Class Documentation

buttonBoard Class Reference

`#include <buttonBoard.h>`
Inherits **hook**.

Public Member Functions

- **buttonBoard** (byte, byte, byte, byte, byte, byte)
  Allocates memory for input and output buffers, sets direction registers on IO pins.
- void **byteWrite** (byte, byte)
  Low level access, writes a byte value to the lamp outputs of an individual board.
- byte **byteRead** (byte)
  Low level access, reads the inputs of an individual board to a byte value.
- void **setLamp** (byte, boolean)
  Sets the state of a lamp output.
- void **setLamp** (boolean)
  Sets the state of all lamps.
- boolean **getButton** (byte)
  Get the state of a button.
- boolean **getLampState** (byte)
  Gets the state of a lamp.
- void **update** ()
  Force an update of the input and output buffers. This is called automatically if autoUpdate is set to true.
- byte * **getInPtr** ()
  Gets a pointer to the input buffer.
- byte * **getOutPtr** ()
  Gets a pointer to the output buffer.
- byte **getSize** ()
  Gets the size of the input and output buffers. Same as the number of boards.
- void **setInputInvert** (boolean)
  Set if the hardware inputs are electrically inverted (default is not inverted). This would normally be called in setup().
- void **setOutputInvert** (boolean)
  Set if the hardware outputs are electrically inverted (default is not inverted). This would normally be called in setup().

Public Attributes

- boolean **autoUpdate**

Protected Attributes

- byte * **inBuffer**
- byte * **outBuffer**
- byte **numBoards**
- boolean **inputInvert**

- boolean **outputInvert**

Additional Inherited Members

Constructor & Destructor Documentation

buttonBoard::buttonBoard (byte *data595Pin*, byte *data165Pin*, byte *clockPin*, byte *latch165Pin*, byte *latch595Pin*, byte *numBoards*)

Allocates memory for input and output buffers, sets direction registers on IO pins.

Parameters:

data595Pin	Data out pin, connect to DI on **buttonBoard**
data165Pin	Data in pin, connect to DO on **buttonBoard**
clockPin	Clock pin, connect to CLK on **buttonBoard**
latch165Pin	Input latch pin, connect to ILT on **buttonBoard**
latch595Pin	Output latch pin, connect to OLT on **buttonBoard**
numBoards	Number of boards in use

Returns:

Return_Description

Member Function Documentation

byte buttonBoard::byteRead (byte *board*)

Low level access, reads the inputs of an individual board to a byte value.

Parameters:

board	Board number

Returns:

Byte read from the button inputs (0x00 would be no buttons pressed)

void buttonBoard::byteWrite (byte *board*, byte *val*)

Low level access, writes a byte value to the lamp outputs of an individual board.

Parameters:

board	Board number
val	Byte to be written to the lamp outputs (0xFF would turn the lamps on)

boolean buttonBoard::getButton (byte *buttonNumber*)

Get the state of a button.

Parameters:

buttonNumber	Button number

Returns:

State of the button, true = pressed

byte * buttonBoard::getInPtr ()

Gets a pointer to the input buffer.

Returns:

Pointer to the input buffer

boolean buttonBoard::getLampState (byte *buttonNumber*)

Gets the state of a lamp.

Parameters:

buttonNumber	Button number

Returns:

State of the lamp true = on

byte * buttonBoard::getOutPtr ()

Gets a pointer to the output buffer.

Returns:

Pointer to the output buffer

byte buttonBoard::getSize ()

Gets the size of the input and output buffers. Same as the number of boards.

Returns:

Number of elements in the buffer

void buttonBoard::setInputInvert (boolean *inputInvert*)

Set if the hardware inputs are electrically inverted (default is not inverted). This would normally be called in setup().

Parameters:

inputInvert	set to true to invert the inputs

void buttonBoard::setLamp (byte *buttonNumber*, boolean *state*)

Sets the state of a lamp output.

Parameters:

buttonNumber	Button number
state	Lamp state, true = on

void buttonBoard::setLamp (boolean *state*)

Sets the state of all lamps.

Parameters:

state	Lamp state, true = on

void buttonBoard::setOutputInvert (boolean *outputInvert*)

Set if the hardware outputs are electrically inverted (default is not inverted). This would normally be called in setup().

Parameters:

outputInvert	Set to true to invert the outputs

void buttonBoard::update ()

Force an update of the input and output buffers. This is called automatically if autoUpdate is set to true.

Member Data Documentation

boolean buttonBoard::autoUpdate

byte* buttonBoard::inBuffer`[protected]`

boolean buttonBoard::inputInvert`[protected]`

byte buttonBoard::numBoards`[protected]`

byte* buttonBoard::outBuffer`[protected]`

boolean buttonBoard::outputInvert`[protected]`

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

- **buttonBoard.h**
- **buttonBoard.cpp**

buttonSelect Class Reference

`#include <buttonBoard.h>`

Public Member Functions

- **buttonSelect** (**buttonBoard** *, byte, byte, boolean)
  Selector functionality for a group of buttons.
- byte **getState** ()
  Gets the current state of the button group.
- void **setState** (byte)
  Set the state of the button group.
- boolean **poll** ()
  Poll the buttons to see if there was a press. This should be called every 10-50ms.
- boolean **event** ()
  Check if there has been a state change event.

Public Attributes

- boolean **defaultState**

Constructor & Destructor Documentation

buttonSelect::buttonSelect (buttonBoard * *bb*, byte *offset*, byte *count*, boolean *defaultState*)

Selector functionality for a group of buttons.

Parameters:

bb	Pointer to **buttonBoard** object
offset	First button number
count	Number of buttons in the group
defaultState	State of the buttons in the reset state, true = on

---

Member Function Documentation

boolean buttonSelect::event ()

Check if there has been a state change event.

Returns:

True if there has been a state change since the last call to this function

byte buttonSelect::getState ()

Gets the current state of the button group.

Returns:

Current state of the button group, buttonReset = reset state, # = button number in the group

boolean buttonSelect::poll ()

Poll the buttons to see if there was a press. This should be called every 10-50ms.

Returns:

True if there was a state change

void buttonSelect::setState (byte *state*)

Set the state of the button group.

Parameters:

state	'buttonReset' or button number

Member Data Documentation

boolean buttonSelect::defaultState

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

- **buttonBoard.h**
- **buttonBoard.cpp**

buttonToggle Class Reference

```
#include <buttonBoard.h>
```

Public Member Functions

- **buttonToggle** (**buttonBoard** *, byte)
  Toggle functionality for an individual button.
- boolean **getState** ()
  Get the state of the toggled button.
- void **setState** (boolean)
  Set the state of the toggled button.
- boolean **poll** ()
  Poll the button to see if there was a press. This should be called every 10-50ms.
- boolean **event** ()
  Check if there has been a state change event.

Constructor & Destructor Documentation

buttonToggle::buttonToggle (buttonBoard * *bb*, byte *buttonNumber*)

Toggle functionality for an individual button.

Parameters:

bb	Pointer to **buttonBoard** object
buttonNumber	Button number to manage

Member Function Documentation

boolean buttonToggle::event ()

Check if there has been a state change event.

Returns:

True if there has been a state change since the last call

boolean buttonToggle::getState ()

Get the state of the toggled button.

Returns:

State of the button, true = active

boolean buttonToggle::poll ()

Poll the button to see if there was a press. This should be called every 10-50ms.

Returns:

True if there was a state change

void buttonToggle::setState (boolean *state*)

Set the state of the toggled button.

Parameters:

state	State of the button, true = active

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

- **buttonBoard.h**
- **buttonBoard.cpp**

buttonToggleNoLamp Class Reference

```
#include <buttonBoard.h>
```

Public Member Functions

- **buttonToggleNoLamp** (**buttonBoard** *, byte)
- **buttonToggleNoLamp** (**buttonBoard** *, byte, byte)
- byte **getState** ()
- void **setState** (byte)
- boolean **poll** ()
- boolean **event** ()

Constructor & Destructor Documentation

buttonToggleNoLamp::buttonToggleNoLamp (buttonBoard * *bb*, byte *buttonNumber*)

buttonToggleNoLamp::buttonToggleNoLamp (buttonBoard * *bb*, byte *buttonNumber*, byte *states*)

Member Function Documentation

boolean buttonToggleNoLamp::event ()

byte buttonToggleNoLamp::getState ()

boolean buttonToggleNoLamp::poll ()

void buttonToggleNoLamp::setState (byte *state*)

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

- **buttonBoard.h**
- **buttonBoard.cpp**

hook Class Reference

`#include <hook.h>`

Inherited by **buttonBoard**.

Public Member Functions

- **hook** ()
- void **attachHook** (void(*eventHook)(void))
- void **detachHook** ()

Protected Member Functions

- void **callHook** ()

Detailed Description

Utility class providing inheritable methods to implement hooks.

Author:

Keegan Morrow

Version:

1 31.01.2014

Constructor & Destructor Documentation

hook::hook () `[inline]`

Member Function Documentation

void hook::attachHook (void(*)(void) *eventHook*) `[inline]`

Attach the function to be called.

Parameters:

eventHook	Function pointer to the function to be attached. In the form void foo().

void hook::callHook () `[inline]`, `[protected]`

Calles the hooked function if there is one. This should be placed in the function in the inheriting class to call the hook.

void hook::detachHook () `[inline]`

Detach the hook.

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

- utility/**hook.h**

File Documentation

buttonBoard.cpp File Reference

```
#include "buttonBoard.h"
```

buttonBoard.h File Reference

Hardware interface for the **buttonBoard** board with interface helpers.

```
#include "WProgram.h"
#include <inttypes.h>
#include "utility/hook.h"
```

Classes

- class **buttonBoard**
- class **buttonSelect**
- class **buttonToggle**
- class **buttonToggleNoLamp**

Macros

- #define **BUTTONBOARD** 6
- #define **buttonReset** 0xFF

Detailed Description

Hardware interface for the **buttonBoard** board with interface helpers.

Author:

Keegan Morrow

Version:

6 01.12.2014

Revision history

Rev 0 - 7/2012 Keegan Morrow

Rev 1 - 8/2012 Keegan Morrow - added comments and example code

Rev 2 - 9/2012 Keegan Morrow - added .event() to **buttonSelect** and **buttonToggle**

Rev 3 - 10/2012 Keegan Morrow - added **buttonToggleNoLamp** class to allow external control of the lamps

Rev 4 - 1/2014 Keegan Morrow - added getSize() and hook utilities to **buttonBoard**

Rev 5 - 8/2014 Keegan Morrow - Bugfix in **buttonSelect::poll()** to fix incorrect return value in **buttonSelect::event()**

Rev 6 - 12/2014 Keegan Morrow - Added setInputInvert() and setOutputInvert() to **buttonBoard**

Macro Definition Documentation

#define BUTTONBOARD 6

#define buttonReset 0xFF

utility/hook.h File Reference

Classes

- class **hook**

Macros

- #define **HOOK**   1

Macro Definition Documentation

#define HOOK   1