Class Documentation

hook Class Reference

`#include <hook.h>`
Inherited by **outputExtend**.

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**

outputExtend Class Reference

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

Public Member Functions

- **outputExtend** (byte, byte, byte, byte)
- **outputExtend** (byte, byte)
- void **update** ()
- void **extendedWrite** (byte, boolean)
- void **byteWrite** (byte *, byte, byte)
- void **byteWrite** (byte, byte)
- boolean **getState** (byte)
- byte * **getPtr** ()
- byte **getSize** ()

Public Attributes

- boolean **autoUpdate**

Protected Attributes

- byte * **boards**
- byte **numChips**

Additional Inherited Members

Detailed Description

Hardware interface class for the **outputExtend** board or other 74HC595 based boards.

Author:
Keegan Morrow

Version:
5 2015.02.06

Constructor & Destructor Documentation

outputExtend::outputExtend (byte *dataPin*, byte *clockPin*, byte *latchPin*, byte *numChips*)

Bitbanging mode constructor. Sets the direction of the io pins and allocates needed memory. This should be used to declare a global object.

Parameters:

dataPin	Pin number attached to the data pin
clockPin	Pin number attached to the data pin
latchPin	Pin number attached to the latch pin
numChips	Number of boards in use

outputExtend::outputExtend (byte *latchPin*, byte *numChips*)

SPI mode constructor. Sets the direction of the io pins and allocates needed memory. Starts the hardware SPI module. Note: data must be on pin 11, clock must be on pin 13, pin 12 cannot be used. This should be used to declare a global object.

Parameters:

latchPin	Pin number attached to the latch pin
numChips	Number of boards in use

Member Function Documentation

void outputExtend::byteWrite (byte * *bytePtr*, byte *offset*, byte *count*)

Writes an array of data to the outputs. Note, if count + offset exceeds the output buffer size, the array will be truncated.

Parameters:

**bytePtr*	pointer to the array of bytes
offset	number of boards to skip
count	Size of the array to be copied

void outputExtend::byteWrite (byte *board*, byte *data*)

Writes bytewise data to a board.

Parameters:

board	Board number to write to
data	Data to be written

void outputExtend::extendedWrite (byte *pinNumber*, boolean *state*)

Writes an individual output pin. Pin numbers are sequential from the first pin on the first board. Pin 0 is board 0 output 0, pin 8 is board 1 output 0.

Parameters:

pinNumber	Input pin to write
state	State of the pin. HIGH or LOW (true or false)

byte * outputExtend::getPtr ()

Returns:

Pointer to the output buffer

byte outputExtend::getSize ()

Returns:

Size in bytes of the output buffer (same as the number of chips)

boolean outputExtend::getState (byte *pinNumber*)

Gets the state of the output pin. Pin numbers are sequential from the first pin on the first board. Pin 0 is board 0 output 0, pin 8 is board 1 output 0. Note, this function reads from the output buffer and does not communicate with any external hardware.

Parameters:

pinNumber	Input pin to read

Returns:

State of the pin. HIGH or LOW (true or false)

void outputExtend::update ()

Update the output pins with the current contents of the output buffer. This function is normally called automatically when needed. In a situation where very fast multi-board writes or synchronized outputs are needed, autoUpdate can be set to false and this can be called manually.

Member Data Documentation

boolean outputExtend::autoUpdate

Determines if inputExtend::update() is called automatically. Default is true.

byte* outputExtend::boards[protected]

Buffer size, derived classes should not modify this.

byte outputExtend::numChips[protected]

Input buffer, derived classes can modify the data, but should not change the pointer address.

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

- **outputExtend.h**
- **outputExtend.cpp**

File Documentation

outputExtend.cpp File Reference

```
#include "outputExtend.h"
```

Macros

- #define **__outputExtend_cpp__**

Macro Definition Documentation

#define __outputExtend_cpp__

outputExtend.h File Reference

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

Classes

- class **outputExtend**

Macros

- #define **OUTPUTEXTEND** 5

Macro Definition Documentation

#define OUTPUTEXTEND 5

utility/hook.h File Reference

Classes

- class **hook**

Macros

- #define **HOOK** 1

Macro Definition Documentation

#define HOOK 1