
00078
          using InputStream::read;
00079
00085
          virtual int read();
00086
```

```
00091     virtual void reset();
00092 };
00093
00094 #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

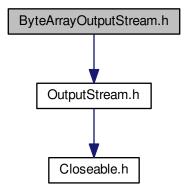
```
00001
00010 #ifndef __ARDUINO_IO_BYTE_ARRAY_OUTPUT_STREAM_CPP__
00011 #define __ARDUINO_IO_BYTE_ARRAY_OUTPUT_STREAM_CPP__ 1
00012
00013 #include "ByteArrayOutputStream.h"
00014
00015 ByteArrayOutputStream::ByteArrayOutputStream(unsigned char* buf
```

```
unsigned int count)
               : buf(buf), count(count) {
00018
          pos = 0;
00019 }
00020 00021 void ByteArrayOutputStream::reset() {
          pos = 0;
00023 }
00024
00025 unsigned int ByteArrayOutputStream::size() {
00026    return count;
           return count;
00027 }
00028
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__ */
```

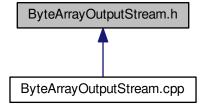
5.15 ByteArrayOutputStream.h File Reference

#include <OutputStream.h>

Include dependency graph for ByteArrayOutputStream.h:



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



Classes

• class ByteArrayOutputStream

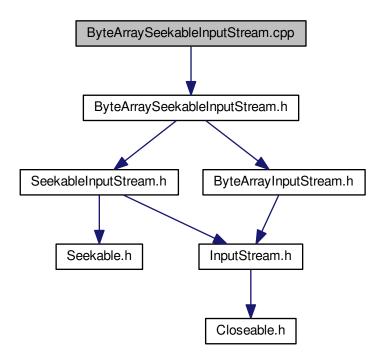
5.16 ByteArrayOutputStream.h

```
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
00017 protected:
00018
00019
          * The buffer where data is stored.
*/
00020
00021
00022
          unsigned char* buf;
00023
00024
          ^{\prime} * The number of valid bytes in the buffer.  
*/
00025
00026
          unsigned int count;
00028
          /*
* Current position
*/
00029
00030
00031
00032
          unsigned int pos;
00034 public:
00035
00042
          ByteArrayOutputStream(unsigned char* buf, unsigned int count);
00043
00047
          virtual ~ByteArrayOutputStream() {
00048
00049
00053
          void reset();
00054
00060
          unsigned int size();
00061
00067
          unsigned char* toByteArray();
00068
00072
          using OutputStream::write;
00073
00079
          virtual void write(unsigned char b);
00080 };
00081
00082 #endif /* __ARDUINO_IO_BYTE_ARRAY_OUTPUT_STREAM_H__ */
```

5.17 ByteArraySeekableInputStream.cpp File Reference

#include "ByteArraySeekableInputStream.h"

Include dependency graph for ByteArraySeekableInputStream.cpp:

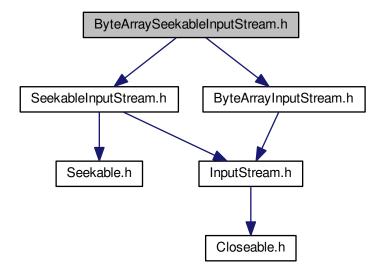


5.18 ByteArraySeekableInputStream.cpp

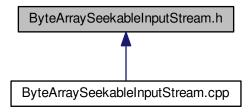
5.19 ByteArraySeekableInputStream.h File Reference

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

Include dependency graph for ByteArraySeekableInputStream.h:



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



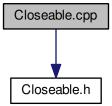
Classes

• class ByteArraySeekableInputStream

5.20 ByteArraySeekableInputStream.h

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

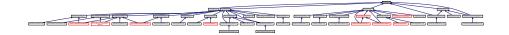
00012 #include "Closeable.h"

00013

00014 #endif /* __ARDUINO_IO_CLOSEABLE_CPP__ */
```

5.23 Closeable.h File Reference

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



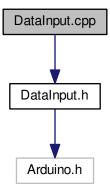
Classes

· class Closeable

5.24 Closeable.h

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

5.26 DataInput.cpp 127

```
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.

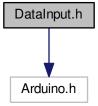
5.26 DataInput.cpp

```
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
```

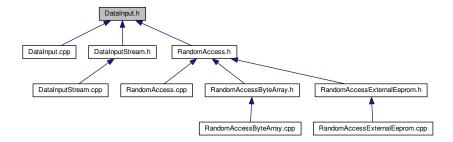
5.27 DataInput.h File Reference

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

Include dependency graph for DataInput.h:



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



Classes

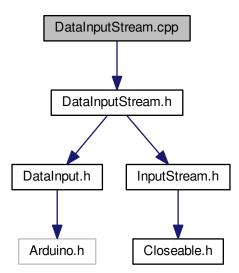
class DataInput

5.28 DataInput.h

```
00001
00011 #ifndef _ARDUINO_IO_DATA_INPUT_H_
00012 #define _ARDUINO_IO_DATA_INPUT_H_ 1
00014 #include <Arduino.h>
00015
00016 class DataInput {
00017 public:
00018
          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
08000
          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.

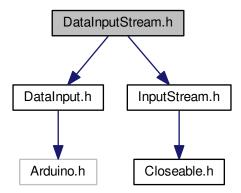
5.30 DataInputStream.cpp

```
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();
00025
00026 char DataInputStream::readChar() {
00027
         return (char) inputStream->read();
00028 }
00029
00030 unsigned char DataInputStream::readUnsignedChar() {
00031
        return (unsigned char) inputStream->read();
00032 }
00033
00034 int DataInputStream::readInt() {
00035
         int v = 0;
00036
         v = inputStream->read();
         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;
00058
         v |= (inputStream->read() & 0xff);
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 }
```

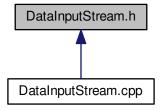
```
00069
00070 double DataInputStream::readDouble() {
00071 return (double) readLong();
          return (double) readLong();
00072 }
00073 00074 void DataInputStream::readFully(unsigned char* b, int len) {
         for (int i = 0; i < len; i++) {
00076
              b[i] = inputStream->read();
00077
00078 }
00079
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

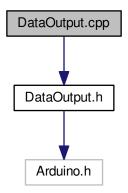
5.32 DataInputStream.h

```
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
00020
          InputStream* inputStream;
00021
00022 public:
00023
00029
          DataInputStream(InputStream* inputStream);
00030
00036
          virtual unsigned char readByte();
00037
00043
          virtual bool readBoolean();
00044
00050
          virtual char readChar();
00051
00057
          virtual unsigned char readUnsignedChar();
00058
00064
          virtual int readInt();
00065
00071
          virtual unsigned int readUnsignedInt();
00072
00078
          virtual 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
00122
          virtual unsigned int skipBytes(unsigned int n);
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

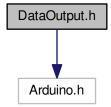
```
00001
00010 #ifndef __ARDUINO_IO_DATA_OUTPUT_CPP__
00011 #define __ARDUINO_IO_DATA_OUTPUT_CPP__ 1
00012
00013 #include "DataOutput.h"
00014
00015 #endif /* __ARDUINO_IO_DATA_OUTPUT_CPP__ */
00016
```

5.35 DataOutput.h File Reference

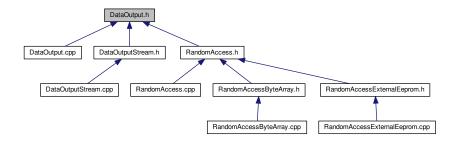
#include <Arduino.h>

5.36 DataOutput.h

Include dependency graph for DataOutput.h:



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



Classes

· class DataOutput

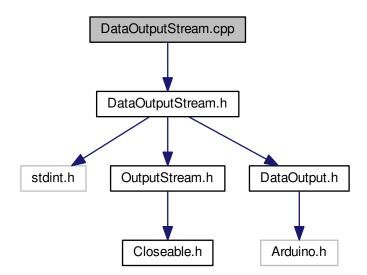
5.36 DataOutput.h

```
00010 #ifndef __ARDUINO_IO_DATA_OUTPUT_H__
00011 #define __ARDUINO_IO_DATA_OUTPUT_H__
00012
00013 #include <Arduino.h>
00014
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
```

```
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

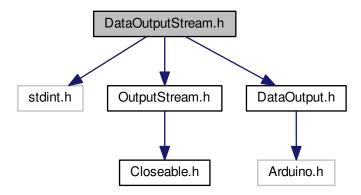
00001

```
00009 #ifndef __ARDUINO_IO_DATA_OUTPUT_STREAM_CPP__ 00010 #define __ARDUINO_IO_DATA_OUTPUT_STREAM_CPP__ 1
00011
00012 #include "DataOutputStream.h"
00013
00014 DataOutputStream::DataOutputStream(
      OutputStream* outputStream) :
00015
              outputStream(outputStream) {
00016 }
00017
00018 void DataOutputStream::write(unsigned char* b, int len) {
00019
          writeBytes(b, len);
00020 }
00021
00022 void DataOutputStream::write(unsigned char b) {
00023
          writeByte(b);
00024 }
00025
00026 void DataOutputStream::writeByte(unsigned char b) {
00027
          outputStream->write(b);
00028 }
00029
00030 void DataOutputStream::writeBytes(unsigned char* b, int len) { 00031 for (int i = 0; i < len; i++) {
00032
              outputStream->write(b[i]);
00033
00034 }
00035
00036 void DataOutputStream::writeBoolean(bool v) {
00037
          outputStream->write((unsigned char) v);
00038 }
00039
00040 void DataOutputStream::writeChar(char c) {
00041
          outputStream->write((unsigned char) c);
00042 }
00043
00044 void DataOutputStream::writeUnsignedChar(unsigned char c) {
00045
          outputStream->write((unsigned char) c);
00046 }
00047
00048 void DataOutputStream::writeInt(int v) {
        outputStream->write((unsigned char) ((v >> 8) & 0xff));
00049
00050
          outputStream->write((unsigned char) (v & 0xff));
00051 }
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));
00063
          outputStream->write((unsigned char) ((v >> 16) & 0xff));
          outputStream->write((unsigned char) ((v >> 8) & 0xff));
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 }
00075
00076 void DataOutputStream::writeDouble(double v) {
00077
          writeLong((long) v);
00078 }
00079
00080 #endif /* __ARDUINO_IO_DATA_OUTPUT_STREAM_CPP__ */
```

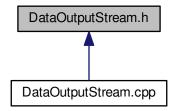
5.39 DataOutputStream.h File Reference

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

Include dependency graph for DataOutputStream.h:



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



Classes

• class DataOutputStream

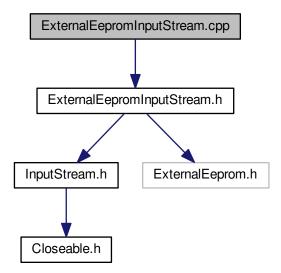
5.40 DataOutputStream.h

```
00001
00009 #ifndef __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 writeBvtes(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
00109
          virtual void writeLong(long v);
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:



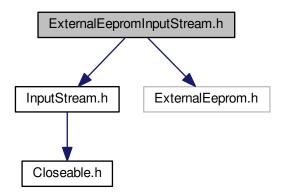
5.42 ExternalEepromInputStream.cpp

```
00014 }
00015
00016 int ExternalEepromInputStream::available() {
         int room = externalEepromSize - pos;
if (room > maxAvailableChunk) {
00017
00018
00019
              return maxAvailableChunk;
00021
          return room;
00022 }
00023
00024 void ExternalEepromInputStream::mark() {
00025
         markpos = pos;
00026 }
00027
00028 bool ExternalEepromInputStream::markSupported() {
00029
         return true;
00030 }
00031
00032 int ExternalEepromInputStream::read() {
00033
        if (pos >= externalEepromSize) {
             return -1;
00034
00035
          return (int) externalEeprom->read(pos++);
00036
00037 }
00038
00039 int ExternalEepromInputStream::read(unsigned char* b, int off, int len) {
00040
          unsigned int available = (externalEepromSize
00041
          int total;
00042
          len = (int) ((unsigned int) len > available) ? available : len;
          total = externalEeprom->readBytes(pos, &b[off], len);
00043
00044
          pos += total;
00045
          return total;
00046 }
00047
00048 void ExternalEepromInputStream::reset() {
00049
         pos = markpos;
00050 }
```

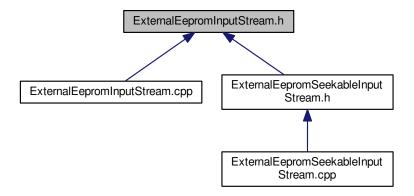
5.43 ExternalEepromInputStream.h File Reference

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

Include dependency graph for ExternalEepromInputStream.h:



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



Classes

· class ExternalEepromInputStream

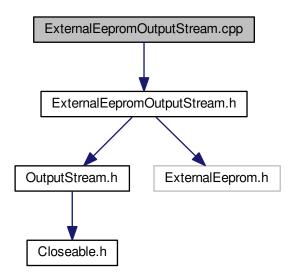
5.44 ExternalEepromInputStream.h

```
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
00018 protected:
00019
00023
          const int maxAvailableChunk;
00024
00025
          \star The externalEeprom where data is stored.
00026
00027
00028
          ExternalEeprom* externalEeprom;
00029
00030
00031
          * Current position
00032
          unsigned int pos;
00033
00034
00035
00036
          * The currently marked position in the stream.
00037
00038
          unsigned int markpos;
00039
00040
00041
          * The size of the externalEeprom.
00042
00043
          unsigned int externalEepromSize;
0\,0\,0\,4\,4
00045 public:
00046
00052
          ExternalEepromInputStream(ExternalEeprom* externalEeprom);
00053
00057
          virtual ~ExternalEepromInputStream() {
00058
00059
00067
          virtual int available();
00068
00072
          virtual void mark();
```

```
virtual bool markSupported();
08000
00084
          using InputStream::read;
00085
00091
          virtual int read();
00092
00101
          virtual int read(unsigned char* b, int off, int len);
00102
00107
          virtual void reset();
00108 };
00109
00110 #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

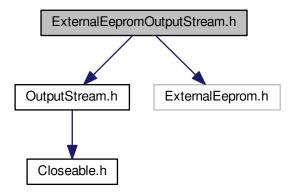
00001

```
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
               {\tt ExternalEeprom*\ externalEeprom})\ :
00016
               externalEeprom(externalEeprom) {
00017
          pos = 0;
00018 }
00019
00020 void ExternalEepromOutputStream::write(unsigned char b) {
         externalEeprom->write(pos++, b);
00021
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__ */
```

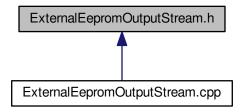
5.47 ExternalEepromOutputStream.h File Reference

```
#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

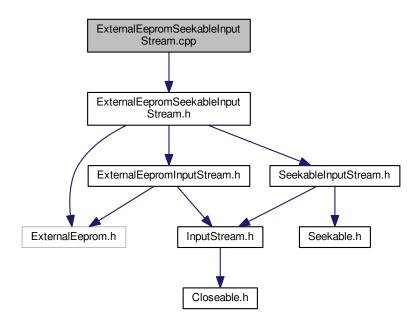
5.48 ExternalEepromOutputStream.h

```
00001
00010 #ifndef __ARDUINO_IO_EXTERNAL_EEPROM_OUTPUT_STREAM_H_
00011 #define __ARDUINO_IO_EXTERNAL_EEPROM_OUTPUT_STREAM_H_
00012
00013 #include <OutputStream.h>
00014 #include <ExternalEeprom.h>
00015
00016 class ExternalEepromOutputStream : public OutputStream {
00017
00021
          ExternalEeprom* externalEeprom;
00022
          unsigned int pos;
00026
00027
00028 public:
00029
00035
          ExternalEepromOutputStream(ExternalEeprom* externalEeprom);
00036
          virtual ~ExternalEepromOutputStream() {
00037
00038
00039
00043
          using OutputStream::write;
00044
00050
          virtual void write(unsigned char b);
00051
00060
          virtual void write(unsigned char* b, int off, int len);
00061 };
00063 #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.

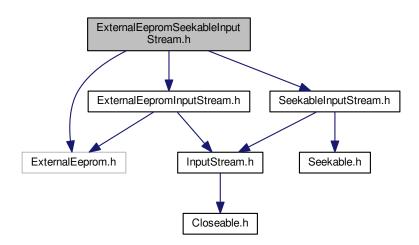
5.50 ExternalEepromSeekableInputStream.cpp

```
00001
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
            ExternalEeprom* externalEeprom) :
00017
            ExternalEepromInputStream(externalEeprom) {
00018 }
00019
00020 void ExternalEepromSeekableInputStream::seek(unsigned int pos) {
00021
        this->pos = pos;
00022 }
00024 #endif /* __ARDUINO_IO_EXTERNAL_EEPROM_SEEKABLE_INPUT_STREAM_CPP__ */
```

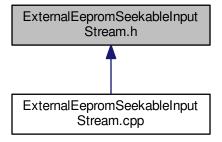
5.51 ExternalEepromSeekableInputStream.h File Reference

```
#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

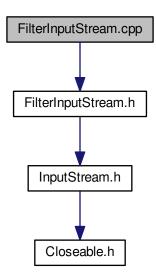
5.52 ExternalEepromSeekableInputStream.h

```
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}
{\tt ExternalEepromInputStream,}
00020
00026
         {\tt ExternalEepromSeekableInputStream}~({\tt ExternalEeprom} \star
      externalEeprom);
00027
00033
          virtual void seek (unsigned int pos);
00034 };
00035
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.

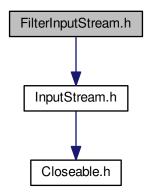
5.54 FilterInputStream.cpp

```
00001
00016 #ifndef __ARDUINO_IO_FILTER_INPUT_STREAM_CPP_
00017 #define __ARDUINO_IO_FILTER_INPUT_STREAM_CPP__ 1
00018
00019 #include "FilterInputStream.h"
00020
00022
             in(in) {
00023 }
00024
00025 int FilterInputStream::read() {
00026
         return in->read();
00027 }
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);
00036
00037 unsigned int FilterInputStream::skip(unsigned int n) {
00038
         return in->skip(n);
00039 }
00040
00041 int FilterInputStream::available() {
00042
         return in->available();
00043 }
00044
00045 void FilterInputStream::close() {
00046
         in->close();
00048
00049 void FilterInputStream::mark() {
00050
         in->mark();
00051 }
00052
00053 void FilterInputStream::reset() {
00054
        in->reset();
00055 }
00056
00057 bool FilterInputStream::markSupported() {
00058
         return in->markSupported();
00059 }
00060
00061 #endif /* __ARDUINO_IO_FILTER_INPUT_STREAM_CPP__ */
```

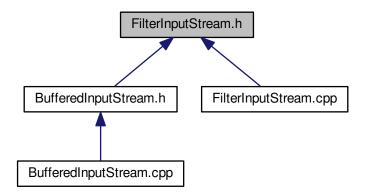
5.55 FilterInputStream.h File Reference

#include <InputStream.h>

Include dependency graph for FilterInputStream.h:



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



Classes

· class FilterInputStream

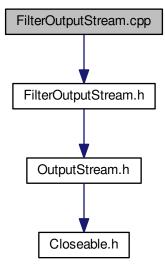
5.56 FilterInputStream.h

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

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

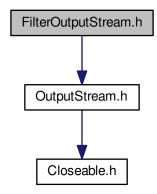
5.58 FilterOutputStream.cpp

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

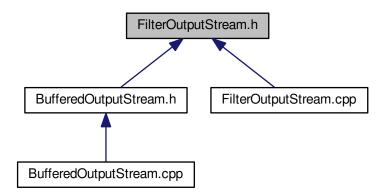
5.59 FilterOutputStream.h File Reference

#include <OutputStream.h>

Include dependency graph for FilterOutputStream.h:



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



Classes

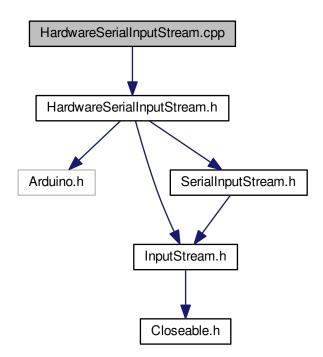
· class FilterOutputStream

5.60 FilterOutputStream.h

```
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.

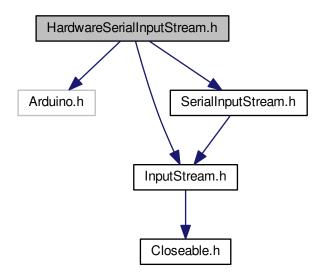
5.62 HardwareSerialInputStream.cpp

```
00001
00009 #ifndef __ARDUINO_IO_HARDWARE_SERIAL_INPUT_STREAM_CPP_
00010 #define __ARDUINO_IO_HARDWARE_SERIAL_INPUT_STREAM_CPP__ 1
00011
00012 #include "HardwareSerialInputStream.h"
00013
00014 HardwareSerialInputStream::HardwareSerialInputStream(
     unsigned int boudRate) {
00015
         Serial.begin(boudRate);
00016 }
00017
00018 int HardwareSerialInputStream::available() {
00019
         return Serial.available();
00020 }
00021
00022 int HardwareSerialInputStream::read() {
00023
       if (available() > 0)
00024
             return Serial.read();
00025
00026
         return -1;
00027 }
00029 #endif /* __ARDUINO_IO_HARDWARE_SERIAL_INPUT_STREAM_CPP__ */
```

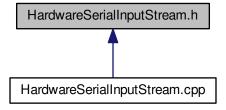
5.63 HardwareSerialInputStream.h File Reference

```
#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

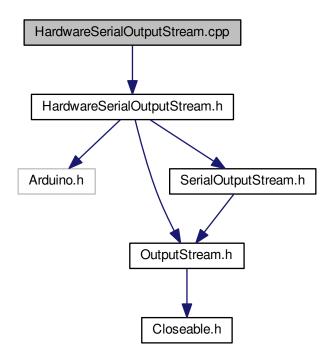
5.64 HardwareSerialInputStream.h

```
00001
00009 #ifndef __ARDUINO_IO_HARDWARE_SERIAL_INPUT_STREAM_H_
00010 #define __ARDUINO_IO_HARDWARE_SERIAL_INPUT_STREAM_H_
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
00030
         virtual int available();
00031
00035
         virtual int read();
00036 };
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.

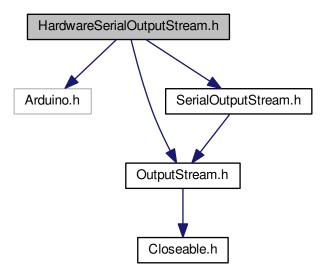
5.66 HardwareSerialOutputStream.cpp

```
00020 }
00021
00022 #endif /* __ARDUINO_IO_HARDWARE_SERIAL_OUTPUT_STREAM_CPP__ */
```

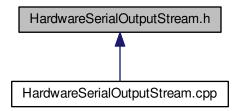
5.67 HardwareSerialOutputStream.h File Reference

```
#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

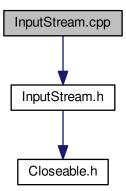
class HardwareSerialOutputStream

5.68 HardwareSerialOutputStream.h

```
00001
00009 #ifndef _ARDUINO_IO_HARDWARE_SERIAL_OUTPUT_STREAM_H_
00010 #define _ARDUINO_IO_HARDWARE_SERIAL_OUTPUT_STREAM_H_ 1
00012 #include <Arduino.h>
00013 #include <OutputStream.h>
00014 #include <SerialOutputStream.h>
00015
00016 class HardwareSerialOutputStream : public
      SerialOutputStream {
00017 public:
00018
00024
           HardwareSerialOutputStream(unsigned int boudRate);
00025
00029
           virtual void write(unsigned char b);
00030 };
00032 #endif /* __ARDUINO_IO_SOFTWARE_SERIAL_OUTPUT_STREAM_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.

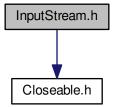
5.70 InputStream.cpp

```
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 }
00032 int InputStream::read(unsigned char* b, int len) {
00033
          return read(b, 0, len);
00034 }
00035
00036 int InputStream::read(unsigned char* b, int off, int len) {
00037
        int i, c;
if (b == (unsigned char*) 0) {
00038
00039
              return 0;
00040
00041
          c = read();
00042
          if (c == -1)
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;
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++) {
    read();
00060
00061
00062
00063
00064
          return i;
00065 }
00066
00067 #endif /* __ARDUINO_IO_INPUT_STREAM_CPP__ */
```

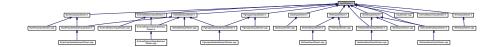
5.71 InputStream.h File Reference

#include <Closeable.h>

Include dependency graph for InputStream.h:



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



Classes

· class InputStream

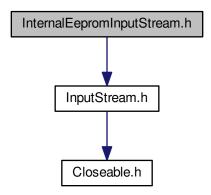
5.72 InputStream.h

```
00013 #ifndef __ARDUINO_IO_INPUT_STREAM_H_
00014 #define __ARDUINO_IO_INPUT_STREAM_H_
00015
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 InternalEepromInputStream.h File Reference

#include <InputStream.h>

Include dependency graph for InternalEepromInputStream.h:



Classes

class InternalEepromInputStream

5.74 InternalEepromInputStream.h

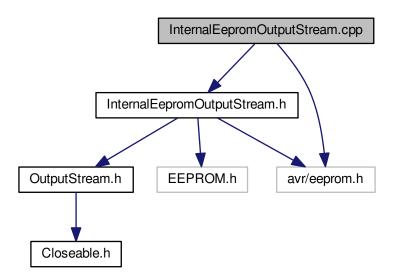
```
00010 #ifndef __ARDUINO_IO_EXTERNAL_EEPROM_INPUT_STREAM_H_
00011 #define __ARDUINO_IO_EXTERNAL_EEPROM_INPUT_STREAM_H_ 1
00012
00013 #include <InputStream.h>
00014
00015 class InternalEepromInputStream: public virtual
      InputStream {
00016
00017 protected:
00018
00022
          const int maxAvailableChunk;
00023
00024
          * Current position
00025
00026
00027
          unsigned int pos;
00028
00029
00030
          * The currently marked position in the stream.
00031
          unsigned int markpos;
00032
00033
00034
          * The size of the eeprom.
*/
00035
00036
00037
          unsigned int eepromSize;
00038
00039 public:
00040
00044
          InternalEepromInputStream();
00045
00053
          virtual int available();
00054
00058
          virtual void mark():
00059
00065
          virtual bool markSupported();
00066
00070
          using InputStream::read;
00071
00077
          virtual int read();
00078
00087
          virtual int read(unsigned char* b, int off, int len);
00088
```

```
00093    virtual void reset();
00094 };
00095
00096 #endif /* __ARDUINO_IO_EXTERNAL_EEPROM_INPUT_STREAM_H__ */
```

5.75 InternalEepromOutputStream.cpp File Reference

```
#include "InternalEepromOutputStream.h"
#include <avr/eeprom.h>
```

Include dependency graph for InternalEepromOutputStream.cpp:



Macros

• #define __ARDUINO_IO_INTERNAL_EEPROM_OUTPUT_STREAM_CPP__ 1

5.75.1 Macro Definition Documentation

```
5.75.1.1 #define __ARDUINO_IO_INTERNAL_EEPROM_OUTPUT_STREAM_CPP__1
```

Arduino IO.

InternalEepromOutputStream

A internal Eeprom output stream is an output stream for writing data to the EEPROM.

Definition at line 10 of file InternalEepromOutputStream.cpp.

5.76 InternalEepromOutputStream.cpp

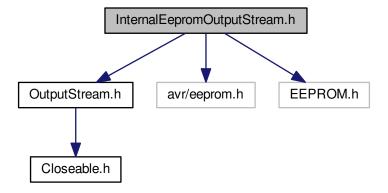
```
00001
00009 #ifndef __ARDUINO_IO_INTERNAL_EEPROM_OUTPUT_STREAM_CPP__
00010 #define __ARDUINO_IO_INTERNAL_EEPROM_OUTPUT_STREAM_CPP__ 1
00011
00012 #include "InternalEepromOutputStream.h"
00013 #include <avr/eeprom.h>
00014
```

```
00015 InternalEepromOutputStream::InternalEepromOutputStream
00016
         pos = 0;
00017 }
00018
00019 void InternalEepromOutputStream::write(unsigned char b) {
         eeprom_write_byte((unsigned char *) (pos++), b);
00021 }
00022
00023 void InternalEepromOutputStream::write(unsigned char* b, int off, int len)
          eeprom_write_block((const void *)pos, (void *)b, len);
00024
00025
          pos += len;
00026 }
00027
00028 #endif /* __ARDUINO_IO_EXTERNAL_EEPROM_OUTPUT_STREAM_CPP__ */
```

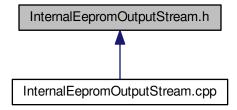
5.77 InternalEepromOutputStream.h File Reference

```
#include <OutputStream.h>
#include <avr/eeprom.h>
#include <EEPROM.h>
```

Include dependency graph for InternalEepromOutputStream.h:



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



Classes

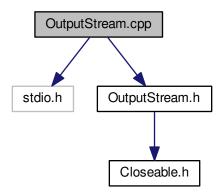
• class InternalEepromOutputStream

5.78 InternalEepromOutputStream.h

```
00001
00010 #ifndef _ARDUINO_IO_INTERNAL_EEPROM_OUTPUT_STREAM_H_
00011 #define _ARDUINO_IO_INTERNAL_EEPROM_OUTPUT_STREAM_H_ 1
00012
00013 #include <OutputStream.h>
00014 #include <avr/eeprom.h>
00015 #include <EEPROM.h>
00016 class InternalEepromOutputStream : public OutputStream {
00017
00021
           unsigned int pos;
00022
00023 public:
00024
00028
           InternalEepromOutputStream();
00029
00033
           using OutputStream::write;
00034
00040
           virtual void write(unsigned char b);
00041
00050
           virtual void write(unsigned char* b, int off, int len);
00051 };
00052
00053 #endif /* __ARDUINO_IO_INTERNAL_EEPROM_OUTPUT_STREAM_H__ */
```

5.79 OutputStream.cpp File Reference

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



Macros

#define __ARDUINO_IO_OUTPUT_STREAM_CPP__ 1

5.79.1 Macro Definition Documentation

5.79.1.1 #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.

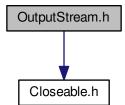
5.80 OutputStream.cpp

```
00001
00011 #ifndef __ARDUINO_IO_OUTPUT_STREAM_CPP_
00012 #define __ARDUINO_IO_OUTPUT_STREAM_CPP_
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__ */
```

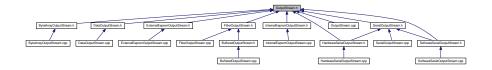
5.81 OutputStream.h File Reference

#include <Closeable.h>

Include dependency graph for OutputStream.h:



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



Classes

· class OutputStream

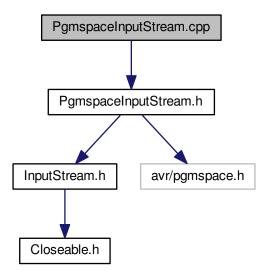
5.82 OutputStream.h

```
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_ */
```

5.83 PgmspaceInputStream.cpp File Reference

#include "PgmspaceInputStream.h"

Include dependency graph for PgmspaceInputStream.cpp:



Macros

#define __ARDUINO_IO_PGMSPACE_INPUT_STREAM_CPP__ 1

5.83.1 Macro Definition Documentation

```
5.83.1.1 #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.

5.84 PgmspaceInputStream.cpp

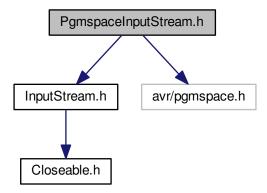
```
00001
00010 #ifndef __ARDUINO_IO_PGMSPACE_INPUT_STREAM_CPP_
00011 #define __ARDUINO_IO_PGMSPACE_INPUT_STREAM_CPP__ 1
00012
00013 #include "PgmspaceInputStream.h"
00014
00015 PgmspaceInputStream::PgmspaceInputStream(char PROGMEM* buf,
     unsigned int count) : buf(buf), count(count) {
00016
         markpos = 0;
00017
         pos = 0;
00018 }
00019
00020 int PgmspaceInputStream::available() {
00021
       if ((count - pos) > 0) {
             return 1;
00022
00023
00024
          return 0;
00025 }
00026
```

```
00027 void PgmspaceInputStream::mark() {
00028
         markpos = pos;
00029 }
00030
00031 bool PgmspaceInputStream::markSupported() {
00032
         return true;
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__ */
```

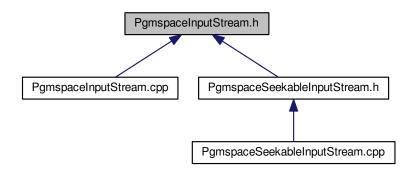
5.85 PgmspaceInputStream.h File Reference

```
#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

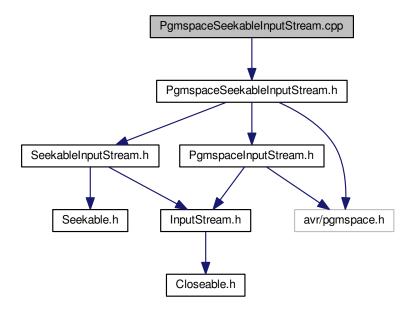
5.86 PgmspaceInputStream.h

```
00001
00010 #ifndef __ARDUINO_IO_PGMSPACE_INPUT_STREAM_H_
00011 #define __ARDUINO_IO_PGMSPACE_INPUT_STREAM_H__ 1
00012
00013 #include <InputStream.h>
00014 #include <avr/pgmspace.h>
00015
00016 class PgmspaceInputStream : public virtual InputStream {
00017 protected:
00018
00019
           \star The buffer where data is stored.
00020
00021
00022
          char PROGMEM* buf;
00023
00024
          00025
00026
00027
          unsigned int count;
00028
00029
00030
          * Current position
00031
          unsigned int pos;
00032
00033
00034
          \star The currently marked position in the stream. 
 \star/
00035
00036
00037
00038
          unsigned int markpos;
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
          virtual void reset();
00080 };
00081
00082 #endif /* __ARDUINO_IO_PGMSPACE_INPUT_STREAM_H__ */
```

5.87 PgmspaceSeekableInputStream.cpp File Reference

#include "PgmspaceSeekableInputStream.h"

Include dependency graph for PgmspaceSeekableInputStream.cpp:



Macros

• #define __ARDUINO_IO_PGMSPACE_SEEKABLE_INPUT_STREAM_CPP__ 1

5.87.1 Macro Definition Documentation

5.87.1.1 #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.

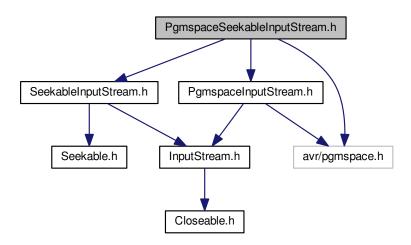
Definition at line 11 of file PgmspaceSeekableInputStream.cpp.

5.88 PgmspaceSeekableInputStream.cpp

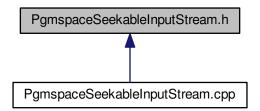
5.89 PgmspaceSeekableInputStream.h File Reference

```
#include <SeekableInputStream.h>
#include <PgmspaceInputStream.h>
#include <avr/pgmspace.h>
```

Include dependency graph for PgmspaceSeekableInputStream.h:



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



Classes

• class PgmspaceSeekableInputStream

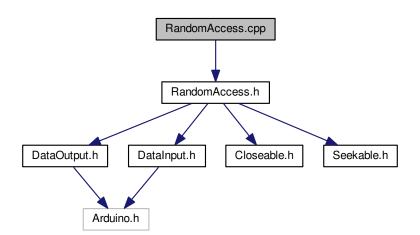
5.90 PgmspaceSeekableInputStream.h

```
00001
00010 #ifndef __ARDUINO_IO_PGMSPACE_SEEKABLE_INPUT_STREAM_H_
00011 #define __ARDUINO_IO_PGMSPACE_SEEKABLE_INPUT_STREAM_H_ 1
00012
00013 #include <SeekableInputStream.h>
00014 #include <PgmspaceInputStream.h>
00015 #include <avr/pgmspace.h>
```

```
00016
00017 class PgmspaceSeekableInputStream: public
      SeekableInputStream, public PgmspaceInputStream {
00018
00019 public:
00020
         PgmspaceSeekableInputStream(char PROGMEM* buf, unsigned int
00021
00022
00023
          virtual ~PgmspaceSeekableInputStream() {
00024
00025
00026
          virtual void seek(unsigned int pos);
00027 };
00028
00029 #endif /* __ARDUINO_IO_PGMSPACE_SEEKABLE_INPUT_STREAM_H__ */
```

5.91 RandomAccess.cpp File Reference

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



Macros

• #define __ARDUINO_IO_RANDOM_ACCESS_CPP__ 1

5.91.1 Macro Definition Documentation

5.91.1.1 #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.

5.92 RandomAccess.cpp

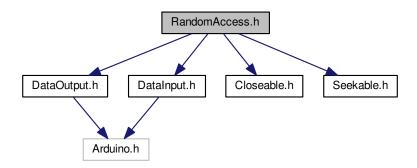
00001

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

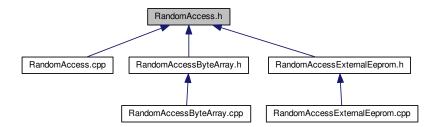
5.93 RandomAccess.h File Reference

```
#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

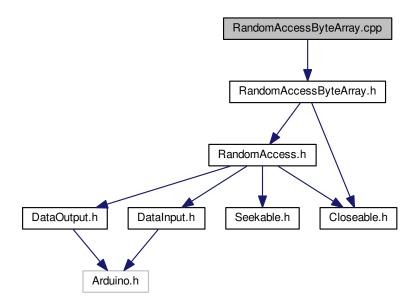
class RandomAccess

5.94 RandomAccess.h

```
00001
00009 #ifndef __ARDUINO_IO_RANDOM_ACCESS_H_
00010 #define __ARDUINO_IO_RANDOM_ACCESS_H_ 1
00011
00012 #include <DataOutput.h>
```

5.95 RandomAccessByteArray.cpp File Reference

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



Macros

#define __ARDUINO_IO_RANDOM_ACCESS_BYTE_ARRAY_CPP__ 1

5.95.1 Macro Definition Documentation

5.95.1.1 #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.

5.96 RandomAccessByteArray.cpp

00001

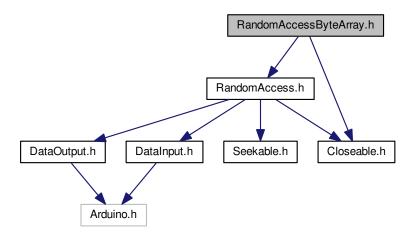
```
00010 #ifndef __ARDUINO_IO_RANDOM_ACCESS_BYTE_ARRAY_CPP_
00011 #define __ARDUINO_IO_RANDOM_ACCESS_BYTE_ARRAY_CPP__ 1
00012
00013 #include "RandomAccessByteArray.h"
00014
00015 RandomAccessByteArray::RandomAccessByteArray(unsigned char* buf
00016
              unsigned int count) :
00017
             buf(buf), count(count) {
         pos = 0;
00018
00019 }
00020
00021 unsigned int RandomAccessByteArray::length() {
00022
         return count;
00023 }
00024
00025 void RandomAccessByteArray::seek(unsigned int pos) {
00026
         this->pos = pos;
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
00047
00048 }
00049
00050 void RandomAccessByteArray::writeBoolean(bool v) {
00051
         buf[pos++] = (unsigned char) v;
00052 }
00054 void RandomAccessByteArray::writeChar(char c) {
00055
         buf[pos++] = (unsigned char) c;
00056 }
00057
00058 void RandomAccessByteArray::writeUnsignedChar(unsigned char c) {
00059
         buf[pos++] = (unsigned char) c;
00060 }
00061
00062 void RandomAccessByteArray::writeInt(int v) {
00063
         buf[pos++] = (unsigned char) ((v >> 8) & 0xff);
          buf[pos++] = (unsigned char) (v & 0xff);
00064
00066
00067 void RandomAccessByteArray::writeUnsignedInt(unsigned int v) {
00068
         writeInt((int) v);
00069 }
00070
00071 void RandomAccessByteArray::writeWord(word v) {
00072
         writeInt((int) v);
00073 }
00074
00075 void RandomAccessByteArray::writeLong(long v) {
00076
         buf[pos++] = (unsigned char) ((v >> 24) & 0xff);
00077
          buf[pos++] = (unsigned char) ((v >> 16) & 0xff);
          buf[pos++] = (unsigned char) ((v >> 8) & 0xff);
00078
00079
          buf[pos++] = (unsigned char) (v & 0xff);
00080 }
00081
00082 void RandomAccessByteArray::writeUnsignedLong(unsigned long v) {
00083
         writeLong((long) v);
00084 }
00085
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() {
         int v = 0;
v = buf[pos++];
00111
00112
00113
          v <<= 8;
          v |= buf[pos++];
00114
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() {
          long v = 0;
00128
          v = (buf[pos++] & 0xff);
00129
          v <<= 8;
00130
          v |= (buf[pos++] & 0xff);
          v <<= 8;
00131
          v |= (buf[pos++] & 0xff);
00132
00133
          v <<= 8;
00134
          v |= (buf[pos++] & 0xff);
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 }
00149
00150 void RandomAccessByteArray::readFully(unsigned char* b, int len) {
         for (int i = 0; i < len; i++) {
    b[i] = buf[pos++];</pre>
00151
00153
00154 }
00155
00156 unsigned int RandomAccessByteArray::skipBytes(unsigned int n) {
00157
         unsigned int skipped;
00158
          unsigned int newpos;
00159
          newpos = pos + n;
          if (newpos > count)
00160
00161
              newpos = count;
00162
00163
          skipped = newpos - pos;
00164
          pos = newpos;
          return skipped;
00165
00166 }
00167
00168 #endif /* __ARDUINO_IO_RANDOM_ACCESS_BYTE_ARRAY_CPP__ */
```

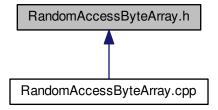
5.97 RandomAccessByteArray.h File Reference

```
#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

5.98 RandomAccessByteArray.h

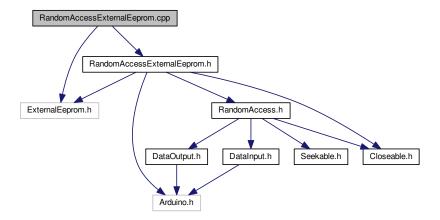
```
00001
00010 #ifndef __ARDUINO_IO_RANDOM_ACCESS_BYTE_ARRAY_H_
00011 #define __ARDUINO_IO_RANDOM_ACCESS_BYTE_ARRAY_H_ 1
00012
00013 #include <RandomAccess.h>
00014 #include <Closeable.h>
00015
00016 class RandomAccessByteArray : public RandomAccess, public virtual
      Closeable {
00017
00021
          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__ */
```

5.99 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

5.99.1 Macro Definition Documentation

5.99.1.1 #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.

5.100 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);
```

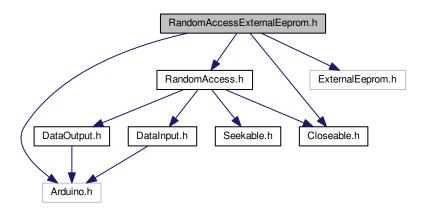
```
00036 }
00037
00038 void RandomAccessExternalEeprom::write(unsigned char b) {
00039
          writeByte(b);
00040 }
00041
00042 void RandomAccessExternalEeprom::writeByte(unsigned char b) {
00043
          externalEeprom->write(pos++, b);
00044 }
00045
00046 void RandomAccessExternalEeprom::writeBytes(unsigned 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) {
         externalEeprom->write(pos++, (unsigned char) ((v >> 8) & 0xff));
externalEeprom->write(pos++, (unsigned char) (v & 0xff));
00065
00066
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 }
00107
00108 unsigned char RandomAccessExternalEeprom::readUnsignedChar() {
00109
          return (unsigned char) externalEeprom->read(pos++);
00110 }
00111
00112 int RandomAccessExternalEeprom::readInt() {
00113
          int v = 0;
00114
          v = externalEeprom->read(pos++);
00115
          v <<= 8;
00116
          v |= (externalEeprom->read(pos++) & 0xff);
00117
          return v;
00118 }
```

```
00119
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++);
         v <<= 8;
00131
00132
         v |= (externalEeprom->read(pos++) & 0xff);
00133
         v <<= 8;
00134
         v |= (externalEeprom->read(pos++) & 0xff);
00135
         v <<= 8;
         v |= (externalEeprom->read(pos++) & 0xff);
00136
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>
             b[i] = externalEeprom->read(pos++);
00154
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__ */
```

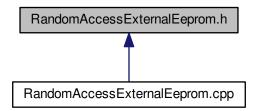
5.101 RandomAccessExternalEeprom.h File Reference

```
#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

5.102 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 */
```

5.103 RandomAccessResource.cpp File Reference

Macros

#define __ARDUINO_IO_RANDOM_ACCESS_RESOURCE_CPP__ 1

5.103.1 Macro Definition Documentation

5.103.1.1 #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.

5.104 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 }
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);
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() {
         return (unsigned char) resource->read();
00108
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() {
         return (word) readInt();
00124
00126
00127 long RandomAccessResource::readLong() {
00128
          long v = 0;
00129
         v = resource->read();
00130
         v <<= 8:
00131
         v |= (resource->read() & 0xff);
         v <<= 8;
00132
00133
         v |= (resource->read() & 0xff);
00134
         v <<= 8;
00135
         v |= (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;
00161
          pos = (unsigned int) resource->tell();
          len = resource->size();
00162
          newpos = pos + n;
00163
00164
          if (newpos > len)
00165
             newpos = len;
00166
00167
          seek (newpos);
00168
          return (unsigned int) (newpos - pos);
00169 }
00170
00171 #endif /* USING_RESOURCE_LIBRARIES */
00172
00173 #endif /* __ARDUINO_IO_RANDOM_ACCESS_RESOURCE_CPP__ */
```

5.105 RandomAccessResource.h File Reference

5.106 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
00139
          virtual void writeUnsignedLong(unsigned long v);
00140
00146
          virtual void writeFloat(float v);
00147
00153
          virtual void writeDouble(double v);
00154
00160
          virtual unsigned char readByte();
00161
00167
          virtual bool readBoolean();
00168
00174
          virtual char readChar();
00175
00181
          virtual unsigned char readUnsignedChar();
00182
00188
          virtual int readInt();
00189
00195
          virtual unsigned int readUnsignedInt();
00196
00202
          virtual word readWord();
00203
00209
          virtual long readLong();
00210
00216
          virtual unsigned long readUnsignedLong();
00217
00223
          virtual float readFloat();
00224
00230
          virtual double readDouble();
00231
00238
          virtual void readFully(unsigned char* b, int len);
00239
00246
          virtual unsigned int skipBytes(unsigned int n);
00247 };
00248
00249 #endif /* USING_RESOURCE_LIBRARIES */
00250
```

```
00251 #endif /* __ARDUINO_IO_RANDOM_ACCESS_RESOURCE_H__ */
```

5.107 ResourceInputStream.cpp File Reference

Macros

#define __ARDUINO_IO_RESOURCE_INPUT_STREAM_CPP__ 1

5.107.1 Macro Definition Documentation

```
5.107.1.1 #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.

5.108 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() {
        if ((resourceSize - pos) > 0) {
00024
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__ */
```

5.109 ResourceInputStream.h File Reference

5.110 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
          \star The currently marked position in the stream.
00031
00032
00033
          unsigned int markpos;
00034
00035
          * The size of the resource.
00036
00037
00038
          unsigned int resourceSize;
00039
00040 public:
00041
00042
          ResourceInputStream(Resource* resource);
00043
         virtual int available():
00049
00050
00055
         virtual void close();
00056
00060
         virtual void mark();
00061
00065
         virtual bool markSupported();
00066
         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__ */
```

5.111 ResourceOutputStream.cpp File Reference

Macros

```
    #define __ARDUINO_IO_RESOURCE_OUTPUT_STREAM_CPP__ 1
```

5.111.1 Macro Definition Documentation

```
5.111.1.1 #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.

5.112 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__ */
```

5.113 ResourceOutputStream.h File Reference

5.114 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__ */
```

5.115 ResourceSeekableInputStream.cpp File Reference

Macros

```
• #define __ARDUINO_IO_RESOURCE_SEEKABLE_INPUT_STREAM_CPP__ 1
```

5.115.1 Macro Definition Documentation

```
5.115.1.1 #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.

5.116 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__ */
```

5.117 ResourceSeekableInputStream.h File Reference

5.118 ResourceSeekableInputStream.h

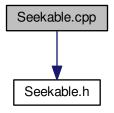
```
00010 #ifndef __ARDUINO_IO_RESOURCE_SEEKABLE_INPUT_STREAM_H_
00011 #define __ARDUINO_IO_RESOURCE_SEEKABLE_INPUT_STREAM_H_
00012
00013 #if USING_RESOURCE_LIBRARIES
00014
00015 #include <SeekableInputStream.h>
00016 #include <ResourceInputStream.h>
00017 #include <Resource.h>
00018
00019 class ResourceSeekableInputStream : 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__ */
```

5.119 Seekable.cpp File Reference

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

5.120 Seekable.cpp 189

Include dependency graph for Seekable.cpp:



Macros

```
• #define __ARDUINO_IO_SEEKABLE_CPP__ 1
```

5.119.1 Macro Definition Documentation

```
5.119.1.1 #define __ARDUINO_IO_SEEKABLE_CPP__ 1
```

Arduino IO.

Seekable

Definition at line 8 of file Seekable.cpp.

5.120 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__ */
```

5.121 Seekable.h File Reference

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



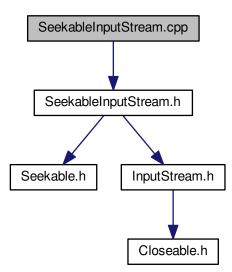
Classes

• class Seekable

5.122 Seekable.h

5.123 SeekableInputStream.cpp File Reference

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



Macros

• #define __ARDUINO_IO_SEEKABLE_INPUT_STREAM_CPP__ 1

5.123.1 Macro Definition Documentation

5.123.1.1 #define __ARDUINO_IO_SEEKABLE_INPUT_STREAM_CPP__1

Arduino IO.

SeekableInputStream

Definition at line 8 of file SeekableInputStream.cpp.

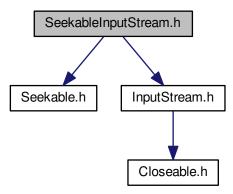
5.124 SeekableInputStream.cpp

00001

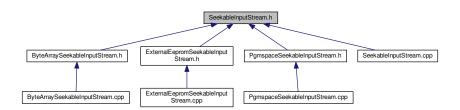
```
00007 #ifndef __ARDUINO_IO_SEEKABLE_INPUT_STREAM_CPP__
00008 #define __ARDUINO_IO_SEEKABLE_INPUT_STREAM_CPP__ 1
00009
00010 #include "SeekableInputStream.h"
00011
00011 #endif /* __ARDUINO_IO_SEEKABLE_INPUT_STREAM_CPP__ */
```

5.125 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

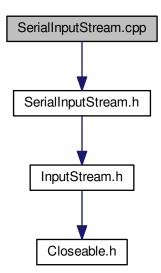
5.126 SeekableInputStream.h

```
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__ */
```

5.127 SerialInputStream.cpp File Reference

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



Macros

#define __ARDUINO_IO_SERIAL_INPUT_STREAM_CPP__ 1

5.127.1 Macro Definition Documentation

5.127.1.1 #define __ARDUINO_IO_SERIAL_INPUT_STREAM_CPP__ 1

Arduino IO.

SerialInputStream

A SerialInputStream obtains input bytes from a serial port.

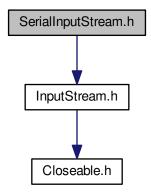
Definition at line 10 of file SerialInputStream.cpp.

5.128 SerialInputStream.cpp

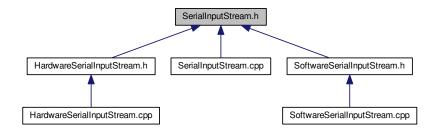
```
00001
00009 #ifndef __ARDUINO_IO_SERIAL_INPUT_STREAM_CPP__
00010 #define __ARDUINO_IO_SERIAL_INPUT_STREAM_CPP__ 1
00011
00012 #include "SerialInputStream.h"
00013
00013 #endif /* __ARDUINO_IO_SERIAL_INPUT_STREAM_CPP__ */
```

5.129 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

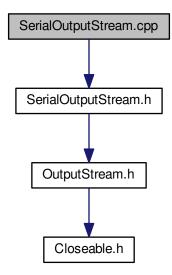
class SerialInputStream

5.130 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__ */
```

5.131 SerialOutputStream.cpp File Reference

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



Macros

• #define __ARDUINO_IO_SERIAL_OUTPUT_STREAM_CPP__ 1

5.131.1 Macro Definition Documentation

5.131.1.1 #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.

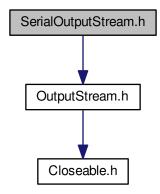
5.132 SerialOutputStream.cpp

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

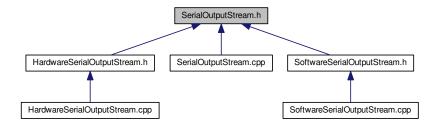
5.133 SerialOutputStream.h File Reference

#include <OutputStream.h>

Include dependency graph for SerialOutputStream.h:



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



Classes

• class SerialOutputStream

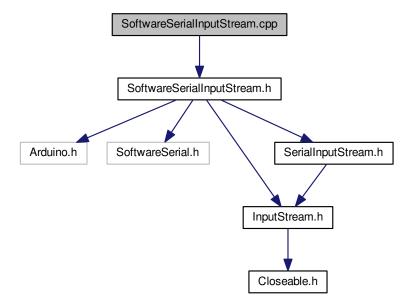
5.134 SerialOutputStream.h

```
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__ */
```

5.135 SoftwareSerialInputStream.cpp File Reference

#include "SoftwareSerialInputStream.h"

Include dependency graph for SoftwareSerialInputStream.cpp:



Macros

#define __ARDUINO_IO_SOFTWARE_SERIAL_INPUT_STREAM_CPP__ 1

5.135.1 Macro Definition Documentation

5.135.1.1 #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.

5.136 SoftwareSerialInputStream.cpp

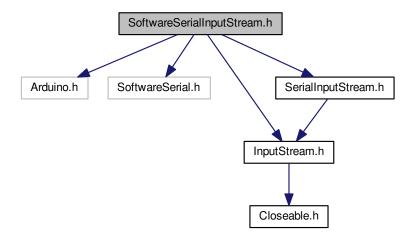
```
00001
00009 #ifndef __ARDUINO_IO_SOFTWARE_SERIAL_INPUT_STREAM_CPP_
00010 #define __ARDUINO_IO_SOFTWARE_SERIAL_INPUT_STREAM_CPP__ 1
00011
00012 #include "SoftwareSerialInputStream.h"
00013
00014 SoftwareSerialInputStream::SoftwareSerialInputStream(
     SoftwareSerial *softwareSerial,
00015
              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__ */
```

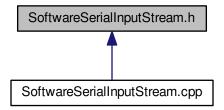
5.137 SoftwareSerialInputStream.h File Reference

```
#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

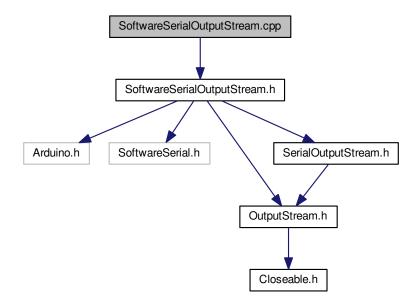
• class SoftwareSerialInputStream

5.138 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__ */
```

5.139 SoftwareSerialOutputStream.cpp File Reference

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



Macros

• #define __ARDUINO_IO_SOFTWARE_SERIAL_OUTPUT_STREAM_CPP__ 1

5.139.1 Macro Definition Documentation

5.139.1.1 #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.

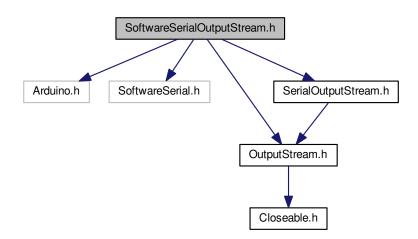
5.140 SoftwareSerialOutputStream.cpp

```
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__ */
```

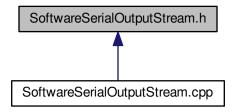
5.141 SoftwareSerialOutputStream.h File Reference

```
#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

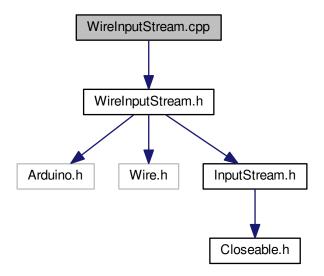
5.142 SoftwareSerialOutputStream.h

```
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
00022
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__ */
```

5.143 WireInputStream.cpp File Reference

#include "WireInputStream.h"

Include dependency graph for WireInputStream.cpp:



Macros

#define __ARDUINO_IO_WIRE_INPUT_STREAM_CPP__ 1

5.143.1 Macro Definition Documentation

5.143.1.1 #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.

5.144 WireInputStream.cpp

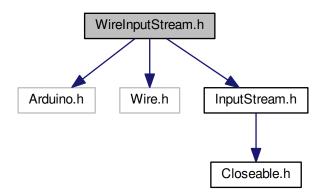
```
00009 #ifndef __ARDUINO_IO_WIRE_INPUT_STREAM_CPP_
00010 #define __ARDUINO_IO_WIRE_INPUT_STREAM_CPP__
00011
00012 #include "WireInputStream.h"
00013
00014 WireInputStream::WireInputStream(unsigned char address) {
00015
         this->address = address;
00016
         Wire.begin();
00017 }
00018
00019 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();
         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__ */
```

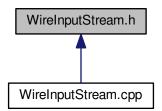
5.145 WireInputStream.h File Reference

```
#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

5.146 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
00052
          virtual int read(unsigned char* b, int off, int len);
00053 };
00054
00055 #endif /* __ARDUINO_IO_WIRE_INPUT_STREAM_H__ */
```

- 5.147 WireOutputStream.cpp File Reference
- 5.148 WireOutputStream.cpp
- 5.149 WireOutputStream.h File Reference
- 5.150 WireOutputStream.h

Index

ARDUINO_IO_BYTE_ARRAY_INPUT_STREAM_	_CPP
CPP	RandomAccessResource.cpp, 181
ByteArrayInputStream.cpp, 117	ARDUINO_IO_RESOURCE_INPUT_STREAM_C
ARDUINO_IO_BYTE_ARRAY_OUTPUT_STREA	PP
M_CPP	ResourceInputStream.cpp, 185
ByteArrayOutputStream.cpp, 120	ARDUINO_IO_RESOURCE_OUTPUT_STREAM_
ARDUINO_IO_CLOSEABLE_CPP	CPP
Closeable.cpp, 125	ResourceOutputStream.cpp, 186
ARDUINO_IO_DATA_INPUT_CPP	ARDUINO_IO_RESOURCE_SEEKABLE_INPUT_
DataInput.cpp, 126	STREAM_CPP
ARDUINO_IO_DATA_INPUT_STREAM_CPP	ResourceSeekableInputStream.cpp, 187
DataInputStream.cpp, 129	ARDUINO_IO_SEEKABLE_CPP
ARDUINO_IO_DATA_OUTPUT_CPP	Seekable.cpp, 189
DataOutput.cpp, 132	ARDUINO_IO_SEEKABLE_INPUT_STREAM_CP
ARDUINO_IO_DATA_OUTPUT_STREAM_CPP	P
DataOutputStream.cpp, 134	SeekableInputStream.cpp, 190
ARDUINO_IO_EXTERNAL_EEPROM_OUTPUT_	ARDUINO_IO_SERIAL_INPUT_STREAM_CPP
STREAM_CPP	SerialInputStream.cpp, 192
ExternalEepromOutputStream.cpp, 140	ARDUINO_IO_SERIAL_OUTPUT_STREAM_CPP
ARDUINO_IO_EXTERNAL_EEPROM_SEEKABL	
E INPUT STREAM CPP	SerialOutputStream.cpp, 194
ExternalEepromSeekableInputStream.cpp, 143	ARDUINO_IO_SOFTWARE_SERIAL_INPUT_ST
ARDUINO_IO_FILTER_INPUT_STREAM_CPP	REAM CPP
FilterInputStream.cpp, 145	SoftwareSerialInputStream.cpp, 196
ARDUINO_IO_FILTER_OUTPUT_STREAM_CPP	ARDUINO_IO_SOFTWARE_SERIAL_OUTPUT_S
	TREAM_CPP
FilterOutputStream.cpp, 148	SoftwareSerialOutputStream.cpp, 198
ARDUINO_IO_HARDWARE_SERIAL_INPUT_ST	ARDUINO_IO_WIRE_INPUT_STREAM_CPP
REAM_CPP	WireInputStream.cpp, 201
HardwareSerialInputStream.cpp, 151	~BufferedInputStream
ARDUINO_IO_HARDWARE_SERIAL_OUTPUT_S	BufferedInputStream, 9
TREAM_CPP	~BufferedOutputStream
HardwareSerialOutputStream.cpp, 154	BufferedOutputStream, 14
ARDUINO_IO_INPUT_STREAM_CPP	~ByteArrayInputStream
InputStream.cpp, 156	ByteArrayInputStream, 17
ARDUINO_IO_INTERNAL_EEPROM_OUTPUT_S	~ByteArrayOutputStream
TREAM_CPP	ByteArrayOutputStream, 20
InternalEepromOutputStream.cpp, 160	\sim ExternalEepromInputStream
ARDUINO_IO_OUTPUT_STREAM_CPP	ExternalEepromInputStream, 46
OutputStream.cpp, 162	\sim ExternalEepromOutputStream
ARDUINO_IO_PGMSPACE_INPUT_STREAM_C	ExternalEepromOutputStream, 49
PP	\sim PgmspaceSeekableInputStream
PgmspaceInputStream.cpp, 165	PgmspaceSeekableInputStream, 79
ARDUINO_IO_PGMSPACE_SEEKABLE_INPUT_	
STREAM_CPP	address
PgmspaceSeekableInputStream.cpp, 168	WireInputStream, 109
ARDUINO_IO_RANDOM_ACCESS_BYTE_ARRA	available
Y_CPP	BufferedInputStream, 9
RandomAccessByteArray.cpp, 172	ByteArrayInputStream, 17
ARDUINO_IO_RANDOM_ACCESS_CPP	ExternalEepromInputStream, 46
RandomAccess.cpp, 170	FilterInputStream, 54
ARDUINO_IO_RANDOM_ACCESS_EXTERNAL_	HardwareSerialInputStream, 62
EEPROM_CPP	InputStream, 66
RandomAccessExternalEeprom.cpp, 177	InternalEepromInputStream, 69
ARDUINO_IO_RANDOM_ACCESS_RESOURCE	PgmspaceInputStream, 76

SoftwareSerialInputStream, 105	\sim ByteArrayOutputStream, 20
WireInputStream, 109	buf, 22
	ByteArrayOutputStream, 20
buf	count, 22
BufferedInputStream, 11	pos, 22
BufferedOutputStream, 15	reset, 20
ByteArrayInputStream, 18	size, 20
ByteArrayOutputStream, 22	toByteArray, 20
PgmspaceInputStream, 77	write, 20
RandomAccessByteArray, 89	ByteArrayOutputStream.cpp, 120
BufferedInputStream, 7	ARDUINO IO BYTE ARRAY OUTPUT ST←
~BufferedInputStream, 9	REAM_CPP, 120
available, 9	ByteArrayOutputStream.h, 121, 122
buf, 11	
BufferedInputStream, 9	ByteArraySeekableInputStream, 22
close, 9	ByteArraySeekableInputStream, 23
count, 11	seek, 23
fill, 10	ByteArraySeekableInputStream.cpp, 122, 123
mark, 10	ByteArraySeekableInputStream.h, 123, 124
markSupported, 10	alasa
marked, 11	close
	BufferedInputStream, 9
markpos, 11	BufferedOutputStream, 14
pos, 11	Closeable, 25
read, 10	FilterInputStream, 54
realineBufferContent, 10	FilterOutputStream, 59
reset, 11	InputStream, 66
size, 12	OutputStream, 73
skip, 11	RandomAccessByteArray, 82
BufferedInputStream.cpp, 110	RandomAccessExternalEeprom, 92
BufferedInputStream.h, 112, 113	Closeable, 24
BufferedOutputStream, 12	close, 25
\sim BufferedOutputStream, 14	Closeable.cpp, 125
buf, 15	ARDUINO_IO_CLOSEABLE_CPP, 125
BufferedOutputStream, 14	Closeable.h, 126
close, 14	count
count, 15	BufferedInputStream, 11
flush, 14	BufferedOutputStream, 15
flushBuffer, 14	ByteArrayInputStream, 18
size, 15	ByteArrayOutputStream, 22
write, 14, 15	PgmspaceInputStream, 77
BufferedOutputStream.cpp, 114	RandomAccessByteArray, 89
BufferedOutputStream.h, 115, 116	
ByteArrayInputStream, 16	DataInput, 25
~ByteArrayInputStream, 17	readBoolean, 26
available, 17	readByte, 26
buf, 18	readChar, 26
ByteArrayInputStream, 17	readDouble, 26
count, 18	readFloat, 26
mark, 17	readFully, 26
markSupported, 18	readint, 27
markpos, 18	readLong, 27
pos, 18	readUnsignedChar, 27
read, 18	readUnsignedInt, 27
	_
reset, 18 ByteArrayInputStream.cpp, 116, 117	readUnsignedLong, 27 readWord, 27
ARDUINO_IO_BYTE_ARRAY_INPUT_STRE↔	skipBytes, 27
AM_CPP, 117	DataInput.cpp, 126, 127
ByteArrayInputStream.h, 118, 119	ARDUINO_IO_DATA_INPUT_CPP, 126
ByteArrayOutputStream, 19	DataInput.h, 127, 128

DataInputStream, 29	InternalEepromInputStream, 70
DataInputStream, 30	externalEeprom
inputStream, 34	ExternalEepromInputStream, 47
•	·
readBoolean, 30	ExternalEepromOutputStream, 50
readByte, 30	RandomAccessExternalEeprom, 98
readChar, 31	ExternalEepromInputStream, 44
readDouble, 31	\sim ExternalEepromInputStream, 46
readFloat, 31	available, 46
readFully, 31	externalEeprom, 47
readInt, 31	ExternalEepromInputStream, 46
readLong, 32	externalEepromSize, 47
readUnsignedChar, 32	mark, 46
readUnsignedInt, 32	markSupported, 47
readUnsignedLong, 32	markpos, 48
readWord, 32	maxAvailableChunk, 48
skipBytes, 32	pos, 48
DataInputStream.cpp, 128, 129	read, 47
ARDUINO_IO_DATA_INPUT_STREAM_CPP	reset, 47
, 129	ExternalEepromInputStream.cpp, 137
DataInputStream.h, 130, 131	ExternalEepromInputStream.h, 138, 139
•	·
DataOutput, 34	External Engrand Output Stream 48
write, 35	~ExternalEepromOutputStream, 49
writeBoolean, 35	externalEeprom, 50
writeByte, 35	ExternalEepromOutputStream, 49
writeBytes, 35	pos, 50
writeChar, 37	write, 50
writeDouble, 37	ExternalEepromOutputStream.cpp, 140
writeFloat, 37	ARDUINO_IO_EXTERNAL_EEPROM_OUTP
writeInt, 37	UT_STREAM_CPP, 140
writeLong, 37	ExternalEepromOutputStream.h, 141, 142
writeUnsignedChar, 37	ExternalEepromSeekableInputStream, 50
writeUnsignedInt, 37	ExternalEepromSeekableInputStream, 52
writeUnsignedLong, 39	seek, 52
writeWord, 39	ExternalEepromSeekableInputStream.cpp, 142, 143
DataOutput.cpp, 131, 132	ARDUINO_IO_EXTERNAL_EEPROM_SEEK
ARDUINO IO DATA OUTPUT CPP , 132	ABLE_INPUT_STREAM_CPP, 143
DataOutput.h, 132, 133	ExternalEepromSeekableInputStream.h, 144
DataOutputStream, 39	externalEepromSize
DataOutputStream, 41	ExternalEepromInputStream, 47
outputStream, 44	External Exprominiputoticam, 47
•	fill
write, 42	BufferedInputStream, 10
writeBoolean, 42	FilterInputStream, 52
writeByte, 42	available, 54
writeBytes, 42	close, 54
writeChar, 43	FilterInputStream, 54
writeDouble, 43	•
writeFloat, 43	in, 57
writeInt, 43	mark, 55
writeLong, 43	markSupported, 55
writeUnsignedChar, 43	read, 55, 56
writeUnsignedInt, 44	reset, 56
writeUnsignedLong, 44	skip, 56
writeWord, 44	FilterInputStream.cpp, 145, 146
DataOutputStream.cpp, 134	ARDUINO_IO_FILTER_INPUT_STREAM_C↔
ARDUINO_IO_DATA_OUTPUT_STREAM_C↔	PP, 145
PP , 134	FilterInputStream.h, 146, 147
DataOutputStream.h, 135, 136	FilterOutputStream, 57
aspects. ca, 100, 100	close, 59
eepromSize	FilterOutputStream, 58

flush, 59	pos, 72
out, 61	write, 72
write, 59	InternalEepromOutputStream.cpp, 160
FilterOutputStream.cpp, 148, 149	ARDUINO_IO_INTERNAL_EEPROM_OUTP
ARDUINO_IO_FILTER_OUTPUT_STREAM_	UT_STREAM_CPP, 160
CPP, 148	InternalEepromOutputStream.h, 161, 162
FilterOutputStream.h, 149, 150	
flush	length
BufferedOutputStream, 14	RandomAccessByteArray, 82
FilterOutputStream, 59	RandomAccessExternalEeprom, 92
OutputStream, 74	
flushBuffer	mark
BufferedOutputStream, 14	BufferedInputStream, 10
	ByteArrayInputStream, 17
HardwareSerialInputStream, 61	ExternalEepromInputStream, 46
available, 62	FilterInputStream, 55
HardwareSerialInputStream, 62	InputStream, 66
read, 63	InternalEepromInputStream, 69
HardwareSerialInputStream.cpp, 151, 152	PgmspaceInputStream, 76
ARDUINO_IO_HARDWARE_SERIAL_INPUT↔	markSupported
_STREAM_CPP, 151	BufferedInputStream, 10
HardwareSerialInputStream.h, 152, 153	ByteArrayInputStream, 18
HardwareSerialOutputStream, 63	ExternalEepromInputStream, 47
HardwareSerialOutputStream, 64	FilterInputStream, 55
write, 64	InputStream, 66
HardwareSerialOutputStream.cpp, 153, 154	InternalEepromInputStream, 69
ARDUINO_IO_HARDWARE_SERIAL_OUTP↔	PgmspaceInputStream, 76
UT_STREAM_CPP, 154	marked
HardwareSerialOutputStream.h, 155, 156	BufferedInputStream, 11
	markpos
in	BufferedInputStream, 11
FilterInputStream, 57	ByteArrayInputStream, 18
InputStream, 65	ExternalEepromInputStream, 48
available, 66	InternalEepromInputStream, 70
close, 66	PgmspaceInputStream, 77
mark, 66	maxAvailableChunk
markSupported, 66	ExternalEepromInputStream, 48
read, 66, 67	InternalEepromInputStream, 70
reset, 67	
skip, 67	out
inputStream	FilterOutputStream, 61
DataInputStream, 34	OutputStream, 72
InputStream.cpp, 156, 157	close, 73
ARDUINO_IO_INPUT_STREAM_CPP, 156	flush, 74
InputStream.h, 157, 158	write, 74
InternalEepromInputStream, 67	outputStream
available, 69	DataOutputStream, 44
eepromSize, 70	OutputStream.cpp, 162, 163
InternalEepromInputStream, 69	ARDUINO_IO_OUTPUT_STREAM_CPP
mark, 69	, 162
markSupported, 69	OutputStream.h, 163, 164
markpos, 70	
maxAvailableChunk, 70	PgmspaceInputStream, 74
pos, 70	available, 76
read, 69	buf, 77
reset, 70	count, 77
InternalEepromInputStream.h, 158, 159	mark, 76
InternalEepromOutputStream, 70	markSupported, 76
InternalEepromOutputStream, 72	markpos, 77
momareopromodipatotroam, 12	manpoo, 77

PgmspaceInputStream, 76	writeDouble, 87
pos, 77	writeFloat, 87
read, 76	writeInt, 87
reset, 77	writeLong, 87
PgmspaceInputStream.cpp, 164, 165	writeUnsignedChar, 89
ARDUINO_IO_PGMSPACE_INPUT_STREA	writeUnsignedInt, 89
M_CPP, 165	writeUnsignedLong, 89
PgmspaceInputStream.h, 166, 167	writeWord, 89
PgmspaceSeekableInputStream, 77	RandomAccessByteArray.cpp, 172
~PgmspaceSeekableInputStream, 79	ARDUINO_IO_RANDOM_ACCESS_BYTE_A
PgmspaceSeekableInputStream, 79	RRAY_CPP, 172
seek, 79	RandomAccessByteArray.h, 174, 175
PgmspaceSeekableInputStream.cpp, 167, 168	RandomAccessExternalEeprom, 90
ARDUINO_IO_PGMSPACE_SEEKABLE_IN	close, 92
PUT_STREAM_CPP, 168	externalEeprom, 98
PgmspaceSeekableInputStream.h, 169	length, 92
pos	pos, 98
BufferedInputStream, 11	RandomAccessExternalEeprom, 91
ByteArrayInputStream, 18	readBoolean, 92
ByteArrayOutputStream, 22	readByte, 92
ExternalEepromInputStream, 48	readChar, 92
ExternalEepromOutputStream, 50	readDouble, 92
InternalEepromInputStream, 70	readFloat, 93
InternalEepromOutputStream, 72	readFully, 93
PgmspaceInputStream, 77	readint, 93
RandomAccessByteArray, 90	readLong, 93
RandomAccessExternalEeprom, 98	readUnsignedChar, 93
	readUnsignedInt, 94
RandomAccess, 79	_
RandomAccess.cpp, 170	readUnsignedLong, 94
ARDUINO_IO_RANDOM_ACCESS_CPP,	readWord, 94
170	seek, 94
RandomAccess.h, 171	skipBytes, 94
RandomAccessByteArray, 80	write, 95
buf, 89	writeBoolean, 95
close, 82	writeByte, 95
count, 89	writeBytes, 95
length, 82	writeChar, 96
pos, 90	writeDouble, 96
RandomAccessByteArray, 82	writeFloat, 96
readBoolean, 82	writeInt, 96
readByte, 82	writeLong, 96
readChar, 83	writeUnsignedChar, 96
readDouble, 83	writeUnsignedInt, 98
readFloat, 83	writeUnsignedLong, 98
readFully, 83	writeWord, 98
readInt, 83	RandomAccessExternalEeprom.cpp, 176, 177
readLong, 84	ARDUINO_IO_RANDOM_ACCESS_EXTER↔
readUnsignedChar, 84	NAL_EEPROM_CPP, 177
readUnsignedInt, 84	RandomAccessExternalEeprom.h, 179, 180
readUnsignedLong, 84	RandomAccessResource.cpp, 181, 182
readWord, 84	ARDUINO_IO_RANDOM_ACCESS_RESOU←
seek, 84	RCE_CPP, 181
skipBytes, 86	RandomAccessResource.h, 184
write, 86	read
writeBoolean, 86	BufferedInputStream, 10
writeByte, 86	ByteArrayInputStream, 18
writeBytes, 87	ExternalEepromInputStream, 47
writeChar, 87	FilterInputStream, 55, 56

HardwareSerialInputStream, 63	DataInputStream, 32
InputStream, 66, 67	RandomAccessByteArray, 84
InternalEepromInputStream, 69	RandomAccessExternalEeprom, 94
PgmspaceInputStream, 76	readWord
SoftwareSerialInputStream, 105	DataInput, 27
WireInputStream, 109	DataInputStream, 32
readBoolean	RandomAccessByteArray, 84
DataInput, 26	RandomAccessExternalEeprom, 94
DataInputStream, 30	realineBufferContent
RandomAccessByteArray, 82	BufferedInputStream, 10
RandomAccessExternalEeprom, 92	reset BufferedInputStream, 11
readByte	ByteArrayInputStream, 18
DataInput, 26	ByteArrayOutputStream, 20
DataInputStream, 30	ExternalEepromInputStream, 47
RandomAccessByteArray, 82	FilterInputStream, 56
RandomAccessExternalEeprom, 92 readChar	InputStream, 67
	InternalEepromInputStream, 70
DataInput, 26 DataInputStream, 31	PgmspaceInputStream, 77
RandomAccessByteArray, 83	ResourceInputStream.cpp, 185
RandomAccessExternalEeprom, 92	ARDUINO_IO_RESOURCE_INPUT_STREA
readDouble	M_CPP, 185
DataInput, 26	ResourceInputStream.h, 186
DataInputStream, 31	ResourceOutputStream.cpp, 186, 187
RandomAccessByteArray, 83	ARDUINO_IO_RESOURCE_OUTPUT_STRE←
RandomAccessExternalEeprom, 92	AM_CPP, 186
readFloat	ResourceOutputStream.h, 187
DataInput, 26	ResourceSeekableInputStream.cpp, 187, 188
DataInputStream, 31	ARDUINO_IO_RESOURCE_SEEKABLE_IN↔
RandomAccessByteArray, 83	PUT_STREAM_CPP, 187
RandomAccessExternalEeprom, 93	ResourceSeekableInputStream.h, 188
readFully	seek
DataInput, 26	ByteArraySeekableInputStream, 23
DataInputStream, 31	ExternalEepromSeekableInputStream, 52
RandomAccessByteArray, 83	PgmspaceSeekableInputStream, 79
RandomAccessExternalEeprom, 93	RandomAccessByteArray, 84
readInt	RandomAccessExternalEeprom, 94
DataInput, 27	Seekable, 99
DataInputStream, 31	Seekable, 98
RandomAccessByteArray, 83	seek, 99
RandomAccessExternalEeprom, 93	Seekable.cpp, 188, 189
readLong	ARDUINO_IO_SEEKABLE_CPP, 189
DataInput, 27	Seekable.h, 189, 190
DataInputStream, 32	SeekableInputStream, 99
RandomAccessByteArray, 84	SeekableInputStream.cpp, 190
RandomAccessExternalEeprom, 93	ARDUINO_IO_SEEKABLE_INPUT_STREAM
readUnsignedChar	_CPP, 190
DataInput, 27	SeekableInputStream.h, 191
DataInputStream, 32	SerialInputStream, 100
RandomAccessByteArray, 84	SerialInputStream.cpp, 192
RandomAccessExternalEeprom, 93	ARDUINO_IO_SERIAL_INPUT_STREAM_C
readUnsignedInt	PP, 192
DataInputStream 32	SerialInputStream.h, 193
DataInputStream, 32 RandomAccessByteArray, 84	SerialOutputStream, 102
RandomAccessByteArray, 84 RandomAccessExternalEeprom, 94	SerialOutputStream.cpp, 194ARDUINO_IO_SERIAL_OUTPUT_STREAM_ \-
readUnsignedLong	ARDUNO_IO_SERIAL_OUTFUT_STREAM_← CPP ,194
DataInput, 27	SerialOutputStream.h, 194, 195
Datamput, 21	Conarcarpatoticamini, 107, 100

size	RandomAccessExternalEeprom, 95
BufferedInputStream, 12	SoftwareSerialOutputStream, 107
BufferedOutputStream, 15	writeBoolean
ByteArrayOutputStream, 20	DataOutput, 35
skip	DataOutputStream, 42
BufferedInputStream, 11	RandomAccessByteArray, 86
FilterInputStream, 56	RandomAccessExternalEeprom, 95
InputStream, 67	writeByte
skipBytes	DataOutput, 35
DataInput, 27	DataOutputStream, 42
DataInputStream, 32	RandomAccessByteArray, 86
RandomAccessByteArray, 86	RandomAccessExternalEeprom, 95
RandomAccessExternalEeprom, 94	writeBytes
softwareSerial	DataOutput, 35
SoftwareSerialInputStream, 105	DataOutputStream, 42
SoftwareSerialOutputStream, 107	RandomAccessByteArray, 87
SoftwareSerialInputStream, 103	RandomAccessExternalEeprom, 95
available, 105	writeChar
read, 105	DataOutput, 37
softwareSerial, 105	DataOutputStream, 43
SoftwareSerialInputStream, 104	RandomAccessByteArray, 87
SoftwareSerialInputStream.cpp, 195, 196	RandomAccessExternalEeprom, 96
ARDUINO IO SOFTWARE SERIAL INPUT↔	writeDouble
STREAM CPP , 196	DataOutput, 37
SoftwareSerialInputStream.h, 197, 198	DataOutput, 37 DataOutputStream, 43
SoftwareSerialOutputStream, 105	RandomAccessByteArray, 87
softwareSerial, 107	
SoftwareSerialOutputStream, 107	RandomAccessExternalEeprom, 96 writeFloat
write, 107	
SoftwareSerialOutputStream.cpp, 198, 199	DataOutput, 37
ARDUINO_IO_SOFTWARE_SERIAL_OUTP	DataOutputStream, 43
UT_STREAM_CPP, 198	RandomAccessByteArray, 87
SoftwareSerialOutputStream.h, 199, 200	RandomAccessExternalEeprom, 96
, , ,	writeInt
toByteArray	DataOutput, 37
ByteArrayOutputStream, 20	DataOutputStream, 43
	RandomAccessByteArray, 87
WireInputStream, 107	RandomAccessExternalEeprom, 96
address, 109	writeLong
available, 109	DataOutput, 37
read, 109	DataOutputStream, 43
WireInputStream, 109	RandomAccessByteArray, 87
WireInputStream.cpp, 200, 201	RandomAccessExternalEeprom, 96
ARDUINO_IO_WIRE_INPUT_STREAM_CPP	writeUnsignedChar
, 201	DataOutput, 37
WireInputStream.h, 202, 203	DataOutputStream, 43
WireOutputStream.cpp, 203	RandomAccessByteArray, 89
WireOutputStream.h, 203	RandomAccessExternalEeprom, 96
write	writeUnsignedInt
BufferedOutputStream, 14, 15	DataOutput, 37
ByteArrayOutputStream, 20	DataOutputStream, 44
DataOutput, 35	RandomAccessByteArray, 89
DataOutputStream, 42	RandomAccessExternalEeprom, 98
ExternalEepromOutputStream, 50	writeUnsignedLong
FilterOutputStream, 59	DataOutput, 39
HardwareSerialOutputStream, 64	DataOutputStream, 44
InternalEepromOutputStream, 72	RandomAccessByteArray, 89
OutputStream, 74	RandomAccessExternalEeprom, 98
RandomAccessByteArray, 86	writeWord

DataOutput, 39
DataOutputStream, 44
RandomAccessByteArray, 89
RandomAccessExternalEeprom, 98