IOOO [ay-oo] Input/Output object oriented

Generated by Doxygen 1.8.3.1

Wed Jul 10 2013 01:25:43

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

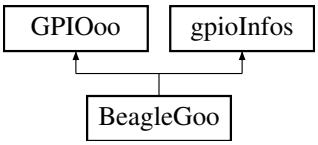
Chapter 4

Class Documentation

4.1 BeagleGoo Class Reference

`#include <BeagleGoo.h>`

Inheritance diagram for BeagleGoo:



Classes

- struct GPIOInfo

Public Member Functions

- virtual ~BeagleGoo ()
- virtual GPIOpin * claim (char *names[], int num, gpioWriteSemantics semantics, gpioFlags flags=gpioFlags-None)

  Method allocates GPIO pins and returns a GPIOPin object. Method allocates a block of pins specified by names passed in names argument. Number of pins in the block is determined by num argument. If flag gpioExclusive is present, only non-allocated pins can be allocated, the pins are marked as exclusive and can not be shared with other blocks. If the gpioExclusive flag has not been specified, pins can be shared with other blocks. If sharing conflict has been detected, no pins must be allocated and the method must return NULL. Argument semantics determines how write operations should be handled. If the requested write semantics is not supported by the hardware platfoom, no pins must be allocated and method must return NULL.
- virtual void release (GPIOpin **gpio)

  Releases a block of GPIO pins. Method releases allocated block of GPIO pins. Methods releases memory allocated for the block, destroys the object and assigns NULL to the referencing variable.

Protected Member Functions

- struct GPIOInfo * _findGpio (char *name)
- BeagleGoo ()

Protected Attributes

- bool active
- int gpioFd
- uint32_t * gpios [4]

Static Protected Attributes

- static struct GPIOInfo gpioInfos []
- static uint16_t addrs []
- static const uint32_t gpioAddrs []
- static const int MaxGpioNameLen =32
- static const int GpioMemBlockLength =0xfff

Friends

- class BeagleGooP
- class GPIOoo

Additional Inherited Members

4.1.1 Detailed Description

pin info

4.1.2 Constructor & Destructor Documentation

4.1.2.1 BeagleGoo::BeagleGoo ( ) [protected]

4.1.2.2 BeagleGoo::~BeagleGoo ( ) [virtual]

4.1.3 Member Function Documentation

4.1.3.1 struct BeagleGoo::GPIOInfo * BeagleGoo::_findGpio (char * *name*) [read], [protected]

4.1.3.2 GPIOpin * BeagleGoo::claim (char * *names[]*, int *num*, gpioWriteSemantics *semantics*, gpioFlags *flags* = gpioFlagsNone) [virtual]

Method allocates GPIO pins and returns a GPIOPin object. Method allocates a block of pins specified by names passed in *names* argument. Number of pins in the block is determined by *num* argument. If flag *gpioExclusive* is present, only non-allocated pins can be allocated, the pins are marked as exclusive and can not be shared with other blocks. If the *gpioExclusive* flag has not been specified, pins can be shared with other blocks. If sharing conflict has been detected, no pins must be allocated and the method must return NULL. Argument *semantics* determines how write operations should be handled. If the requested write semantics is not supported by the hardware platfoom, no pins must be allocated and method must return NULL.

Parameters

names	- an array of system names of pins in the block. Pin names are implementation-dependent. The array should have *num* entries.
num	- number of pins in the block.
semantics	- write semantics. Uses constants defined by *gpioWriteSemantics* enum.
flags	- Flags governing pin allocation. Defined by *gpioFlags* enum. Optional parameter. Default value is no flags.

Returns

Implements GPIOoo.

4.1.3.3 void BeagleGoo::release (GPIOpin ** *gpio*) [virtual]

Releases a block of GPIO pins. Method releases allocated block of GPIO pins. Methods releases memory allocated for the block, destroys the object and assigns NULL to the referencing variable.

Parameters

gpio	- pointer to a variable with reference to an object describing a block of GPIO pins.

Implements GPIOoo.

4.1.4 Friends And Related Function Documentation

4.1.4.1 friend class BeagleGooP [friend]

4.1.4.2 friend class GPIOoo [friend]

4.1.5 Member Data Documentation

4.1.5.1 bool BeagleGoo::active [protected]

4.1.5.2 uint16_t BeagleGoo::addrs [static], [protected]

Initial value:

```
=
        { 0x0818, 0x081C, 0x0808,
                        0x080C, 0x0890, 0x0894, 0x089C, 0x0898,
                        0x0834, 0x0830, 0x0824, 0x0828, 0x083C,
                        0x0838, 0x082C, 0x088C, 0x0820, 0x0884,
                        0x0880, 0x0814, 0x0810, 0x0804, 0x0800,
                        0x087C, 0x08E0, 0x08E8, 0x08E4, 0x08EC,
                        0x08D8, 0x08DC, 0x08D4, 0x08CC, 0x08D0,
                        0x08C8, 0x08C0, 0x08C4, 0x08B8, 0x08BC,
                        0x08B0, 0x08B4, 0x08A8, 0x08AC, 0x08A0,
                        0x08A4
                }
```

4.1.5.3 const uint32_t BeagleGoo::gpioAddrs [static], [protected]

Initial value:

```
=
        { 0x44E07000, 0x4804C000, 0x481AC000, 0x481AE000 }
```

Base addresses for GPIO blocks in memory

4.1.5.4 int BeagleGoo::gpioFd [protected]

4.1.5.5 struct GPIOInfo BeagleGoo::gpioInfos[] [static], [protected]

4.1.5.6 const int BeagleGoo::GpioMemBlockLength =0xfff [static], [protected]

4.1.5.7 uint32_t* BeagleGoo::gpios[4] [protected]

4.1.5.8 const int BeagleGoo::MaxGpioNameLen =32 [static], [protected]

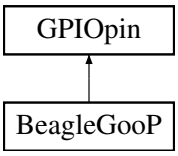
The documentation for this class was generated from the following files:

- src/BeagleGoo.cpp
- include/beaglebone/BeagleGoo.h

4.2 BeagleGooP Class Reference

`#include <BeagleGooP.h>`

Inheritance diagram for BeagleGooP:



Public Member Functions

- virtual void namePin (int i, char *name)
- virtual void namePins (char *names[])
- virtual int findPinIndex (char *name)
- virtual void enableOutput (bool enable)
- virtual void enableOutput (int i, bool enable)
- virtual void enableOutput (int *outs, int num)
- virtual void enableOutput (char **outNames, int num)
- virtual void write (uint32_t v)
- virtual void set (uint32_t v)
- virtual void setBit (int bit)
- virtual void clear (uint32_t v)
- virtual void clearBit (int bit)
- virtual uint32_t read ()

Friends

- class BeagleGoo

Additional Inherited Members

4.2.1 Member Function Documentation

4.2.1.1 void BeagleGooP::clear (uint32_t *v*) [virtual]

Method clears GPIO lines for which corresponding bit in the parameter *v* is set to 1. Lines whose bits are set to 0 remain unchanged.

Parameters

v	

Implements GPIOpin.

4.2.1.2 void BeagleGooP::clearBit (int *bit*) [virtual]

Method clears bit-th bit.

Parameters

bit	

Implements GPIOpin.

4.2.1.3 void BeagleGooP::enableOutput (bool *enable*) [virtual]

Method enables (if enable==true) or disables (enable==false) output buffers on all lines in the block

Parameters

enable	

Implements GPIOpin.

4.2.1.4 void BeagleGooP::enableOutput (int *i*, bool *enable*) [virtual]

Method enables (if enable==true) or disables (enable==false) output buffers on i-th line in the block

Parameters

i	
enable	

Implements GPIOpin.

4.2.1.5 void BeagleGooP::enableOutput (int * *outs*, int *num*) [virtual]

Method enables output buffers on lines listed in the *outs* array. Output buffers on all pins from the block not listed in the array will be disabled. Array contains indexes of the output lines, number of elements in the array is *num*.

Parameters

outs	
num	

Implements GPIOpin.

4.2.1.6 void BeagleGooP::enableOutput (char ** *outNames*, int *num*) [virtual]

Method enables output buffers on lines listed in the *ootNames* array. Output buffers on all pins from the block not listed in the array will be disabled. Array contains references to strings with names of the output lines, number of elements in the array is *num*.

Parameters

outNames	
num	

Implements GPIOpin.

4.2.1.7 int BeagleGooP::findPinIndex (char * *name*) [virtual]

Implements GPIOpin.

4.2.1.8 void BeagleGooP::namePin (int *i*, char * *name*) [virtual]

Implements GPIOpin.

4.2.1.9 void BeagleGooP::namePins (char * *names[]*) [virtual]

Implements GPIOpin.

4.2.1.10 uint32_t BeagleGooP::read ( ) [virtual]

Function returns value read from the GPIO block.

Returns

Implements GPIOpin.

4.2.1.11 void BeagleGooP::set (uint32_t *v*) [virtual]

Method sets GPIO lines for which corresponding bit in the parameter *v* is set to 1. Lines whose bits are set to 0 remain unchanged.

Parameters

v	

Implements GPIOpin.

4.2.1.12 void BeagleGooP::setBit (int *bit*) [virtual]

Method sets i-th bit.

Parameters

bit	

Implements GPIOpin.

4.2.1.13 void BeagleGooP::write (uint32_t *v*) [virtual]

Function writes the value to the pin. Write semantics is determined by the semantics parameter when GPIOs are claimed.

Parameters

v	

Implements GPIOpin.

4.2.2 Friends And Related Function Documentation

4.2.2.1 friend class BeagleGoo [friend]

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

- include/beaglebone/BeagleGooP.h
- src/BeagleGooP.cpp

4.3 BeagleGoo::GPIOInfo Struct Reference

`#include <BeagleGoo.h>`

Public Attributes

- char * name
- int gpioNum
- int bitNum
- int refCounter
- int flags

4.3.1 Member Data Documentation

4.3.1.1 int BeagleGoo::GPIOInfo::bitNum

4.3.1.2 int BeagleGoo::GPIOInfo::flags

4.3.1.3 int BeagleGoo::GPIOInfo::gpioNum

4.3.1.4 char* BeagleGoo::GPIOInfo::name

4.3.1.5 int BeagleGoo::GPIOInfo::refCounter

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

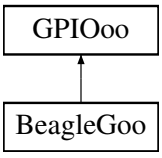
- include/beaglebone/BeagleGoo.h

4.4 GPIOoo Class Reference

Object-oriented implementation of GPIO Class defines interface for object-oriented handling of GPIO operations. Should be used as parent class for platform-specific implementations.

`#include <GPIOoo.h>`

Inheritance diagram for GPIOoo:



Public Types

- enum gpioFlags { gpioFlagsNone = 0, gpioExclusive = 1 }
- enum gpioWriteSemantics { gpioWrite = 1, gpioWriteAtomic, gpioWriteSetBeforeClear, gpioWriteClear-BeforeSet }

Public Member Functions

- virtual ~GPIOoo ()
- virtual GPIOpin * claim (char *names[], int num)

  Simplified version of GPIOpin::claim() Simplified version of GPIOpin::claim(). Assumes gpioWrite semantics and no options.
- virtual GPIOpin * claim (char *names[], int num, gpioWriteSemantics semantics, gpioFlags flags=gpioFlags-None)=0

  Method allocates GPIO pins and returns a GPIOPin object. Method allocates a block of pins specified by names passed in names argument. Number of pins in the block is determined by num argument. If flag gpioExclusive is present, only non-allocated pins can be allocated, the pins are marked as exclusive and can not be shared with other blocks. If the gpioExclusive flag has not been specified, pins can be shared with other blocks. If sharing conflict has been detected, no pins must be allocated and the method must return NULL. Argument semantics determines how write operations should be handled. If the requested write semantics is not supported by the hardware platfoom, no pins must be allocated and method must return NULL.
- virtual void release (GPIOpin **gpio)=0

  Releases a block of GPIO pins. Method releases allocated block of GPIO pins. Methods releases memory allocated for the block, destroys the object and assigns NULL to the referencing variable.

Static Public Member Functions

- static GPIOoo * getInstance ()

Static Protected Member Functions

- static class BeagleGoo inst ()

Friends

- class GPIOpin

4.4.1 Detailed Description

Object-oriented implementation of GPIO Class defines interface for object-oriented handling of GPIO operations. Should be used as parent class for platform-specific implementations.

4.4.2 Member Enumeration Documentation

4.4.2.1 enum GPIOoo::gpioFlags

Options flags for GPIO pin allocation.

Enumerator

gpioFlagsNone gpioFlagsNone - No flags

gpioExclusive gpioExclusive - GPIOs allocated exclusively. Allocating with this flag disables sharing with other blocks.

4.4.2.2 enum GPIOoo::gpioWriteSemantics

Enum defines semantics of write operation to GPIOs.

Enumerator

gpioWrite State of the port can be affected by writes to the pins on the same GPIO port. gpioWrite - Simple write to the port. Prone to race conditions, offers no multi-process safety.

gpioWriteAtomic gpioWriteAtomic - Atomic write to the port. Write to the port must be guaranteed to be successful and effective.

gpioWriteSetBeforeClear gpioWriteSetBeforeClear - In two-step implementation of writing to the pins, pins with value '1' are set before pins with value '0' are cleared. For a short period of time the state of the pins in the GPIO block will be equal to bitwise OR of the previous and next states.

gpioWriteClearBeforeSet gpioWriteClearBeforeSet - In two-step implementation of writing to the pins, pins with value '0' are cleared before pins with value '1' are set. For a short period of time the state of the pins in the GPIO block will be equal to bitwise AND of the previous and next states.

4.4.3 Constructor & Destructor Documentation

4.4.3.1 GPIOoo::~GPIOoo ( ) [virtual]

4.4.4 Member Function Documentation

4.4.4.1 virtual GPIOpin* GPIOoo::claim (char * *names[],* int *num*) [inline], [virtual]

Simplified version of GPIOpin::claim() Simplified version of GPIOpin::claim(). Assumes *gpioWrite* semantics and no options.

Parameters

names	- an array of system names of pins in the block. Pin names are implementation-dependent. The array should have *num* entries.
num	- number of pins in the block.

Returns

4.4.4.2 virtual GPIOpin* GPIOoo::claim (char * *names[],* int *num,* gpioWriteSemantics *semantics,* gpioFlags *flags* = gpioFlagsNone) [pure virtual]

Method allocates GPIO pins and returns a GPIOPin object. Method allocates a block of pins specified by names passed in *names* argument. Number of pins in the block is determined by *num* argument. If flag *gpioExclusive* is present, only non-allocated pins can be allocated, the pins are marked as exclusive and can not be shared with other blocks. If the *gpioExclusive* flag has not been specified, pins can be shared with other blocks. If sharing conflict has been detected, no pins must be allocated and the method must return NULL. Argument *semantics* determines how write operations should be handled. If the requested write semantics is not supported by the hardware platfoom, no pins must be allocated and method must return NULL.

Parameters

names	- an array of system names of pins in the block. Pin names are implementation-dependent. The array should have *num* entries.
num	- number of pins in the block.
semantics	- write semantics. Uses constants defined by *gpioWriteSemantics* enum.
flags	- Flags governing pin allocation. Defined by *gpioFlags* enum. Optional parameter. Default value is no flags.

Returns

Implemented in BeagleGoo.

4.4.4.3 class GPIOoo * GPIOoo::getInstance ( ) [static]

4.4.4.4 static class BeagleGoo GPIOoo::inst ( ) [static], [protected]

4.4.4.5 virtual void GPIOoo::release (GPIOpin ** *gpio*) [pure virtual]

Releases a block of GPIO pins. Method releases allocated block of GPIO pins. Methods releases memory allocated for the block, destroys the object and assigns NULL to the referencing variable.

Parameters

gpio	- pointer to a variable with reference to an object describing a block of GPIO pins.

Implemented in BeagleGoo.

4.4.5 Friends And Related Function Documentation

4.4.5.1 friend class GPIOpin [friend]

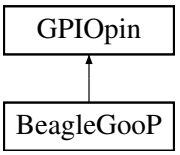
The documentation for this class was generated from the following files:

- include/GPIOoo.h
- src/GPIOoo.cpp

4.5 GPIOpin Class Reference

`#include <GPIOpin.h>`

Inheritance diagram for GPIOpin:



Public Member Functions

- virtual ~GPIOpin ()
- virtual void namePin (int i, char *name)=0
- virtual void namePins (char *names[])=0
- virtual int findPinIndex (char *name)=0
- virtual void enableOutput (bool enable)=0
- virtual void enableOutput (int i, bool enable)=0
- virtual void enableOutput (int *outs, int num)=0
- virtual void enableOutput (char **outNames, int num)=0
- virtual void write (uint32_t v)=0

- virtual void set (uint32_t v)=0
- virtual void setBit (int bit)=0
- virtual void clear (uint32_t v)=0
- virtual void clearBit (int bit)=0
- virtual uint32_t read ()=0
- bool isValid ()

Protected Member Functions

- GPIOpin ()

Protected Attributes

- bool active

Friends

- class GPIO

4.5.1 Constructor & Destructor Documentation

4.5.1.1 GPIOpin::GPIOpin ( ) [inline], [protected]

4.5.1.2 virtual GPIOpin::~GPIOpin ( ) [inline], [virtual]

4.5.2 Member Function Documentation

4.5.2.1 virtual void GPIOpin::clear (uint32_t *v*) [pure virtual]

Method clears GPIO lines for which corresponding bit in the parameter *v* is set to 1. Lines whose bits are set to 0 remain unchanged.

Parameters

v	

Implemented in BeagleGooP.

4.5.2.2 virtual void GPIOpin::clearBit (int *bit*) [pure virtual]

Method clears bit-th bit.

Parameters

bit	

Implemented in BeagleGooP.

4.5.2.3 virtual void GPIOpin::enableOutput (bool *enable*) [pure virtual]

Method enables (if enable==true) or disables (enable==false) output buffers on all lines in the block

Parameters

enable	

Implemented in BeagleGooP.

4.5.2.4 virtual void GPIOpin::enableOutput (int *i*, bool *enable*) [pure virtual]

Method enables (if enable==true) or disables (enable==false) output buffers on i-th line in the block

Parameters

i	
enable	

Implemented in BeagleGooP.

4.5.2.5 virtual void GPIOpin::enableOutput (int * *outs*, int *num*) [pure virtual]

Method enables output buffers on lines listed in the *outs* array. Output buffers on all pins from the block not listed in the array will be disabled. Array contains indexes of the output lines, number of elements in the array is *num*.

Parameters

outs	
num	

Implemented in BeagleGooP.

4.5.2.6 virtual void GPIOpin::enableOutput (char ** *outNames*, int *num*) [pure virtual]

Method enables output buffers on lines listed in the *ootNames* array. Output buffers on all pins from the block not listed in the array will be disabled. Array contains references to strings with names of the output lines, number of elements in the array is *num*.

Parameters

outNames	
num	

Implemented in BeagleGooP.

4.5.2.7 virtual int GPIOpin::findPinIndex (char * *name*) [pure virtual]

Implemented in BeagleGooP.

4.5.2.8 bool GPIOpin::isValid ( ) [inline]

Method returns true of the block describes a valid set of GPIO lines. If method returns false, the block is useless and should not be expected to perform any operations.

Returns

4.5.2.9 virtual void GPIOpin::namePin (int *i*, char * *name*) [pure virtual]

Implemented in BeagleGooP.

4.5.2.10 virtual void GPIOpin::namePins (char * *names[]*) [pure virtual]

Implemented in BeagleGooP.

4.5.2.11 virtual uint32_t GPIOpin::read ( ) [pure virtual]

Function returns value read from the GPIO block.

Returns

Implemented in BeagleGooP.

4.5.2.12 virtual void GPIOpin::set (uint32_t *v*) [pure virtual]

Method sets GPIO lines for which corresponding bit in the parameter *v* is set to 1. Lines whose bits are set to 0 remain unchanged.

Parameters

v	

Implemented in BeagleGooP.

4.5.2.13 virtual void GPIOpin::setBit (int *bit*) [pure virtual]

Method sets i-th bit.

Parameters

bit	

Implemented in BeagleGooP.

4.5.2.14 virtual void GPIOpin::write (uint32_t *v*) [pure virtual]

Function writes the value to the pin. Write semantics is determined by the semantics parameter when GPIOs are claimed.

Parameters

v	

Implemented in BeagleGooP.

4.5.3 Friends And Related Function Documentation

4.5.3.1 friend class GPIO [friend]

4.5.4 Member Data Documentation

4.5.4.1 bool GPIOpin::active [protected]

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

- include/GPIOpin.h

4.6 HD44780 Class Reference

`#include <HD44780.h>`

Public Member Functions

- HD44780 (HD44780phy *phy, int sizeX, int sizeY)
- virtual ~HD44780 ()
- void init ()

  Function initializes the display.
- void clear ()

  Function clears the LCD and moves the cursor to home position.
- void home ()

  Function moves the cursor to home location.
- void gotoXY (uint8_t x, uint8_t y)

  Function moves the cursor to location (x,y)
- void putcc (char c)

  Function prints one character at current cursor position. If the cursor is at the end of the line, the.
- void puts (char *s)

  Function prints a string of characters at current cursor location. Control characters are not interpreted.
- void print (char *s)

  Function prints a string of characters at current cursor location. Function interprets basic ANSI control characters:
- void defineCustomCharacter (uint8_t c, uint8_t *def)

  Function defined a custom user character.

4.6.1 Constructor & Destructor Documentation

4.6.1.1 HD44780::HD44780 (HD44780phy * *phy*, int *sizeX*, int *sizeY*)

4.6.1.2 HD44780::~HD44780 ( ) [virtual]

4.6.2 Member Function Documentation

4.6.2.1 void HD44780::clear ( )

Function clears the LCD and moves the cursor to home position.

4.6.2.2 void HD44780::defineCustomCharacter (uint8_t *c*, uint8_t * *def*)

Function defined a custom user character.

Parameters

c	index of the character.
def	pointer to an array of 8 bytes holding definition of the character.

4.6.2.3 void HD44780::gotoXY (uint8_t *x*, uint8_t *y*)

Function moves the cursor to location (x,y)

4.6.2.4 void HD44780::home ( )

Function moves the cursor to home location.

4.6.2.5 void HD44780::init ( )

Function initializes the display.

4.6.2.6 void HD44780::print (char * *s*)

Function prints a string of characters at current cursor location. Function interprets basic ANSI control characters:

- Line feed '\n'
- Carriage return '\r'

4.6.2.7 void HD44780::putcc (char *c*)

Function prints one character at current cursor position. If the cursor is at the end of the line, the.

4.6.2.8 void HD44780::puts (char * *s*)

Function prints a string of characters at current cursor location. Control characters are not interpreted.

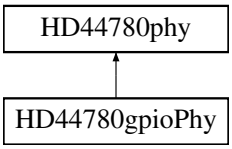
The documentation for this class was generated from the following files:

- include/device/HD44780.h
- src/HD44780.cpp

4.7 HD44780gpioPhy Class Reference

`#include <HD44780gpioPhy.h>`

Inheritance diagram for HD44780gpioPhy:



Public Member Functions

- HD44780gpioPhy (GPIOpin *wires)
- virtual ~HD44780gpioPhy ()
- virtual void write (uint8_t n, uint8_t x)

Method writes byte v to the display. Method writes byte v to the n-th chip on the display. It must implement all operations necessary to write the data. If the display uses 4bit interface, the method must split the value into nibbles and write them in correct order.

- virtual uint8_t read (uint8_t n)

  Method reads the data from the display Method reads the data lines from the the n-th chip on the display. Register read by the method is selected by the RS line. If the hardware interface does not support reading from the display, value returned by the read function is not determined. Ability to read from the display should be tested with supportRead().

- virtual bool busy (uint8_t n)

  Method checks status of BUSY flag If the implementation does not support readback, method should always return false;.

- virtual bool supportsRead ()

  Method reports status of support for reading from the display. Method should return true if the higher level implementation can relay on reading from the display (e.g. status checking v.s. delays between operations.

- virtual uint8_t currentDataAddress (uint8_t n)

  Method reads value of the internal.

- virtual void setE (uint8_t num, uint8_t v)

  Function sets selected E line to requested state. Function sets selected E line to requested state. Selection of enable line allows to support big displays with more than one controllers on board. Enable lines are numbered staring from 0. Value of num can not be ignored. If the hardware interface uses only one enable line, the method should respond only to num set to 0.

- virtual void setRS (uint8_t v)

  Method sets status of RS line. Method sets status of RS line selecting instruction (for write), status (for read) registers or data memory (R/W). Data: RS = 1 Instruction: RS = 0.

- virtual void setRW (uint8_t v)

  Method sets status of RW line. Method sets status of RW line. If the hardware interface does not support readback, this method can be empty. Read: RW=1 Write: RW=0.

Additional Inherited Members

4.7.1 Detailed Description

Implementation of physical interface to text displays based on HD44780, using GPIOoo. Displays with up to 8 chips are supported.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 HD44780gpioPhy::HD44780gpioPhy (GPIOpin * *wires*)

Constructor initializes interface to text display based on HD44780. The display is controlled by named GPIO lines from the wires block. The class supports 4 and 8 bit interfaces. Width of the interface is determined by the number of data lines in the block. Implementation supports multiple chips on the display board. Chips are selected by lines from a block of ENABLE lines.

Wires in the GPIO block are identified by their names:

- D[0] ... D[7] - for data wires. Width of the bus is determined by the number of data wires defined in the block. If 8 wires are present, 8 bit interface will be used. Otherwise if 4 or more wires are present 4-bit interface will be used.
- RS - for wire controlling RS line
- RW - for wire controlling R/W line
- E - for enable wire if only one chip is present (alternative for E[0])

- E[0] ... E[7] for Enable wires. Number of chips is determined by the finding the index of first non-defined E[x] label. For a set of enable wires (E[0], E[1], E[3]) number of chips will be 2, because wire E[2] is not defined. If wire "E" is found, wires E[x] will be ignored and only one chip will be supported.

For multi-wire buses the names consist of the bus symbol (D for data, E for enable), opening square bracket, one digit of wire index and closing square bracket.

Parameters

wires	- block of GPIOs intefacing the display.

4.7.2.2 HD44780gpioPhy::~HD44780gpioPhy ( ) [virtual]

4.7.3 Member Function Documentation

4.7.3.1 bool HD44780gpioPhy::busy (uint8_t *n*) [virtual]

Method checks status of BUSY flag If the implementation does not support readback, method should always return false;.

DNI

Parameters

n	- index of the chip on the display

Returns

true if the display reports BUSY state. False if the display is busy performing internal operations or readback is not supported.

Implements HD44780phy.

4.7.3.2 uint8_t HD44780gpioPhy::currentDataAddress (uint8_t *n*) [virtual]

Method reads value of the internal.

Returns

Implements HD44780phy.

4.7.3.3 uint8_t HD44780gpioPhy::read (uint8_t *n*) [virtual]

Method reads the data from the display Method reads the data lines from the the n-th chip on the display. Register read by the method is selected by the RS line. If the hardware interface does not support reading from the display, value returned by the read function is not determined. Ability to read from the display should be tested with *supportRead()*.

Parameters

n	- index of the chip on the display

Returns

value read from the display.

Implements HD44780phy.

4.7.3.4 void HD44780gpioPhy::setE (uint8_t *num*, uint8_t *v*) [virtual]

Function sets selected E line to requested state. Function sets selected E line to requested state. Selection of enable line allows to support big displays with more than one controllers on board. Enable lines are numbered staring from 0. Value of *num* can not be ignored. If the hardware interface uses only one enable line, the method should respond only to num set to 0.

Parameters

num	Index of E line.
v	Status of E line.

Implements HD44780phy.

4.7.3.5 void HD44780gpioPhy::setRS (uint8_t *v*) [virtual]

Method sets status of RS line. Method sets status of RS line selecting instruction (for write), status (for read) registers or data memory (R/W). Data: RS = 1 Instruction: RS = 0.

Parameters

v	requested status of RS line.

Implements HD44780phy.

4.7.3.6 void HD44780gpioPhy::setRW (uint8_t *v*) [virtual]

Method sets status of RW line. Method sets status of RW line. If the hardware interface does not support readback, this method can be empty. Read: RW=1 Write: RW=0.

Parameters

v	requested status of RS line.

Implements HD44780phy.

4.7.3.7 virtual bool HD44780gpioPhy::supportsRead ( ) [inline], [virtual]

Method reports status of support for reading from the display. Method should return true if the higher level implementation can relay on reading from the display (e.g. status checking v.s. delays between operations.

Returns

true of reading BUSY flag is supported.

Implements HD44780phy.

4.7.3.8 void HD44780gpioPhy::write (uint8_t *n*, uint8_t *x*) [virtual]

Method writes byte v to the display. Method writes byte v to the n-th chip on the display. It must implement all operations necessary to write the data. If the display uses 4bit interface, the method must split the value into nibbles and write them in correct order.

Parameters

n	- index of the chip on the display
x	value to be written

Implements HD44780phy.

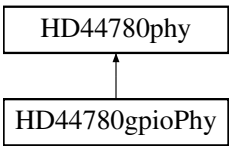
The documentation for this class was generated from the following files:

- include/device/HD44780gpioPhy.h
- src/HD44780gpioPhy.cpp

4.8 HD44780phy Class Reference

`#include <HD44780phy.h>`

Inheritance diagram for HD44780phy:



Public Types

- enum { RScommand =0, RSdata =1 }
- enum { RWwrite =0, RWread =1 }

Public Member Functions

- HD44780phy ()
- virtual ~HD44780phy ()
- virtual void write (uint8_t n, uint8_t x)=0

  Method writes byte v to the display. Method writes byte v to the n-th chip on the display. It must implement all operations necessary to write the data. If the display uses 4bit interface, the method must split the value into nibbles and write them in correct order.
- virtual bool supportsRead ()=0

  Method reports status of support for reading from the display. Method should return true if the higher level implementation can relay on reading from the display (e.g. status checking v.s. delays between operations.
- virtual uint8_t read (uint8_t n)=0

  Method reads the data from the display Method reads the data lines from the the n-th chip on the display. Register read by the method is selected by the RS line. If the hardware interface does not support reading from the display, value returned by the read function is not determined. Ability to read from the display should be tested with supportRead().
- virtual bool busy (uint8_t n)=0

  Method checks status of BUSY flag If the implementation does not support readback, method should always return false;.
- virtual uint8_t currentDataAddress (uint8_t n)=0

Method reads value of the internal.

- virtual void setE (uint8_t num, uint8_t v)=0

  Function sets selected E line to requested state. Function sets selected E line to requested state. Selection of enable line allows to support big displays with more than one controllers on board. Enable lines are numbered staring from 0. Value of num can not be ignored. If the hardware interface uses only one enable line, the method should respond only to num set to 0.

- virtual void setRS (uint8_t v)=0

  Method sets status of RS line. Method sets status of RS line selecting instruction (for write), status (for read) registers or data memory (R/W). Data: RS = 1 Instruction: RS = 0.

- virtual void setRW (uint8_t v)=0

  Method sets status of RW line. Method sets status of RW line. If the hardware interface does not support readback, this method can be empty. Read: RW=1 Write: RW=0.

- int getBits () const

Protected Attributes

- int bits

4.8.1 Detailed Description

Class defines interface to the display. Higher level functions use the interface for low level communication. Implementation of the interface should support 8 or 4 bit communication (depending on hardware implementation), exposing to the higher layer only 8-bit interface. The implementation must honor multiple chip select (enable) lines, for displays with multiple HD44780 chips on board. All requests to chips with indexes beyond the number of supported enable lines should be ignored.

The class should not be use to drive multiple displays connected to the data and control bus, unless they are supposed to act as one composite display. To drive multiple independent displays sharing data and control bus each display, each display should have a separate instance of HD44780phy object, controlled by separate blocks of IO lines sharing the data and control lines, with different enable lines.

4.8.2 Member Enumeration Documentation

4.8.2.1 anonymous enum

Constants for setRS

Enumerator

RScommand RScommand - sets command mode.

RSdata RSdata - sets data mode.

4.8.2.2 anonymous enum

constants for setRW

Enumerator

RWwrite RWwrite - sets write mode.

RWread RWread - sets read mode.

4.8.3 Constructor & Destructor Documentation

4.8.3.1 HD44780phy::HD44780phy ( ) [inline]

4.8.3.2 virtual HD44780phy::~HD44780phy ( ) [inline], [virtual]

4.8.4 Member Function Documentation

4.8.4.1 virtual bool HD44780phy::busy (uint8_t *n*) [pure virtual]

Method checks status of BUSY flag If the implementation does not support readback, method should always return false;.

DNI

Parameters

n	- index of the chip on the display

Returns

true if the display reports BUSY state. False if the display is busy performing internal operations or readback is not supported.

Implemented in HD44780gpioPhy.

4.8.4.2 virtual uint8_t HD44780phy::currentDataAddress (uint8_t *n*) [pure virtual]

Method reads value of the internal.

Returns

Implemented in HD44780gpioPhy.

4.8.4.3 int HD44780phy::getBits ( ) const [inline]

4.8.4.4 virtual uint8_t HD44780phy::read (uint8_t *n*) [pure virtual]

Method reads the data from the display Method reads the data lines from the the n-th chip on the display. Register read by the method is selected by the RS line. If the hardware interface does not support reading from the display, value returned by the read function is not determined. Ability to read from the display should be tested with *supportRead()*.

Parameters

n	- index of the chip on the display

Returns

value read from the display.

Implemented in HD44780gpioPhy.

4.8.4.5 virtual void HD44780phy::setE (uint8_t *num*, uint8_t *v*) [pure virtual]

Function sets selected E line to requested state. Function sets selected E line to requested state. Selection of enable line allows to support big displays with more than one controllers on board. Enable lines are numbered staring from 0. Value of *num* can not be ignored. If the hardware interface uses only one enable line, the method should respond only to num set to 0.

Parameters

num	Index of E line.
v	Status of E line.

Implemented in HD44780gpioPhy.

4.8.4.6 virtual void HD44780phy::setRS (uint8_t *v*) [pure virtual]

Method sets status of RS line. Method sets status of RS line selecting instruction (for write), status (for read) registers or data memory (R/W). Data: RS = 1 Instruction: RS = 0.

Parameters

v	requested status of RS line.

Implemented in HD44780gpioPhy.

4.8.4.7 virtual void HD44780phy::setRW (uint8_t *v*) [pure virtual]

Method sets status of RW line. Method sets status of RW line. If the hardware interface does not support readback, this method can be empty. Read: RW=1 Write: RW=0.

Parameters

v	requested status of RS line.

Implemented in HD44780gpioPhy.

4.8.4.8 virtual bool HD44780phy::supportsRead ( ) [pure virtual]

Method reports status of support for reading from the display. Method should return true if the higher level implementation can relay on reading from the display (e.g. status checking v.s. delays between operations.

Returns

true of reading BUSY flag is supported.

Implemented in HD44780gpioPhy.

4.8.4.9 virtual void HD44780phy::write (uint8_t *n*, uint8_t *x*) [pure virtual]

Method writes byte v to the display. Method writes byte v to the n-th chip on the display. It must implement all operations necessary to write the data. If the display uses 4bit interface, the method must split the value into nibbles and write them in correct order.

Parameters

n	- index of the chip on the display
x	value to be written

Implemented in HD44780gpioPhy.

4.8.5 Member Data Documentation

4.8.5.1 int HD44780phy::bits [protected]

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

- include/device/HD44780phy.h

4.9 pru_data Struct Reference

Public Attributes

- uint8_t * prumem

4.9.1 Member Data Documentation

4.9.1.1 uint8_t* pru_data::prumem

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

- examples/pru_loader.c

4.10 SPI Class Reference

```
#include <SPI.h>
```

Public Member Functions

- SPI ()
- int open (int bus, int channel)
- int close ()
- int setMode (uint8_t mode)
- int setClockPolarity (uint8_t pol)
- int setClockPhase (uint8_t phase)
- int setLSBFirst (bool lsb_first)
- int setBitsPerWord (int bits)
- int setSpeed (uint32_t speed)
- int write (uint8_t wbuf[], int len)
- int read (uint8_t rbuf[], int len)
- int xfer1 (uint8_t wbuf[], uint8_t rbuf[], int len)
- virtual ~SPI ()

4.10.1 Constructor & Destructor Documentation

4.10.1.1 SPI::SPI ( )

4.10.1.2 SPI::~SPI ( ) [virtual]

4.10.2 Member Function Documentation

4.10.2.1 int SPI::close ( )

4.10.2.2 int SPI::open (int *bus*, int *channel*)

4.10.2.3 int SPI::read (uint8_t *rbuf[]*, int *len*)

4.10.2.4 int SPI::setBitsPerWord (int *bits*)

4.10.2.5 int SPI::setClockPhase (uint8_t *phase*)

4.10.2.6 int SPI::setClockPolarity (uint8_t *pol*)

4.10.2.7 int SPI::setLSBFirst (bool *lsb_first*)

4.10.2.8 int SPI::setMode (uint8_t *mode*)

4.10.2.9 int SPI::setSpeed (uint32_t *speed*)

4.10.2.10 int SPI::write (uint8_t *wbuf[]*, int *len*)

4.10.2.11 int SPI::xfer1 (uint8_t *wbuf[]*, uint8_t *rbuf[]*, int *len*)

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

- include/SPI.h
- src/SPI.cpp

4.11 TestGPIOButtons Class Reference

`#include <TestGPIOButtons.h>`

Public Member Functions

- TestGPIOButtons ()
- virtual ~TestGPIOButtons ()
- virtual void loop ()

Protected Attributes

- GPIOoo * gp
- GPIOpin * blockButton

4.11.1 Constructor & Destructor Documentation

4.11.1.1 TestGPIOButtons::TestGPIOButtons ( )

4.11.1.2 TestGPIOButtons::~TestGPIOButtons ( ) [virtual]

4.11.2 Member Function Documentation

4.11.2.1 void TestGPIOButtons::loop ( ) [virtual]

4.11.3 Member Data Documentation

4.11.3.1 GPIOpin* TestGPIOButtons::blockButton [protected]

4.11.3.2 GPIOoo* TestGPIOButtons::gp [protected]

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

- examples/TestGPIOButtons.h
- examples/TestGPIOButtons.cpp

4.12 TestGPIOLeds Class Reference

`#include <TestGPIOLeds.h>`

Public Member Functions

- TestGPIOLeds ()
- virtual ~TestGPIOLeds ()
- virtual void loop ()
- virtual void loop (int iterations)

4.12.1 Constructor & Destructor Documentation

4.12.1.1 TestGPIOLeds::TestGPIOLeds ( )

4.12.1.2 TestGPIOLeds::~TestGPIOLeds ( ) [virtual]

4.12.2 Member Function Documentation

4.12.2.1 virtual void TestGPIOLeds::loop ( ) [inline], [virtual]

4.12.2.2 void TestGPIOLeds::loop (int *iterations*) [virtual]

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

- examples/TestGPIOLeds.h
- examples/TestGPIOLeds.cpp

4.13 TestLCD Class Reference

`#include <TestLCD.h>`

Public Member Functions

- TestLCD (int bits)
- virtual ~TestLCD ()
- void loop ()

4.13.1 Constructor & Destructor Documentation

4.13.1.1 TestLCD::TestLCD (int *bits*)

4.13.1.2 TestLCD::~TestLCD ( ) [virtual]

4.13.2 Member Function Documentation

4.13.2.1 void TestLCD::loop ( )

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

- examples/TestLCD.h
- examples/TestLCD.cpp

4.14 TestTLC5946 Class Reference

`#include <TestTLC5946.h>`

Public Member Functions

- TestTLC5946 (SPI *spi, char *pruBinFile)
- virtual ~TestTLC5946 ()
- void loop ()

4.14.1 Constructor & Destructor Documentation

4.14.1.1 TestTLC5946::TestTLC5946 (SPI * *spi,* char * *pruBinFile*)

4.14.1.2 TestTLC5946::~TestTLC5946 ( ) [virtual]

4.14.2 Member Function Documentation

4.14.2.1 void TestTLC5946::loop ( )

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

- examples/TestTLC5946.h
- examples/TestTLC5946.cpp

4.15 TLC5946chain Class Reference

`#include <TLC5946chain.h>`

Public Member Functions

- TLC5946chain (TLC5946phy *_phy, int num)
- virtual ~TLC5946chain ()
- void setBrightness (int i, uint16_t b)
- void setDOT (int i, uint16_t dot)
- void blank (int b)
- void commit ()

4.15.1 Constructor & Destructor Documentation

4.15.1.1 TLC5946chain::TLC5946chain (TLC5946phy * _phy, int num)

4.15.1.2 TLC5946chain::~TLC5946chain ( ) [virtual]

4.15.2 Member Function Documentation

4.15.2.1 void TLC5946chain::blank (int b)

4.15.2.2 void TLC5946chain::commit ( )

4.15.2.3 void TLC5946chain::setBrightness (int i, uint16_t b)

4.15.2.4 void TLC5946chain::setDOT (int i, uint16_t dot)

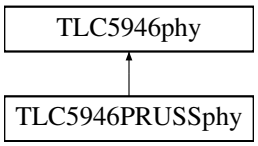
The documentation for this class was generated from the following files:

- include/device/TLC5946chain.h
- src/TLC5946chain.cpp

4.16 TLC5946phy Class Reference

`#include <TLC5946phy.h>`

Inheritance diagram for TLC5946phy:



Public Member Functions

- TLC5946phy (SPI *_spi, GPIOpin *ctrl)
- virtual ~TLC5946phy ()
- virtual void setBlank (uint8_t blank)
- virtual void setMode (uint8_t mode)
- virtual void setXhalf (uint8_t xhalf)
- virtual uint8_t getXerr ()
- virtual int setBitsPerWord (int bits)
- virtual int setLSBFirst (bool lsb_first)
- virtual int xfer (uint8_t buf_out[], uint8_t buf_in[], int len)

Protected Attributes

- SPI * spi
- GPIOpin * ctrl
- bool active
- int blank_pin_pin
- int mode_pin_pin
- int xhalf_pin_pin
- int xerr_pin_pin

4.16.1 Detailed Description

The class acts as a layer abstracting physical interface to TLC5946. Higher level class controlling behavior of the chain of the chips can call functions of the interface without needing to know how they interface with the actual hardware. This allows to implement high-level functionality without knowing how the hardware interface works.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 TLC5946phy::TLC5946phy (SPI * *_spi,* GPIOpin * *ctrl*)

Constructor initializes TLC5946 physical interface using provided SPI bus for data communication and GPIO lines as control signals. Constructor assumes that the clocking and blanking signals for the chip are generated externally and will not attempt to set them up. The "blank" line acts as "true" blank: setting the line active will blank all the outputs.

Control signals should be assigned the following names:

- mode
- xhalf
- xerr
- blank

GSCLK line will not be used and does not need to be allocated.

Parameters

_spi	
ctrl	

4.16.2.2 TLC5946phy::~TLC5946phy ( ) [virtual]

4.16.3 Member Function Documentation

4.16.3.1 uint8_t TLC5946phy::getXerr ( ) [virtual]

4.16.3.2 int TLC5946phy::setBitsPerWord (int *bits*) [virtual]

4.16.3.3 void TLC5946phy::setBlank (uint8_t *blank*) [virtual]

Reimplemented in TLC5946PRUSSphy.

4.16.3.4 int TLC5946phy::setLSBFirst (bool *lsb_first*) [virtual]

4.16.3.5 void TLC5946phy::setMode (uint8_t *mode*) [virtual]

4.16.3.6 void TLC5946phy::setXhalf (uint8_t *xhalf*) [virtual]

4.16.3.7 int TLC5946phy::xfer (uint8_t *buf_out[],* uint8_t *buf_in[],* int *len*) [virtual]

4.16.4 Member Data Documentation

4.16.4.1 bool TLC5946phy::active [protected]

4.16.4.2 int TLC5946phy::blank_pin_pin [protected]

4.16.4.3 GPIOpin* TLC5946phy::ctrl [protected]

4.16.4.4 int TLC5946phy::mode_pin_pin [protected]

4.16.4.5 SPI* TLC5946phy::spi [protected]

4.16.4.6 int TLC5946phy::xerr_pin_pin [protected]

4.16.4.7 int TLC5946phy::xhalf_pin_pin [protected]

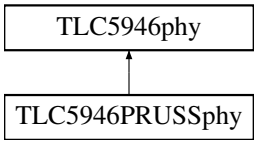
The documentation for this class was generated from the following files:

- include/device/TLC5946phy.h
- src/TLC5946phy.cpp

4.17 TLC5946PRUSSphy Class Reference

`#include <TLC5946PRUSSphy.h>`

Inheritance diagram for TLC5946PRUSSphy:



Public Member Functions

- TLC5946PRUSSphy (SPI *_spi, GPIOpin *ctrl, char *pruBinFile)
- virtual ~TLC5946PRUSSphy ()
- virtual void setBlank (uint8_t blank)

Additional Inherited Members

4.17.1 Detailed Description

The class acts as a layer abstracting physical interface to TLC5946. Higher level class controlling behavior of the chain of the chips can call functions of the interface without needing to know how they interface with the actual hardware. This allows to implement high-level functionality without knowing how the hardware interface works.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 TLC5946PRUSSphy::TLC5946PRUSSphy (SPI * *_spi,* GPIOpin * *ctrl,* char * *pruBinFile*)

Constructor initializes TLC5946 physical interface using provided SPI bus for data communication and GPIO lines as control signals. Constructor initializes PRU0 unit with microcode read from the file whose name is provided as the last parameter. The microcode should generate clocking and blanking sequences required by TLC5946 chip. If the microcode can not be read from the file or PRUSS can not be initialized, the module will not be activated.

Control signals should be assigned the following names:

- mode
- xhalf
- xerr
- blank
- gsclk

This function is Beaglebone-specific.

Parameters

_spi	
ctrl	
pruBinFile	

4.17.2.2 TLC5946PRUSSphy::~TLC5946PRUSSphy ( ) [virtual]

4.17.3 Member Function Documentation

4.17.3.1 void TLC5946PRUSSphy::setBlank (uint8_t *blank*) [virtual]

Reimplemented from TLC5946phy.

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

- include/device/TLC5946PRUSSphy.h
- src/TLC5946PRUSSphy.cpp

Chapter 5

File Documentation

5.1 examples/BeagleboneSPI.cpp File Reference

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <pruss/prussdrv.h>
#include <pruss/pruss_intc_mapping.h>
#include <errno.h>
#include "SPI.h"
#include "GPIOoo.h"
#include "GPIOpin.h"
#include "device/HD44780gpioPhy.h"
#include "device/HD44780.h"
#include "device/TLC5946phy.h"
#include "device/TLC5946chain.h"
#include "TestTLC5946.h"
#include "TestLCD.h"
#include "TestGPIOLeds.h"
#include "debug.h"
```

Macros

- #define SEQ_LEN 10

Functions

- int testSPI (SPI &spi)
- SPI * setupSPI ()
- int main ()

5.1.1 Macro Definition Documentation

5.1.1.1 #define SEQ_LEN 10

5.1.2 Function Documentation

5.1.2.1 int main ( )

5.1.2.2 SPI* setupSPI ( )

5.1.2.3 int testSPI (SPI & *spi*)

5.2 examples/gpio_buttons.cpp File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include "TestGPIOButtons.h"
```

Functions

- int main ()

5.2.1 Function Documentation

5.2.1.1 int main ( )

5.3 examples/gpio_lcd.cpp File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "GPIOoo.h"
#include "GPIOpin.h"
#include "device/HD44780gpioPhy.h"
#include "device/HD44780.h"
#include "TestLCD.h"
#include "debug.h"
```

Functions

- int main ()

5.3.1 Function Documentation

5.3.1.1 int main ( )

5.4 examples/gpio_leds.cpp File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include "TestGPIOLeds.h"
#include "debug.h"
```

Functions

- int main ()

5.4.1 Function Documentation

5.4.1.1 int main ( )

5.5 examples/pru_loader.c File Reference

```
#include <errno.h>
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <unistd.h>
#include <pruss/prussdrv.h>
#include <pruss/pruss_intc_mapping.h>
#include <sys/fcntl.h>
#include <sys/stat.h>
#include <time.h>
```

Classes

- struct pru_data

Functions

- int main (int argc, char **argv)

5.5.1 Function Documentation

5.5.1.1 int main (int *argc,* char ** *argv*)

5.6 examples/TestGPIOButtons.cpp File Reference

```
#include "TestGPIOButtons.h"
#include "stdio.h"
#include "unistd.h"
```

5.7 examples/TestGPIOButtons.h File Reference

```
#include "GPIOoo.h"
#include "GPIOpin.h"
```

Classes

- class TestGPIOButtons

5.8 examples/TestGPIOLeds.cpp File Reference

```
#include "TestGPIOLeds.h"
#include <unistd.h>
```

5.9 examples/TestGPIOLeds.h File Reference

```
#include "GPIOoo.h"
#include "GPIOpin.h"
```

Classes

- class TestGPIOLeds

5.10 examples/TestLCD.cpp File Reference

```
#include <stdio.h>
#include "TestLCD.h"
#include "debug.h"
```

5.11 examples/TestLCD.h File Reference

```
#include "GPIOoo.h"
#include "GPIOpin.h"
#include "device/HD44780gpioPhy.h"
#include "device/HD44780.h"
```

Classes

- class TestLCD

5.12 examples/TestTLC5946.cpp File Reference

```
#include "TestTLC5946.h"
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
```

5.13 examples/TestTLC5946.h File Reference

```
#include "GPIOoo.h"
#include "GPIOpin.h"
#include "device/TLC5946PRUSSphy.h"
#include "device/TLC5946chain.h"
```

Classes

- class TestTLC5946

5.14 examples/tlc5946.cpp File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include "SPI.h"
#include "GPIOoo.h"
#include "GPIOpin.h"
#include "TestTLC5946.h"
#include "debug.h"
```

Functions

- SPI * setupSPI ()
- int main ()

5.14.1 Function Documentation

5.14.1.1 int main ( )

5.14.1.2 SPI* setupSPI ( )

5.15 include/beaglebone/BeagleGoo.h File Reference

```
#include "../GPIOoo.h"
#include <stdint.h>
```

Classes

- class BeagleGoo
- struct BeagleGoo::GPIOInfo

5.16 include/beaglebone/BeagleGooP.h File Reference

```
#include "GPIOpin.h"
```

```
#include "BeagleGoo.h"
#include <stdint.h>
```

Classes

- class BeagleGooP

5.17 include/debug.h File Reference

Macros

- #define DEBUG_LEVEL -1
- #define debug(level, format,...) {if(level<=DEBUG_LEVEL){printf("%s:%i:"#format"\n",__FILE__,__LINE__-,##__VA_ARGS__);}}

5.17.1 Macro Definition Documentation

5.17.1.1 #define debug(*level, format, ...*) {if(level<=DEBUG_LEVEL){printf("%s:%i:"#format"\n",__FILE__,__LINE__,##-__VA_ARGS__);}}

Debug macro Params:

Parameters

level	- level of details of debug message.
format	- Formatting string for the debug message
...	-

5.17.1.2 #define DEBUG_LEVEL -1

5.18 include/device/HD44780.h File Reference

```
#include "HD44780phy.h"
```

Classes

- class HD44780

5.19 include/device/HD44780gpioPhy.h File Reference

```
#include "GPIOpin.h"
#include "HD44780phy.h"
```

Classes

- class HD44780gpioPhy

5.20 include/device/HD44780phy.h File Reference

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

Classes

- class HD44780phy

5.21 include/device/TLC5946chain.h File Reference

```
#include "TLC5946phy.h"
```

Classes

- class TLC5946chain

5.22 include/device/TLC5946phy.h File Reference

```
#include "SPI.h"
#include "GPIOpin.h"
```

Classes

- class TLC5946phy

5.23 include/device/TLC5946PRUSSphy.h File Reference

```
#include "SPI.h"
#include "GPIOpin.h"
#include "TLC5946phy.h"
```

Classes

- class TLC5946PRUSSphy

5.24 include/GPIOoo.h File Reference

```
#include "GPIOpin.h"
```

Classes

- class GPIOoo

  Object-oriented implementation of GPIO Class defines interface for object-oriented handling of GPIO operations. Should be used as parent class for platform-specific implementations.

5.25 include/GPIOpin.h File Reference

```
#include <stdlib.h>
#include <stdint.h>
```

Classes

- class GPIOpin

5.26 include/SPI.h File Reference

```
#include <stdint.h>
#include <linux/spi/spidev.h>
```

Classes

- class SPI

5.27 src/BeagleGoo.cpp File Reference

```
#include "beaglebone/BeagleGoo.h"
#include "beaglebone/BeagleGooP.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/mman.h>
#include "debug.h"
```

Classes

- class BeagleGoo

5.28 src/BeagleGooP.cpp File Reference

```
#include "beaglebone/BeagleGooP.h"
#include <string.h>
#include <stdio.h>
#include "debug.h"
```

Macros

- #define DATA_OUT_REG 0x13C
- #define DATA_IN_REG 0x138
- #define GPIO_OE_REG 0x134
- #define DATA_CLEAR_REG 0x190
- #define DATA_SET_REG 0x194

5.28.1 Macro Definition Documentation

5.28.1.1 #define DATA_CLEAR_REG 0x190

5.28.1.2 #define DATA_IN_REG 0x138

5.28.1.3 #define DATA_OUT_REG 0x13C

5.28.1.4 #define DATA_SET_REG 0x194

5.28.1.5 #define GPIO_OE_REG 0x134

5.29 src/GPIOoo.cpp File Reference

```
#include "GPIOoo.h"
#include "GPIOpin.h"
#include <stdio.h>
```

5.30 src/gpiotest.c File Reference

```
#include <stdio.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <unistd.h>
```

Macros

- #define BUILD_TESTER
- #define DATA_OUT_REG 0x13C
- #define GPIO_OE_REG 0x134
- #define DATA_CLEAR_REG 0x190
- #define DATA_SET_REG 0x194

Functions

- void print_gpio_conf_info (char *name, unsigned int info)
- void print_gpio_infos (unsigned long *m_controlModule)
- int _main ()

5.30.1 Macro Definition Documentation

5.30.1.1 #define BUILD_TESTER

5.30.1.2 #define DATA_CLEAR_REG 0x190

5.30.1.3 #define DATA_OUT_REG 0x13C

5.30.1.4 #define DATA_SET_REG 0x194

5.30.1.5 #define GPIO_OE_REG 0x134

5.30.2 Function Documentation

5.30.2.1 int _main ( )

5.30.2.2 void print_gpio_conf_info (char * *name*, unsigned int *info*)

5.30.2.3 void print_gpio_infos (unsigned long * *m_controlModule*)

5.31 src/HD44780.cpp File Reference

```
#include "device/HD44780.h"
#include <unistd.h>
```

5.32 src/HD44780gpioPhy.cpp File Reference

```
#include "device/HD44780gpioPhy.h"
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include "debug.h"
```

Macros

- #define BUSY_BIT 0x80
- #define ADDRESS_BITS 0x7f

5.32.1 Macro Definition Documentation

5.32.1.1 #define ADDRESS_BITS 0x7f

5.32.1.2 #define BUSY_BIT 0x80

5.33 src/SPI.cpp File Reference

```
#include "SPI.h"
#include <errno.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/ioctl.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
```

Macros

- #define MAX_PATH_LEN 40
- #define SPI_DEVICE_PATH_BASE "/dev/spidev"

5.33.1 Macro Definition Documentation

5.33.1.1 #define MAX_PATH_LEN 40

5.33.1.2 #define SPI_DEVICE_PATH_BASE "/dev/spidev"

5.34 src/TLC5946chain.cpp File Reference

```
#include "device/TLC5946chain.h"
#include <string.h>
#include <stdio.h>
#include "debug.h"
```

5.35 src/TLC5946phy.cpp File Reference

```
#include "device/TLC5946phy.h"
#include <iostream>
#include <errno.h>
#include <stdio.h>
#include <pruss/prussdrv.h>
#include <pruss/pruss_intc_mapping.h>
#include "debug.h"
```

5.36 src/TLC5946PRUSSphy.cpp File Reference

```
#include "device/TLC5946PRUSSphy.h"
#include <iostream>
#include <errno.h>
#include <stdio.h>
#include <pruss/prussdrv.h>
#include <pruss/pruss_intc_mapping.h>
#include "debug.h"
```

Index