Arduino Gyroscope Driver

Generated by Doxygen 1.8.9.1

Tue Aug 18 2015 22:52:32

Contents

1 Hierarchical Index

1.1 Class Hierarchy

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

2 Class Index

2.1 Class List

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

3 File Index

3.1 File List

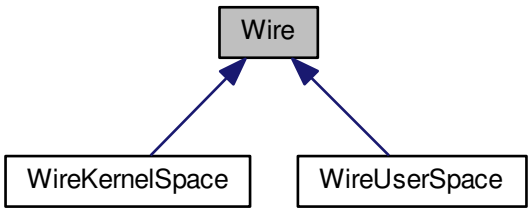
Here is a list of all files with brief descriptions:

4 Class Documentation

4.1 Wire Class Reference

`#include <Wire.h>`

Inheritance diagram for Wire:



Public Member Functions

- virtual ~Wire ()
- virtual void begin ()=0
- virtual void stop ()=0
- virtual void beginTransmission (int address)=0
- virtual unsigned char endTransmission ()=0
- virtual unsigned char requestFrom (int address, unsigned int len)=0
- virtual unsigned int write (unsigned char b)=0
- virtual unsigned int write (const unsigned char *buf, unsigned int len)=0
- virtual int available ()=0
- virtual int read ()=0
- virtual void flush ()=0

4.1.1 Detailed Description

This is a siple Wire library to Raspberry interface.

Definition at line 9 of file Wire.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 virtual Wire::~Wire ( ) [inline], [virtual]

Destructor.

Definition at line 16 of file Wire.h.

4.1.3 Member Function Documentation

4.1.3.1 virtual int Wire::available ( ) [pure virtual]

Returns 1 if there is one or more bytes to be read.

Returns

0 if there is no bytes to be read in the internal FIFO.

Implemented in WireKernelSpace, and WireUserSpace.

4.1.3.2 virtual void Wire::begin ( ) [pure virtual]

Initiate the library.

(Only as a master) This should normally be called only once. It maps the BSC0 (0x7E20_5000) registers.

Implemented in WireKernelSpace, and WireUserSpace.

4.1.3.3 virtual void Wire::beginTransmission (int *address*) [pure virtual]

Begin a transmission to the I2C slave device with the given address.

Subsequently, queue bytes for transmission with the write() function and transmit them by calling endTransmission().

Parameters

address	The device address.

Implemented in WireKernelSpace, and WireUserSpace.

4.1.3.4 virtual unsigned char Wire::endTransmission ( ) [pure virtual]

Begin a transmission to the I2C slave device with the given address.

Subsequently, queue bytes for transmission with the write() function and transmit them by calling endTransmission().

Parameters

address	The device address. Ends a transmission to a slave device that was begun by beginTransmission() and transmits the bytes that were queued by write().

Returns

Nothing for now.

Implemented in WireKernelSpace, and WireUserSpace.

4.1.3.5 virtual void Wire::flush ( ) [pure virtual]

For now, does nothing.

Implemented in WireKernelSpace, and WireUserSpace.

4.1.3.6 virtual int Wire::read ( ) [pure virtual]

Reads a byte that was transmitted from a slave device to a master after a call to requestFrom()

Returns

The byte read.

Implemented in WireKernelSpace, and WireUserSpace.

4.1.3.7 virtual unsigned char Wire::requestFrom (int *address*, unsigned int *len*) [pure virtual]

Used to request bytes from a slave device.

The bytes may then be retrieved with the available() and read() functions.

Parameters

address	The slave address.
len	The length of data. Need be <= 16 due the FIFO limits. Used to request bytes from a slave device. The bytes may then be retrieved with the available() and read() functions.
address	The slave address.
len	The length of data. Need be <= 16 due the FIFO limits.

Implemented in WireKernelSpace, and WireUserSpace.

4.1.3.8 virtual void Wire::stop ( ) [pure virtual]

Unmap the BSC0 registers.

Implemented in WireKernelSpace, and WireUserSpace.

4.1.3.9 virtual unsigned int Wire::write (unsigned char *b*) [pure virtual]

Queues a single byte for transmission to slave device (in-between calls to beginTransmission() and end↩ Transmission()).

Parameters

b	The byte to be queued.

Returns

1 if the byte was accepted or 0 if the internal FIFO does not accepted.

Implemented in WireKernelSpace, and WireUserSpace.

4.1.3.10 virtual unsigned int Wire::write (const unsigned char * *buf*, unsigned int *len*) [pure virtual]

Queues bytes for transmission to slave device (in-between calls to beginTransmission() and endTransmission()).

Parameters

buf	The bytes to be queued.
len	The number of byte to be queued.

Returns

The number of accepted bytes.

Implemented in WireKernelSpace, and WireUserSpace.

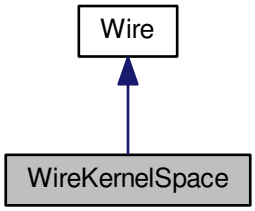
The documentation for this class was generated from the following file:

- Wire.h

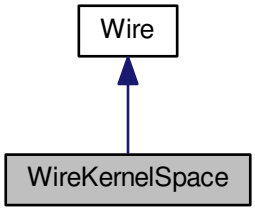
4.2 WireKernelSpace Class Reference

```
#include <WireKernelSpace.h>
```

Inheritance diagram for WireKernelSpace:



Collaboration diagram for WireKernelSpace:



Public Member Functions

- WireKernelSpace (unsigned char channel)
- virtual void begin ()
- virtual void stop ()
- virtual void beginTransmission (int address)
- virtual unsigned char endTransmission ()
- virtual unsigned char requestFrom (int address, unsigned int len)
- virtual unsigned int write (unsigned char b)
- virtual unsigned int write (const unsigned char ∗buf, unsigned int len)
- virtual int available ()
- virtual int read ()
- virtual void flush ()
- void dumpStatus ()

Private Member Functions

- bool isDone ()
- void waitDone ()

Private Attributes

- Bcm2835::Peripheral bsc
- int txSize
- unsigned char channel

4.2.1 Detailed Description

Definition at line 57 of file WireKernelSpace.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 WireKernelSpace::WireKernelSpace (unsigned char *channel*)

Definition at line 4 of file WireKernelSpace.cpp.

4.2.3 Member Function Documentation

4.2.3.1 int WireKernelSpace::available ( ) [virtual]

Returns 1 if there is one or more bytes to be read.

Returns

0 if there is no bytes to be read in the internal FIFO.

Implements Wire.

Definition at line 60 of file WireKernelSpace.cpp.

4.2.3.2 void WireKernelSpace::begin ( ) [virtual]

Initiate the library.

(Only as a master) This should normally be called only once. It maps the BSC0 (0x7E20_5000) registers.

Implements Wire.

Definition at line 9 of file WireKernelSpace.cpp.

4.2.3.3 void WireKernelSpace::beginTransmission (int *address*) [virtual]

Begin a transmission to the I2C slave device with the given address.

Subsequently, queue bytes for transmission with the write() function and transmit them by calling endTransmission().

Parameters

address	The device address.

Implements Wire.

Definition at line 18 of file WireKernelSpace.cpp.

4.2.3.4 void WireKernelSpace::dumpStatus ( )

Prints the status register.

Definition at line 88 of file WireKernelSpace.cpp.

4.2.3.5 unsigned char WireKernelSpace::endTransmission (void) [virtual]

Ends a transmission to a slave device that was begun by beginTransmission() and transmits the bytes that were queued by write().

Returns

Nothing for now.

Implements Wire.

Definition at line 24 of file WireKernelSpace.cpp.

4.2.3.6 void WireKernelSpace::flush ( ) [virtual]

For now, does nothing.

Implements Wire.

Definition at line 71 of file WireKernelSpace.cpp.

4.2.3.7 bool WireKernelSpace::isDone ( ) [private]

Checks if the transmission in complete.

Definition at line 74 of file WireKernelSpace.cpp.

4.2.3.8 int WireKernelSpace::read ( ) [virtual]

Reads a byte that was transmitted from a slave device to a master after a call to requestFrom()

Returns

The byte read.

Implements Wire.

Definition at line 64 of file WireKernelSpace.cpp.

4.2.3.9 unsigned char WireKernelSpace::requestFrom (int *address*, unsigned int *len*) [virtual]

Used to request bytes from a slave device.

The bytes may then be retrieved with the available() and read() functions.

Parameters

address	The slave address.
len	The length of data. Need be <= 16 due the FIFO limits.

Implements Wire.

Definition at line 32 of file WireKernelSpace.cpp.

4.2.3.10 void WireKernelSpace::stop ( ) [virtual]

Unmap the BSC0 registers.

Implements Wire.

Definition at line 14 of file WireKernelSpace.cpp.

4.2.3.11 void WireKernelSpace::waitDone ( ) [private]

Waits for the transmission to be complete.

Definition at line 78 of file WireKernelSpace.cpp.

4.2.3.12 unsigned int WireKernelSpace::write (unsigned char *b*) [virtual]

Queues a single byte for transmission to slave device (in-between calls to beginTransmission() and endTransmission()).

Parameters

b	The byte to be queued.

Returns

1 if the byte was accepted or 0 if the internal FIFO does not accepted.

Implements Wire.

Definition at line 41 of file WireKernelSpace.cpp.

4.2.3.13 unsigned int WireKernelSpace::write (const unsigned char * *buf*, unsigned int *len*) [virtual]

Queues bytes for transmission to slave device (in-between calls to beginTransmission() and endTransmission()).

Parameters

buf	The bytes to be queued.
len	The number of byte to be queued.

Returns

The number of accepted bytes.

Implements Wire.

Definition at line 50 of file WireKernelSpace.cpp.

4.2.4 Member Data Documentation

4.2.4.1 Bcm2835::Peripheral WireKernelSpace::bsc [private]

Definition at line 59 of file WireKernelSpace.h.

4.2.4.2 unsigned char WireKernelSpace::channel [private]

BSC channel (0 or 1)

Definition at line 69 of file WireKernelSpace.h.

4.2.4.3 int WireKernelSpace::txSize [private]

This cannot be bigger then 16.

Definition at line 64 of file WireKernelSpace.h.

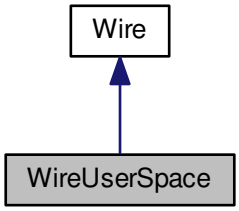
The documentation for this class was generated from the following files:

- WireKernelSpace.h
- WireKernelSpace.cpp

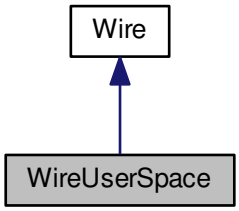
4.3 WireUserSpace Class Reference

```
#include <WireUserSpace.h>
```

Inheritance diagram for WireUserSpace:



Collaboration diagram for WireUserSpace:



Public Member Functions

- WireUserSpace (unsigned char channel)
- virtual void begin ()
- virtual void stop ()
- virtual void beginTransmission (int address)
- virtual unsigned char endTransmission ()
- virtual unsigned char requestFrom (int address, unsigned int len)
- virtual unsigned int write (unsigned char b)
- virtual unsigned int write (const unsigned char ∗buf, unsigned int len)
- virtual int available ()
- virtual int read ()
- virtual void flush ()

Private Attributes

- unsigned char channel
- int fd

4.3.1 Detailed Description

This is a siple Wire library to Raspberry.

It doesn't use the specific i2c module (i2c_dev or i2c_bcm2708) it maps the memory (the BSC0 chunk) into the virtual memory space and handles directly the register.

Thanks to this blog: http://www.susa.net/wordpress/2012/06/raspberry-pi-pcf8563-real-time-clock-

Definition at line 25 of file WireUserSpace.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 WireUserSpace::WireUserSpace (unsigned char *channel*)

Definition at line 4 of file WireUserSpace.cpp.

4.3.3 Member Function Documentation

4.3.3.1 int WireUserSpace::available ( ) [virtual]

Returns 1 if there is one or more bytes to be read.

Returns

0 if there is no bytes to be read in the internal FIFO.

Implements Wire.

Definition at line 50 of file WireUserSpace.cpp.

4.3.3.2 void WireUserSpace::begin ( ) [virtual]

Initiate the library.

(Only as a master) This should normally be called only once. It maps the BSC0 (0x7E20_5000) registers.

Implements Wire.

Definition at line 9 of file WireUserSpace.cpp.

4.3.3.3 void WireUserSpace::beginTransmission (int *address*) [virtual]

Begin a transmission to the I2C slave device with the given address.

Subsequently, queue bytes for transmission with the write() function and transmit them by calling endTransmission().

Parameters

address	The device address.

Implements Wire.

Definition at line 23 of file WireUserSpace.cpp.

4.3.3.4 unsigned char WireUserSpace::endTransmission (void) [virtual]

Ends a transmission to a slave device that was begun by beginTransmission() and transmits the bytes that were queued by write().

Returns

Nothing for now.

Implements Wire.

Definition at line 30 of file WireUserSpace.cpp.

4.3.3.5 void WireUserSpace::flush ( ) [virtual]

For now, does nothing.

Implements Wire.

Definition at line 64 of file WireUserSpace.cpp.

4.3.3.6 int WireUserSpace::read ( ) [virtual]

Reads a byte that was transmitted from a slave device to a master after a call to requestFrom()

Returns

The byte read.

Implements Wire.

Definition at line 54 of file WireUserSpace.cpp.

4.3.3.7 unsigned char WireUserSpace::requestFrom (int *address*, unsigned int *len*) [virtual]

Used to request bytes from a slave device.

The bytes may then be retrieved with the available() and read() functions.

Parameters

address	The slave address.
len	The length of data. Need be <= 16 due the FIFO limits.

Implements Wire.

Definition at line 34 of file WireUserSpace.cpp.

4.3.3.8 void WireUserSpace::stop ( ) [virtual]

Unmap the BSC0 registers.

Implements Wire.

Definition at line 19 of file WireUserSpace.cpp.

4.3.3.9 unsigned int WireUserSpace::write (unsigned char *b*) [virtual]

Queues a single byte for transmission to slave device (in-between calls to beginTransmission() and end↩ Transmission()).

Parameters

b	The byte to be queued.

Returns

1 if the byte was accepted or 0 if the internal FIFO does not accepted.

Implements Wire.

Definition at line 42 of file WireUserSpace.cpp.

4.3.3.10 unsigned int WireUserSpace::write (const unsigned char * *buf*, unsigned int *len*) [virtual]

Queues bytes for transmission to slave device (in-between calls to beginTransmission() and endTransmission()).

Parameters

buf	The bytes to be queued.
len	The number of byte to be queued.

Returns

The number of accepted bytes.

Implements Wire.

Definition at line 46 of file WireUserSpace.cpp.

4.3.4 Member Data Documentation

4.3.4.1 unsigned char WireUserSpace::channel [private]

Channel (0 or 1)

Definition at line 30 of file WireUserSpace.h.

4.3.4.2 int WireUserSpace::fd [private]

File descriptor.

Definition at line 35 of file WireUserSpace.h.

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

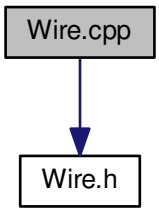
- WireUserSpace.h
- WireUserSpace.cpp

5 File Documentation

5.1 Wire.cpp File Reference

```
#include "Wire.h"
```

Include dependency graph for Wire.cpp:



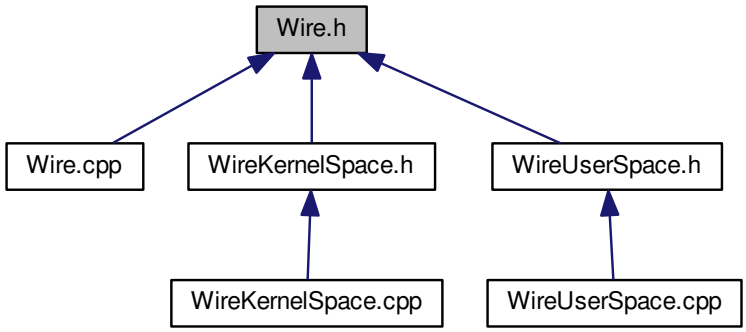
5.2 Wire.cpp

```
00001
00002 #include "Wire.h"
```

```
00003 /*
00004 void Wire::beginTransmission(unsigned char address) {
00005     beginTransmission((int) address);
00006 }
00007
00008 unsigned char Wire::requestFrom(unsigned char address, unsigned char len) {
00009     return requestFrom((int) address, (int) len);
00010 }
00011 */
```

5.3 Wire.h File Reference

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



Classes

- class Wire

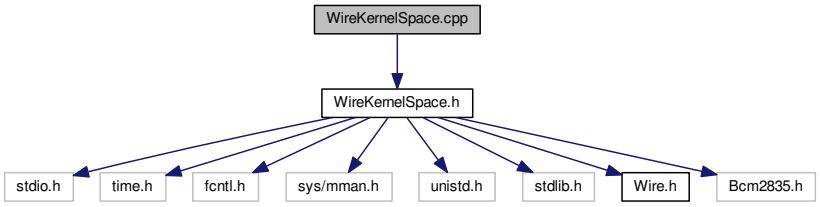
5.4 Wire.h

```
00001
00006 #ifndef __RASPBERRY_WIRE_H__
00007 #define __RASPBERRY_WIRE_H__ 1
00008
00009 class Wire {
00010
00011 public:
00012
00016     virtual ~Wire() {
00017     }
00018
00024     virtual void begin() = 0;
00025
00029     virtual void stop() = 0;
00030
00038     virtual void beginTransmission(int address) = 0;
00039
00047     //void beginTransmission(unsigned char address);
00048
00056     virtual unsigned char endTransmission() = 0;
00057
00066     //unsigned char requestFrom(unsigned char address, unsigned char len);
00067
00076     virtual unsigned char requestFrom(int address, unsigned int len) = 0;
00077
00086     virtual unsigned int write(unsigned char b) = 0;
00087
00096     virtual unsigned int write(const unsigned char* buf, unsigned int len) = 0;
00097
00104     virtual int available() = 0;
```

```
00105
00112     virtual int read() = 0;
00113
00117     virtual void flush() = 0;
00118 };
00119
00120 #endif /* __RASPBERRY_WIRE_H__ */
```

5.5 WireKernelSpace.cpp File Reference

`#include "WireKernelSpace.h"`
Include dependency graph for WireKernelSpace.cpp:



Variables

- WireKernelSpace WireKS0 (0)
- WireKernelSpace WireKS1 (1)

5.5.1 Variable Documentation

5.5.1.1 WireKernelSpace WireKS0(0)

5.5.1.2 WireKernelSpace WireKS1(1)

5.6 WireKernelSpace.cpp

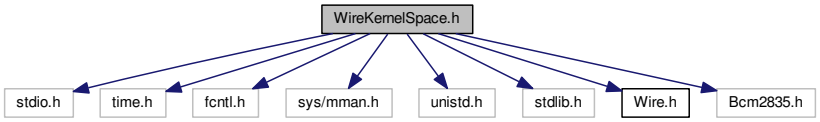
```
00001
00002 #include "WireKernelSpace.h"
00003
00004 WireKernelSpace::WireKernelSpace(unsigned char channel) {
00005     this->channel = (channel & 0x01);
00006     this->txSize = 0;
00007 }
00008
00009 void WireKernelSpace::begin() {
00010     bsc.address = (this->channel == 0) ? BSC0_ADDRESS :
      BSC1_ADDRESS;
00011     Bcm2835::mapPeripheral(&bsc);
00012 }
00013
00014 void WireKernelSpace::stop() {
00015     Bcm2835::unmapPeripheral(&bsc);
00016 }
00017
00018 void WireKernelSpace::beginTransmission(int address) {
00019     BSC_A = (address & 0x7ff);
00020     BSC_DLEN = 0;
00021     txSize = 0;
00022 }
00023
00024 unsigned char WireKernelSpace::endTransmission(void) {
00025     BSC_DLEN = txSize;
00026     BSC_S = CLEAR_STATUS;
00027     BSC_C = START_WRITE;
00028     waitDone();
00029     return txSize;
```

```
00030 }
00031
00032 unsigned char WireKernelSpace::requestFrom(int address, unsigned int len) {
00033     BSC_A = (address & 0x7ff);
00034     BSC_DLEN = len;
00035     BSC_S = CLEAR_STATUS;
00036     BSC_C = START_READ;
00037     waitDone();
00038     return 0;
00039 }
00040
00041 unsigned int WireKernelSpace::write(unsigned char b) {
00042     if (BSC_S & BSC_S_TXD) {
00043         txSize++;
00044         BSC_FIFO = b;
00045         return 1;
00046     }
00047     return 0;
00048 }
00049
00050 unsigned int WireKernelSpace::write(const unsigned char* buf, unsigned int len) {
00051     unsigned int i;
00052     for (i = 0; i < len; i++) {
00053         if (!write(buf[i])) {
00054             break;
00055         }
00056     }
00057     return i;
00058 }
00059
00060 int WireKernelSpace::available() {
00061     return (bool) (BSC_S & BSC_S_RXD);
00062 }
00063
00064 int WireKernelSpace::read() {
00065     dumpStatus();
00066     unsigned char b = BSC_FIFO;
00067     printf("read: %x\n", b);
00068     return b;
00069 }
00070
00071 void WireKernelSpace::flush() {
00072 }
00073
00074 bool WireKernelSpace::isDone() {
00075     return (bool) (BSC_S & BSC_S_DONE);
00076 }
00077
00078 void WireKernelSpace::waitDone() {
00079     int timeout = 60;
00080     while(!isDone() && --timeout) {
00081         usleep(1000);
00082     }
00083     if(timeout == 0) {
00084         perror("#waitDone: Timeout! Something went wrong.\n");
00085     }
00086 }
00087
00088 void WireKernelSpace::dumpStatus() {
00089     unsigned int s = BSC_S;
00090     printf("BSC_S: ERR=%d  RXF=%d  TXE=%d  RXD=%d  TXD=%d  RXR=%d  TXW=%d  DONE=%d  TA=%d\n",
00091         (s & BSC_S_ERR) != 0,
00092         (s & BSC_S_RXF) != 0,
00093         (s & BSC_S_TXE) != 0,
00094         (s & BSC_S_RXD) != 0,
00095         (s & BSC_S_TXD) != 0,
00096         (s & BSC_S_RXR) != 0,
00097         (s & BSC_S_TXW) != 0,
00098         (s & BSC_S_DONE) != 0,
00099         (s & BSC_S_TA) != 0);
00100 }
00101
00102 WireKernelSpace WireKS0(0);
00103 WireKernelSpace WireKS1(1);
```

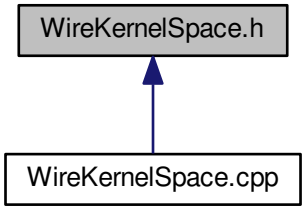
5.7 WireKernelSpace.h File Reference

```
#include <stdio.h>
#include <time.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <unistd.h>
#include <stdlib.h>
#include <Wire.h>
#include <Bcm2835.h>
```

Include dependency graph for WireKernelSpace.h:



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



Classes

- class WireKernelSpace

Macros

- #define BSC0_ADDRESS 0x205000
- #define BSC1_ADDRESS 0x804000
- #define BSC_C *((unsigned int *)(bsc.mem) + 0x00)
- #define BSC_S *((unsigned int *)(bsc.mem) + 0x01)
- #define BSC_DLEN *((unsigned int *)(bsc.mem) + 0x02)
- #define BSC_A *((unsigned int *)(bsc.mem) + 0x03)
- #define BSC_FIFO *((unsigned int *)(bsc.mem) + 0x04)
- #define BSC_C_I2CEN (0x01 << 15)
- #define BSC_C_INTR (0x01 << 10)
- #define BSC_C_INTT (0x01 << 9)
- #define BSC_C_INTD (0x01 << 8)

- #define BSC_C_ST (0x01 << 7)
- #define BSC_C_CLEAR (0x01 << 4)
- #define BSC_C_READ (0x01 << 0)
- #define START_READ BSC_C_I2CEN | BSC_C_ST | BSC_C_CLEAR | BSC_C_READ
- #define START_WRITE BSC_C_I2CEN | BSC_C_ST
- #define BSC_S_CLKT (0x01 << 9)
- #define BSC_S_ERR (0x01 << 8)
- #define BSC_S_RXF (0x01 << 7)
- #define BSC_S_TXE (0x01 << 6)
- #define BSC_S_RXD (0x01 << 5)
- #define BSC_S_TXD (0x01 << 4)
- #define BSC_S_RXR (0x01 << 3)
- #define BSC_S_TXW (0x01 << 2)
- #define BSC_S_DONE (0x01 << 1)
- #define BSC_S_TA (0x01 << 0)
- #define CLEAR_STATUS BSC_S_CLKT | BSC_S_ERR | BSC_S_DONE

Variables

- WireKernelSpace WireKS0
- WireKernelSpace WireKS1

5.7.1 Macro Definition Documentation

5.7.1.1 #define BSC0_ADDRESS 0x205000

This is a siple Wire library to Raspberry.

It doesn't use the specific i2c module (i2c_dev or i2c_bcm2708) it maps the memory (the BSC0 chunk) into the virtual memory space and handles directly the register.

Thanks to this blog: http://www.susa.net/wordpress/2012/06/raspberry-pi-pcf8563-real-time-clock-

Definition at line 24 of file WireKernelSpace.h.

5.7.1.2 #define BSC1_ADDRESS 0x804000

Definition at line 25 of file WireKernelSpace.h.

5.7.1.3 #define BSC_A *((unsigned int *)(bsc.mem) + 0x03)

Definition at line 30 of file WireKernelSpace.h.

5.7.1.4 #define BSC_C *((unsigned int *)(bsc.mem) + 0x00)

Definition at line 27 of file WireKernelSpace.h.

5.7.1.5 #define BSC_C_CLEAR (0x01 << 4)

Definition at line 38 of file WireKernelSpace.h.

5.7.1.6 #define BSC_C_I2CEN (0x01 << 15)

Definition at line 33 of file WireKernelSpace.h.

5.7.1.7 #define BSC_C_INTD (0x01 << 8)

Definition at line 36 of file WireKernelSpace.h.

5.7.1.8 #define BSC_C_INTR (0x01 << 10)

Definition at line 34 of file WireKernelSpace.h.

5.7.1.9 #define BSC_C_INTT (0x01 << 9)

Definition at line 35 of file WireKernelSpace.h.

5.7.1.10 #define BSC_C_READ (0x01 << 0)

Definition at line 39 of file WireKernelSpace.h.

5.7.1.11 #define BSC_C_ST (0x01 << 7)

Definition at line 37 of file WireKernelSpace.h.

5.7.1.12 #define BSC_DLEN *((unsigned int *)(bsc.mem) + 0x02)

Definition at line 29 of file WireKernelSpace.h.

5.7.1.13 #define BSC_FIFO *((unsigned int *)(bsc.mem) + 0x04)

Definition at line 31 of file WireKernelSpace.h.

5.7.1.14 #define BSC_S *((unsigned int *)(bsc.mem) + 0x01)

Definition at line 28 of file WireKernelSpace.h.

5.7.1.15 #define BSC_S_CLKT (0x01 << 9)

Definition at line 44 of file WireKernelSpace.h.

5.7.1.16 #define BSC_S_DONE (0x01 << 1)

Definition at line 52 of file WireKernelSpace.h.

5.7.1.17 #define BSC_S_ERR (0x01 << 8)

Definition at line 45 of file WireKernelSpace.h.

5.7.1.18 #define BSC_S_RXD (0x01 << 5)

Definition at line 48 of file WireKernelSpace.h.

5.7.1.19 #define BSC_S_RXF (0x01 << 7)

Definition at line 46 of file WireKernelSpace.h.

5.7.1.20 #define BSC_S_RXR (0x01 << 3)

Definition at line 50 of file WireKernelSpace.h.

5.7.1.21 #define BSC_S_TA (0x01 << 0)

Definition at line 53 of file WireKernelSpace.h.

5.7.1.22 #define BSC_S_TXD (0x01 << 4)

Definition at line 49 of file WireKernelSpace.h.

5.7.1.23 #define BSC_S_TXE (0x01 << 6)

Definition at line 47 of file WireKernelSpace.h.

5.7.1.24 #define BSC_S_TXW (0x01 << 2)

Definition at line 51 of file WireKernelSpace.h.

5.7.1.25 #define CLEAR_STATUS BSC_S_CLKT | BSC_S_ERR | BSC_S_DONE

Definition at line 55 of file WireKernelSpace.h.

5.7.1.26 #define START_READ BSC_C_I2CEN | BSC_C_ST | BSC_C_CLEAR | BSC_C_READ

Definition at line 41 of file WireKernelSpace.h.

5.7.1.27 #define START_WRITE BSC_C_I2CEN | BSC_C_ST

Definition at line 42 of file WireKernelSpace.h.

5.7.2 Variable Documentation

5.7.2.1 WireKernelSpace WireKS0

5.7.2.2 WireKernelSpace WireKS1

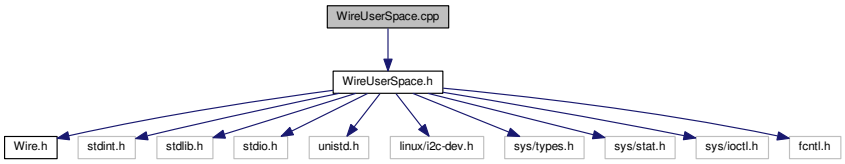
5.8 WireKernelSpace.h

```
00001
00011 #ifndef __RASPBERRY_WIRE_KERNEL_SPACE_H__
00012 #define __RASPBERRY_WIRE_KERNEL_SPACE_H__ 1
00013
00014 #include <stdio.h>
00015 #include <time.h>
00016 #include <fcntl.h>
00017 #include <sys/mman.h>
00018 #include <unistd.h>
00019 #include <stdlib.h>
00020
00021 #include <Wire.h>
00022 #include <Bcm2835.h>
00023
00024 #define BSC0_ADDRESS    0x205000
00025 #define BSC1_ADDRESS    0x804000
00026
00027 #define BSC_C          *((unsigned int *)(bsc.mem) + 0x00)
00028 #define BSC_S          *((unsigned int *)(bsc.mem) + 0x01)
00029 #define BSC_DLEN       *((unsigned int *)(bsc.mem) + 0x02)
00030 #define BSC_A          *((unsigned int *)(bsc.mem) + 0x03)
00031 #define BSC_FIFO       *((unsigned int *)(bsc.mem) + 0x04)
00032
00033 #define BSC_C_I2CEN     (0x01 << 15)
00034 #define BSC_C_INTR      (0x01 << 10)
00035 #define BSC_C_INTT      (0x01 << 9)
00036 #define BSC_C_INTD      (0x01 << 8)
00037 #define BSC_C_ST        (0x01 << 7)
00038 #define BSC_C_CLEAR     (0x01 << 4)
00039 #define BSC_C_READ      (0x01 << 0)
00040
00041 #define START_READ      BSC_C_I2CEN | BSC_C_ST | BSC_C_CLEAR | BSC_C_READ
00042 #define START_WRITE     BSC_C_I2CEN | BSC_C_ST
00043
00044 #define BSC_S_CLKT      (0x01 << 9)
00045 #define BSC_S_ERR       (0x01 << 8)
00046 #define BSC_S_RXF       (0x01 << 7)
00047 #define BSC_S_TXE       (0x01 << 6)
00048 #define BSC_S_RXD       (0x01 << 5)
00049 #define BSC_S_TXD       (0x01 << 4)
00050 #define BSC_S_RXR       (0x01 << 3)
00051 #define BSC_S_TXW       (0x01 << 2)
00052 #define BSC_S_DONE      (0x01 << 1)
00053 #define BSC_S_TA        (0x01 << 0)
00054
00055 #define CLEAR_STATUS    BSC_S_CLKT | BSC_S_ERR | BSC_S_DONE
```

```
00056
00057 class WireKernelSpace : public Wire {
00058
00059     Bcm2835::Peripheral bsc;
00060
00064     int txSize;
00065
00069     unsigned char channel;
00070
00071 public:
00072
00073     WireKernelSpace(unsigned char channel);
00074
00080     virtual void begin();
00081
00085     virtual void stop();
00086
00094     virtual void beginTransmission(int address);
00095
00103     virtual unsigned char endTransmission();
00104
00113     virtual unsigned char requestFrom(int address, unsigned int len);
00114
00123     virtual unsigned int write(unsigned char b);
00124
00133     virtual unsigned int write(const unsigned char* buf, unsigned int len);
00134
00141     virtual int available();
00142
00149     virtual int read();
00150
00154     virtual void flush();
00155
00159     void dumpStatus();
00160
00161 private:
00162
00166     bool isDone();
00167
00171     void waitDone();
00172 };
00173
00174 extern WireKernelSpace WireKS0;
00175 extern WireKernelSpace WireKS1;
00176
00177 #endif /* __RASPBERRY_WIRE_KERNEL_SPACE_H__ */
```

5.9 WireUserSpace.cpp File Reference

`#include "WireUserSpace.h"`
Include dependency graph for WireUserSpace.cpp:



Variables

- WireUserSpace WireUS0 (0)
- WireUserSpace WireUS1 (1)

5.9.1 Variable Documentation

5.9.1.1 WireUserSpace WireUS0(0)

5.9.1.2 WireUserSpace WireUS1(1)

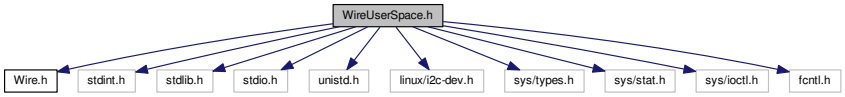
5.10 WireUserSpace.cpp

```
00001
00002 #include "WireUserSpace.h"
00003
00004 WireUserSpace::WireUserSpace(unsigned char channel) {
00005     this->channel = channel;
00006     this->fd = 0;
00007 }
00008
00009 void WireUserSpace::begin() {
00010     char f[11] = "/dev/i2c-0";
00011     f[9] = '0' + channel;
00012     fd = open(f, O_RDWR);
00013     if (fd < 0) {
00014         perror("Cannot open i2c bus.");
00015         exit(1);
00016     }
00017 }
00018
00019 void WireUserSpace::stop() {
00020     close(fd);
00021 }
00022
00023 void WireUserSpace::beginTransmission(int address) {
00024     if (ioctl(fd, I2C_SLAVE, address) < 0) {
00025         perror("Cannot set i2c address.");
00026         exit(1);
00027     }
00028 }
00029
00030 unsigned char WireUserSpace::endTransmission(void) {
00031     return 0;
00032 }
00033
00034 unsigned char WireUserSpace::requestFrom(int address, unsigned int len) {
00035     if (ioctl(fd, I2C_SLAVE, address) < 0) {
00036         perror("Cannot set i2c address.");
00037         exit(1);
00038     }
00039     return len;
00040 }
00041
00042 unsigned int WireUserSpace::write(unsigned char b) {
00043     return write(&b, 1);
00044 }
00045
00046 unsigned int WireUserSpace::write(const unsigned char* buf, unsigned int len) {
00047     return ::write(fd, buf, (int)len);
00048 }
00049
00050 int WireUserSpace::available() {
00051     return 1;
00052 }
00053
00054 int WireUserSpace::read() {
00055     char buf[1];
00056     if (::read(fd, buf, 1) != 1) {
00057         perror("Cannot read i2c.");
00058         exit(1);
00059     }
00060     return buf[0];
00061 }
00062
00063
00064 void WireUserSpace::flush() {
00065 }
00066
00067 WireUserSpace WireUS0(0);
00068 WireUserSpace WireUS1(1);
```

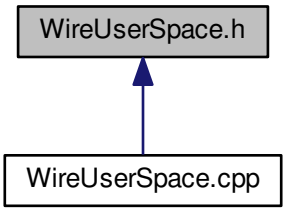
5.11 WireUserSpace.h File Reference

```
#include <Wire.h>
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <linux/i2c-dev.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <fcntl.h>
```

Include dependency graph for WireUserSpace.h:



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



Classes

- class WireUserSpace

Variables

- WireUserSpace WireUS0
- WireUserSpace WireUS1

5.11.1 Variable Documentation

5.11.1.1 WireUserSpace WireUS0

5.11.1.2 WireUserSpace WireUS1

5.12 WireUserSpace.h

```
00001
```

```
00011 #ifndef __RASPBERRY_WIRE_USER_SPACE_H__
00012 #define __RASPBERRY_WIRE_USER_SPACE_H__ 1
00013
00014 #include <Wire.h>
00015 #include <stdint.h>
00016 #include <stdlib.h>
00017 #include <stdio.h>
00018 #include <unistd.h>
00019 #include <linux/i2c-dev.h>
00020 #include <sys/types.h>
00021 #include <sys/stat.h>
00022 #include <sys/ioctl.h>
00023 #include <fcntl.h>
00024
00025 class WireUserSpace : public Wire {
00026
00030     unsigned char channel;
00031
00035     int fd;
00036
00037 public:
00038
00039     WireUserSpace(unsigned char channel);
00040
00046     virtual void begin();
00047
00051     virtual void stop();
00052
00060     virtual void beginTransmission(int address);
00061
00069     virtual unsigned char endTransmission();
00070
00079     virtual unsigned char requestFrom(int address, unsigned int len);
00080
00089     virtual unsigned int write(unsigned char b);
00090
00099     virtual unsigned int write(const unsigned char* buf, unsigned int len);
00100
00107     virtual int available();
00108
00115     virtual int read();
00116
00120     virtual void flush();
00121 };
00122
00123 extern WireUserSpace WireUS0;
00124 extern WireUserSpace WireUS1;
00125
00126 #endif /* __RASPBERRY_WIRE_USER_SPACE_H__ */
```

Index