TITLE

AUTHOR
Version
CREATEDATE

Table of Contents

Table of contents

Hierarchical Index

Class Hierarchy

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

Class Index

Class List

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

Class Documentation

mbed::BusIn Class Reference

`#include <BusIn.h>`

Public Member Functions

- **BusIn** (PinName p0, PinName p1=NC, PinName p2=NC, PinName p3=NC, PinName p4=NC, PinName p5=NC, PinName p6=NC, PinName p7=NC, PinName p8=NC, PinName p9=NC, PinName p10=NC, PinName p11=NC, PinName p12=NC, PinName p13=NC, PinName p14=NC, PinName p15=NC)
- **BusIn** (PinName pins[16])
- int **read** ()
- **operator int** ()

Protected Attributes

- **DigitalIn** * **_pin** [16]

Detailed Description

A digital input bus, used for reading the state of a collection of pins

---

Constructor & Destructor Documentation

mbed::BusIn::BusIn (PinName *p0*, PinName *p1* = NC, PinName *p2* = NC, PinName *p3* = NC, PinName *p4* = NC, PinName *p5* = NC, PinName *p6* = NC, PinName *p7* = NC, PinName *p8* = NC, PinName *p9* = NC, PinName *p10* = NC, PinName *p11* = NC, PinName *p12* = NC, PinName *p13* = NC, PinName *p14* = NC, PinName *p15* = NC)

Create an **BusIn**, connected to the specified pins

Parameters:

<*n*>	**DigitalIn** pin to connect to bus bit <n> (p5-p30, NC)

Note:

It is only required to specify as many pin variables as is required for the bus; the rest will default to NC (not connected)

Member Function Documentation

mbed::BusIn::operator int ()

A shorthand for **read()**

int mbed::BusIn::read ()

Read the value of the input bus

Returns:

An integer with each bit corresponding to the value read from the associated **DigitalIn** pin

---

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

- BusIn.h

mbed::BusInOut Class Reference

`#include <BusInOut.h>`

Public Member Functions

- **BusInOut** (PinName p0, PinName p1=NC, PinName p2=NC, PinName p3=NC, PinName p4=NC, PinName p5=NC, PinName p6=NC, PinName p7=NC, PinName p8=NC, PinName p9=NC, PinName p10=NC, PinName p11=NC, PinName p12=NC, PinName p13=NC, PinName p14=NC, PinName p15=NC)
- **BusInOut** (PinName pins[16])
- void **write** (int value)
- int **read** ()
- void **output** ()
- void **input** ()
- void **mode** (PinMode pull)
- **BusInOut** & **operator=** (int v)
- **BusInOut** & **operator=** (**BusInOut** &rhs)
- **operator int** ()

Protected Attributes

- **DigitalInOut** * **_pin** [16]

Detailed Description

A digital input output bus, used for setting the state of a collection of pins

---

Constructor & Destructor Documentation

mbed::BusInOut::BusInOut (PinName *p0*, PinName *p1* = NC, PinName *p2* = NC, PinName *p3* = NC, PinName *p4* = NC, PinName *p5* = NC, PinName *p6* = NC, PinName *p7* = NC, PinName *p8* = NC, PinName *p9* = NC, PinName *p10* = NC, PinName *p11* = NC, PinName *p12* = NC, PinName *p13* = NC, PinName *p14* = NC, PinName *p15* = NC)

Create an **BusInOut**, connected to the specified pins

Parameters:

p<n>	**DigitalInOut** pin to connect to bus bit p<n> (p5-p30, NC)

Note:

It is only required to specify as many pin variables as is required for the bus; the rest will default to NC (not connected)

Member Function Documentation

void mbed::BusInOut::input ()

Set as an input

void mbed::BusInOut::mode (PinMode *pull*)

Set the input pin mode

Parameters:

mode	PullUp, PullDown, PullNone

mbed::BusInOut::operator int ()

A shorthand for **read()**

BusInOut& mbed::BusInOut::operator= (int *v*)

A shorthand for **write()**

void mbed::BusInOut::output ()

Set as an output

int mbed::BusInOut::read ()

Read the value currently output on the bus

Returns:

An integer with each bit corresponding to associated **DigitalInOut** pin setting

void mbed::BusInOut::write (int *value*)

Write the value to the output bus

Parameters:

value	An integer specifying a bit to write for every corresponding **DigitalInOut** pin

---

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

- BusInOut.h

mbed::BusOut Class Reference

`#include <BusOut.h>`

Public Member Functions

- **BusOut** (PinName p0, PinName p1=NC, PinName p2=NC, PinName p3=NC, PinName p4=NC, PinName p5=NC, PinName p6=NC, PinName p7=NC, PinName p8=NC, PinName p9=NC, PinName p10=NC, PinName p11=NC, PinName p12=NC, PinName p13=NC, PinName p14=NC, PinName p15=NC)
- **BusOut** (PinName pins[16])
- void **write** (int value)
- int **read** ()
- **BusOut** & **operator=** (int v)
- **BusOut** & **operator=** (**BusOut** &rhs)
- **operator int** ()

Protected Attributes

- **DigitalOut** * **_pin** [16]

Detailed Description

A digital output bus, used for setting the state of a collection of pins

---

Constructor & Destructor Documentation

mbed::BusOut::BusOut (PinName *p0*, PinName *p1* = NC, PinName *p2* = NC, PinName *p3* = NC, PinName *p4* = NC, PinName *p5* = NC, PinName *p6* = NC, PinName *p7* = NC, PinName *p8* = NC, PinName *p9* = NC, PinName *p10* = NC, PinName *p11* = NC, PinName *p12* = NC, PinName *p13* = NC, PinName *p14* = NC, PinName *p15* = NC)

Create an **BusOut**, connected to the specified pins

Parameters:

p<n>	**DigitalOut** pin to connect to bus bit <n> (p5-p30, NC)

Note:

It is only required to specify as many pin variables as is required for the bus; the rest will default to NC (not connected)

Member Function Documentation

mbed::BusOut::operator int ()

A shorthand for **read()**

BusOut& mbed::BusOut::operator= (int *v*)

A shorthand for **write()**

int mbed::BusOut::read ()

Read the value currently output on the bus

Returns:

An integer with each bit corresponding to associated **DigitalOut** pin setting

void mbed::BusOut::write (int *value*)

Write the value to the output bus

Parameters:

value	An integer specifying a bit to write for every corresponding **DigitalOut** pin

---

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

- BusOut.h

mbed::CallChain Class Reference

Public Member Functions

- **CallChain** (int **size**=4)
- **pFunctionPointer_t add** (void(*function)(void))
- template<typename T > **pFunctionPointer_t add** (T *tptr, void(T::*mptr)(void))
- **pFunctionPointer_t add_front** (void(*function)(void))
- template<typename T > **pFunctionPointer_t add_front** (T *tptr, void(T::*mptr)(void))
- int **size** () const
- **pFunctionPointer_t get** (int i) const
- int **find** (**pFunctionPointer_t** f) const
- void **clear** ()
- bool **remove** (**pFunctionPointer_t** f)
- void **call** ()

Constructor & Destructor Documentation

mbed::CallChain::CallChain (int *size* = 4)

Create an empty chain

Parameters:

size	(optional) Initial size of the chain

Member Function Documentation

pFunctionPointer_t mbed::CallChain::add (void(*)(void) *function*)

Add a function at the end of the chain

Parameters:

function	A pointer to a void function

Returns:

The function object created for 'function'

template<typename T > pFunctionPointer_t mbed::CallChain::add (T * *tptr*, void(T::*)(void) *mptr*) `[inline]`

Add a function at the end of the chain

Parameters:

tptr	pointer to the object to call the member function on
mptr	pointer to the member function to be called

Returns:

The function object created for 'tptr' and 'mptr'

pFunctionPointer_t mbed::CallChain::add_front (void(*)(void) *function*)

Add a function at the beginning of the chain

Parameters:

function	A pointer to a void function

Returns:

The function object created for 'function'

template<typename T > pFunctionPointer_t mbed::CallChain::add_front (T * *tptr*, void(T::*)(void) *mptr*) [inline]

Add a function at the beginning of the chain

Parameters:

tptr	pointer to the object to call the member function on
mptr	pointer to the member function to be called

Returns:

The function object created for 'tptr' and 'mptr'

void mbed::CallChain::call ()

Call all the functions in the chain in sequence

void mbed::CallChain::clear ()

Clear the call chain (remove all functions in the chain).

int mbed::CallChain::find (pFunctionPointer_t *f*) const

Look for a function object in the call chain

Parameters:

f	the function object to search

Returns:

The index of the function object if found, -1 otherwise.

pFunctionPointer_t mbed::CallChain::get (int *i*) const

Get a function object from the chain

Parameters:

i	function object index

Returns:

The function object at position 'i' in the chain

bool mbed::CallChain::remove (pFunctionPointer_t *f*)

Remove a function object from the chain

- f the function object to remove

Returns:

true if the function object was found and removed, false otherwise.

int mbed::CallChain::size () const

Get the number of functions in the chain

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

- CallChain.h

mbed::DigitalIn Class Reference

`#include <DigitalIn.h>`

Public Member Functions

- **DigitalIn** (PinName pin)
- int **read** ()
- void **mode** (PinMode pull)
- **operator int** ()

Protected Attributes

- gpio_t **gpio**

Detailed Description

A digital input, used for reading the state of a pin

Example:

```
* // Flash an LED while a DigitalIn is true
*
* #include "mbed.h"
*
* DigitalIn enable(p5);
* DigitalOut led(LED1);
*
* int main() {
*     while(1) {
*         if(enable) {
*             led = !led;
*         }
*         wait(0.25);
*     }
* }
*
```

Constructor & Destructor Documentation

mbed::DigitalIn::DigitalIn (PinName *pin*) `[inline]`

Create a **DigitalIn** connected to the specified pin

Parameters:

pin	**DigitalIn** pin to connect to
name	(optional) A string to identify the object

Member Function Documentation

void mbed::DigitalIn::mode (PinMode *pull*) `[inline]`

Set the input pin mode

Parameters:

mode	PullUp, PullDown, PullNone, OpenDrain

mbed::DigitalIn::operator int () `[inline]`

An operator shorthand for **read()**

int mbed::DigitalIn::read () `[inline]`

Read the input, represented as 0 or 1 (int)

Returns:

An integer representing the state of the input pin, 0 for logical 0, 1 for logical 1

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

- DigitalIn.h

mbed::DigitalInOut Class Reference

`#include <DigitalInOut.h>`

Public Member Functions

- **DigitalInOut** (PinName pin)
- void **write** (int value)
- int **read** ()
- void **output** ()
- void **input** ()
- void **mode** (PinMode pull)
- **DigitalInOut** & **operator=** (int value)
- **DigitalInOut** & **operator=** (**DigitalInOut** &rhs)
- **operator int** ()

Protected Attributes

- gpio_t **gpio**

Detailed Description

A digital input/output, used for setting or reading a bi-directional pin

---

Constructor & Destructor Documentation

mbed::DigitalInOut::DigitalInOut (PinName *pin*) `[inline]`

Create a **DigitalInOut** connected to the specified pin

Parameters:

pin	**DigitalInOut** pin to connect to

Member Function Documentation

void mbed::DigitalInOut::input () `[inline]`

Set as an input

void mbed::DigitalInOut::mode (PinMode *pull*) `[inline]`

Set the input pin mode

Parameters:

mode	PullUp, PullDown, PullNone, OpenDrain

mbed::DigitalInOut::operator int () `[inline]`

A shorthand for **read()**

DigitalInOut& mbed::DigitalInOut::operator= (int *value*) `[inline]`

A shorthand for **write()**

void mbed::DigitalInOut::output () `[inline]`

Set as an output

int mbed::DigitalInOut::read () `[inline]`

Return the output setting, represented as 0 or 1 (int)

Returns:

an integer representing the output setting of the pin if it is an output, or read the input if set as an input

void mbed::DigitalInOut::write (int *value*) `[inline]`

Set the output, specified as 0 or 1 (int)

Parameters:

value	An integer specifying the pin output value, 0 for logical 0, 1 (or any other non-zero value) for logical 1

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

- DigitalInOut.h

mbed::DigitalOut Class Reference

`#include <DigitalOut.h>`

Public Member Functions

- **DigitalOut** (PinName pin)
- void **write** (int value)
- int **read** ()
- **DigitalOut** & **operator=** (int value)
- **DigitalOut** & **operator=** (**DigitalOut** &rhs)
- **operator int** ()

Protected Attributes

- gpio_t **gpio**

Detailed Description

A digital output, used for setting the state of a pin

Example:

```
* // Toggle a LED
* #include "mbed.h"
*
* DigitalOut led(LED1);
*
* int main() {
*     while(1) {
*         led = !led;
*         wait(0.2);
*     }
* }
*
```

Constructor & Destructor Documentation

mbed::DigitalOut::DigitalOut (PinName *pin*) `[inline]`

Create a **DigitalOut** connected to the specified pin

Parameters:

pin	**DigitalOut** pin to connect to

Member Function Documentation

mbed::DigitalOut::operator int () `[inline]`

A shorthand for **read()**

DigitalOut& mbed::DigitalOut::operator= (int *value*) `[inline]`

A shorthand for **write()**

int mbed::DigitalOut::read () `[inline]`

Return the output setting, represented as 0 or 1 (int)

Returns:

an integer representing the output setting of the pin, 0 for logical 0, 1 for logical 1

void mbed::DigitalOut::write (int *value*) `[inline]`

Set the output, specified as 0 or 1 (int)

Parameters:

value	An integer specifying the pin output value, 0 for logical 0, 1 (or any other non-zero value) for logical 1

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

- DigitalOut.h

dirent Struct Reference

Public Attributes

- char **d_name** [NAME_MAX+1]

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

- DirHandle.h

mbed::DirHandle Class Reference

`#include <DirHandle.h>`

Public Member Functions

- virtual int **closedir** ()=0
- virtual struct **dirent** * **readdir** ()=0
- virtual void **rewinddir** ()=0
- virtual off_t **telldir** ()
- virtual void **seekdir** (off_t location)

Detailed Description

Represents a directory stream. Objects of this type are returned by a **FileSystemLike**'s opendir method. Implementations must define at least closedir, readdir and rewinddir.

If a **FileSystemLike** class defines the opendir method, then the directories of an object of that type can be accessed by DIR *d = opendir("/example/directory") (or opendir("/example") to open the root of the filesystem), and then using readdir(d) etc.

The root directory is considered to contain all **FileLike** and **FileSystemLike** objects, so the DIR* returned by opendir("/") will reflect this.

Member Function Documentation

virtual int mbed::DirHandle::closedir () `[pure virtual]`

Closes the directory.

Returns:

0 on success, -1 on error.

virtual struct dirent* mbed::DirHandle::readdir () `[pure virtual]`

Return the directory entry at the current position, and advances the position to the next entry.

Returns:

A pointer to a dirent structure representing the directory entry at the current position, or NULL on reaching end of directory or error.

virtual void mbed::DirHandle::rewinddir () `[pure virtual]`

Resets the position to the beginning of the directory.

virtual void mbed::DirHandle::seekdir (off_t *location*) `[inline]`, `[virtual]`

Sets the position of the **DirHandle**.

Parameters:

location	The location to seek to. Must be a value returned by telldir.

virtual off_t mbed::DirHandle::telldir () `[inline], [virtual]`

Returns the current position of the **DirHandle**.

Returns:

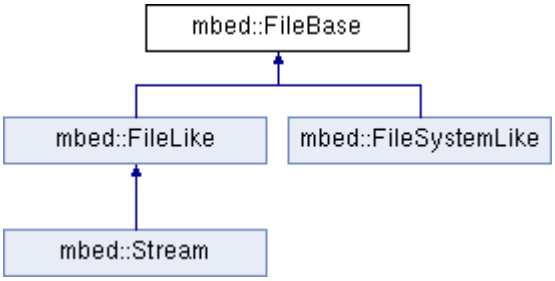
the current position, -1 on error.

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

- DirHandle.h

mbed::FileBase Class Reference

Inheritance diagram for mbed::FileBase:



Public Member Functions

- **FileBase** (const char *name, PathType t)
- const char * **getName** (void)
- PathType **getPathType** (void)

Static Public Member Functions

- static **FileBase** * **lookup** (const char *name, unsigned int len)
- static **FileBase** * **get** (int n)

Protected Attributes

- **FileBase** * **_next**
- const char * **_name**
- PathType **_path_type**

Static Protected Attributes

- static **FileBase** * **_head**

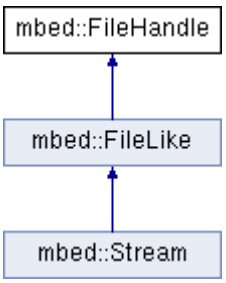
The documentation for this class was generated from the following file:

- FileBase.h

mbed::FileHandle Class Reference

`#include <FileHandle.h>`

Inheritance diagram for mbed::FileHandle:



Public Member Functions

- virtual ssize_t **write** (const void *buffer, size_t length)=0
- virtual int **close** ()=0
- virtual ssize_t **read** (void *buffer, size_t length)=0
- virtual int **isatty** ()=0
- virtual off_t **lseek** (off_t offset, int whence)=0
- virtual int **fsync** ()=0
- virtual off_t **flen** ()

Detailed Description

An OO equivalent of the internal FILEHANDLE variable and associated *sys* * functions.

FileHandle is an abstract class, needing at least sys_write and sys_read to be implmented for a simple interactive device.

No one ever directly tals to/instanciates a **FileHandle** - it gets created by FileSystem, and wrapped up by stdio.

Member Function Documentation

virtual int mbed::FileHandle::close () `[pure virtual]`

Close the file

Returns:

Zero on success, -1 on error.

Implemented in **mbed::Stream** (*p.35*).

virtual int mbed::FileHandle::fsync () `[pure virtual]`

Flush any buffers associated with the **FileHandle**, ensuring it is up to date on disk

Returns:

0 on success or un-needed, -1 on error

Implemented in **mbed::Stream** (*p.36*).

virtual int mbed::FileHandle::isatty () [pure virtual]

Check if the handle is for a interactive terminal device. If so, line buffered behaviour is used by default

Returns:

1 if it is a terminal, 0 otherwise

Implemented in **mbed::Stream** (*p.36*).

virtual off_t mbed::FileHandle::lseek (off_t *offset*, int *whence*) [pure virtual]

Move the file position to a given offset from a given location.

Parameters:

offset	The offset from whence to move to
whence	SEEK_SET for the start of the file, SEEK_CUR for the current file position, or SEEK_END for the end of the file.

Returns:

new file position on success, -1 on failure or unsupported

Implemented in **mbed::Stream** (*p.36*).

virtual ssize_t mbed::FileHandle::read (void * *buffer*, size_t *length*) [pure virtual]

Function read Reads the contents of the file into a buffer

Parameters:

buffer	the buffer to read in to
length	the number of characters to read

Returns:

The number of characters read (zero at end of file) on success, -1 on error.

Implemented in **mbed::Stream** (*p.36*).

virtual ssize_t mbed::FileHandle::write (const void * *buffer*, size_t *length*) [pure virtual]

Write the contents of a buffer to the file

Parameters:

buffer	the buffer to write from
length	the number of characters to write

Returns:

The number of characters written (possibly 0) on success, -1 on error.

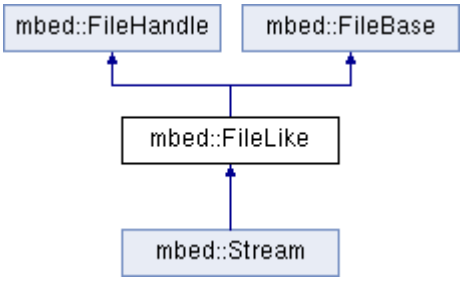
Implemented in **mbed::Stream** (*p.36*).

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

- FileHandle.h

mbed::FileLike Class Reference

Inheritance diagram for mbed::FileLike:



Public Member Functions

- **FileLike** (const char *name)

Additional Inherited Members

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

- FileLike.h

mbed::FilePath Class Reference

Public Member Functions

- **FilePath** (const char *file_path)
- const char * **fileName** (void)
- bool **isFileSystem** (void)
- **FileSystemLike** * **fileSystem** (void)
- bool **isFile** (void)
- **FileLike** * **file** (void)
- bool **exists** (void)

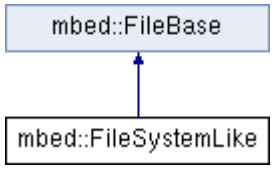
The documentation for this class was generated from the following file:

- FilePath.h

mbed::FileSystemLike Class Reference

`#include <FileSystemLike.h>`

Inheritance diagram for mbed::FileSystemLike:



Public Member Functions

- **FileSystemLike** (const char *name)
- virtual **FileHandle** * **open** (const char *filename, int flags)=0
- virtual int **remove** (const char *filename)
- virtual int **rename** (const char *oldname, const char *newname)
- virtual **DirHandle** * **opendir** (const char *name)
- virtual int **mkdir** (const char *name, mode_t mode)

Static Public Member Functions

- static **DirHandle** * **opendir** ()

Friends

- class **BaseDirHandle**

Additional Inherited Members

Detailed Description

A filesystem-like object is one that can be used to open files though it by fopen("/name/filename", mode)

Implementations must define at least open (the default definitions of the rest of the functions just return error values).

Constructor & Destructor Documentation

mbed::FileSystemLike::FileSystemLike (const char * *name*)

FileSystemLike constructor

Parameters:

name	The name to use for the filesystem.

Member Function Documentation

virtual int mbed::FileSystemLike::mkdir (const char * *name*, mode_t *mode*) `[inline], [virtual]`

Creates a directory in the filesystem.

Parameters:

name	The name of the directory to create.
mode	The permissions to create the directory with.

Returns:

0 on success, -1 on failure.

virtual FileHandle* mbed::FileSystemLike::open (const char * *filename*, int *flags*) [pure virtual]

Opens a file from the filesystem

Parameters:

filename	The name of the file to open.
flags	One of O_RDONLY, O_WRONLY, or O_RDWR, OR'd with zero or more of O_CREAT, O_TRUNC, or O_APPEND.

Returns:

A pointer to a **FileHandle** object representing the file on success, or NULL on failure.

virtual DirHandle* mbed::FileSystemLike::opendir (const char * *name*) [inline], [virtual]

Opens a directory in the filesystem and returns a **DirHandle** representing the directory stream.

Parameters:

name	The name of the directory to open.

Returns:

A **DirHandle** representing the directory stream, or NULL on failure.

virtual int mbed::FileSystemLike::remove (const char * *filename*) [inline], [virtual]

Remove a file from the filesystem.

Parameters:

filename	the name of the file to remove.
returns	0 on success, -1 on failure.

virtual int mbed::FileSystemLike::rename (const char * *oldname*, const char * *newname*) [inline], [virtual]

Rename a file in the filesystem.

Parameters:

oldname	the name of the file to rename.
newname	the name to rename it to.

Returns:

0 on success, -1 on failure.

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

- FileSystemLike.h

mbed::FunctionPointer Class Reference

`#include <FunctionPointer.h>`

Public Member Functions

- **FunctionPointer** (void(*function)(void)=0)
- template<typename T > **FunctionPointer** (T *object, void(T::*member)(void))
- void **attach** (void(*function)(void)=0)
- template<typename T > void **attach** (T *object, void(T::*member)(void))
- void **call** ()
- pvoidf_t **get_function** () const

Detailed Description

A class for storing and calling a pointer to a static or member void function

---

Constructor & Destructor Documentation

mbed::FunctionPointer::FunctionPointer (void(*)(void) *function* = 0)

Create a **FunctionPointer**, attaching a static function

Parameters:

function	The void static function to attach (default is none)

template<typename T > mbed::FunctionPointer::FunctionPointer (T * *object*, void(T::*)(void) *member*) [inline]

Create a **FunctionPointer**, attaching a member function

Parameters:

object	The object pointer to invoke the member function on (i.e. the this pointer)
function	The address of the void member function to attach

Member Function Documentation

void mbed::FunctionPointer::attach (void(*)(void) *function* = 0)

Attach a static function

Parameters:

function	The void static function to attach (default is none)

template<typename T > void mbed::FunctionPointer::attach (T * *object*, void(T::*)(void) *member*) [inline]

Attach a member function

Parameters:

object	The object pointer to invoke the member function on (i.e. the this pointer)

function	The address of the void member function to attach

void mbed::FunctionPointer::call ()

Call the attached static or member function

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

- FunctionPointer.h

mbed::InterruptManager Class Reference

`#include <InterruptManager.h>`

Public Member Functions

- **pFunctionPointer_t add_handler** (void(*function)(void), IRQn_Type irq)
- **pFunctionPointer_t add_handler_front** (void(*function)(void), IRQn_Type irq)
- template<typename T > **pFunctionPointer_t add_handler** (T *tptr, void(T::*mptr)(void), IRQn_Type irq)
- template<typename T > **pFunctionPointer_t add_handler_front** (T *tptr, void(T::*mptr)(void), IRQn_Type irq)
- bool **remove_handler** (**pFunctionPointer_t** handler, IRQn_Type irq)

Static Public Member Functions

- static **InterruptManager** * **get** ()
- static void **destroy** ()

Detailed Description

Use this singleton if you need to chain interrupt handlers.

Example (for LPC1768):

```
* #include "InterruptManager.h"
* #include "mbed.h"
*
* Ticker flipper;
* DigitalOut led1(LED1);
* DigitalOut led2(LED2);
*
* void flip(void) {
*     led1 = !led1;
* }
*
* void handler(void) {
*     led2 = !led1;
* }
*
* int main() {
*     led1 = led2 = 0;
*     flipper.attach(&flip, 1.0);
*     InterruptManager::get()->add handler(handler, TIMER3 IRQn);
* }
*
```

Member Function Documentation

pFunctionPointer_t mbed::InterruptManager::add_handler (void(*)(void) *function*, IRQn_Type *irq*) [inline]

Add a handler for an interrupt at the end of the handler list

Parameters:

function	the handler to add
irq	interrupt number

Returns:

The function object created for 'function'

template<typename T > pFunctionPointer_t mbed::InterruptManager::add_handler (T * *tptr*, void(T::*)(void) *mptr*, IRQn_Type *irq*) [inline]

Add a handler for an interrupt at the end of the handler list

Parameters:

tptr	pointer to the object that has the handler function
mptr	pointer to the actual handler function
irq	interrupt number

Returns:

The function object created for 'tptr' and 'mptr'

pFunctionPointer_t mbed::InterruptManager::add_handler_front (void(*)(void) *function*, IRQn_Type *irq*) [inline]

Add a handler for an interrupt at the beginning of the handler list

Parameters:

function	the handler to add
irq	interrupt number

Returns:

The function object created for 'function'

template<typename T > pFunctionPointer_t mbed::InterruptManager::add_handler_front (T * *tptr*, void(T::*)(void) *mptr*, IRQn_Type *irq*) [inline]

Add a handler for an interrupt at the beginning of the handler list

Parameters:

tptr	pointer to the object that has the handler function
mptr	pointer to the actual handler function
irq	interrupt number

Returns:

The function object created for 'tptr' and 'mptr'

static void mbed::InterruptManager::destroy () [static]

Destroy the current instance of the interrupt manager

static InterruptManager* mbed::InterruptManager::get () [static]

Return the only instance of this class

bool mbed::InterruptManager::remove_handler (pFunctionPointer_t *handler*, IRQn_Type *irq*)

Remove a handler from an interrupt

Parameters:

handler	the function object for the handler to remove
irq	the interrupt number

Returns:

true if the handler was found and removed, false otherwise

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

- InterruptManager.h

PinMap Struct Reference

Public Attributes

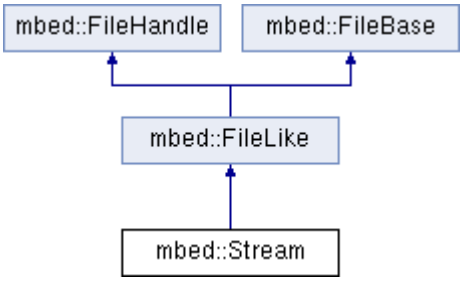
- PinName **pin**
- int **peripheral**
- int **function**

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

- pinmap.h

mbed::Stream Class Reference

Inheritance diagram for mbed::Stream:



Public Member Functions

- **Stream** (const char *name=NULL)
- int **putc** (int c)
- int **puts** (const char *s)
- int **getc** ()
- char * **gets** (char *s, int size)
- int **printf** (const char *format,...)
- int **scanf** (const char *format,...)
- **operator std::FILE *** ()

Protected Member Functions

- virtual int **close** ()
- virtual ssize_t **write** (const void *buffer, size_t length)
- virtual ssize_t **read** (void *buffer, size_t length)
- virtual off_t **lseek** (off_t offset, int whence)
- virtual int **isatty** ()
- virtual int **fsync** ()
- virtual off_t **flen** ()
- virtual int **_putc** (int c)=0
- virtual int **_getc** ()=0

Protected Attributes

- std::FILE * **_file**

Additional Inherited Members

Member Function Documentation

virtual int mbed::Stream::close () `[protected], [virtual]`

Close the file

Returns:

Zero on success, -1 on error.

Implements **mbed::FileHandle** (*p.23*).

virtual int mbed::Stream::fsync () [protected], [virtual]

Flush any buffers associated with the **FileHandle**, ensuring it is up to date on disk

Returns:

0 on success or un-needed, -1 on error

Implements **mbed::FileHandle** (*p.23*).

virtual int mbed::Stream::isatty () [protected], [virtual]

Check if the handle is for a interactive terminal device. If so, line buffered behaviour is used by default

Returns:

1 if it is a terminal, 0 otherwise

Implements **mbed::FileHandle** (*p.24*).

virtual off_t mbed::Stream::lseek (off_t *offset*, int *whence*) [protected], [virtual]

Move the file position to a given offset from a given location.

Parameters:

offset	The offset from whence to move to
whence	SEEK_SET for the start of the file, SEEK_CUR for the current file position, or SEEK_END for the end of the file.

Returns:

new file position on success, -1 on failure or unsupported

Implements **mbed::FileHandle** (*p.24*).

virtual ssize_t mbed::Stream::read (void * *buffer*, size_t *length*) [protected], [virtual]

Function read Reads the contents of the file into a buffer

Parameters:

buffer	the buffer to read in to
length	the number of characters to read

Returns:

The number of characters read (zero at end of file) on success, -1 on error.

Implements **mbed::FileHandle** (*p.24*).

virtual ssize_t mbed::Stream::write (const void * *buffer*, size_t *length*) [protected], [virtual]

Write the contents of a buffer to the file

Parameters:

buffer	the buffer to write from
length	the number of characters to write

Returns:

The number of characters written (possibly 0) on success, -1 on error.

Implements **mbed::FileHandle** (*p.24*).

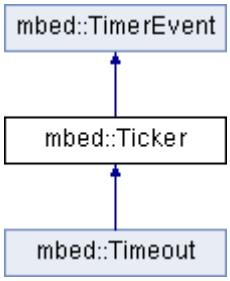
The documentation for this class was generated from the following file:

- Stream.h

mbed::Ticker Class Reference

`#include <Ticker.h>`

Inheritance diagram for mbed::Ticker:



Public Member Functions

- **pFunctionPointer_t attach** (void(*fptr)(void), float t)
- **pFunctionPointer_t add_function** (void(*fptr)(void))
- **pFunctionPointer_t add_function_front** (void(*fptr)(void))
- template<typename T > **pFunctionPointer_t attach** (T *tptr, void(T::*mptr)(void), float t)
- template<typename T > **pFunctionPointer_t add_function** (T *tptr, void(T::*mptr)(void))
- template<typename T > **pFunctionPointer_t add_function_front** (T *tptr, void(T::*mptr)(void))
- **pFunctionPointer_t attach_us** (void(*fptr)(void), unsigned int t)
- template<typename T > **pFunctionPointer_t attach_us** (T *tptr, void(T::*mptr)(void), unsigned int t)
- void **detach** ()
- bool **remove_function** (**pFunctionPointer_t** pf)

Protected Member Functions

- void **setup** (unsigned int t)
- **pFunctionPointer_t add_function_helper** (void(*fptr)(void), bool front=false)
- virtual void **handler** ()
- template<typename T > **pFunctionPointer_t add_function_helper** (T *tptr, void(T::*mptr)(void), bool front=false)

Protected Attributes

- unsigned int **_delay**
- **CallChain _chain**

Additional Inherited Members

Detailed Description

A **Ticker** is used to call a function at a recurring interval

You can use as many seperate **Ticker** objects as you require.

Example:

```
* // Toggle the blinking led after 5 seconds
*
* #include "mbed.h"
*
* Ticker timer;
* DigitalOut led1(LED1);
```

```
* DigitalOut led2(LED2);
*
* int flip = 0;
*
* void attime() {
*     flip = !flip;
* }
*
* int main() {
*     timer.attach(&attime, 5);
*     while(1) {
*         if(flip == 0) {
*             led1 = !led1;
*         } else {
*             led2 = !led2;
*         }
*         wait(0.2);
*     }
* }
*
```

Member Function Documentation

pFunctionPointer_t mbed::Ticker::add_function (void(*)(void) *fptr*) [inline]

Add a function to be called by the **Ticker** at the end of the call chain

Parameters:

fptr	the function to add

Returns:

The function object created for 'fptr'

template<typename T > pFunctionPointer_t mbed::Ticker::add_function (T * *tptr*, void(T::*)(void) *mptr*) [inline]

Add a function to be called by the **Ticker** at the end of the call chain

Parameters:

tptr	pointer to the object to call the member function on
mptr	pointer to the member function to be called

Returns:

The function object created for 'tptr' and 'mptr'

pFunctionPointer_t mbed::Ticker::add_function_front (void(*)(void) *fptr*) [inline]

Add a function to be called by the **Ticker** at the beginning of the call chain

Parameters:

fptr	the function to add

Returns:

The function object created for 'fptr'

template<typename T > pFunctionPointer_t mbed::Ticker::add_function_front (T * *tptr*, void(T::*)(void) *mptr*) [inline]

Add a function to be called by the **Ticker** at the beginning of the call chain

Parameters:

tptr	pointer to the object to call the member function on
mptr	pointer to the member function to be called

Returns:

The function object created for 'tptr' and 'mptr'

pFunctionPointer_t mbed::Ticker::attach (void(*)(void) *fptr*, float *t*) `[inline]`

Attach a function to be called by the **Ticker**, specifiying the interval in seconds

Parameters:

fptr	pointer to the function to be called
t	the time between calls in seconds

Returns:

The function object created for 'fptr'

template<typename T > pFunctionPointer_t mbed::Ticker::attach (T * *tptr*, void(T::*)(void) *mptr*, float *t*) `[inline]`

Attach a member function to be called by the **Ticker**, specifiying the interval in seconds

Parameters:

tptr	pointer to the object to call the member function on
mptr	pointer to the member function to be called
t	the time between calls in seconds

Returns:

The function object created for 'tptr' and 'mptr'

pFunctionPointer_t mbed::Ticker::attach_us (void(*)(void) *fptr*, unsigned int *t*) `[inline]`

Attach a function to be called by the **Ticker**, specifiying the interval in micro-seconds

Parameters:

fptr	pointer to the function to be called
t	the time between calls in micro-seconds

Returns:

The function object created for 'fptr'

template<typename T > pFunctionPointer_t mbed::Ticker::attach_us (T * *tptr*, void(T::*)(void) *mptr*, unsigned int *t*) `[inline]`

Attach a member function to be called by the **Ticker**, specifiying the interval in micro-seconds

Parameters:

tptr	pointer to the object to call the member function on
mptr	pointer to the member function to be called
t	the time between calls in micro-seconds

Returns:

The function object created for 'tptr' and 'mptr'

void mbed::Ticker::detach ()

Detach the function

bool mbed::Ticker::remove_function (pFunctionPointer_t *pf*) `[inline]`

Remove a function from the **Ticker**'s call chain

Parameters:

pf	the function object to remove

Returns:

true if the function was found and removed, false otherwise

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

- Ticker.h

ticker_event_s Struct Reference

Public Attributes

- uint32_t **timestamp**
- uint32_t **id**
- struct **ticker_event_s** * **next**

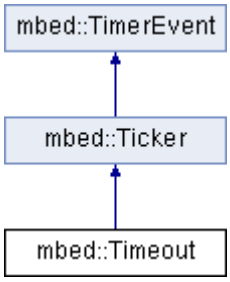
The documentation for this struct was generated from the following file:

- us_ticker_api.h

mbed::Timeout Class Reference

`#include <Timeout.h>`

Inheritance diagram for mbed::Timeout:



Protected Member Functions

- virtual void **handler** ()

Additional Inherited Members

Detailed Description

A **Timeout** is used to call a function at a point in the future

You can use as many seperate **Timeout** objects as you require.

Example:

```
* // Blink until timeout.
*
* #include "mbed.h"
*
* Timeout timeout;
* DigitalOut led(LED1);
*
* int on = 1;
*
* void attimeout() {
*     on = 0;
* }
*
* int main() {
*     timeout.attach(&attimeout, 5);
*     while(on) {
*         led = !led;
*         wait(0.2);
*     }
* }
*
```

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

- Timeout.h

mbed::Timer Class Reference

`#include <Timer.h>`

Public Member Functions

- void **start** ()
- void **stop** ()
- void **reset** ()
- float **read** ()
- int **read_ms** ()
- int **read_us** ()
- **operator float** ()

Protected Member Functions

- int **slicetime** ()

Protected Attributes

- int **_running**
- unsigned int **_start**
- int **_time**

Detailed Description

A general purpose timer

Example:

```
* // Count the time to toggle a LED
*
* #include "mbed.h"
*
* Timer timer;
* DigitalOut led(LED1);
* int begin, end;
*
* int main() {
*     timer.start();
*     begin = timer.read_us();
*     led = !led;
*     end = timer.read_us();
*     printf("Toggle the led takes %d us", end - begin);
* }
*
```

Member Function Documentation

float mbed::Timer::read ()

Get the time passed in seconds

int mbed::Timer::read_ms ()

Get the time passed in mili-seconds

int mbed::Timer::read_us ()

Get the time passed in micro-seconds

void mbed::Timer::reset ()

Reset the timer to 0.

If it was already counting, it will continue

void mbed::Timer::start ()

Start the timer

void mbed::Timer::stop ()

Stop the timer

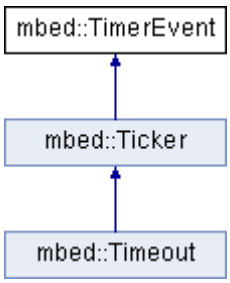
The documentation for this class was generated from the following file:

- Timer.h

mbed::TimerEvent Class Reference

`#include <TimerEvent.h>`

Inheritance diagram for mbed::TimerEvent:



Public Member Functions

- virtual **~TimerEvent** ()

Static Public Member Functions

- static void **irq** (uint32_t id)

Protected Member Functions

- virtual void **handler** ()=0
- void **insert** (unsigned int timestamp)
- void **remove** ()

Protected Attributes

- **ticker_event_t event**

Detailed Description

Base abstraction for timer interrupts

Constructor & Destructor Documentation

virtual mbed::TimerEvent::~TimerEvent () `[virtual]`

Destruction removes it...

Member Function Documentation

static void mbed::TimerEvent::irq (uint32_t *id*) `[static]`

The handler registered with the underlying timer interrupt

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

- TimerEvent.h

Index

INDEX