Exploring BeagleBone

V1.0

Generated by Doxygen 1.8.8

Sun Apr 5 2015 02:15:33

Contents

Chapter 1

Exploring BeagleBone Main Documentation

1.1 Introduction

This is the API documentation for the Exploring BeagleBone library. On this web page, click on the **Classes tab** at the top of the page. You can also click on the **Files tab** to get a list of the files in the project. To return to the main page for the book follow the link Exploring BeagleBone

You can get a full list of the available classes: Annotated Classes

You can get a full list of the files at: Annotated Files

A PDF version of this documentation is available here: Exploring BeagleBone API Documentation (PDF)

1.2 Installation

To clone this library use the command: git clone https://github.com/derekmolloy/exploringBB.git

To build this documentation enter the exploringBB/library/docs directory and type: **doxygen ExploringBB.Doxyfile**.

This generates the html documents in the html directory and LaTeX files in the latex directory. Do not try to build this on the BeagleBone as the packages required are too large.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 exploringBB Namespace Reference

Data Structures

- class ADXL345

  Specific class for the ADXL345 Accelerometer.
- class BMA180

  A class to control a BMA180 accelerometer (untested)
- class BusDevice

  This class is the parent of I2C and SPI devices, so that devices that use both SPI and I2C interfaces can use those interfaces interchangeably. Because it contains abstract methods, the child classes MUST implement the methods that are listed in this class.
- class DCMotor

  A generic DC motor class that controls a motor driver board using a PWM signal,and a GPIO state to control the motor direction.
- class GPIO

  GPIO class for input and output functionality on a single GPIO pin.
- class I2CDevice

  Generic I2C Device class that can be used to connect to any type of I2C device and read or write to its registers.
- class ITG3200

  A class to interface with the ITG3200 gyroscope (untested)
- class LCDCharacterDisplay

  A class that provides an interface to an LCD character module. It provices support for multiple rows and columns and provides methods for formatting and printing text. You should use a 4 wire interface and a 74XX595 to communicate with the display module.
- class PWM

  A class to control a basic PWM output – you must know the exact sysfs filename for the PWM output.
- class Servo

  An extremely basic Servo class stub – does nothing more than the PWM class but is here for future use.
- class SevenSegmentDisplay

  A class that allows you to drive an array of 7 segment displays using an array of 74XX595 ICs.
- class SocketClient

  A class that encapsulates a socket client to be used for network communication.
- class SocketServer

  A class that encapsulates a server socket for network communication.
- class SPIDevice

  Generic SPI Device class that can be used to connect to any type of SPI device and read or write to its registers.

- class StepperMotor

  A class to control a stepper motor using a motor driver board, such as the Easy Driver board, or compatible. The class uses five GPIOs to control each motor.

Typedefs

- typedef int(∗ CallbackType)(int)

Functions

- void ∗ threadedToggle (void ∗value)
- void ∗ threadedPoll (void ∗value)
- int write (string path, string filename, string value)
- string read (string path, string filename)
- int write (string path, string filename, int value)
- void ∗ threadedStep (void ∗value)

6.1.1 Typedef Documentation

6.1.1.1 typedef int(∗ exploringBB::CallbackType)(int)

6.1.2 Function Documentation

6.1.2.1 string exploringBB::read (string *path*, string *filename*)

Helper read function that reads a single string value to a file from the path provided

Parameters

path	The sysfs path of the file to be read
filename	Filename The file to be written to in that path

Returns

```
57                                          {
58     ifstream fs;
59     fs.open((path + filename).c_str());
60     if (!fs.is_open()){
61             perror("GPIO: read failed to open file ");
62      }
63     string input;
64     getline(fs,input);
65     fs.close();
66     return input;
67 }
```

6.1.2.2 void ∗ exploringBB::threadedPoll (void ∗ *value*)

```
272                                 {
273         GPIO *gpio = static_cast<GPIO*>(value);
274         while(gpio->threadRunning){
275                 gpio->callbackFunction(gpio->waitForEdge());
276                 usleep(gpio->debounceTime * 1000);
277         }
278         return 0;
279 }
```

6.1.2.3 void * exploringBB::threadedStep (void * *value*)

```
179                                 {
180         StepperMotor *stepper = static_cast<StepperMotor*>(value);
181         while(stepper->threadRunning){
182                 stepper->step();
183                 usleep(stepper->threadedStepPeriod * 1000);  // convert from ms to us
184                 if(stepper->threadedStepNumber>0) stepper->threadedStepNumber--;
185                 if(stepper->threadedStepNumber==0) stepper->threadRunning = false;
186         }
187         return 0;
188 }
```

6.1.2.4 void * exploringBB::threadedToggle (void * *value*)

```
218                                  {
219         GPIO *gpio = static_cast<GPIO*>(value);
220         bool isHigh = (bool) gpio->getValue(); //find current value
221         while(gpio->threadRunning){
222                 if (isHigh)     gpio->setValue(GPIO::HIGH);
223                 else gpio->setValue(GPIO::LOW);
224                 usleep(gpio->togglePeriod * 500);
225                 isHigh=!isHigh;
226                 if(gpio->toggleNumber>0) gpio->toggleNumber--;
227                 if(gpio->toggleNumber==0) gpio->threadRunning=false;
228         }
229         return 0;
230 }
```

6.1.2.5 int exploringBB::write (string *path*, string *filename*, string *value*)

Helper write function that writes a single string value to a file in the path provided

Parameters

path	The sysfs path of the file to be modified
filename	The file to be written to in that path
value	The value to be written to the file

Returns

```
40                                                  {
41     ofstream fs;
42     fs.open((path + filename).c_str());
43     if (!fs.is_open()){
44             perror("GPIO: write failed to open file ");
45             return -1;
46     }
47     fs << value;
48     fs.close();
49     return 0;
50 }
```

6.1.2.6 int exploringBB::write (string *path*, string *filename*, int *value*)

Private write method that writes a single int value to a file in the path provided

Parameters

path	The sysfs path of the file to be modified

filename	The file to be written to in that path
value	The int value to be written to the file

Returns

```
76                                                  {
77     stringstream s;
78     s << value;
79     return write(path,filename,s.str());
80 }
```

Chapter 7

Data Structure Documentation

7.1 exploringBB::ADXL345 Class Reference

Specific class for the ADXL345 Accelerometer.

`#include <ADXL345.h>`

Public Types

- enum RANGE { PLUSMINUS_2_G = 0, PLUSMINUS_4_G = 1, PLUSMINUS_8_G = 2, PLUSMINUS_16_G = 3 }

  An enumeration to define the gravity range of the sensor.
- enum RESOLUTION { NORMAL = 0, HIGH = 1 }

  The resolution of the sensor. High is only available in +/- 16g range.

Public Member Functions

- ADXL345 (BusDevice *busDevice)
- virtual int readSensorState ()
- virtual void setRange (ADXL345::RANGE range)
- virtual ADXL345::RANGE getRange ()
- virtual void setResolution (ADXL345::RESOLUTION resolution)
- virtual ADXL345::RESOLUTION getResolution ()
- virtual short getAccelerationX ()
- virtual short getAccelerationY ()
- virtual short getAccelerationZ ()
- virtual float getPitch ()
- virtual float getRoll ()
- virtual void displayPitchAndRoll (int iterations=600)
- virtual ~ADXL345 ()

7.1.1 Detailed Description

Specific class for the ADXL345 Accelerometer.

7.1.2 Member Enumeration Documentation

7.1.2.1 enum exploringBB::ADXL345::RANGE

An enumeration to define the gravity range of the sensor.

Enumerator

PLUSMINUS_2_G plus/minus 2g

PLUSMINUS_4_G plus/minus 4g

PLUSMINUS_8_G plus/minus 8g

PLUSMINUS_16_G plus/minus 16g

```
43                  {
44                PLUSMINUS_2_G              = 0,
45                PLUSMINUS_4_G              = 1,
46                PLUSMINUS_8_G              = 2,
47                PLUSMINUS_16_G            = 3
48        };
```

7.1.2.2 enum exploringBB::ADXL345::RESOLUTION

The resolution of the sensor. High is only available in +/- 16g range.

Enumerator

NORMAL NORMAL 10-bit resolution.

HIGH HIGH 13-bit resolution.

```
50                  {
51                NORMAL = 0,
52                HIGH = 1
53        };
```

7.1.3 Constructor & Destructor Documentation

7.1.3.1 exploringBB::ADXL345::ADXL345 (BusDevice * *busDevice*)

The constructor for the ADXL345 accelerometer object. It passes the bus number and the device address (with is 0x53 by default) to the constructor of I2CDevice. All of the states are initialized and the registers are updated.

Parameters

I2CBus	The bus number that the ADXL345 device is on - typically 0 or 1
I2CAddress	The address of the ADLX345 device (default 0x53, but can be altered)

```
127                                 {
128         //this->I2CAddress = I2CAddress;
129         //this->I2CBus = I2CBus;
130         this->device = busDevice;
131         this->accelerationX = 0;
132         this->accelerationY = 0;
133         this->accelerationZ = 0;
134         this->pitch = 0.0f;
135         this->roll = 0.0f;
136         this->registers = NULL;
137         this->range = ADXL345::PLUSMINUS_16_G;
138         this->resolution = ADXL345::HIGH;
139         this->device->writeRegister(POWER_CTL, 0x08);
140         this->updateRegisters();
141 }
```

7.1.3.2 exploringBB::ADXL345::~ADXL345 ( ) [virtual]

```
196 {}
```

7.1.4 Member Function Documentation

7.1.4.1 void exploringBB::ADXL345::displayPitchAndRoll (int *iterations* = 600) [virtual]

Useful debug method to display the pitch and roll values in degrees on a single standard output line

Parameters

iterations	The number of 0.1s iterations to take place.

```
186                                                 {
187         int count = 0;
188         while(count < iterations){
189                 cout << "Pitch:"<< this->getPitch() << " Roll:" << this->
      getRoll() << "     \r"<<flush;
190                 usleep(100000);
191                 this->readSensorState();
192                 count++;
193         }
194 }
```

7.1.4.2 virtual short exploringBB::ADXL345::getAccelerationX ( ) [inline], [virtual]

```
75 { return accelerationX; }
```

7.1.4.3 virtual short exploringBB::ADXL345::getAccelerationY ( ) [inline], [virtual]

```
76 { return accelerationY; }
```

7.1.4.4 virtual short exploringBB::ADXL345::getAccelerationZ ( ) [inline], [virtual]

```
77 { return accelerationZ; }
```

7.1.4.5 virtual float exploringBB::ADXL345::getPitch ( ) [inline], [virtual]

```
78 { return pitch; }
```

7.1.4.6 virtual ADXL345::RANGE exploringBB::ADXL345::getRange ( ) [inline], [virtual]

```
71 { return this->range; }
```

7.1.4.7 virtual ADXL345::RESOLUTION exploringBB::ADXL345::getResolution ( ) [inline], [virtual]

```
73 { return this->resolution; }
```

7.1.4.8 virtual float exploringBB::ADXL345::getRoll ( ) [inline], [virtual]

```
79 { return roll; }
```

7.1.4.9 int exploringBB::ADXL345::readSensorState ( ) [virtual]

Read the sensor state. This method checks that the device is being correctly read by using the device ID of the ADXL345 sensor. It will read in the accelerometer registers and pass them to the combineRegisters() method to be processed.

Returns

0 if the registers are successfully read and -1 if the device ID is incorrect.

```
149                             {
150         this->registers = this->device->readRegisters(BUFFER_SIZE, 0x00);
151         if(*this->registers!=0xe5){
152                 perror("ADXL345: Failure Condition - Sensor ID not Verified");
153                 return -1;
154         }
155         this->accelerationX = this->combineRegisters(*(registers+DATAX1), *(registers+
    DATAX0));
156         this->accelerationY = this->combineRegisters(*(registers+DATAY1), *(registers+
    DATAY0));
157         this->accelerationZ = this->combineRegisters(*(registers+DATAZ1), *(registers+
    DATAZ0));
158         this->resolution = (ADXL345::RESOLUTION) (((*(registers+
    DATA_FORMAT))&0x08)>>3);
159         this->range = (ADXL345::RANGE) ((*(registers+DATA_FORMAT))&0x03);
160         this->calculatePitchAndRoll();
161         return 0;
162 }
```

7.1.4.10 void exploringBB::ADXL345::setRange (ADXL345::RANGE *range*) [virtual]

Set the ADXL345 gravity range according to the RANGE enumeration

Parameters

range	One of the four possible gravity ranges defined by the RANGE enumeration

```
168                                         {
169         this->range = range;
170         updateRegisters();
171 }
```

7.1.4.11 void exploringBB::ADXL345::setResolution (ADXL345::RESOLUTION *resolution*) [virtual]

Set the ADXL345 resolution according to the RESOLUTION enumeration

Parameters

resolution	either HIGH or NORMAL resolution. HIGH resolution is only available if the range is set to +/- 16g

```
177                                                 {
178         this->resolution = resolution;
179         updateRegisters();
180 }
```

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

- /home/molloyd/exploringBB/library/sensor/ADXL345.h
- /home/molloyd/exploringBB/library/sensor/ADXL345.cpp

7.2 exploringBB::BMA180 Class Reference

A class to control a BMA180 accelerometer (untested)

`#include <BMA180.hxx>`

Public Member Functions

- BMA180 (BusDevice *device)
- void displayMode (int iterations)
- int readFullSensorState ()
- int setRange (BMA180::RANGE range)
- BMA180::RANGE getRange ()
- int setBandwidth (BMA180::BANDWIDTH bandwidth)
- BMA180::BANDWIDTH getBandwidth ()
- int setModeConfig (BMA180::MODECONFIG mode)
- BMA180::MODECONFIG getModeConfig ()
- float getTemperature ()
- int getAccelerationX ()
- int getAccelerationY ()
- int getAccelerationZ ()
- float getPitch ()
- float getRoll ()
- virtual ~BMA180 ()

7.2.1 Detailed Description

A class to control a BMA180 accelerometer (untested)

7.2.2 Constructor & Destructor Documentation

7.2.2.1 exploringBB::BMA180::BMA180 (BusDevice * *device*)

```
43                                  {
44          this->device = device;
45          readFullSensorState();
46  }
```

7.2.2.2 virtual exploringBB::BMA180::~BMA180 ( ) [virtual]

7.2.3 Member Function Documentation

7.2.3.1 void exploringBB::BMA180::displayMode (int *iterations*)

```
80                                      {
81          for(int i=0; i<iterations; i++){
82                  this->readFullSensorState();
83                  printf("Rotation (%d, %d, %d)", accelerationX, accelerationY, accelerationZ);
84          }
85  }
```

7.2.3.2 int exploringBB::BMA180::getAccelerationX ( ) [inline]

```
100 { return accelerationX; }
```

7.2.3.3 int exploringBB::BMA180::getAccelerationY ( ) [inline]

```
101 { return accelerationY; }
```

7.2.3.4 int exploringBB::BMA180::getAccelerationZ ( ) [inline]

```
102 { return accelerationZ; }
```

7.2.3.5 BMA180::BANDWIDTH exploringBB::BMA180::getBandwidth ( )

```
139                                     {
140         this->readFullSensorState();
141         char temp = *(registers+BMA_BANDWIDTH);   //bits 7->4
142         //char temp = this->readI2CDeviceByte(BANDWIDTH);  //bits 7,6,5,4
143 //      cout << "The value of bandwidth returned is: " << (int)temp << endl;
144         temp = temp & 0b11110000;
145         temp = temp>>4;
146 //      cout << "The current bandwidth is: " << (int)temp << endl;
147         this->bandwidth = (BMA180::BANDWIDTH) temp;
148         return this->bandwidth;
149 }
```

7.2.3.6 BMA180::MODECONFIG exploringBB::BMA180::getModeConfig ( )

```
164                                       {
165         //char temp = dataBuffer[MODE_CONFIG];   //bits 1,0
166         //char temp = this->readI2CDeviceByte(MODE_CONFIG);  //bits 1,0
167         this->readFullSensorState();
168     char temp = *(registers+MODE_CONFIG);
169         temp = temp & 0b00000011;
170         this->modeConfig = (BMA180::MODECONFIG) temp;
171         return this->modeConfig;
172 }
```

7.2.3.7 float exploringBB::BMA180::getPitch ( ) [inline]

```
104 { return pitch; }  // in degrees
```

7.2.3.8 BMA180::RANGE exploringBB::BMA180::getRange ( )

```
114                                 {
115         this->readFullSensorState();
116         char temp = *(registers+BMA_RANGE);
117         //char temp = this->readI2CDeviceByte(RANGE);  //bits 3,2,1
118         temp = temp & 0b00001110;
119         temp = temp>>1;
120         //cout << "The current range is: " << (int)temp << endl;
121         this->range = (BMA180::RANGE) temp;
122         return this->range;
123 }
```

7.2.3.9 float exploringBB::BMA180::getRoll ( ) [inline]

```
105 { return roll; }  // in degrees
```

7.2.3.10 float exploringBB::BMA180::getTemperature ( )

```
91                                {
92
93         int offset = -40;  // -40 degrees C
94         this->readFullSensorState();
95         char temp = *(registers+BMA_TEMP);   // = -80C 0b10000000  0b00000010; = +25C
96         //char temp = this->readI2CDeviceByte(TEMP);
97         //this->readFullSensorState();
98     //char temp = dataBuffer[TEMP];
99         int temperature;
100         if(temp&0x80)   {
```

```
101                 temp = ~temp + 0b00000001;
102                 temperature = 128 - temp;
103         }
104         else {
105                 temperature = 128 + temp;
106         }
107         this->temperature = offset + ((float)temperature*0.5f);
108         //cout << "The temperature is " << this->temperature << endl;
109         //int temp_off = dataBuffer[0x37]>>1;
110         //cout << "Temperature offset raw value is: " << temp_off << endl;
111         return this->temperature;
112 }
```

7.2.3.11 int exploringBB::BMA180::readFullSensorState ( )

```
56                              {
57         this->registers = this->device->readRegisters(BUFFER_SIZE, 0x00);
58         if(*this->registers!=0x03){
59                 perror("BMA180: Failure Condition - Sensor ID not Verified");
60                 return -1;
61         }
62     this->accelerationX = convertAcceleration(ACC_X_MSB, ACC_X_LSB);
63     this->accelerationY = convertAcceleration(ACC_Y_MSB, ACC_Y_LSB);
64     this->accelerationZ = convertAcceleration(ACC_Z_MSB, ACC_Z_LSB);
65     this->calculatePitchAndRoll();
66     //cout << "Pitch:" << this->getPitch() << "   Roll:" << this->getRoll() <<  endl;
67     return 0;
68 }
```

7.2.3.12 int exploringBB::BMA180::setBandwidth (BMA180::BANDWIDTH *bandwidth*)

```
151                                                 {
152         this->readFullSensorState();
153     char current = *(registers+BMA_BANDWIDTH);  //bits 7->4
154         char temp = bandwidth << 4; //move value into bits 7,6,5,4
155         current = current & 0b00001111; //clear the current bits 7,6,5,4
156         temp = current | temp;
157         if(this->device->writeRegister(BMA_BANDWIDTH,temp)!=0){
158                 perror("Failure to update BANDWIDTH value\n");
159                 return 1;
160         }
161         return 0;
162 }
```

7.2.3.13 int exploringBB::BMA180::setModeConfig (BMA180::MODECONFIG *mode*)

7.2.3.14 int exploringBB::BMA180::setRange (BMA180::RANGE *range*)

```
125                                     {
126         //char current = this->readI2CDeviceByte(RANGE);  //bits 3,2,1
127         this->readFullSensorState();
128         char current = *(registers+BMA_RANGE);
129         char temp = range << 1; //move value into bits 3,2,1
130         current = current & 0b11110001; //clear the current bits 3,2,1
131         temp = current | temp;
132         if(this->device->writeRegister(BMA_RANGE,temp)!=0){
133                 perror("Failure to update RANGE value\n");
134                 return 1;
135         }
136         return 0;
137 }
```

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

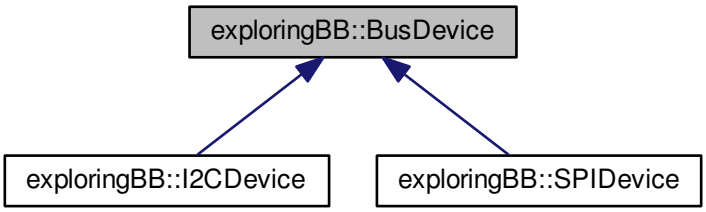
- /home/molloyd/exploringBB/library/sensor/BMA180.hxx
- /home/molloyd/exploringBB/library/sensor/BMA180.cxx

7.3 exploringBB::BusDevice Class Reference

This class is the parent of I2C and SPI devices, so that devices that use both SPI and I2C interfaces can use those interfaces interchangeably. Because it contains abstract methods, the child classes MUST implement the methods that are listed in this class.

`#include <BusDevice.h>`

Inheritance diagram for exploringBB::BusDevice:



Public Member Functions

- BusDevice (unsigned int bus, unsigned int device)
- virtual int open ()=0
- virtual unsigned char readRegister (unsigned int registerAddress)=0
- virtual unsigned char * readRegisters (unsigned int number, unsigned int fromAddress=0)=0
- virtual int write (unsigned char value)=0
- virtual int writeRegister (unsigned int registerAddress, unsigned char value)=0
- virtual void debugDumpRegisters (unsigned int number=0xff)=0
- virtual void close ()=0
- virtual ~BusDevice ()

Protected Attributes

- unsigned int bus
- unsigned int device
- int file

7.3.1 Detailed Description

This class is the parent of I2C and SPI devices, so that devices that use both SPI and I2C interfaces can use those interfaces interchangeably. Because it contains abstract methods, the child classes MUST implement the methods that are listed in this class.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 exploringBB::BusDevice::BusDevice (unsigned int *bus*, unsigned int *device*)

Constructor for a generic bus device

Parameters

bus	the bus number
device	the device number

```
34                                                      {
35          this->bus = bus;
36          this->device = device;
37          this->file=-1;
38  }
```

7.3.2.2 exploringBB::BusDevice::~BusDevice ( ) [virtual]

Destructor is unused

```
43 {}
```

7.3.3 Member Function Documentation

7.3.3.1 virtual void exploringBB::BusDevice::close ( ) [pure virtual]

Implemented in exploringBB::SPIDevice, and exploringBB::I2CDevice.

7.3.3.2 virtual void exploringBB::BusDevice::debugDumpRegisters (unsigned int *number* = 0xff) [pure virtual]

Implemented in exploringBB::SPIDevice, and exploringBB::I2CDevice.

7.3.3.3 virtual int exploringBB::BusDevice::open ( ) [pure virtual]

Implemented in exploringBB::SPIDevice, and exploringBB::I2CDevice.

7.3.3.4 virtual unsigned char exploringBB::BusDevice::readRegister (unsigned int *registerAddress*) [pure virtual]

Implemented in exploringBB::SPIDevice, and exploringBB::I2CDevice.

7.3.3.5 virtual unsigned char* exploringBB::BusDevice::readRegisters (unsigned int *number,* unsigned int *fromAddress* = 0) [pure virtual]

Implemented in exploringBB::SPIDevice, and exploringBB::I2CDevice.

7.3.3.6 virtual int exploringBB::BusDevice::write (unsigned char *value*) [pure virtual]

Implemented in exploringBB::SPIDevice, and exploringBB::I2CDevice.

7.3.3.7 virtual int exploringBB::BusDevice::writeRegister (unsigned int *registerAddress,* unsigned char *value*) [pure virtual]

Implemented in exploringBB::SPIDevice, and exploringBB::I2CDevice.

7.3.4 Field Documentation

7.3.4.1 unsigned int exploringBB::BusDevice::bus [protected]

the bus number

7.3.4.2 unsigned int exploringBB::BusDevice::device [protected]

the device number on the bus

7.3.4.3 int exploringBB::BusDevice::file [protected]

the file handle to the device

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

- /home/molloyd/exploringBB/library/bus/BusDevice.h
- /home/molloyd/exploringBB/library/bus/BusDevice.cpp

7.4 exploringBB::DCMotor Class Reference

A generic DC motor class that controls a motor driver board using a PWM signal,and a GPIO state to control the motor direction.

`#include <DCMotor.h>`

Public Types

- enum DIRECTION { CLOCKWISE, ANTICLOCKWISE }

Public Member Functions

- DCMotor (PWM *pwm, GPIO *gpio)
- DCMotor (PWM *pwm, int gpioNumber)
- DCMotor (PWM *pwm, GPIO *gpio, DCMotor::DIRECTION direction)
- DCMotor (PWM *pwm, int gpioNumber, DCMotor::DIRECTION direction)
- DCMotor (PWM *pwm, GPIO *gpio, DCMotor::DIRECTION direction, float speedPercent)
- DCMotor (PWM *pwm, int gpioNumber, DCMotor::DIRECTION direction, float speedPercent)
- virtual void go ()
- virtual void setSpeedPercent (float speedPercent)
- virtual float getSpeedPercent ()
- virtual void setDirection (DIRECTION direction)
- virtual DIRECTION getDirection ()
- virtual void reverseDirection ()
- virtual void stop ()
- virtual void setDutyCyclePeriod (unsigned int period_ns)
- virtual ~DCMotor ()

7.4.1 Detailed Description

A generic DC motor class that controls a motor driver board using a PWM signal,and a GPIO state to control the motor direction.

7.4.2 Member Enumeration Documentation

7.4.2.1 enum exploringBB::DCMotor::DIRECTION

Enumerator

CLOCKWISE

ANTICLOCKWISE

```
42 { CLOCKWISE, ANTICLOCKWISE };
```

7.4.3 Constructor & Destructor Documentation

7.4.3.1 exploringBB::DCMotor::DCMotor (PWM * *pwm*, GPIO * *gpio*)

```
29                                      {
30         init(pwm, gpio, CLOCKWISE, DEFAULT_DCMOTOR_SPEED);
31 }
```

7.4.3.2 exploringBB::DCMotor::DCMotor (PWM * *pwm*, int *gpioNumber*)

```
33                                      {
34         this->gpio = new GPIO(gpioNumber);
35         this->gpio->setDirection(GPIO::OUTPUT);
36         init(pwm, this->gpio, CLOCKWISE, DEFAULT_DCMOTOR_SPEED);
37 }
```

7.4.3.3 exploringBB::DCMotor::DCMotor (PWM * *pwm*, GPIO * *gpio*, DCMotor::DIRECTION *direction*)

```
39                                                              {
40         init(pwm, gpio, direction, DEFAULT_DCMOTOR_SPEED);
41 }
```

7.4.3.4 exploringBB::DCMotor::DCMotor (PWM * *pwm*, int *gpioNumber*, DCMotor::DIRECTION *direction*)

```
43                                                                      {
44         this->gpio = new GPIO(gpioNumber);
45         this->gpio->setDirection(GPIO::OUTPUT);
46         init(pwm, this->gpio, direction, DEFAULT_DCMOTOR_SPEED);
47 }
```

7.4.3.5 exploringBB::DCMotor::DCMotor (PWM * *pwm*, GPIO * *gpio*, DCMotor::DIRECTION *direction*, float *speedPercent*)

```
49                                                                                      {
50         init(pwm, gpio, direction, speedPercent);
51 }
```

7.4.3.6 exploringBB::DCMotor::DCMotor (PWM * *pwm*, int *gpioNumber*, DCMotor::DIRECTION *direction*, float *speedPercent*)

```
53                                                                                              {
54         this->gpio = new GPIO(gpioNumber);
55         this->gpio->setDirection(GPIO::OUTPUT);
56         init(pwm, this->gpio, direction, speedPercent);
57 }
```

7.4.3.7 exploringBB::DCMotor::~DCMotor ( ) [virtual]

```
108                     {
109         delete gpio;
110 }
```

7.4.4 Member Function Documentation

7.4.4.1 virtual DIRECTION exploringBB::DCMotor::getDirection ( ) [inline], [virtual]

```
60 { return this->direction; }
```

7.4.4.2 virtual float exploringBB::DCMotor::getSpeedPercent ( ) [inline], [virtual]

```
58 { return this->speedPercent; }
```

7.4.4.3 void exploringBB::DCMotor::go ( ) [virtual]

```
100                 {
101         this->pwm->run();
102 }
```

7.4.4.4 void exploringBB::DCMotor::reverseDirection ( ) [virtual]

```
87                              {
88         if(this->direction == CLOCKWISE){
89                 this->setDirection(ANTICLOCKWISE);
90         }
91         else{
92                 this->setDirection(CLOCKWISE);
93         }
94 }
```

7.4.4.5 void exploringBB::DCMotor::setDirection (DIRECTION *direction*) [virtual]

```
77                                              {
78         if(direction == CLOCKWISE){
79                 this->gpio->setValue(GPIO::HIGH);
80         }
81         else{
82                 this->gpio->setValue(GPIO::LOW);
83         }
84         this->direction = direction;
85 }
```

7.4.4.6 void exploringBB::DCMotor::setDutyCyclePeriod (unsigned int *period_ns*) [virtual]

```
104                                                     {
105         this->pwm->setPeriod(period_ns);
106 }
```

7.4.4.7 void exploringBB::DCMotor::setSpeedPercent (float *speedPercent*) [virtual]

```
72                                              {
73         this->pwm->setDutyCycle(speedPercentage);
74         this->speedPercent = speedPercentage;
75 }
```

7.4.4.8 void exploringBB::DCMotor::stop ( ) [virtual]

```
96                      {
97          this->pwm->stop();
98  }
```

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

- /home/molloyd/exploringBB/library/motor/DCMotor.h
- /home/molloyd/exploringBB/library/motor/DCMotor.cpp

7.5 exploringBB::GPIO Class Reference

GPIO class for input and output functionality on a single GPIO pin.

`#include <GPIO.h>`

Public Types

- enum DIRECTION { INPUT, OUTPUT }
- enum VALUE { LOW =0, HIGH =1 }
- enum EDGE { NONE, RISING, FALLING, BOTH }

Public Member Functions

- GPIO (int number)
- virtual int getNumber ()
- virtual int setDirection (GPIO::DIRECTION)
- virtual GPIO::DIRECTION getDirection ()
- virtual int setValue (GPIO::VALUE)
- virtual int toggleOutput ()
- virtual GPIO::VALUE getValue ()
- virtual int setActiveLow (bool isLow=true)
- virtual int setActiveHigh ()
- virtual void setDebounceTime (int time)
- virtual int streamOpen ()
- virtual int streamWrite (GPIO::VALUE)
- virtual int streamClose ()
- virtual int toggleOutput (int time)
- virtual int toggleOutput (int numberOfTimes, int time)
- virtual void changeToggleTime (int time)
- virtual void toggleCancel ()
- virtual int setEdgeType (GPIO::EDGE)
- virtual GPIO::EDGE getEdgeType ()
- virtual int waitForEdge ()
- virtual int waitForEdge (CallbackType callback)
- virtual void waitForEdgeCancel ()
- virtual ~GPIO ()

Friends

- void * threadedPoll (void *value)
- void * threadedToggle (void *value)

7.5.1 Detailed Description

GPIO class for input and output functionality on a single GPIO pin.

7.5.2 Member Enumeration Documentation

7.5.2.1 enum exploringBB::GPIO::DIRECTION

Enumerator

INPUT

OUTPUT

```
48 { INPUT, OUTPUT };
```

7.5.2.2 enum exploringBB::GPIO::EDGE

Enumerator

NONE

RISING

FALLING

BOTH

```
50 { NONE, RISING, FALLING, BOTH };
```

7.5.2.3 enum exploringBB::GPIO::VALUE

Enumerator

LOW

HIGH

```
49 { LOW=0, HIGH=1 };
```

7.5.3 Constructor & Destructor Documentation

7.5.3.1 exploringBB::GPIO::GPIO (int *number*)

The constructor will set up the states and export the pin.

Parameters

number	The GPIO number to be exported

```
47                      {
48          this->number = number;
49          this->debounceTime = 0;
50          this->togglePeriod=100;
51          this->toggleNumber=-1; //infinite number
52          this->callbackFunction = NULL;
53          this->threadRunning = false;
54
55          ostringstream s;
56          s << "gpio" << number;
57          this->name = string(s.str());
58          this->path = GPIO_PATH + this->name + "/";
59          this->exportGPIO();
60          // need to give Linux time to set up the sysfs structure
61          usleep(250000); // 250ms delay
62  }
```

7.5.3.2 exploringBB::GPIO::~GPIO ( ) [virtual]

```
293             {
294         this->unexportGPIO();
295 }
```

7.5.4 Member Function Documentation

7.5.4.1 virtual void exploringBB::GPIO::changeToggleTime (int *time*) [inline], [virtual]

```
80 { this->togglePeriod = time; }
```

7.5.4.2 GPIO::DIRECTION exploringBB::GPIO::getDirection ( ) [virtual]

```
169                                 {
170         string input = read(this->path, "direction");
171         if (input == "in") return INPUT;
172         else return OUTPUT;
173 }
```

7.5.4.3 GPIO::EDGE exploringBB::GPIO::getEdgeType ( ) [virtual]

```
175                             {
176         string input = read(this->path, "edge");
177         if (input == "rising") return RISING;
178         else if (input == "falling") return FALLING;
179         else if (input == "both") return BOTH;
180         else return NONE;
181 }
```

7.5.4.4 virtual int exploringBB::GPIO::getNumber ( ) [inline], [virtual]

Returns the GPIO number as an int.

7.5.4.5 GPIO::VALUE exploringBB::GPIO::getValue ( ) [virtual]

```
163                         {
164         string input = read(this->path, "value");
165         if (input == "0") return LOW;
166         else return HIGH;
167 }
```

7.5.4.6 int exploringBB::GPIO::setActiveHigh ( ) [virtual]

```
159                         {
160     return this->setActiveLow(false);
161 }
```

7.5.4.7 int exploringBB::GPIO::setActiveLow (bool *isLow* = true) [virtual]

```
154                                 {
155     if(isLow) return write(this->path, "active_low", "1");
156     else return write(this->path, "active_low", "0");
157 }
```

7.5.4.8 virtual void exploringBB::GPIO::setDebounceTime (int *time*) [inline], [virtual]

```
71 { this->debounceTime = time; }
```

7.5.4.9 int exploringBB::GPIO::setDirection (GPIO::DIRECTION *dir*) [virtual]

```
120                                         {
121     switch(dir){
122     case INPUT: return write(this->path, "direction", "in");
123        break;
124     case OUTPUT:return write(this->path, "direction", "out");
125        break;
126     }
127     return -1;
128 }
```

7.5.4.10 int exploringBB::GPIO::setEdgeType (GPIO::EDGE *value*) [virtual]

```
140                                       {
141     switch(value){
142     case NONE: return write(this->path, "edge", "none");
143        break;
144     case RISING: return write(this->path, "edge", "rising");
145        break;
146     case FALLING: return write(this->path, "edge", "falling");
147        break;
148     case BOTH: return write(this->path, "edge", "both");
149        break;
150     }
151     return -1;
152 }
```

7.5.4.11 int exploringBB::GPIO::setValue (GPIO::VALUE *value*) [virtual]

```
130                                     {
131     switch(value){
132     case HIGH: return write(this->path, "value", "1");
133        break;
134     case LOW: return write(this->path, "value", "0");
135        break;
136     }
137     return -1;
138 }
```

7.5.4.12 int exploringBB::GPIO::streamClose ( ) [virtual]

```
191                         {
192         stream.close();
193         return 0;
194 }
```

7.5.4.13 int exploringBB::GPIO::streamOpen ( ) [virtual]

```
183                        {
184         stream.open((path + "value").c_str());
185         return 0;
186 }
```

7.5.4.14 int exploringBB::GPIO::streamWrite (GPIO::VALUE *value*) [virtual]

```
187                                       {
188         stream << value << std::flush;
189         return 0;
190 }
```

7.5.4.15 virtual void exploringBB::GPIO::toggleCancel ( ) [inline], [virtual]

```
81 { this->threadRunning = false; }
```

7.5.4.16 int exploringBB::GPIO::toggleOutput ( ) [virtual]

```
196                         {
197         this->setDirection(OUTPUT);
198         if ((bool) this->getValue()) this->setValue(LOW);
199         else this->setValue(HIGH);
200     return 0;
201 }
```

7.5.4.17 int exploringBB::GPIO::toggleOutput (int *time*) [virtual]

```
203 { return this->toggleOutput(-1, time); }
```

7.5.4.18 int exploringBB::GPIO::toggleOutput (int *numberOfTimes*, int *time*) [virtual]

```
204                                                 {
205         this->setDirection(OUTPUT);
206         this->toggleNumber = numberOfTimes;
207         this->togglePeriod = time;
208         this->threadRunning = true;
209     if(pthread_create(&this->thread, NULL, &threadedToggle, static_cast<void*>(this))){
210         perror("GPIO: Failed to create the toggle thread");
211         this->threadRunning = false;
212         return -1;
213     }
214     return 0;
215 }
```

7.5.4.19 int exploringBB::GPIO::waitForEdge ( ) [virtual]

```
233                         {
234         this->setDirection(INPUT); // must be an input pin to poll its value
235         int fd, i, epollfd, count=0;
236         struct epoll_event ev;
237         epollfd = epoll_create(1);
238     if (epollfd == -1) {
239             perror("GPIO: Failed to create epollfd");
240             return -1;
241     }
242     if ((fd = open((this->path + "value").c_str(), O_RDONLY | O_NONBLOCK)) == -1) {
243        perror("GPIO: Failed to open file");
244        return -1;
245     }
246
247     //ev.events = read operation | edge triggered | urgent data
248     ev.events = EPOLLIN | EPOLLET | EPOLLPRI;
249     ev.data.fd = fd;  // attach the file file descriptor
250
251     //Register the file descriptor on the epoll instance, see: man epoll_ctl
252     if (epoll_ctl(epollfd, EPOLL_CTL_ADD, fd, &ev) == -1) {
253        perror("GPIO: Failed to add control interface");
254        return -1;
255     }
256         while(count<=1){  // ignore the first trigger
257                 i = epoll_wait(epollfd, &ev, 1, -1);
258                 if (i==-1){
259                         perror("GPIO: Poll Wait fail");
260                         count=5; // terminate loop
261                 }
262                 else {
263                         count++; // count the triggers up
264                 }
265         }
266     close(fd);
267     if (count==5) return -1;
268         return 0;
269 }
```

7.5.4.20 int exploringBB::GPIO::waitForEdge (CallbackType *callback*) [virtual]

```
281                                         {
282         this->threadRunning = true;
283         this->callbackFunction = callback;
284     // create the thread, pass the reference, address of the function and data
285     if(pthread_create(&this->thread, NULL, &threadedPoll, static_cast<void*>(this))){
286         perror("GPIO: Failed to create the poll thread");
287         this->threadRunning = false;
288         return -1;
289     }
290     return 0;
291 }
```

7.5.4.21 virtual void exploringBB::GPIO::waitForEdgeCancel ( ) [inline], [virtual]

```
88 { this->threadRunning = false; }
```

7.5.5 Friends And Related Function Documentation

7.5.5.1 void* threadedPoll (void * *value*) [friend]

```
272                                 {
273         GPIO *gpio = static_cast<GPIO*>(value);
274         while(gpio->threadRunning){
275                 gpio->callbackFunction(gpio->waitForEdge());
276                 usleep(gpio->debounceTime * 1000);
277         }
278         return 0;
279 }
```

7.5.5.2 void* threadedToggle (void * *value*) [friend]

```
218                                 {
219         GPIO *gpio = static_cast<GPIO*>(value);
220         bool isHigh = (bool) gpio->getValue(); //find current value
221         while(gpio->threadRunning){
222                 if (isHigh)     gpio->setValue(GPIO::HIGH);
223                 else gpio->setValue(GPIO::LOW);
224                 usleep(gpio->togglePeriod * 500);
225                 isHigh=!isHigh;
226                 if(gpio->toggleNumber>0) gpio->toggleNumber--;
227                 if(gpio->toggleNumber==0) gpio->threadRunning=false;
228         }
229         return 0;
230 }
```

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

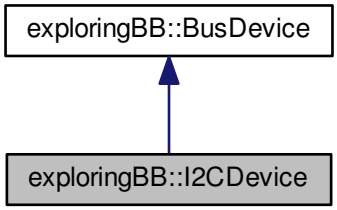
- /home/molloyd/exploringBB/library/gpio/GPIO.h
- /home/molloyd/exploringBB/library/gpio/GPIO.cpp

7.6 exploringBB::I2CDevice Class Reference

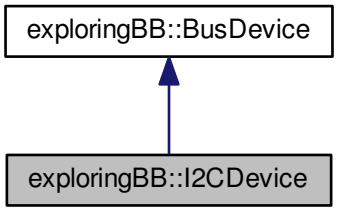
Generic I2C Device class that can be used to connect to any type of I2C device and read or write to its registers.

`#include <I2CDevice.h>`

Inheritance diagram for exploringBB::I2CDevice:



Collaboration diagram for exploringBB::I2CDevice:



Public Member Functions

- I2CDevice (unsigned int bus, unsigned int device)
- virtual int open ()
- virtual int write (unsigned char value)
- virtual unsigned char readRegister (unsigned int registerAddress)
- virtual unsigned char * readRegisters (unsigned int number, unsigned int fromAddress=0)
- virtual int writeRegister (unsigned int registerAddress, unsigned char value)
- virtual void debugDumpRegisters (unsigned int number=0xff)
- virtual void close ()
- virtual ~I2CDevice ()

Additional Inherited Members

7.6.1 Detailed Description

Generic I2C Device class that can be used to connect to any type of I2C device and read or write to its registers.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 exploringBB::I2CDevice::I2CDevice (unsigned int *bus*, unsigned int *device*)

Constructor for the I2CDevice class. It requires the bus number and device number. The constructor opens a file handle to the I2C device, which is destroyed when the destructor is called

Parameters

bus	The bus number. Usually 0 or 1 on the BBB
device	The device ID on the bus.

```
47                                                      :
48          BusDevice(bus,device){
49          this->open();
50  }
```

7.6.2.2 exploringBB::I2CDevice::~I2CDevice ( ) [virtual]

Closes the file on destruction, provided that it has not already been closed.

```
168                         {
169         if(file!=-1) this->close();
170  }
```

7.6.3 Member Function Documentation

7.6.3.1 void exploringBB::I2CDevice::close ( ) [virtual]

Close the file handles and sets a temporary state to -1.

Implements exploringBB::BusDevice.

```
160                         {
161         ::close(this->file);
162         this->file = -1;
163  }
```

7.6.3.2 void exploringBB::I2CDevice::debugDumpRegisters (unsigned int *number* = 0xff) [virtual]

Method to dump the registers to the standard output. It inserts a return character after every 16 values and displays the results in hexadecimal to give a standard output using the HEX() macro that is defined at the top of this file. The standard output will stay in hexadecimal format, hence the call on the last like.

Parameters

number	the total number of registers to dump, defaults to 0xff

Implements exploringBB::BusDevice.

```
147                                                     {
148         cout << "Dumping Registers for Debug Purposes:" << endl;
149         unsigned char *registers = this->readRegisters(number);
150         for(int i=0; i<(int)number; i++){
151                 cout << HEX(*(registers+i)) << " ";
152                 if (i%16==15) cout << endl;
153         }
154         cout << dec;
155  }
```

7.6.3.3 int exploringBB::I2CDevice::open ( ) [virtual]

Open a connection to an I2C device

Returns

1 on failure to open to the bus or device, 0 on success.

Implements exploringBB::BusDevice.

```
56                      {
57     string name;
58     if(this->bus==0) name = BBB_I2C_0;
59     else name = BBB_I2C_1;
60
61     if((this->file=::open(name.c_str(), O_RDWR)) < 0){
62        perror("I2C: failed to open the bus\n");
63            return 1;
64     }
65     if(ioctl(this->file, I2C_SLAVE, this->device) < 0){
66        perror("I2C: Failed to connect to the device\n");
67            return 1;
68     }
69     return 0;
70 }
```

7.6.3.4 unsigned char exploringBB::I2CDevice::readRegister (unsigned int *registerAddress*) [virtual]

Read a single register value from the address on the device.

Parameters

registerAddress	the address to read from

Returns

the byte value at the register address.

Implements exploringBB::BusDevice.

```
111                                                                 {
112     this->write(registerAddress);
113     unsigned char buffer[1];
114     if(::read(this->file, buffer, 1)!=1){
115        perror("I2C: Failed to read in the value.\n");
116        return 1;
117     }
118     return buffer[0];
119 }
```

7.6.3.5 unsigned char * exploringBB::I2CDevice::readRegisters (unsigned int *number*, unsigned int *fromAddress* = 0) [virtual]

Method to read a number of registers from a single device. This is much more efficient than reading the registers individually. The from address is the starting address to read from, which defaults to 0x00.

Parameters

number	the number of registers to read from the device
fromAddress	the starting address to read from

Returns

a pointer of type unsigned char* that points to the first element in the block of registers

Implements exploringBB::BusDevice.

```
129                                                                                 {
130         this->write(fromAddress);
131         unsigned char* data = new unsigned char[number];
132     if(::read(this->file, data, number)!=(int)number){
133        perror("IC2: Failed to read in the full buffer.\n");
134            return NULL;
135     }
136         return data;
137 }
```

7.6.3.6 int exploringBB::I2CDevice::write (unsigned char *value*) [virtual]

Write a single value to the I2C device. Used to set up the device to read from a particular address.

Parameters

value	the value to write to the device

Returns

1 on failure to write, 0 on success.

Implements exploringBB::BusDevice.

```
96                                        {
97    unsigned char buffer[1];
98    buffer[0]=value;
99    if (::write(this->file, buffer, 1)!=1){
100       perror("I2C: Failed to write to the device\n");
101       return 1;
102    }
103    return 0;
104 }
```

7.6.3.7 int exploringBB::I2CDevice::writeRegister (unsigned int *registerAddress*, unsigned char *value*) [virtual]

Write a single byte value to a single register.

Parameters

registerAddress	The register address
value	The value to be written to the register

Returns

1 on failure to write, 0 on success.

Implements exploringBB::BusDevice.

```
79                                                                                  {
80    unsigned char buffer[2];
81    buffer[0] = registerAddress;
82    buffer[1] = value;
83    if(::write(this->file, buffer, 2)!=2){
84       perror("I2C: Failed write to the device\n");
85       return 1;
86    }
87    return 0;
88 }
```

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

- /home/molloyd/exploringBB/library/bus/I2CDevice.h
- /home/molloyd/exploringBB/library/bus/I2CDevice.cpp

7.7 exploringBB::ITG3200 Class Reference

A class to interface with the ITG3200 gyroscope (untested)

`#include <ITG3200.h>`

Public Member Functions

- ITG3200 (BusDevice *device)
- float getGyroscopeRoll ()
- float getGyroscopePitch ()
- float getGyroscopeYaw ()
- float getOffsetRollOffset ()
- float getOffsetPitchOffset ()
- float getOffsetYawOffset ()
- int getTemperature ()
- virtual ~ITG3200 ()
- int readFullSensorState ()
- int convertGyroscopeValue (int msb_reg_addr, int lsb_reg_addr)
- int setDigitalLowPassFilter (ITG3200_LPFILTER filterValue)
- int setSampleRateDivider (char divider)
- int zeroCalibrate (int numberSamples, int sampleDelayMs)

7.7.1 Detailed Description

A class to interface with the ITG3200 gyroscope (untested)

7.7.2 Constructor & Destructor Documentation

7.7.2.1 exploringBB::ITG3200::ITG3200 (BusDevice * *device*)

```
59                                  {
60         this->device = device;
61         for (int i=0; i<3; i++) {
62                 this->gyroscope[i] = 0.0;
63                 this->offsets[i] = 0.0;
64         }
65         this->registers = NULL;
66         this->temperature = 0;
67         this->readFullSensorState();
68 }
```

7.7.2.2 exploringBB::ITG3200::~ITG3200 ( ) [virtual]

```
129 {}
```

7.7.3 Member Function Documentation

7.7.3.1 int exploringBB::ITG3200::convertGyroscopeValue (int *msb_reg_addr,* int *lsb_reg_addr*)

```
83                                                                  {
84         short temp = *(registers+msb_reg_addr);
85         temp = (temp<<8) | *(registers+lsb_reg_addr);
86         temp = ~temp + 1;
87         return temp;
88 }
```

7.7.3.2 float exploringBB::ITG3200::getGyroscopePitch ( ) [inline]

```
65 { return gyroscope[1]/SENSITIVITY; }
```

7.7.3.3 float exploringBB::ITG3200::getGyroscopeRoll ( ) [inline]

```
64 { return gyroscope[0]/SENSITIVITY; }
```

7.7.3.4 float exploringBB::ITG3200::getGyroscopeYaw ( ) [inline]

```
66 { return gyroscope[2]/SENSITIVITY; }
```

7.7.3.5 float exploringBB::ITG3200::getOffsetPitchOffset ( ) [inline]

```
69 { return offsets[1]/SENSITIVITY; }
```

7.7.3.6 float exploringBB::ITG3200::getOffsetRollOffset ( ) [inline]

```
68 { return offsets[0]/SENSITIVITY; }
```

7.7.3.7 float exploringBB::ITG3200::getOffsetYawOffset ( ) [inline]

```
70 { return offsets[2]/SENSITIVITY; }
```

7.7.3.8 int exploringBB::ITG3200::getTemperature ( ) [inline]

```
72 { return temperature; }
```

7.7.3.9 int exploringBB::ITG3200::readFullSensorState ( )

```
70                                {
71         this->registers = this->device->readRegisters(BUFFER_SIZE, 0x00);
72         if(*this->registers!=0x69){
73                 perror("ITG3200: Failure Condition - Sensor ID not Verified");
74                 return -1;
75         }
76     this->gyroscope[0] = convertGyroscopeValue(GYRO_X_MSB,
       GYRO_X_LSB);
77     this->gyroscope[1] = convertGyroscopeValue(GYRO_Y_MSB,
       GYRO_Y_LSB);
78     this->gyroscope[2] = convertGyroscopeValue(GYRO_Z_MSB,
       GYRO_Z_LSB);
79     this->temperature = convertGyroscopeValue(TEMP_MSB,
       TEMP_LSB);
80     return 0;
81 }
```

7.7.3.10 int exploringBB::ITG3200::setDigitalLowPassFilter (ITG3200_LPFILTER *filterValue*)

```
90                                                                {
91         //bits 7,6,5 are 0;  bits 4,3 are = 03h for proper operation, i.e. +/- 2000 deg/sec full range
92         char temp = 0b00011000;
93         temp = temp | filterValue; // OR with incoming value (between 00 and 06h)
94         if(this->device->writeRegister(DLPF_FS, temp)!=0){
95                 cout << "Failure to update ITG3200 DLPF_FS value" << endl;
96                 return 1;
97         }
98         return 0;
99 }
```

7.7.3.11 int exploringBB::ITG3200::setSampleRateDivider (char *divider*)

```
117                                                 {
118         //bits 7-0 set the divider which sets the sample rate of the gyro according to the equation
119         // Fsample = Finternal / (divider+1)
120         // Finternal is set according to the low pass filter ITG3200_LPFILTER
121         //        which is 1kHz for all values except the highest LPF of DLPF_CFG
122         if(this->device->writeRegister(SMPLRT_DIV, divider)!=0){
123                 cout << "Failure to update ITG3200 SMPLRT_DIV value" << endl;
124                 return 1;
125         }
126         return 0;
127 }
```

7.7.3.12 int exploringBB::ITG3200::zeroCalibrate (int *numberSamples*, int *sampleDelayMs*)

```
101                                                                 {
102
103         double total[3] = {0.0, 0.0, 0.0};
104         for(int i=0; i<numberSamples; i++){
105                 this->readFullSensorState();
106                 for(int j=0; j<3; j++){
107                         total[j] = total[j] + this->gyroscope[j];
108                 }
109                 usleep(sampleDelayMs*1000);
110         }
111         for(int i=0; i<=2; i++) {
112                 this->offsets[i] = total[i]/numberSamples;
113         }
114         return 0;
115 }
```

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

- /home/molloyd/exploringBB/library/sensor/ITG3200.h
- /home/molloyd/exploringBB/library/sensor/ITG3200.cpp

7.8 exploringBB::LCDCharacterDisplay Class Reference

A class that provides an interface to an LCD character module. It provices support for multiple rows and columns and provides methods for formatting and printing text. You should use a 4 wire interface and a 74XX595 to communicate with the display module.

`#include <LCDCharacterDisplay.h>`

Public Member Functions

- LCDCharacterDisplay (SPIDevice *device, int width, int height)
- virtual void write (char c)
- virtual void print (std::string message)
- virtual void clear ()
- virtual void home ()
- virtual int setCursorPosition (int row, int column)
- virtual void setDisplayOff (bool displayOff)
- virtual void setCursorOff (bool cursorOff)
- virtual void setCursorBlink (bool isBlink)
- virtual void setCursorMoveOff (bool cursorMoveOff)
- virtual void setCursorMoveLeft (bool cursorMoveLeft)
- virtual void setAutoscroll (bool isAutoscroll)
- virtual void setScrollDisplayLeft (bool scrollLeft)
- virtual ~LCDCharacterDisplay ()

7.8.1 Detailed Description

A class that provides an interface to an LCD character module. It provices support for multiple rows and columns and provides methods for formatting and printing text. You should use a 4 wire interface and a 74XX595 to communicate with the display module.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 exploringBB::LCDCharacterDisplay::LCDCharacterDisplay (SPIDevice * *device,* int *width,* int *height*)

```
65                                                                                  {
66          this->device  = device;
67          this->width   = width;
68          this->height  = height;
69
70          //Default Cursor, Display and Entry states
71          this->cursorState  = LCD_CURSOR_DISPLAY;
72          this->displayState = LCD_DISPLAY_ON_OFF |
      DISPLAY_ENTIRE | DISPLAY_CURSOR | DISPLAY_CURSOR_POS;
73          this->entryState   = LCD_ENTRY_MODE_SET |
      ENTRY_MODE_LEFT;
74
75          this->setup4bit();
76  }
```

7.8.2.2 exploringBB::LCDCharacterDisplay::~LCDCharacterDisplay ( ) [virtual]

```
334                                         {
335          this->device->close();
336  }
```

7.8.3 Member Function Documentation

7.8.3.1 void exploringBB::LCDCharacterDisplay::clear ( ) [virtual]

```
191                                 {
192          this->command(LCD_CLEAR_DISPLAY);
193          usleep(LCD_LONG_DELAY);  //data sheets states that a delay of 1.52ms is required
194  }
```

7.8.3.2 void exploringBB::LCDCharacterDisplay::home ( ) [virtual]

```
199                                 {
200          this->command(LCD_RETURN_HOME);
201          usleep(LCD_LONG_DELAY);  //data sheets states that a delay of 1.52ms is required
202  }
```

7.8.3.3 void exploringBB::LCDCharacterDisplay::print (std::string *message*) [virtual]

```
155                                                 {
156          for(unsigned int i=0; i<message.length(); i++){
157                  this->write(message[i]);
158          }
159  }
```

7.8.3.4 void exploringBB::LCDCharacterDisplay::setAutoscroll (bool *isAutoscroll*) [virtual]

Parameters

setAutoscroll	

```
304                                                     {
305         if (!isAutoscroll){
306                 this->entryState = entryState & (~ENTRY_MODE_S);
307                 this->writeEntryState();
308         }
309         else{
310                 this->entryState = entryState | ENTRY_MODE_S;
311                 this->writeEntryState();
312         }
313 }
```

7.8.3.5 void exploringBB::LCDCharacterDisplay::setCursorBlink (bool *isBlink*) [virtual]

Turn the blink on or off.

Parameters

isBlink	pass true to turn the blink on, false to turn it off

```
259                                                     {
260         if (!isBlink){
261                 this->displayState = displayState & (~DISPLAY_CURSOR_POS); //bit
      inversion of DISPLAY_ENTIRE
262                 this->writeDisplayState();
263         }
264         else{
265                 this->displayState = displayState | DISPLAY_CURSOR_POS;
266                 this->writeDisplayState();
267         }
268 }
```

7.8.3.6 void exploringBB::LCDCharacterDisplay::setCursorMoveLeft (bool *cursorMoveLeft*) [virtual]

Parameters

cursorMoveLeft	

```
289                                                     {
290         if (!cursorMoveLeft){
291                 this->cursorState = cursorState & (~CURSOR_DISPLAY_RL);
292                 this->writeCursorState();
293         }
294         else{
295                 this->cursorState = cursorState | CURSOR_DISPLAY_RL;
296                 this->writeCursorState();
297         }
298 }
```

7.8.3.7 void exploringBB::LCDCharacterDisplay::setCursorMoveOff (bool *cursorMoveOff*) [virtual]

Turn the cursor moving On or Off

Parameters

cursorOff	

```
274                                                     {
275         if (!cursorMoveOff){
276                 this->cursorState = cursorState & (~CURSOR_DISPLAY_SC);
277                 this->writeCursorState();
278         }
279         else{
280                 this->cursorState = cursorState | CURSOR_DISPLAY_SC;
281                 this->writeCursorState();
282         }
283 }
```

7.8.3.8 void exploringBB::LCDCharacterDisplay::setCursorOff (bool *cursorOff*) [virtual]

Turn the cursor on or off.

Parameters

cursorOff	pass true to turn the cursor off, false to turn it back on

```
244                                                 {
245         if (cursorOff){
246                 this->displayState = displayState & (~DISPLAY_CURSOR);
247                 this->writeDisplayState();
248         }
249         else{
250                 this->displayState = displayState | DISPLAY_CURSOR;
251                 this->writeDisplayState();
252         }
253 }
```

7.8.3.9 int exploringBB::LCDCharacterDisplay::setCursorPosition (int *row*, int *column*) [virtual]

```
215                                                         {
216
217         if ((column>=this->width)||(row>=this->height)) return -1;
218     row = row * LCD_ROW_OFFSET_ADDR;
219     int value = (row + column) | LCD_DDRAM_ADDR;
220         this->command(value);
221         //printf("[%02x]", value);
222     return 0;
223 }
```

7.8.3.10 void exploringBB::LCDCharacterDisplay::setDisplayOff (bool *displayOff*) [virtual]

Turn the display on or off.

Parameters

displayOff	pass true to turn the display off, false to turn it back on

```
229                                                 {
230         if (displayOff){
231                 this->displayState = displayState & (~DISPLAY_ENTIRE); //bit inversion of
      DISPLAY_ENTIRE
232                 this->writeDisplayState();
233         }
234         else{
235                 this->displayState = displayState | DISPLAY_ENTIRE;
236                 this->writeDisplayState();
237         }
238 }
```

7.8.3.11 void exploringBB::LCDCharacterDisplay::setScrollDisplayLeft (bool *scrollLeft*) [virtual]

Parameters

scrollDisplayLeft	

```
319                                                         {
320         if (scrollLeft){
321                 this->entryState = entryState & (~ENTRY_MODE_LEFT);
322                 this->writeEntryState();
323         }
324         else{
325                 this->entryState = entryState | ENTRY_MODE_LEFT;
326                 this->writeEntryState();
327         }
328 }
```

7.8.3.12 void exploringBB::LCDCharacterDisplay::write (char *c*) [virtual]

```
174                                     {
175         // 4-bit mode. Send lower 4 bits followed by higher 4 bits
176         char upper = (c << 4) & 0b11110000;
177         char lower = c & 0b11110000;
178         // need to write the lower data and toggle the E bit
179         this->device->write(lower | 0b00000011);  //lower 4 bits
180         usleep(1);                                //sleep for at least 300ns
181         this->device->write(lower | 0b00000001);  //lower 4 bits
182         // need to write the upper data and toggle the E bit
183         this->device->write(upper | 0b00000011);  //lower 4 bits
184         usleep(1);                                //sleep for at least 300ns
185         this->device->write(upper | 0b00000001);  //lower 4 bits
186 }
```

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

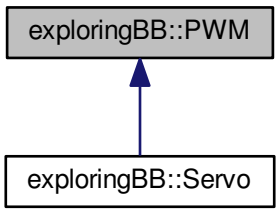
- /home/molloyd/exploringBB/library/display/LCDCharacterDisplay.h
- /home/molloyd/exploringBB/library/display/LCDCharacterDisplay.cpp

7.9 exploringBB::PWM Class Reference

A class to control a basic PWM output – you must know the exact sysfs filename for the PWM output.

`#include <PWM.h>`

Inheritance diagram for exploringBB::PWM:



Public Types

- enum POLARITY { ACTIVE_LOW =0, ACTIVE_HIGH =1 }

Public Member Functions

- PWM (string pinName)
- virtual int setPeriod (unsigned int period_ns)
- virtual unsigned int getPeriod ()
- virtual int setFrequency (float frequency_hz)
- virtual float getFrequency ()
- virtual int setDutyCycle (unsigned int duration_ns)
- virtual int setDutyCycle (float percentage)
- virtual unsigned int getDutyCycle ()
- virtual float getDutyCyclePercent ()

- virtual int setPolarity (PWM::POLARITY)
- virtual void invertPolarity ()
- virtual PWM::POLARITY getPolarity ()
- virtual void setAnalogFrequency (float frequency_hz)
- virtual int calibrateAnalogMax (float analogMax)
- virtual int analogWrite (float voltage)
- virtual int run ()
- virtual bool isRunning ()
- virtual int stop ()
- virtual ~PWM ()

7.9.1 Detailed Description

A class to control a basic PWM output – you must know the exact sysfs filename for the PWM output.

7.9.2 Member Enumeration Documentation

7.9.2.1 enum exploringBB::PWM::POLARITY

Enumerator

ACTIVE_LOW

ACTIVE_HIGH

```
45 { ACTIVE_LOW=0, ACTIVE_HIGH=1 };
```

7.9.3 Constructor & Destructor Documentation

7.9.3.1 exploringBB::PWM::PWM (string *pinName*)

```
31                          {
32         this->name = pinName;
33         this->path = PWM_PATH + this->name + "/";
34         this->analogFrequency = 100000;
35         this->analogMax = 3.3;
36 }
```

7.9.3.2 exploringBB::PWM::~PWM ( ) [virtual]

```
127 {}
```

7.9.4 Member Function Documentation

7.9.4.1 int exploringBB::PWM::analogWrite (float *voltage*) [virtual]

```
106                                     {
107         if ((voltage<0.0f)||(voltage>3.3f)) return -1;
108         this->setFrequency(this->analogFrequency);
109         this->setPolarity(PWM::ACTIVE_LOW);
110         this->setDutyCycle((100.0f*voltage)/this->analogMax);
111         return this->run();
112 }
```

7.9.4.2 int exploringBB::PWM::calibrateAnalogMax (float *analogMax*) [virtual]

```
100                                         { //must be between 3.2 and 3.4
101         if((analogMax<3.2f) || (analogMax>3.4f)) return -1;
102         else this->analogMax = analogMax;
103         return 0;
104 }
```

7.9.4.3 unsigned int exploringBB::PWM::getDutyCycle ( ) [virtual]

```
76                                  {
77         return atoi(read(this->path, PWM_DUTY).c_str());
78 }
```

7.9.4.4 float exploringBB::PWM::getDutyCyclePercent ( ) [virtual]

```
80                                  {
81         unsigned int period_ns = this->getPeriod();
82         unsigned int duty_ns = this->getDutyCycle();
83         return 100.0f * (float)duty_ns/(float)period_ns;
84 }
```

7.9.4.5 float exploringBB::PWM::getFrequency ( ) [virtual]

```
60                          {
61         return this->period_nsToFrequency(this->getPeriod());
62 }
```

7.9.4.6 unsigned int exploringBB::PWM::getPeriod ( ) [virtual]

```
42                              {
43         return atoi(read(this->path, PWM_PERIOD).c_str());
44 }
```

7.9.4.7 PWM::POLARITY exploringBB::PWM::getPolarity ( ) [virtual]

```
95                                  {
96         if (atoi(read(this->path, PWM_POLARITY).c_str())==0) return
      PWM::ACTIVE_LOW;
97         else return PWM::ACTIVE_HIGH;
98 }
```

7.9.4.8 void exploringBB::PWM::invertPolarity ( ) [virtual]

```
90                          {
91         if (this->getPolarity()==PWM::ACTIVE_LOW) this->
      setPolarity(PWM::ACTIVE_HIGH);
92         else this->setPolarity(PWM::ACTIVE_LOW);
93 }
```

7.9.4.9 bool exploringBB::PWM::isRunning ( ) [virtual]

```
118                     {
119         string running = read(this->path, PWM_RUN);
120         return (running=="1");
121 }
```

7.9.4.10 int exploringBB::PWM::run ( ) [virtual]

```
114             {
115          return write(this->path, PWM_RUN, 1);
116 }
```

7.9.4.11 virtual void exploringBB::PWM::setAnalogFrequency (float *frequency_hz*) [inline], [virtual]

```
68 { this->analogFrequency = frequency_hz; }
```

7.9.4.12 int exploringBB::PWM::setDutyCycle (unsigned int *duration_ns*) [virtual]

```
64                                          {
65          return write(this->path, PWM_DUTY, duty_ns);
66 }
```

7.9.4.13 int exploringBB::PWM::setDutyCycle (float *percentage*) [virtual]

```
68                                      {
69          if ((percentage>100.0f)||(percentage<0.0f)) return -1;
70          unsigned int period_ns = this->getPeriod();
71          float duty_ns = period_ns * (percentage/100.0f);
72          this->setDutyCycle((unsigned int) duty_ns );
73          return 0;
74 }
```

7.9.4.14 int exploringBB::PWM::setFrequency (float *frequency_hz*) [virtual]

```
56                                      {
57          return this->setPeriod(this->frequencyToPeriod_ns(frequency_hz));
58 }
```

7.9.4.15 int exploringBB::PWM::setPeriod (unsigned int *period_ns*) [virtual]

```
38                                      {
39          return write(this->path, PWM_PERIOD, period_ns);
40 }
```

7.9.4.16 int exploringBB::PWM::setPolarity (PWM::POLARITY *polarity*) [virtual]

```
86                                      {
87          return write(this->path, PWM_POLARITY, polarity);
88 }
```

7.9.4.17 int exploringBB::PWM::stop ( ) [virtual]

```
123             {
124          return write(this->path, PWM_RUN, 0);
125 }
```

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

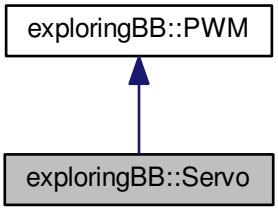
- /home/molloyd/exploringBB/library/gpio/PWM.h
- /home/molloyd/exploringBB/library/gpio/PWM.cpp

7.10 exploringBB::Servo Class Reference

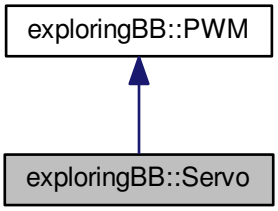
An extremely basic Servo class stub – does nothing more than the PWM class but is here for future use.

`#include <Servo.h>`

Inheritance diagram for exploringBB::Servo:



Collaboration diagram for exploringBB::Servo:



Public Member Functions

- Servo (string pinName)
- virtual ~Servo ()

Additional Inherited Members

7.10.1 Detailed Description

An extremely basic Servo class stub – does nothing more than the PWM class but is here for future use.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 exploringBB::Servo::Servo (string *pinName*)

```
29                          : PWM(pinName)
```

```
30 {
31         // TODO Auto-generated constructor stub
32
33 }
```

7.10.2.2 exploringBB::Servo::~Servo ( ) [virtual]

```
35              {
36         // TODO Auto-generated destructor stub
37 }
```

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

- /home/molloyd/exploringBB/library/motor/Servo.h
- /home/molloyd/exploringBB/library/motor/Servo.cpp

7.11 exploringBB::SevenSegmentDisplay Class Reference

A class that allows you to drive an array of 7 segment displays using an array of 74XX595 ICs.

`#include <SevenSegmentDisplay.h>`

Public Member Functions

- SevenSegmentDisplay (SPIDevice *device, int numberSegments)
- virtual int write (int number)
- virtual int write (float number, int places)
- virtual int setNumberBase (int base)
- virtual int getNumberBase ()
- virtual int getNumberSegments ()
- virtual void setCommonAnode (bool isCommonAnode)
- virtual ~SevenSegmentDisplay ()

7.11.1 Detailed Description

A class that allows you to drive an array of 7 segment displays using an array of 74XX595 ICs.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 exploringBB::SevenSegmentDisplay::SevenSegmentDisplay (SPIDevice * *device*, int *numberSegments*)

The constructor for the 7-segment display that defines the number of segments.

Parameters

device	The pointer to the SPI device bus
numberSegments	The number of 7-segment modules attached to the bus

```
44                                                                          {
45         this->spidevice = device;
46         this->numberSegments = numberSegments;
47         this->numberBase = 10; //decimal by default
48         this->isCommonAnode = false;
49 }
```

7.11.2.2 exploringBB::SevenSegmentDisplay::~SevenSegmentDisplay ( ) [virtual]

```
88                                          {
89          this->spidevice->close();
90  }
```

7.11.3 Member Function Documentation

7.11.3.1 virtual int exploringBB::SevenSegmentDisplay::getNumberBase ( ) [inline], [virtual]

```
47  { return this->numberBase; }
```

7.11.3.2 virtual int exploringBB::SevenSegmentDisplay::getNumberSegments ( ) [inline], [virtual]

```
48  { return this->numberSegments; }
```

7.11.3.3 virtual void exploringBB::SevenSegmentDisplay::setCommonAnode (bool *isCommonAnode*) [inline], [virtual]

```
49  { this->isCommonAnode = isCommonAnode; }
```

7.11.3.4 int exploringBB::SevenSegmentDisplay::setNumberBase (int *base*) [virtual]

```
51                                                  {
52          if (base>16 || base<2) return -1;
53          return (this->numberBase = base);
54  }
```

7.11.3.5 int exploringBB::SevenSegmentDisplay::write (int *number*) [virtual]

```
56                                          {
57      // going to send one character for each segment
58          unsigned char output[this->numberSegments];
59          // output least-significant digit and divide by base
60      for(int i=0; i<this->numberSegments; i++){
61         output[i] = this->symbols[number%this->numberBase];
62         if(this->isCommonAnode) output[i]=~output[i]; //invert the bits for common anode
63         number = number/this->numberBase;
64      }
65      this->spidevice->write(output, this->numberSegments);
66          return 0;
67  }
```

7.11.3.6 int exploringBB::SevenSegmentDisplay::write (float *number,* int *places*) [virtual]

```
69                                                      {
70          // if the number of places is greater than the number of segments -1, stop.
71          if (places>(this->numberSegments-1)) return -1;
72          // can display non-decimal floats
73          int intNumber = (int) number;
74          if (places>0) intNumber = (int)(number * places * this->numberBase);
75      // going to send one character for each segment
76          unsigned char output[this->numberSegments];
77          // output least-significant digit and divide by base
78      for(int i=0; i<this->numberSegments; i++){
79         output[i] = this->symbols[intNumber%this->numberBase];
80         if(i==places) output[i] = output[i] | 0b10000000; // turn on "decimal point"
81         if(this->isCommonAnode) output[i]=~output[i]; //invert the bits for common anode
82         intNumber = intNumber/this->numberBase;
83      }
84      this->spidevice->write(output, this->numberSegments);
85          return 0;
86  }
```

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

- /home/molloyd/exploringBB/library/display/SevenSegmentDisplay.h
- /home/molloyd/exploringBB/library/display/SevenSegmentDisplay.cpp

7.12 exploringBB::SocketClient Class Reference

A class that encapsulates a socket client to be used for network communication.

`#include <SocketClient.h>`

Public Member Functions

- SocketClient (std::string serverName, int portNumber)
- virtual int connectToServer ()
- virtual int disconnectFromServer ()
- virtual int send (std::string message)
- virtual std::string receive (int size)
- bool isClientConnected ()
- virtual ~SocketClient ()

7.12.1 Detailed Description

A class that encapsulates a socket client to be used for network communication.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 exploringBB::SocketClient::SocketClient (std::string *serverName,* int *portNumber*)

```
33                                                                  {
34         this->socketfd = -1;
35         this->server = NULL;
36         this->serverName = serverName;
37         this->portNumber = portNumber;
38         this->isConnected = false;
39 }
```

7.12.2.2 exploringBB::SocketClient::~SocketClient ( ) [virtual]

```
111                             {
112         if (this->isConnected == true){
113                 disconnectFromServer();
114         }
115 }
```

7.12.3 Member Function Documentation

7.12.3.1 int exploringBB::SocketClient::connectToServer ( ) [virtual]

```
41                                  {
42
43     socketfd = socket(AF_INET, SOCK_STREAM, 0);
44     if (socketfd < 0){
45         perror("Socket Client: error opening socket.\n");
46         return 1;
47     }
48     server = gethostbyname(serverName.data());
49     if (server == NULL) {
```

```
50          perror("Socket Client: error - no such host.\n");
51          return 1;
52      }
53      bzero((char *) &serverAddress, sizeof(serverAddress));
54      serverAddress.sin_family = AF_INET;
55      bcopy((char *)server->h_addr,(char *)&serverAddress.sin_addr.s_addr, server->h_length);
56      serverAddress.sin_port = htons(portNumber);
57
58      if (connect(socketfd, (struct sockaddr *) &serverAddress, sizeof(serverAddress)) < 0){
59          perror("Socket Client: error connecting to the server.\n");
60          return 1;
61      }
62      this->isConnected = true;
63      return 0;
64  }
```

7.12.3.2 int exploringBB::SocketClient::disconnectFromServer ( ) [virtual]

```
105                                         {
106          this->isConnected = false;
107          close(this->socketfd);
108          return 0;
109  }
```

7.12.3.3 bool exploringBB::SocketClient::isClientConnected ( ) [inline]

```
57 { return this->isConnected; }
```

7.12.3.4 string exploringBB::SocketClient::receive (int *size* = 1024) [virtual]

```
77                                              {
78      char readBuffer[size];
79      int n = read(this->socketfd, readBuffer, sizeof(readBuffer));
80      if (n < 0){
81         perror("Socket Client: error reading from socket");
82      }
83      return string(readBuffer);
84  }
```

7.12.3.5 int exploringBB::SocketClient::send (std::string *message*) [virtual]

```
66                                          {
67          const char *writeBuffer = message.data();
68          int length = message.length();
69      int n = write(this->socketfd, writeBuffer, length);
70      if (n < 0){
71         perror("Socket Client: error writing to socket");
72         return 1;
73      }
74      return 0;
75  }
```

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

- /home/molloyd/exploringBB/library/network/SocketClient.h
- /home/molloyd/exploringBB/library/network/SocketClient.cpp

7.13 exploringBB::SocketServer Class Reference

A class that encapsulates a server socket for network communication.

`#include <SocketServer.h>`

Public Member Functions

- SocketServer (int portNumber)
- virtual int listen ()
- virtual int send (std::string message)
- virtual std::string receive (int size)
- virtual ~SocketServer ()

7.13.1 Detailed Description

A class that encapsulates a server socket for network communication.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 exploringBB::SocketServer::SocketServer (int *portNumber*)

```
34                                          {
35         this->socketfd = -1;
36         this->clientSocketfd = -1;
37         this->portNumber = portNumber;
38         this->clientConnected = false;
39 }
```

7.13.2.2 exploringBB::SocketServer::~SocketServer ( ) [virtual]

```
87                              {
88         close(this->socketfd);
89         close(this->clientSocketfd);
90 }
```

7.13.3 Member Function Documentation

7.13.3.1 int exploringBB::SocketServer::listen ( ) [virtual]

```
41                          {
42     this->socketfd = socket(AF_INET, SOCK_STREAM, 0);
43     if (this->socketfd < 0){
44         perror("Socket Server: error opening socket.\n");
45         return 1;
46     }
47     bzero((char *) &serverAddress, sizeof(serverAddress));
48     serverAddress.sin_family = AF_INET;
49     serverAddress.sin_addr.s_addr = INADDR_ANY;
50     serverAddress.sin_port = htons(this->portNumber);
51     if (bind(socketfd, (struct sockaddr *) &serverAddress, sizeof(serverAddress)) < 0){
52         perror("Socket Server: error on binding the socket.\n");
53         return 1;
54     }
55     ::listen(this->socketfd, 5);
56     socklen_t clientLength = sizeof(this->clientAddress);
57     this->clientSocketfd = accept(this->socketfd,
58                 (struct sockaddr *) &this->clientAddress,
59                 &clientLength);
60     if (this->clientSocketfd < 0){
61         perror("Socket Server: Failed to bind the client socket properly.\n");
62         return 1;
63     }
64     return 0;
65 }
```

7.13.3.2 string exploringBB::SocketServer::receive (int *size* = 1024) [virtual]

```
78                                          {
79     char readBuffer[size];
80     int n = read(this->clientSocketfd, readBuffer, sizeof(readBuffer));
```

```
81     if (n < 0){
82        perror("Socket Server: error reading from server socket.");
83     }
84     return string(readBuffer);
85 }
```

7.13.3.3 int exploringBB::SocketServer::send (std::string *message*) [virtual]

```
67                                      {
68         const char *writeBuffer = message.data();
69         int length = message.length();
70     int n = write(this->clientSocketfd, writeBuffer, length);
71     if (n < 0){
72        perror("Socket Server: error writing to server socket.");
73        return 1;
74     }
75     return 0;
76 }
```

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

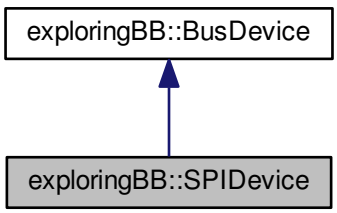
- /home/molloyd/exploringBB/library/network/SocketServer.h
- /home/molloyd/exploringBB/library/network/SocketServer.cpp

7.14 exploringBB::SPIDevice Class Reference

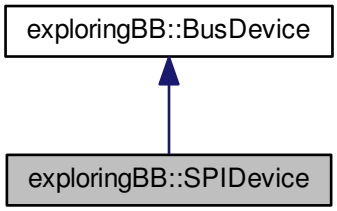
Generic SPI Device class that can be used to connect to any type of SPI device and read or write to its registers.

`#include <SPIDevice.h>`

Inheritance diagram for exploringBB::SPIDevice:



Collaboration diagram for exploringBB::SPIDevice:



Public Types

- enum SPIMODE { MODE0 = 0, MODE1 = 1, MODE2 = 2, MODE3 = 3 }
  The SPI Mode.

Public Member Functions

- SPIDevice (unsigned int bus, unsigned int device)
- virtual int open ()
- virtual unsigned char readRegister (unsigned int registerAddress)
- virtual unsigned char * readRegisters (unsigned int number, unsigned int fromAddress=0)
- virtual int writeRegister (unsigned int registerAddress, unsigned char value)
- virtual void debugDumpRegisters (unsigned int number=0xff)
- virtual int write (unsigned char value)
- virtual int write (unsigned char value[], int length)
- virtual int setSpeed (uint32_t speed)
- virtual int setMode (SPIDevice::SPIMODE mode)
- virtual int setBitsPerWord (uint8_t bits)
- virtual void close ()
- virtual ~SPIDevice ()
- virtual int transfer (unsigned char read[], unsigned char write[], int length)

Additional Inherited Members

7.14.1 Detailed Description

Generic SPI Device class that can be used to connect to any type of SPI device and read or write to its registers.

7.14.2 Member Enumeration Documentation

7.14.2.1 enum exploringBB::SPIDevice::SPIMODE

The SPI Mode.

Enumerator

MODE0 Low at idle, capture on rising clock edge.

MODE1 Low at idle, capture on falling clock edge.

MODE2 High at idle, capture on falling clock edge.

MODE3 High at idle, capture on rising clock edge.

```
42                      {
43                  MODE0 = 0,
44                  MODE1 = 1,
45                  MODE2 = 2,
46                  MODE3 = 3
47          };
```

7.14.3 Constructor & Destructor Documentation

7.14.3.1 exploringBB::SPIDevice::SPIDevice (unsigned int *bus*, unsigned int *device*)

The constructor for the SPIDevice that sets up and opens the SPI connection. The destructor will close the SPI file connection.

Parameters

bus	The SPI bus number X (first digit after spidevX.Y)
device	The device on the bus Y (second digit after spidevX.Y)

```
51                                                      :
52          BusDevice(bus,device) {
53          stringstream s;
54          s << SPI_PATH << bus << "." << device;
55          this->filename = string(s.str());
56          this->mode = SPIDevice::MODE3;
57          this->bits = 8;
58          this->speed = 500000;
59          this->delay = 0;
60          this->open();
61  }
```

7.14.3.2 exploringBB::SPIDevice::~SPIDevice ( ) [virtual]

The destructor closes the SPI bus device

```
253                         {
254          this->close();
255  }
```

7.14.4 Member Function Documentation

7.14.4.1 void exploringBB::SPIDevice::close ( ) [virtual]

Close the SPI device

Implements exploringBB::BusDevice.

```
245                         {
246          ::close(this->file);
247          this->file = -1;
248  }
```

7.14.4.2 void exploringBB::SPIDevice::debugDumpRegisters (unsigned int *number* = 0xff) [virtual]

A simple method to dump the registers to the standard output – useful for debugging

Parameters

number	the number of registers to dump

Implements exploringBB::BusDevice.

```
178                                                 {
179         cout << "SPI Mode: " << this->mode << endl;
180         cout << "Bits per word: " << (int)this->bits << endl;
181         cout << "Max speed: " << this->speed << endl;
182         cout << "Dumping Registers for Debug Purposes:" << endl;
183         unsigned char *registers = this->readRegisters(number);
184         for(int i=0; i<(int)number; i++){
185                 cout << HEX(*(registers+i)) << " ";
186                 if (i%16==15) cout << endl;
187         }
188         cout << dec;
189 }
```

7.14.4.3 int exploringBB::SPIDevice::open ( ) [virtual]

This method opens the file connection to the SPI device.

Returns

0 on a successful open of the file

Implements exploringBB::BusDevice.

```
67                      {
68         //cout << "Opening the file: " << filename.c_str() << endl;
69         if ((this->file = ::open(filename.c_str(), O_RDWR))<0){
70                 perror("SPI: Can't open device.");
71                 return -1;
72         }
73         if (this->setMode(this->mode)==-1) return -1;
74         if (this->setSpeed(this->speed)==-1) return -1;
75         if (this->setBitsPerWord(this->bits)==-1) return -1;
76         return 0;
77 }
```

7.14.4.4 unsigned char exploringBB::SPIDevice::readRegister (unsigned int *registerAddress*) [virtual]

A method to read a single register at the SPI address

Parameters

registerAddress	the address of the register from the device datasheet

Returns

the character that is returned from the address

Implements exploringBB::BusDevice.

```
108                                                             {
109         unsigned char send[2], receive[2];
110         memset(send, 0, sizeof send);
111         memset(receive, 0, sizeof receive);
112         send[0] = (unsigned char) (0x80 | registerAddress);
113         this->transfer(send, receive, 2);
114         //cout << "The value that was received is: " << (int) receive[1] << endl;
115         return receive[1];
116 }
```

7.14.4.5 unsigned char * exploringBB::SPIDevice::readRegisters (unsigned int *number*, unsigned int *fromAddress* = 0) [virtual]

A method to read a number of registers as a data array

Parameters

number	the number of registers to read
fromAddress	the starting address of the block of data

Returns

the data array that is returned (memory allocated by the method)

Implements exploringBB::BusDevice.

```
124                                                                                 {
125         unsigned char* data = new unsigned char[number];
126         unsigned char send[number+1], receive[number+1];
127         memset(send, 0, sizeof send);
128         send[0] =  (unsigned char) (0x80 | 0x40 | fromAddress); //set read bit and MB bit
129         this->transfer(send, receive, number+1);
130         memcpy(data, receive+1, number);  //ignore the first (address) byte in the array returned
131         return data;
132 }
```

7.14.4.6 int exploringBB::SPIDevice::setBitsPerWord (uint8_t *bits*) [virtual]

Set the number of bits per word of the SPI bus

Parameters

bits	the number of bits per word

```
229                                         {
230         this->bits = bits;
231         if (ioctl(this->file, SPI_IOC_WR_BITS_PER_WORD, &this->bits)==-1){
232                 perror("SPI: Can't set bits per word.");
233                 return -1;
234         }
235         if (ioctl(this->file, SPI_IOC_RD_BITS_PER_WORD, &this->bits)==-1){
236                 perror("SPI: Can't get bits per word.");
237                 return -1;
238         }
239         return 0;
240 }
```

7.14.4.7 int exploringBB::SPIDevice::setMode (SPIDevice::SPIMODE *mode*) [virtual]

Set the mode of the SPI bus

Parameters

mode	the enumerated SPI mode

```
212                                         {
213         this->mode = mode;
214         if (ioctl(this->file, SPI_IOC_WR_MODE, &this->mode)==-1){
215                 perror("SPI: Can't set SPI mode.");
216                 return -1;
217         }
218         if (ioctl(this->file, SPI_IOC_RD_MODE, &this->mode)==-1){
219                 perror("SPI: Can't get SPI mode.");
220                 return -1;
221         }
222         return 0;
223 }
```

7.14.4.8 int exploringBB::SPIDevice::setSpeed (uint32_t *speed*) [virtual]

Set the speed of the SPI bus

Parameters

speed	the speed in Hz

```
195                                     {
196         this->speed = speed;
197         if (ioctl(this->file, SPI_IOC_WR_MAX_SPEED_HZ, &this->speed)==-1){
198                 perror("SPI: Can't set max speed HZ");
199                 return -1;
200         }
201         if (ioctl(this->file, SPI_IOC_RD_MAX_SPEED_HZ, &this->speed)==-1){
202                 perror("SPI: Can't get max speed HZ.");
203                 return -1;
204         }
205         return 0;
206 }
```

7.14.4.9 int exploringBB::SPIDevice::transfer (unsigned char *send[]*, unsigned char *receive[]*, int *length*) [virtual]

Generic method to transfer data to and from the SPI device. It is used by the following methods to read and write registers.

Parameters

send	The array of data to send to the SPI device
receive	The array of data to receive from the SPI device
length	The length of the array to send

Returns

-1 on failure

```
87                                                                                      {
88         struct spi_ioc_transfer transfer;
89         transfer.tx_buf = (unsigned long) send;
90         transfer.rx_buf = (unsigned long) receive;
91         transfer.len = length;
92         transfer.speed_hz = this->speed;
93         transfer.bits_per_word = this->bits;
94         transfer.delay_usecs = this->delay;
95         int status = ioctl(this->file, SPI_IOC_MESSAGE(1), &transfer);
96         if (status < 0) {
97                 perror("SPI: SPI_IOC_MESSAGE Failed");
98                 return -1;
99         }
100         return status;
101 }
```

7.14.4.10 int exploringBB::SPIDevice::write (unsigned char *value*) [virtual]

A write method that writes a single character to the SPI bus

Parameters

value	the value to write to the bus

Returns

returns 0 if successful

Implements exploringBB::BusDevice.

```
139                                     {
140         unsigned char null_return = 0x00;
141         //printf("[%02x]", value);
142         this->transfer(&value, &null_return, 1);
143         return 0;
144 }
```

7.14.4.11 int exploringBB::SPIDevice::write (unsigned char *value[],* int *length*) [virtual]

A write method that writes a block of data of the length to the bus.

Parameters

value	the array of data to write to the device
length	the length of the data array

Returns

returns 0 if successful

```
152                                                 {
153         unsigned char null_return = 0x00;
154         this->transfer(value, &null_return, length);
155         return 0;
156 }
```

7.14.4.12 int exploringBB::SPIDevice::writeRegister (unsigned int *registerAddress*, unsigned char *value*) [virtual]

Writes a value to a defined register address (check the datasheet for the device)

Parameters

registerAddress	the address of the register to write to
value	the value to write to the register

Returns

returns 0 if successful

Implements exploringBB::BusDevice.

```
164                                                                                 {
165         unsigned char send[2], receive[2];
166         memset(receive, 0, sizeof receive);
167         send[0] = (unsigned char) registerAddress;
168         send[1] = value;
169         //cout << "The value that was written is: " << (int) send[1] << endl;
170         this->transfer(send, receive, 2);
171         return 0;
172 }
```

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

- /home/molloyd/exploringBB/library/bus/SPIDevice.h
- /home/molloyd/exploringBB/library/bus/SPIDevice.cpp

7.15 exploringBB::StepperMotor Class Reference

A class to control a stepper motor using a motor driver board, such as the Easy Driver board, or compatible. The class uses five GPIOs to control each motor.

`#include <StepperMotor.h>`

Public Types

- enum STEP_MODE { STEP_FULL, STEP_HALF, STEP_QUARTER, STEP_EIGHT }
- enum DIRECTION { CLOCKWISE, ANTICLOCKWISE }

Public Member Functions

- StepperMotor (GPIO *gpio_MS1, GPIO *gpio_MS2, GPIO *gpio_STEP, GPIO *gpio_SLP, GPIO *gpio_DIR, int speedRPM=60, int stepsPerRevolution=200)
- StepperMotor (int gpio_MS1, int gpio_MS2, int gpio_STEP, int gpio_SLP, int gpio_DIR, int speedRPM=60, int stepsPerRevolution=200)
- virtual void step ()
- virtual void step (int numberOfSteps)
- virtual int threadedStepForDuration (int numberOfSteps, int duration_ms)
- virtual void threadedStepCancel ()
- virtual void rotate (float degrees)
- virtual void setDirection (DIRECTION direction)
- virtual DIRECTION getDirection ()
- virtual void reverseDirection ()
- virtual void setStepMode (STEP_MODE mode)
- virtual STEP_MODE getStepMode ()
- virtual void setSpeed (float rpm)
- virtual float getSpeed ()
- virtual void setStepsPerRevolution (int steps)
- virtual int getStepsPerRevolution ()
- virtual void sleep ()
- virtual void wake ()
- virtual bool isAsleep ()
- virtual ~StepperMotor ()

Friends

- void * threadedStep (void *value)

7.15.1 Detailed Description

A class to control a stepper motor using a motor driver board, such as the Easy Driver board, or compatible. The class uses five GPIOs to control each motor.

7.15.2 Member Enumeration Documentation

7.15.2.1 enum exploringBB::StepperMotor::DIRECTION

Enumerator

CLOCKWISE

ANTICLOCKWISE

```
40 { CLOCKWISE, ANTICLOCKWISE };
```

7.15.2.2 enum exploringBB::StepperMotor::STEP_MODE

Enumerator

STEP_FULL

STEP_HALF

STEP_QUARTER

STEP_EIGHT

```
39 { STEP_FULL, STEP_HALF, STEP_QUARTER, STEP_EIGHT };
```

7.15.3 Constructor & Destructor Documentation

7.15.3.1 exploringBB::StepperMotor::StepperMotor (GPIO * *gpio_MS1*, GPIO * *gpio_MS2*, GPIO * *gpio_STEP*, GPIO * *gpio_SLP*, GPIO * *gpio_DIR*, int *speedRPM* = 60, int *stepsPerRevolution* = 200)

```
34                                                                                  {
35          this->gpio_MS1  = gpio_MS1;
36          this->gpio_MS2  = gpio_MS2;
37          this->gpio_STEP = gpio_STEP;
38          this->gpio_SLP  = gpio_SLP;
39          this->gpio_DIR  = gpio_DIR;
40          // the default speed in rpm
41          this->setSpeed(speedRPM);
42          this->stepsPerRevolution = stepsPerRevolution;
43          this->init(speedRPM, stepsPerRevolution);
44  }
```

7.15.3.2 exploringBB::StepperMotor::StepperMotor (int *gpio_MS1*, int *gpio_MS2*, int *gpio_STEP*, int *gpio_SLP*, int *gpio_DIR*, int *speedRPM* = 60, int *stepsPerRevolution* = 200)

```
47                                                                                      {
48          this->gpio_MS1 = new GPIO(gpio_MS1);
49          this->gpio_MS2 = new GPIO(gpio_MS2);
50          this->gpio_STEP = new GPIO(gpio_STEP);
51          this->gpio_SLP = new GPIO(gpio_SLP);
52          this->gpio_DIR = new GPIO(gpio_DIR);
53          this->gpio_MS1->setDirection(GPIO::OUTPUT);
54          this->gpio_MS2->setDirection(GPIO::OUTPUT);
55          this->gpio_STEP->setDirection(GPIO::OUTPUT);
56          this->gpio_SLP->setDirection(GPIO::OUTPUT);
57          this->gpio_DIR->setDirection(GPIO::OUTPUT);
58          this->init(speedRPM, stepsPerRevolution);
59  }
```

7.15.3.3 exploringBB::StepperMotor::~StepperMotor ( ) [virtual]

```
176 {}
```

7.15.4 Member Function Documentation

7.15.4.1 virtual DIRECTION exploringBB::StepperMotor::getDirection ( ) [inline], [virtual]

```
67 { return this->direction; }
```

7.15.4.2 virtual float exploringBB::StepperMotor::getSpeed ( ) [inline], [virtual]

```
72 { return speed; }
```

7.15.4.3 virtual STEP_MODE exploringBB::StepperMotor::getStepMode ( ) [inline], [virtual]

```
70 { return stepMode; }
```

7.15.4.4 virtual int exploringBB::StepperMotor::getStepsPerRevolution ( ) [inline], [virtual]

```
74 { return stepsPerRevolution; }
```

7.15.4.5 virtual bool exploringBB::StepperMotor::isAsleep ( ) [inline], [virtual]

```
77 { return asleep; }
```

7.15.4.6 void exploringBB::StepperMotor::reverseDirection ( ) [virtual]

```
151                                     {
152         if(this->direction==CLOCKWISE){
153                 this->setDirection(ANTICLOCKWISE);
154         }
155         else this->setDirection(CLOCKWISE);
156 }
```

7.15.4.7 void exploringBB::StepperMotor::rotate (float *degrees*) [virtual]

```
158                                         {
159         float degreesPerStep = 360.0f/getStepsPerRevolution();
160         int numberOfSteps = floor(((this->delayFactor*degrees)/degreesPerStep)+0.5);
161         //cout << "The number of steps is " << numberOfSteps << endl;
162         //cout << "The delay factor is " << delayFactor << endl;
163         step(numberOfSteps);
164 }
```

7.15.4.8 void exploringBB::StepperMotor::setDirection (DIRECTION *direction*) [virtual]

```
145                                                 {
146         this->direction = direction;
147         if(this->direction==CLOCKWISE) this->gpio_DIR->setValue(
    GPIO::HIGH);
148                 else this->gpio_DIR->setValue(GPIO::LOW);
149 }
```

7.15.4.9 void exploringBB::StepperMotor::setSpeed (float *rpm*) [virtual]

```
108                                     {
109         this->speed = rpm;
110         float delayPerSec = (60/rpm)/stepsPerRevolution;    // delay per step in seconds
111         this->uSecDelay = (int)(delayPerSec * 1000 * 1000); // in microseconds
112 }
```

7.15.4.10 void exploringBB::StepperMotor::setStepMode (STEP_MODE *mode*) [virtual]

```
82                                              {
83         this->stepMode = mode;
84         switch(stepMode){
85         case STEP_FULL:
86                 this->gpio_MS1->setValue(GPIO::LOW);
87                 this->gpio_MS2->setValue(GPIO::LOW);
88                 this->delayFactor = 1;
89                 break;
90         case STEP_HALF:
91                 this->gpio_MS1->setValue(GPIO::HIGH);
92                 this->gpio_MS2->setValue(GPIO::LOW);
93                 this->delayFactor = 2;
94                 break;
95         case STEP_QUARTER:
96                 this->gpio_MS1->setValue(GPIO::LOW);
97                 this->gpio_MS2->setValue(GPIO::HIGH);
98                 this->delayFactor = 4;
99                 break;
100         case STEP_EIGHT:
101                 this->gpio_MS1->setValue(GPIO::HIGH);
102                 this->gpio_MS2->setValue(GPIO::HIGH);
103                 this->delayFactor = 8;
104                 break;
105         }
106 }
```

7.15.4.11 virtual void exploringBB::StepperMotor::setStepsPerRevolution (int *steps*) [inline], [virtual]

```
73 { stepsPerRevolution = steps; }
```

7.15.4.12 void exploringBB::StepperMotor::sleep ( ) [virtual]

```
166                          {
167         this->asleep = true;
168         this->gpio_SLP->setValue(GPIO::LOW);
169 }
```

7.15.4.13 void exploringBB::StepperMotor::step ( ) [virtual]

```
127                         {
128     this->gpio_STEP->setValue(GPIO::LOW);
129     this->gpio_STEP->setValue(GPIO::HIGH);
130 }
```

7.15.4.14 void exploringBB::StepperMotor::step (int *numberOfSteps*) [virtual]

```
114                                         {
115         //cout << "Doing "<< numberOfSteps << " steps and going to sleep for " << uSecDelay/delayFactor <<
        "uS\n";
116         int sleepDelay = uSecDelay/delayFactor;
117         if(numberOfSteps<0) {
118                 this->reverseDirection();
119                 numberOfSteps = -numberOfSteps;
120         }
121         for(int i=0; i<numberOfSteps; i++){
122                 this->step();
123                 usleep(sleepDelay);
124         }
125 }
```

7.15.4.15 virtual void exploringBB::StepperMotor::threadedStepCancel ( ) [inline], [virtual]

```
64 { this->threadRunning = false; }
```

7.15.4.16 int exploringBB::StepperMotor::threadedStepForDuration (int *numberOfSteps*, int *duration_ms*) [virtual]

```
133                                                                                 {
134         this->threadedStepNumber = numberOfSteps;
135         this->threadedStepPeriod = duration_ms/numberOfSteps;
136         this->threadRunning = true;
137     if(pthread_create(&this->thread, NULL, &threadedStep, static_cast<void*>(this))){
138         perror("StepperMotor: Failed to create the stepping thread");
139         this->threadRunning = false;
140         return -1;
141     }
142     return 0;
143 }
```

7.15.4.17 void exploringBB::StepperMotor::wake ( ) [virtual]

```
171                          {
172         this->asleep = false;
173         this->gpio_SLP->setValue(GPIO::HIGH);
174 }
```

7.15.5 Friends And Related Function Documentation

7.15.5.1 void* threadedStep (void * *value*) [friend]

```
179                                 {
180         StepperMotor *stepper = static_cast<StepperMotor*>(value);
181         while(stepper->threadRunning){
182                 stepper->step();
```

```
183                 usleep(stepper->threadedStepPeriod * 1000);  // convert from ms to us
184                 if(stepper->threadedStepNumber>0) stepper->threadedStepNumber--;
185                 if(stepper->threadedStepNumber==0) stepper->threadRunning = false;
186         }
187         return 0;
188 }
```

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

- /home/molloyd/exploringBB/library/motor/StepperMotor.h
- /home/molloyd/exploringBB/library/motor/StepperMotor.cpp

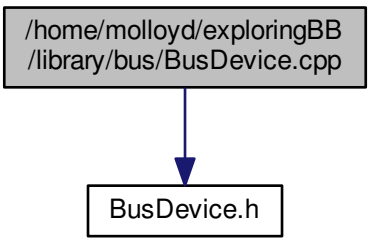
Chapter 8

File Documentation

8.1 /home/molloyd/exploringBB/library/bus/BusDevice.cpp File Reference

```
#include "BusDevice.h"
```

Include dependency graph for BusDevice.cpp:

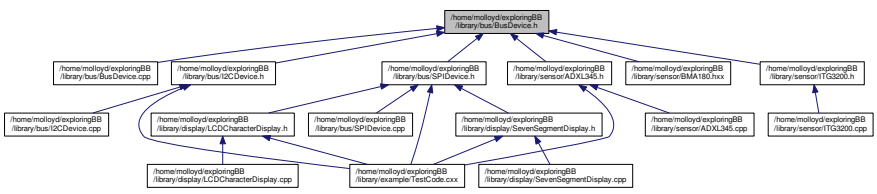


Namespaces

- exploringBB

8.2 /home/molloyd/exploringBB/library/bus/BusDevice.h File Reference

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



Data Structures

- class exploringBB::BusDevice

  This class is the parent of I2C and SPI devices, so that devices that use both SPI and I2C interfaces can use those interfaces interchangeably. Because it contains abstract methods, the child classes MUST implement the methods that are listed in this class.

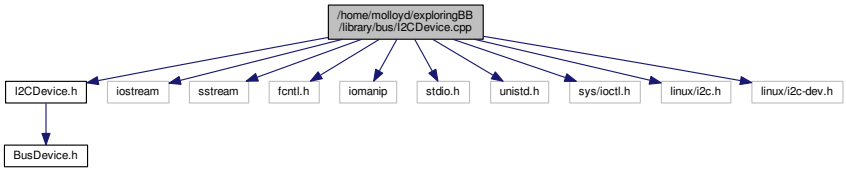
Namespaces

- exploringBB

8.3 /home/molloyd/exploringBB/library/bus/I2CDevice.cpp File Reference

```
#include "I2CDevice.h"
#include <iostream>
#include <sstream>
#include <fcntl.h>
#include <iomanip>
#include <stdio.h>
#include <unistd.h>
#include <sys/ioctl.h>
#include <linux/i2c.h>
#include <linux/i2c-dev.h>
```

Include dependency graph for I2CDevice.cpp:



Namespaces

- exploringBB

Macros

- #define HEX(x) setw(2) << setfill('0') << hex << (int)(x)

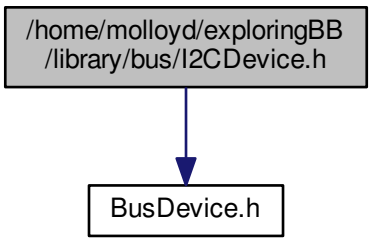
8.3.1 Macro Definition Documentation

8.3.1.1 #define HEX(*x*) setw(2) << setfill('0') << hex << (int)(x)

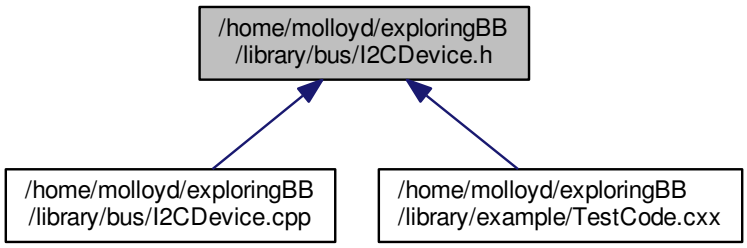
8.4 /home/molloyd/exploringBB/library/bus/I2CDevice.h File Reference

```
#include "BusDevice.h"
```

Include dependency graph for I2CDevice.h:



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



Data Structures

- class exploringBB::I2CDevice

  Generic I2C Device class that can be used to connect to any type of I2C device and read or write to its registers.

Namespaces

- exploringBB

Macros

- #define BBB_I2C_0 "/dev/i2c-0"
- #define BBB_I2C_1 "/dev/i2c-1"

8.4.1 Macro Definition Documentation

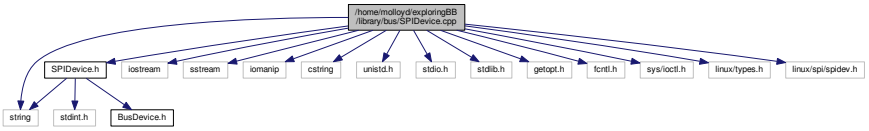
8.4.1.1 #define BBB_I2C_0 "/dev/i2c-0"

8.4.1.2 #define BBB_I2C_1 "/dev/i2c-1"

8.5 /home/molloyd/exploringBB/library/bus/SPIDevice.cpp File Reference

```
#include "SPIDevice.h"
#include <iostream>
#include <sstream>
#include <iomanip>
#include <cstring>
#include <string>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <getopt.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <linux/types.h>
#include <linux/spi/spidev.h>
```

Include dependency graph for SPIDevice.cpp:



Namespaces

- exploringBB

Macros

- #define HEX(x) setw(2) << setfill('0') << hex << (int)(x)

  Macro for filling in leading 0 on HEX outputs.

8.5.1 Macro Definition Documentation

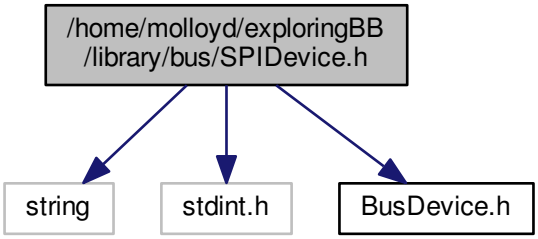
8.5.1.1 #define HEX(*x*) setw(2) << setfill('0') << hex << (int)(x)

Macro for filling in leading 0 on HEX outputs.

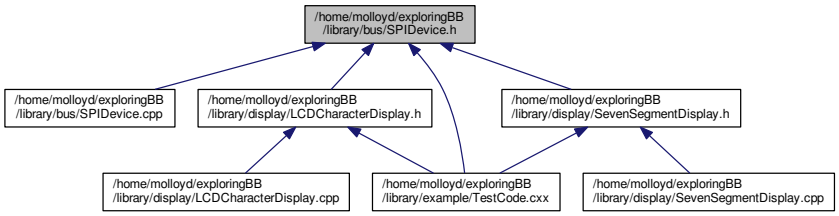
8.6 /home/molloyd/exploringBB/library/bus/SPIDevice.h File Reference

```
#include <string>
#include <stdint.h>
#include "BusDevice.h"
```

Include dependency graph for SPIDevice.h:



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



Data Structures

- class exploringBB::SPIDevice

  Generic SPI Device class that can be used to connect to any type of SPI device and read or write to its registers.

Namespaces

- exploringBB

Macros

- #define SPI_PATH "/dev/spidev"

8.6.1 Macro Definition Documentation

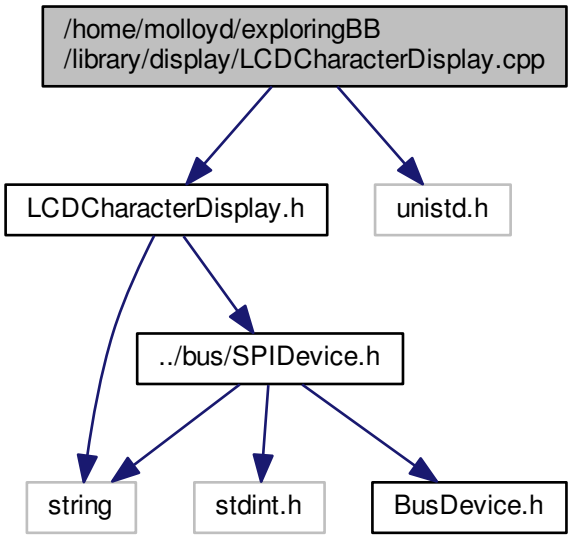
8.6.1.1 #define SPI_PATH "/dev/spidev"

The general path to an SPI device

8.7 /home/molloyd/exploringBB/library/display/LCDCharacterDisplay.cpp File Reference

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

Include dependency graph for LCDCharacterDisplay.cpp:



Namespaces

- exploringBB

Macros

- #define LCD_CLEAR_DISPLAY 0b00000001
- #define LCD_RETURN_HOME 0b00000010
- #define LCD_ENTRY_MODE_SET 0b00000100
- #define ENTRY_MODE_LEFT 0b00000010
- #define ENTRY_MODE_S 0b00000001
- #define LCD_DISPLAY_ON_OFF 0b00001000
- #define DISPLAY_ENTIRE 0b00000100
- #define DISPLAY_CURSOR 0b00000010
- #define DISPLAY_CURSOR_POS 0b00000001
- #define LCD_CURSOR_DISPLAY 0b00010000
- #define CURSOR_DISPLAY_SC 0b00001000
- #define CURSOR_DISPLAY_RL 0b00000100
- #define LCD_FUNCTION_SET 0b00100000
- #define LCD_CGRAM_ADDR 0b01000000
- #define LCD_DDRAM_ADDR 0b10000000
- #define LCD_LONG_DELAY 1520
- #define LCD_SHORT_DELAY 37
- #define LCD_ROW_OFFSET_ADDR 0x40

8.7.1 Macro Definition Documentation

8.7.1.1 #define CURSOR_DISPLAY_RL 0b00000100

8.7.1.2 #define CURSOR_DISPLAY_SC 0b00001000

8.7.1.3 #define DISPLAY_CURSOR 0b00000010

8.7.1.4 #define DISPLAY_CURSOR_POS 0b00000001

8.7.1.5 #define DISPLAY_ENTIRE 0b00000100

8.7.1.6 #define ENTRY_MODE_LEFT 0b00000010

8.7.1.7 #define ENTRY_MODE_S 0b00000001

8.7.1.8 #define LCD_CGRAM_ADDR 0b01000000

8.7.1.9 #define LCD_CLEAR_DISPLAY 0b00000001

8.7.1.10 #define LCD_CURSOR_DISPLAY 0b00010000

8.7.1.11 #define LCD_DDRAM_ADDR 0b10000000

8.7.1.12 #define LCD_DISPLAY_ON_OFF 0b00001000

8.7.1.13 #define LCD_ENTRY_MODE_SET 0b00000100

8.7.1.14 #define LCD_FUNCTION_SET 0b00100000

8.7.1.15 #define LCD_LONG_DELAY 1520

8.7.1.16 #define LCD_RETURN_HOME 0b00000010

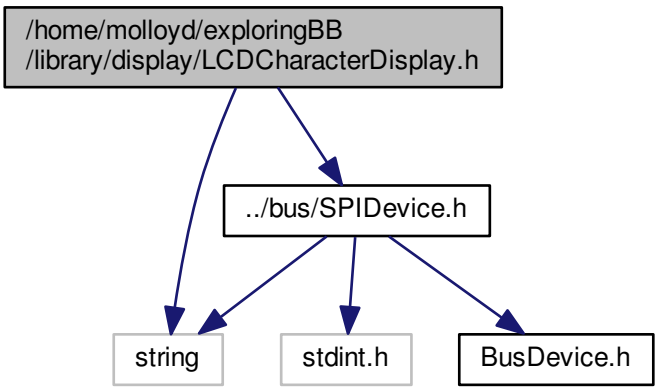
8.7.1.17 #define LCD_ROW_OFFSET_ADDR 0x40

8.7.1.18 #define LCD_SHORT_DELAY 37

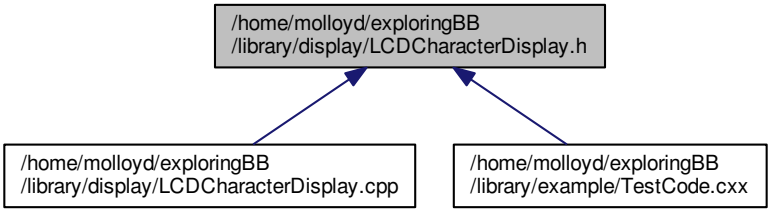
8.8 /home/molloyd/exploringBB/library/display/LCDCharacterDisplay.h File Reference

```
#include "../bus/SPIDevice.h"
#include <string>
```

Include dependency graph for LCDCharacterDisplay.h:



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



Data Structures

- class exploringBB::LCDCharacterDisplay

  A class that provides an interface to an LCD character module. It provices support for multiple rows and columns and provides methods for formatting and printing text. You should use a 4 wire interface and a 74XX595 to communicate with the display module.

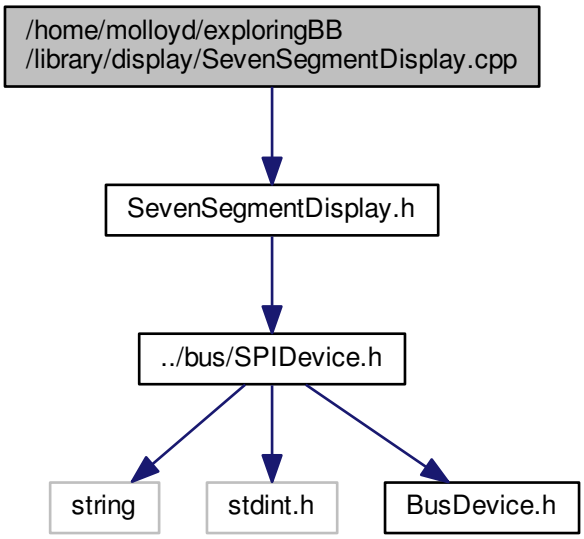
Namespaces

- exploringBB

8.9 /home/molloyd/exploringBB/library/display/SevenSegmentDisplay.cpp File Reference

```
#include "SevenSegmentDisplay.h"
```

Include dependency graph for SevenSegmentDisplay.cpp:



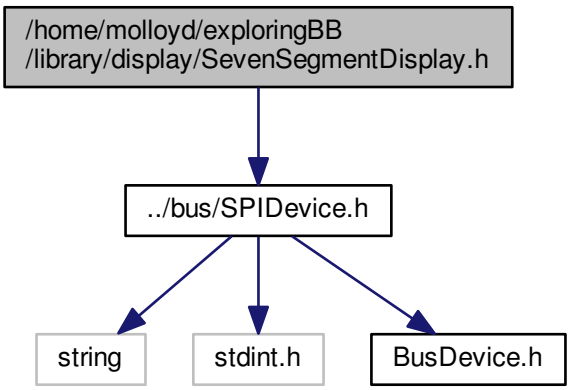
Namespaces

- exploringBB

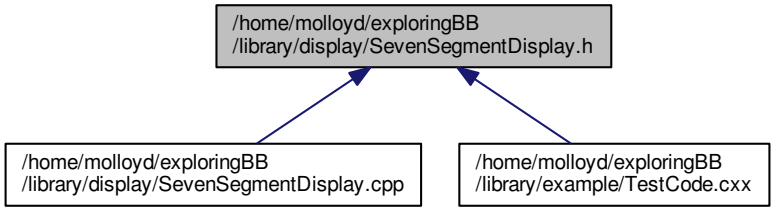
8.10 /home/molloyd/exploringBB/library/display/SevenSegmentDisplay.h File Reference

```
#include "../bus/SPIDevice.h"
```

Include dependency graph for SevenSegmentDisplay.h:



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



Data Structures

- class exploringBB::SevenSegmentDisplay

  A class that allows you to drive an array of 7 segment displays using an array of 74XX595 ICs.

Namespaces

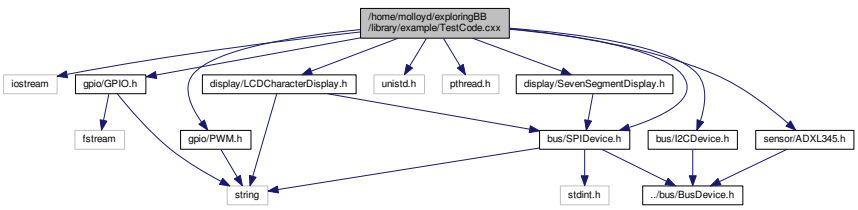
- exploringBB

8.11 front_page.cpp File Reference

8.12 /home/molloyd/exploringBB/library/example/TestCode.cxx File Reference

```
#include <iostream>
#include "gpio/GPIO.h"
#include "gpio/PWM.h"
#include "sensor/ADXL345.h"
#include <unistd.h>
#include <pthread.h>
#include "bus/I2CDevice.h"
#include "bus/SPIDevice.h"
#include "display/SevenSegmentDisplay.h"
#include "display/LCDCharacterDisplay.h"
```

Include dependency graph for TestCode.cxx:



Functions

- int callbackFunction (int var)
- int main ()

8.12.1 Function Documentation

8.12.1.1 int callbackFunction (int *var*)

```
46                              {
47          cout << "BBB Button Pressed!" << var << endl;
48          return var;
49  }
```

8.12.1.2 int main ( )

```
51             {
52
53          int32_t x = 54;
54
55
56          /*if(getuid()!=0){
57                cout << "You must run this program as root. Exiting." << endl;
58                return -1;
59      }*/
60
61          /*SPIDevice spi(1,0);
62          ADXL345 acc1(&spi);
63          acc1.displayPitchAndRoll(100);
64
65          I2CDevice i2c(1,0x53);
66          ADXL345 acc2(&i2c);
67          acc2.displayPitchAndRoll(100);*/
68
69
70          LCDCharacterDisplay display(new SPIDevice(2,0), 16, 2);
71          display.clear();
72          display.setAutoscroll(true);
73
74          //display.setScrollDisplayLeft(true);
75          //display.setCursorPosition(0,2);
76          display.print("Exploring BB");
77          //usleep(2000000);
78
79          //display.setCursorPosition(1,0);
80          //display.print("by Derek Molloy");
81          //usleep(2000000);
82
83  /*
84          SevenSegmentDisplay disp(new SPIDevice(1,0), 2);
85          //display.setNumberBase(7);
86          for(int i=80; i>=0; i--){
87                  disp.write(i);
88                  usleep(250000);
89          }*/
90
91
92
93          //   SPIDevice spi(1,0);             // chip select 0 on bus 1
94          //   spi.setSpeed(1000000);          // set the speed to 1 MHz
95          //   cout << "The device ID is: " << (int) spi.readRegister(0x00) << endl;
96          //   spi.setMode(SPIDevice::MODE3);  // set the mode to Mode3
97          //   spi.writeRegister(0x2D, 0x08);  // POWER_CTL for the ADXL345
98          //   spi.debugDumpRegisters(0x40);   // Dump the 64 registers from 0x00
99
100
101
102          /*for(int i=0; i<=255; i++){
103                  spi.write((unsigned int)i);
104                  usleep(500000);
105          }*/
106
107
108
109
110          //cout << "The device ID is: " << (int) spi.readRegister(0x00) << endl;
111          //spi.writeRegister(0x2D, 0x08); //POWER_CTL
112      //spi.debugDumpRegisters(0x40);
```

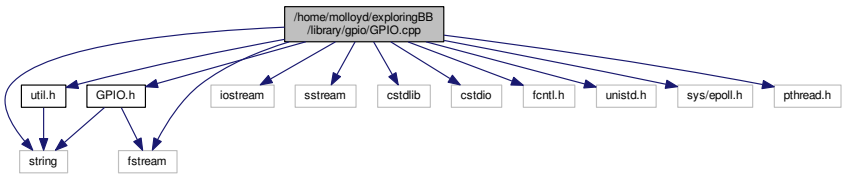
```
113
114
115
116         //spi.readRegister(0x80 + 0x32);
117         //spi.readRegister(0x80 + 0x33);
118
119
120         /*I2C i2c(1,0x53);
121         cout << "The address is: " << (int)i2c.readRegister(0x00) << endl;
122         unsigned char* data = i2c.readRegisters(0x40);
123         cout << "The value of the first address is: " << (int) *data << endl;*/
124
125         /* I2C
126         ADXL345 sensor(1, 0x53);
127         sensor.readSensorState();
128         cout << "The x acceleration is " << sensor.getAccelerationX() << endl;
129         cout << "The y acceleration is " << sensor.getAccelerationY() << endl;
130         cout << "The z acceleration is " << sensor.getAccelerationZ() << endl;
131         */
132         //sensor.updateSensorState();
133
134         //sensor.debugDumpRegisters();
135
136         /*sensor.setResolution(ADXL345::NORMAL);
137         sensor.setRange(ADXL345::PLUSMINUS_4_G);
138         sensor.readSensorState();
139         //sensor.calculatePitchAndRoll();
140
141         sensor.setResolution(ADXL345::HIGH);
142         sensor.setRange(ADXL345::PLUSMINUS_16_G);
143         sensor.readSensorState();*/
144
145     // sensor.displayPitchAndRoll();
146
147
148
149
150         //cout << "**Resolution is: " << (int)sensor.getResolution() << " and Range is: " <<
    (int)sensor.getRange() << endl;
151
152
153
154         /*PWM pwm("pwm_test_P9_22.15");
155         //pwm.calibrateAnalogMax(3.318);
156         //pwm.analogWrite(1.25);
157
158         pwm.setPeriod(10000);
159         pwm.setDutyCycle(50.0f);
160         pwm.setPolarity(PWM::ACTIVE_LOW);
161         pwm.run();*/
162
163         /*pwm.setPeriod(10000);
164         cout << "The period is: " << pwm.getPeriod() << endl;
165     pwm.setFrequency(1000);
166     cout << "The frequency is: " << pwm.getFrequency() << endl;
167     cout << "The period is: " << pwm.getPeriod() << endl;
168     pwm.setDutyCycle(66.66f);
169     cout << "The duty cycle is: " << pwm.getDutyCycle() << endl;
170     cout << "The duty cycle% is: " << pwm.getDutyCyclePercent() << endl;
171     cout << "The polarity is: " << pwm.getPolarity() << endl;
172     pwm.invertPolarity();
173     cout << "The polarity is: " << pwm.getPolarity() << endl;
174     cout << "Is running? " << pwm.isRunning() << endl;
175     pwm.stop();
176     cout << "Is running? " << pwm.isRunning() << endl;*/
177
178 /*      cout << "BeagleBone Poll Test" << endl;
179
180         GPIO inGPIO(48);
181         GPIO outGPIO(60);
182
183         inGPIO.setDirection(GPIO::INPUT);
184         inGPIO.setEdgeType(GPIO::RISING);
185         outGPIO.setDirection(GPIO::OUTPUT);
186
187         cout << "GPIO(48) has value: " << inGPIO.getValue() << endl;
188         inGPIO.setDebounceTime(200);
189         inGPIO.waitForEdge(&callbackFunction);
190         outGPIO.toggleOutput(100);
191     cout << "Poll Started: Press the button:" << endl;
192     usleep(10000000);
193     cout << "Finished sleeping for 10 seconds" << endl;*/
194
195
196         //GPIO outgpio(60);
197         /*gpio.setDirection(OUTPUT);
198         gpio.setEdge(NONE);
```

```
199
200         for (int i=0; i<100000; i++){
201                 gpio.setValue(HIGH);
202                 gpio.setValue(LOW);
203         }*/
204
205         /*gpio.streamOpen();
206         for (int i=0; i<1000000; i++){
207                 gpio.streamWrite(HIGH);
208                 gpio.streamWrite(LOW);
209         }
210         gpio.streamClose();*/
211
212
213         /*GPIO ingpio(48);
214         ingpio.setDirection(INPUT);
215         ingpio.setEdge(FALLING);
216         for (int i=0; i<10; i++){
217                 usleep(1000000);
218                 cout << "GPIO(48) has value: " << ingpio.getValue() << endl;
219         }*/
220
221         //cout << "finished" << endl;
222         return 0;
223 }
```

8.13 /home/molloyd/exploringBB/library/gpio/GPIO.cpp File Reference

```
#include "GPIO.h"
#include "util.h"
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <cstdlib>
#include <cstdio>
#include <fcntl.h>
#include <unistd.h>
#include <sys/epoll.h>
#include <pthread.h>
```

Include dependency graph for GPIO.cpp:



Namespaces

- exploringBB

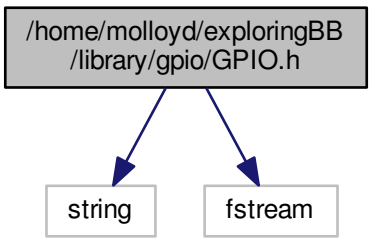
Functions

- void * exploringBB::threadedToggle (void *value)
- void * exploringBB::threadedPoll (void *value)

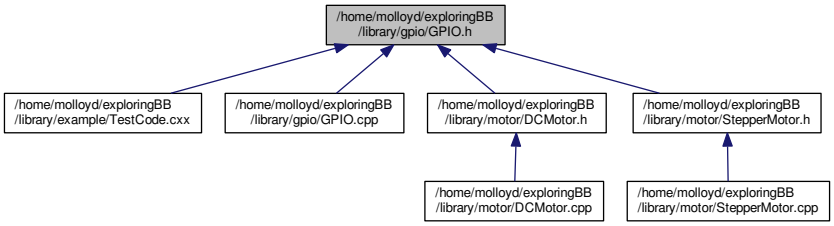
8.14 /home/molloyd/exploringBB/library/gpio/GPIO.h File Reference

```
#include <string>
#include <fstream>
```

Include dependency graph for GPIO.h:



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



Data Structures

- class exploringBB::GPIO

  GPIO class for input and output functionality on a single GPIO pin.

Namespaces

- exploringBB

Macros

- #define GPIO_PATH "/sys/class/gpio/"

Typedefs

- typedef int(∗ exploringBB::CallbackType)(int)

Functions

- void * exploringBB::threadedPoll (void *value)
- void * exploringBB::threadedToggle (void *value)

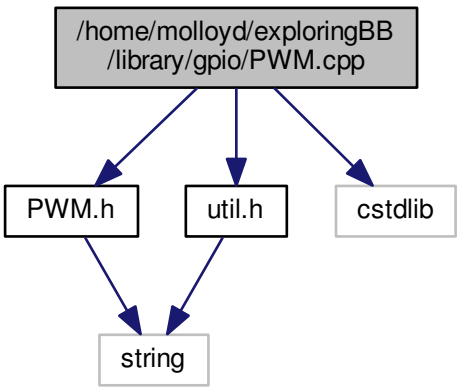
8.14.1 Macro Definition Documentation

8.14.1.1 #define GPIO_PATH "/sys/class/gpio/"

8.15 /home/molloyd/exploringBB/library/gpio/PWM.cpp File Reference

```
#include "PWM.h"
#include "util.h"
#include <cstdlib>
```

Include dependency graph for PWM.cpp:



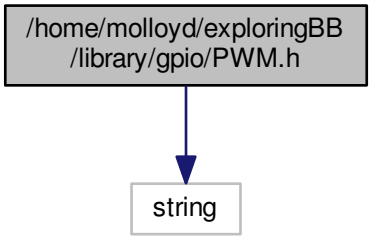
Namespaces

- exploringBB

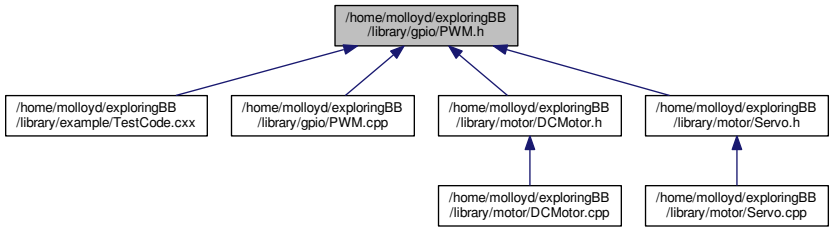
8.16 /home/molloyd/exploringBB/library/gpio/PWM.h File Reference

```
#include <string>
```

Include dependency graph for PWM.h:



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



Data Structures

- class exploringBB::PWM

  A class to control a basic PWM output – you must know the exact sysfs filename for the PWM output.

Namespaces

- exploringBB

Macros

- #define PWM_PATH "/sys/devices/ocp.3/"
- #define PWM_PERIOD "period"
- #define PWM_DUTY "duty"
- #define PWM_POLARITY "polarity"
- #define PWM_RUN "run"

8.16.1 Macro Definition Documentation

8.16.1.1 #define PWM_DUTY "duty"

8.16.1.2 #define PWM_PATH "/sys/devices/ocp.3/"

8.16.1.3 #define PWM_PERIOD "period"

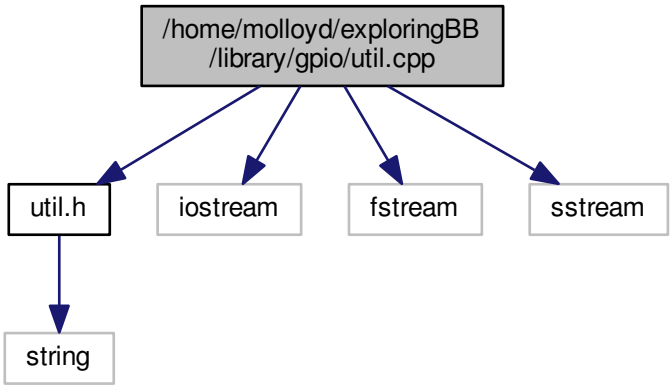
8.16.1.4 #define PWM_POLARITY "polarity"

8.16.1.5 #define PWM_RUN "run"

8.17 /home/molloyd/exploringBB/library/gpio/util.cpp File Reference

```
#include "util.h"
#include <iostream>
#include <fstream>
#include <sstream>
```

Include dependency graph for util.cpp:



Namespaces

- exploringBB

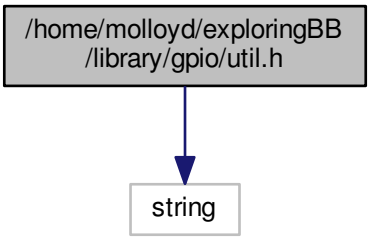
Functions

- int exploringBB::write (string path, string filename, string value)
- string exploringBB::read (string path, string filename)
- int exploringBB::write (string path, string filename, int value)

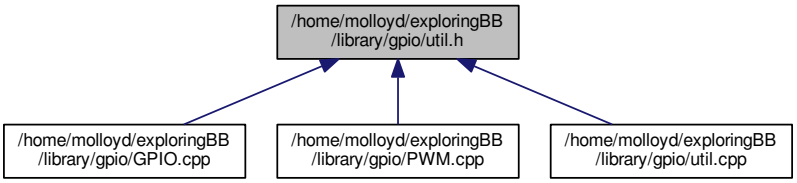
8.18 /home/molloyd/exploringBB/library/gpio/util.h File Reference

```
#include <string>
```

Include dependency graph for util.h:



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



Namespaces

- exploringBB

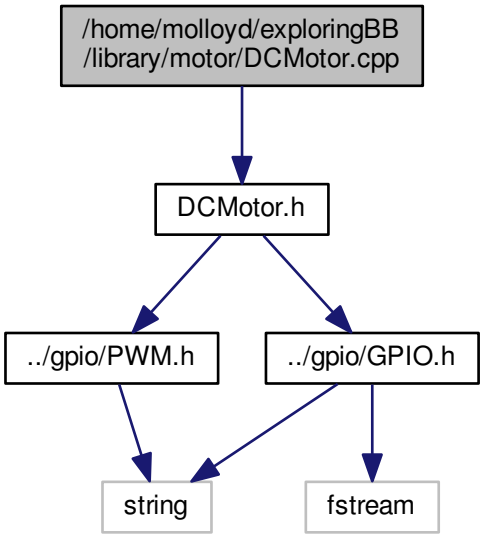
Functions

- int exploringBB::write (string path, string filename, string value)
- int exploringBB::write (string path, string filename, int value)
- string exploringBB::read (string path, string filename)

8.19 /home/molloyd/exploringBB/library/motor/DCMotor.cpp File Reference

```
#include "DCMotor.h"
```

Include dependency graph for DCMotor.cpp:



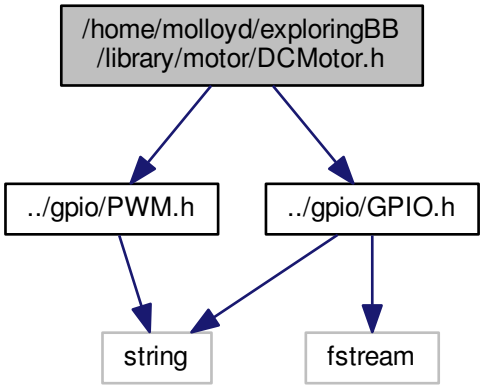
Namespaces

- exploringBB

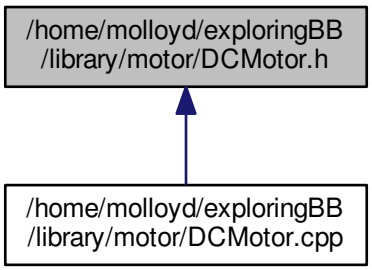
8.20 /home/molloyd/exploringBB/library/motor/DCMotor.h File Reference

```
#include "../gpio/GPIO.h"
#include "../gpio/PWM.h"
```

Include dependency graph for DCMotor.h:



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



Data Structures

- class exploringBB::DCMotor

  A generic DC motor class that controls a motor driver board using a PWM signal,and a GPIO state to control the motor direction.

Namespaces

- exploringBB

Macros

- #define DEFAULT_DCMOTOR_PWM_PERIOD 4000
- #define DEFAULT_DCMOTOR_SPEED 50.0f

8.20.1 Macro Definition Documentation

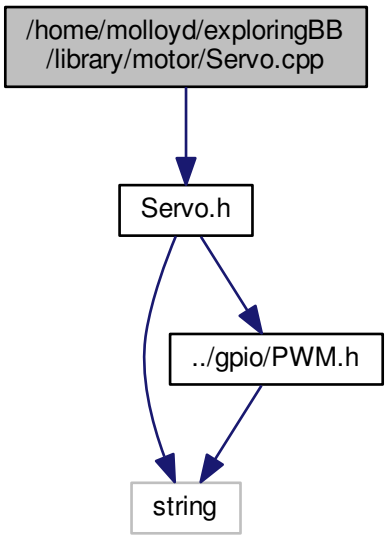
8.20.1.1 #define DEFAULT_DCMOTOR_PWM_PERIOD 4000

8.20.1.2 #define DEFAULT_DCMOTOR_SPEED 50.0f

8.21 /home/molloyd/exploringBB/library/motor/Servo.cpp File Reference

```
#include "Servo.h"
```

Include dependency graph for Servo.cpp:



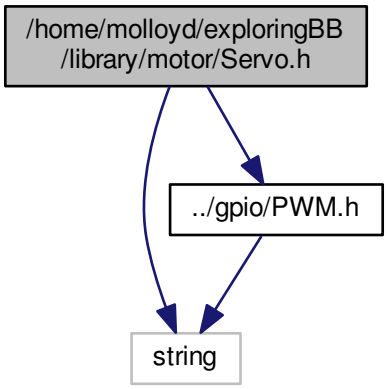
Namespaces

- exploringBB

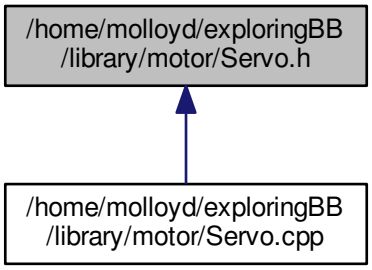
8.22 /home/molloyd/exploringBB/library/motor/Servo.h File Reference

```
#include <string>
#include "../gpio/PWM.h"
```

Include dependency graph for Servo.h:



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



Data Structures

- class exploringBB::Servo

  An extremely basic Servo class stub – does nothing more than the PWM class but is here for future use.

Namespaces

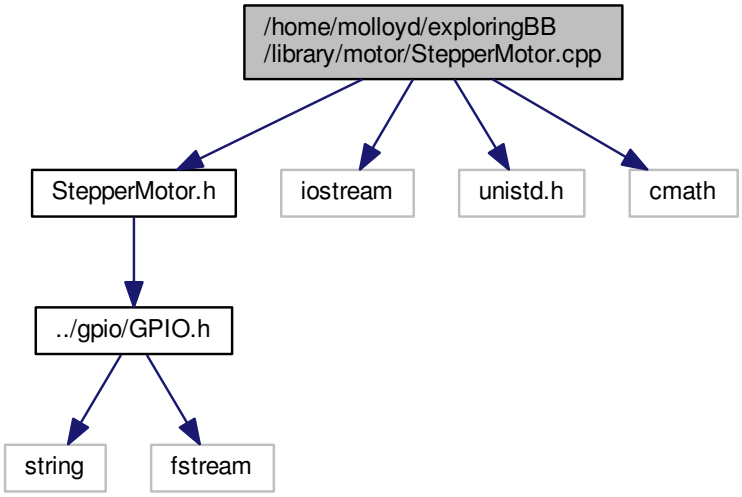
- exploringBB

8.23 /home/molloyd/exploringBB/library/motor/StepperMotor.cpp File Reference

```
#include "StepperMotor.h"
```

```
#include <iostream>
#include <unistd.h>
#include <cmath>
```

Include dependency graph for StepperMotor.cpp:



Namespaces

- exploringBB

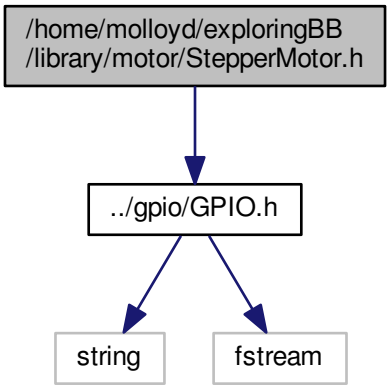
Functions

- void * exploringBB::threadedStep (void *value)

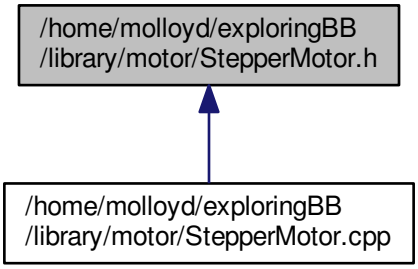
8.24 /home/molloyd/exploringBB/library/motor/StepperMotor.h File Reference

```
#include "../gpio/GPIO.h"
```

Include dependency graph for StepperMotor.h:



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



Data Structures

- class exploringBB::StepperMotor

  A class to control a stepper motor using a motor driver board, such as the Easy Driver board, or compatible. The class uses five GPIOs to control each motor.

Namespaces

- exploringBB

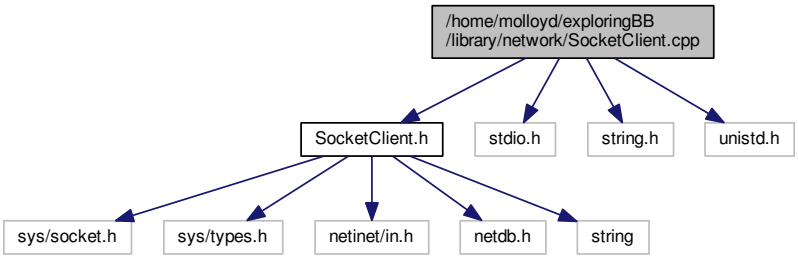
Functions

- void * exploringBB::threadedStep (void *value)

8.25 /home/molloyd/exploringBB/library/network/SocketClient.cpp File Reference

```
#include "SocketClient.h"
#include <stdio.h>
#include <string.h>
#include <unistd.h>
```

Include dependency graph for SocketClient.cpp:



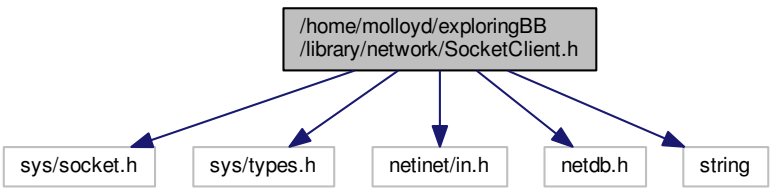
Namespaces

- exploringBB

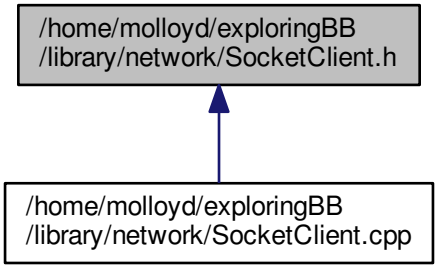
8.26 /home/molloyd/exploringBB/library/network/SocketClient.h File Reference

```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <string>
```

Include dependency graph for SocketClient.h:



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



Data Structures

- class exploringBB::SocketClient

  A class that encapsulates a socket client to be used for network communication.

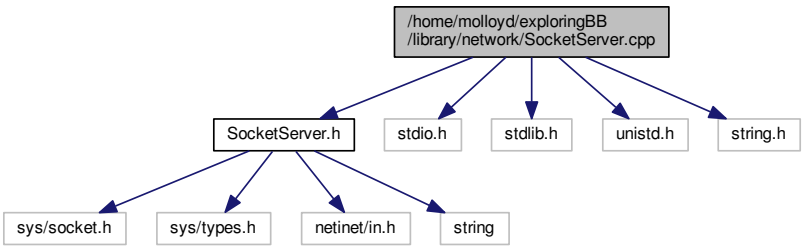
Namespaces

- exploringBB

8.27 /home/molloyd/exploringBB/library/network/SocketServer.cpp File Reference

```
#include "SocketServer.h"
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
```

Include dependency graph for SocketServer.cpp:



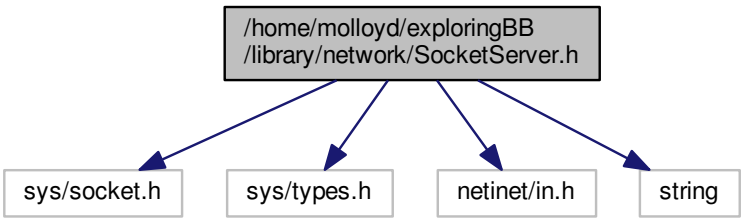
Namespaces

- exploringBB

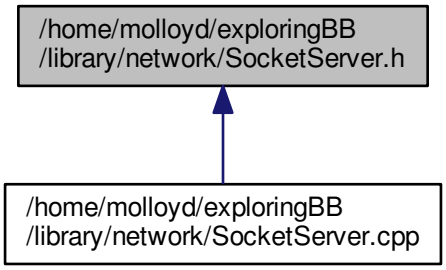
8.28 /home/molloyd/exploringBB/library/network/SocketServer.h File Reference

```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <string>
```

Include dependency graph for SocketServer.h:



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



Data Structures

- class exploringBB::SocketServer

  A class that encapsulates a server socket for network communication.

Namespaces

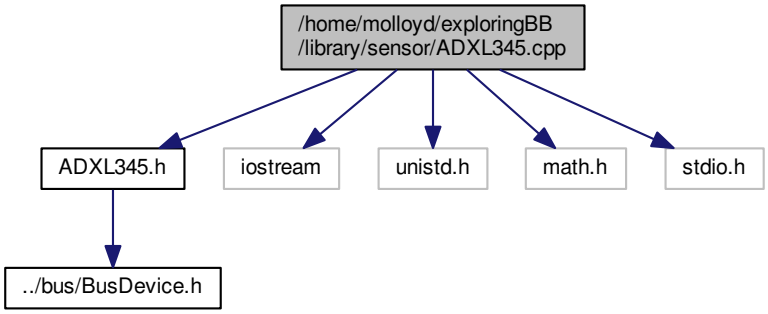
- exploringBB

8.29 /home/molloyd/exploringBB/library/sensor/ADXL345.cpp File Reference

```
#include "ADXL345.h"
```

```
#include <iostream>
#include <unistd.h>
#include <math.h>
#include <stdio.h>
```

Include dependency graph for ADXL345.cpp:



Namespaces

- exploringBB

Macros

- #define DEVID 0x00
- #define THRESH_TAP 0x1D
- #define OFSX 0x1E
- #define OFSY 0x1F
- #define OFSZ 0x20
- #define DUR 0x21
- #define LATENT 0x22
- #define WINDOW 0x23
- #define THRESH_ACT 0x24
- #define THRESH_INACT 0x25
- #define TIME_INACT 0x26
- #define ACT_INACT_CTL 0x27
- #define THRESH_FF 0x28
- #define TIME_FF 0x29
- #define TAP_AXES 0x2A
- #define ACT_TAP_STATUS 0x2B
- #define BW_RATE 0x2C
- #define POWER_CTL 0x2D
- #define INT_ENABLE 0x2E
- #define INT_MAP 0x2F
- #define INT_SOURCE 0x30
- #define DATA_FORMAT 0x31
- #define DATAX0 0x32
- #define DATAX1 0x33
- #define DATAY0 0x34
- #define DATAY1 0x35

- #define DATAZ0 0x36
- #define DATAZ1 0x37
- #define FIFO_CTL 0x38
- #define FIFO_STATUS 0x39

8.29.1 Macro Definition Documentation

8.29.1.1 #define ACT_INACT_CTL 0x27

8.29.1.2 #define ACT_TAP_STATUS 0x2B

8.29.1.3 #define BW_RATE 0x2C

8.29.1.4 #define DATA_FORMAT 0x31

8.29.1.5 #define DATAX0 0x32

8.29.1.6 #define DATAX1 0x33

8.29.1.7 #define DATAY0 0x34

8.29.1.8 #define DATAY1 0x35

8.29.1.9 #define DATAZ0 0x36

8.29.1.10 #define DATAZ1 0x37

8.29.1.11 #define DEVID 0x00

8.29.1.12 #define DUR 0x21

8.29.1.13 #define FIFO_CTL 0x38

8.29.1.14 #define FIFO_STATUS 0x39

8.29.1.15 #define INT_ENABLE 0x2E

8.29.1.16 #define INT_MAP 0x2F

8.29.1.17 #define INT_SOURCE 0x30

8.29.1.18 #define LATENT 0x22

8.29.1.19 #define OFSX 0x1E

8.29.1.20 #define OFSY 0x1F

8.29.1.21 #define OFSZ 0x20

8.29.1.22 #define POWER_CTL 0x2D

8.29.1.23 #define TAP_AXES 0x2A

8.29.1.24 #define THRESH_ACT 0x24

8.29.1.25 #define THRESH_FF 0x28

8.29.1.26 #define THRESH_INACT 0x25

8.29.1.27 #define THRESH_TAP 0x1D

8.29.1.28 #define TIME_FF 0x29

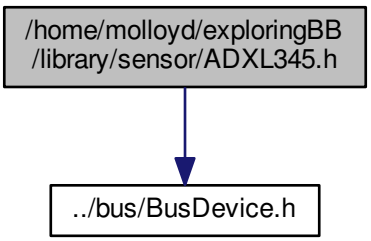
8.29.1.29 #define TIME_INACT 0x26

8.29.1.30 #define WINDOW 0x23

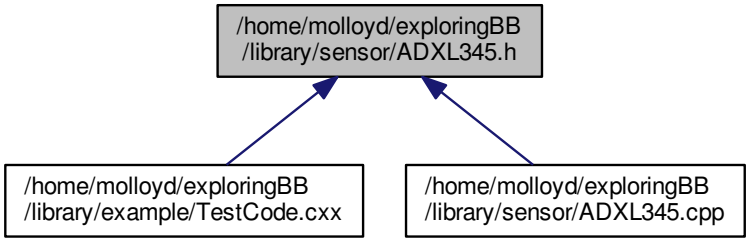
8.30 /home/molloyd/exploringBB/library/sensor/ADXL345.h File Reference

```
#include "../bus/BusDevice.h"
```

Include dependency graph for ADXL345.h:



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



Data Structures

- class exploringBB::ADXL345

  Specific class for the ADXL345 Accelerometer.

Namespaces

- exploringBB

Macros

- #define BUFFER_SIZE 0x40

  The ADXL345 has 0x40 registers (0x01 to 0x1C are reserved and should not be accessed)

8.30.1 Macro Definition Documentation

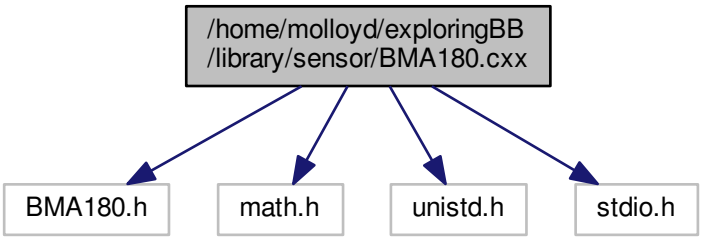
8.30.1.1 #define BUFFER_SIZE 0x40

The ADXL345 has 0x40 registers (0x01 to 0x1C are reserved and should not be accessed)

8.31 /home/molloyd/exploringBB/library/sensor/BMA180.cxx File Reference

```
#include "BMA180.h"
#include <math.h>
#include <unistd.h>
#include <stdio.h>
```

Include dependency graph for BMA180.cxx:



Namespaces

- exploringBB

Macros

- #define ACC_X_LSB 0x02
- #define ACC_X_MSB 0x03
- #define ACC_Y_LSB 0x04
- #define ACC_Y_MSB 0x05
- #define ACC_Z_LSB 0x06
- #define ACC_Z_MSB 0x07
- #define BMA_TEMP 0x08
- #define BMA_RANGE 0x35

- #define BMA_BANDWIDTH 0x20
- #define MODE_CONFIG 0x30

8.31.1 Macro Definition Documentation

8.31.1.1 #define ACC_X_LSB 0x02

8.31.1.2 #define ACC_X_MSB 0x03

8.31.1.3 #define ACC_Y_LSB 0x04

8.31.1.4 #define ACC_Y_MSB 0x05

8.31.1.5 #define ACC_Z_LSB 0x06

8.31.1.6 #define ACC_Z_MSB 0x07

8.31.1.7 #define BMA_BANDWIDTH 0x20

8.31.1.8 #define BMA_RANGE 0x35

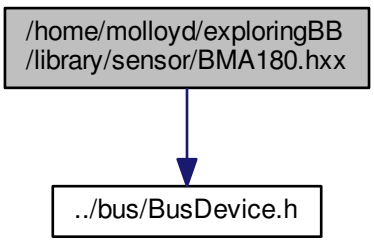
8.31.1.9 #define BMA_TEMP 0x08

8.31.1.10 #define MODE_CONFIG 0x30

8.32 /home/molloyd/exploringBB/library/sensor/BMA180.hxx File Reference

```
#include "../bus/BusDevice.h"
```

Include dependency graph for BMA180.hxx:



Data Structures

- class exploringBB::BMA180

  A class to control a BMA180 accelerometer (untested)

Namespaces

- exploringBB

Macros

- #define BUFFER_SIZE 0x80

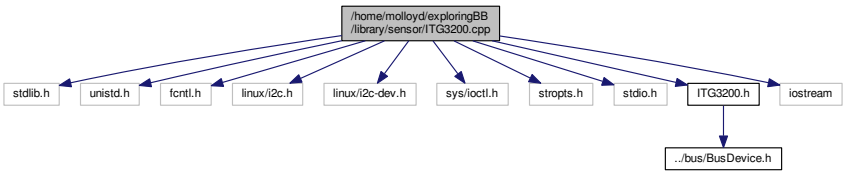
8.32.1 Macro Definition Documentation

8.32.1.1 #define BUFFER_SIZE 0x80

8.33 /home/molloyd/exploringBB/library/sensor/ITG3200.cpp File Reference

```
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <linux/i2c.h>
#include <linux/i2c-dev.h>
#include <sys/ioctl.h>
#include <stropts.h>
#include <stdio.h>
#include "ITG3200.h"
#include <iostream>
```

Include dependency graph for ITG3200.cpp:



Namespaces

- exploringBB

Macros

- #define WHOAMI 0x00
- #define GYRO_X_MSB 0x1D
- #define GYRO_X_LSB 0x1E
- #define GYRO_Y_MSB 0x1F
- #define GYRO_Y_LSB 0x20
- #define GYRO_Z_MSB 0x21
- #define GYRO_Z_LSB 0x22
- #define TEMP_MSB 0x1B
- #define TEMP_LSB 0x1C
- #define INT_CFG 0x17
- #define INT_STATUS 0x1A
- #define PWR_MGM 0x3E
- #define SMPLRT_DIV 0x15
- #define DLPF_FS 0x16
- #define MAX_BUS 64

8.33.1 Macro Definition Documentation

8.33.1.1 #define DLPF_FS 0x16

8.33.1.2 #define GYRO_X_LSB 0x1E

8.33.1.3 #define GYRO_X_MSB 0x1D

8.33.1.4 #define GYRO_Y_LSB 0x20

8.33.1.5 #define GYRO_Y_MSB 0x1F

8.33.1.6 #define GYRO_Z_LSB 0x22

8.33.1.7 #define GYRO_Z_MSB 0x21

8.33.1.8 #define INT_CFG 0x17

8.33.1.9 #define INT_STATUS 0x1A

8.33.1.10 #define MAX_BUS 64

8.33.1.11 #define PWR_MGM 0x3E

8.33.1.12 #define SMPLRT_DIV 0x15

8.33.1.13 #define TEMP_LSB 0x1C

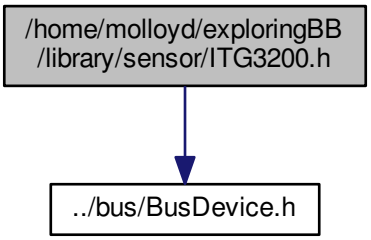
8.33.1.14 #define TEMP_MSB 0x1B

8.33.1.15 #define WHOAMI 0x00

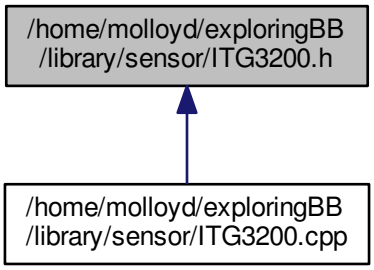
8.34 /home/molloyd/exploringBB/library/sensor/ITG3200.h File Reference

```
#include "../bus/BusDevice.h"
```

Include dependency graph for ITG3200.h:



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



Data Structures

- class exploringBB::ITG3200

  A class to interface with the ITG3200 gyroscope (untested)

Namespaces

- exploringBB

Macros

- #define BUFFER_SIZE 0x3E
- #define SENSITIVITY 14.375

8.34.1 Macro Definition Documentation

8.34.1.1 #define BUFFER_SIZE 0x3E

8.34.1.2 #define SENSITIVITY 14.375

Index