CageControl

Control waveplates inside tomography cages

Generated by Doxygen 1.9.1

Chapter 1

Bug List

File debug.h

Printing to console does not work on Windows. Workaround: Redirect stderr to stdout and redirect stdout to a file.

File defines.h

There are no known bugs.

Namespace helper

There are no known bugs.

Class Motor

There are no known bugs.

Class PCBMotor

There are no known bugs.

Class UDPlistener

There are no known bugs

File version.h

There are no known bugs.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 helper Namespace Reference

contains small functions to display messages

Functions

- void message (QString msg)

  message displays a message box
- void error (QString msg)

  error displays an error-messagebox and writes a debug_error message to stdout
- void warning (QString msg)

  warning displays warning-messagebox and writes a debug_warning message to stdout
- void info (QString msg)

  info displays an info-messagebox and writes a debug_info message to stdout
- std::string **str_tolower** (std::string s)

6.1.1 Detailed Description

contains small functions to display messages

Bug There are no known bugs.

6.1.2 Function Documentation

6.1.2.1 error()

```
void helper::error (
            QString msg )
```

error displays an error-messagebox and writes a debug_error message to stdout

Parameters

msg	the message to be displayed

6.1.2.2 info()

```
void helper::info (
            QString msg )
```

info displays an info-messagebox and writes a debug_info message to stdout

Parameters

msg	the message to be displayed

6.1.2.3 message()

```
void helper::message (
            QString msg )
```

message displays a message box

Parameters

msg	the message to be displayed

6.1.2.4 warning()

```
void helper::warning (
            QString msg )
```

warning displays warning-messagebox and writes a debug_warning message to stdout

Parameters

msg	the message to be displayed

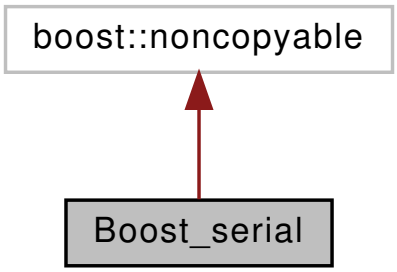
Chapter 7

Class Documentation

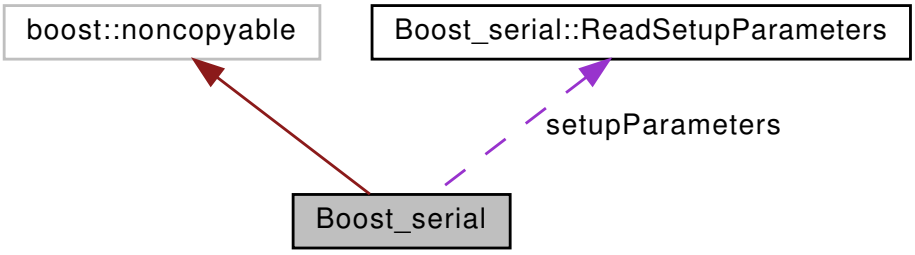
7.1 Boost_serial Class Reference

`#include <boost_serial.h>`

Inheritance diagram for Boost_serial:



Collaboration diagram for Boost_serial:



Classes

- class ReadSetupParameters

Public Member Functions

- Boost_serial (const std::string &devname, unsigned int baud_rate, boost::asio::serial_port_base::parity opt_parity=boost::asio::serial_port_base::parity(boost::asio::serial_port_base::parity::none), boost::asio::serial_port_base::character_size opt_csize=boost::asio::serial_port_base::character_size(8), boost::asio::serial_port_base::flow_control opt_flow=boost::asio::serial_port_base::flow_control(boost::asio::serial_port_base::flow_control::none), boost::asio::serial_port_base::stop_bits opt_stop=boost::asio::serial_port_base::stop_bits(boost::asio::serial_port_base::stop_bits::one))
- void open (const std::string &devname, unsigned int baud_rate, boost::asio::serial_port_base::parity opt_parity=boost::asio::serial_port_base::parity(boost::asio::serial_port_base::parity::none), boost::asio::serial_port_base::character_size opt_csize=boost::asio::serial_port_base::character_size(8), boost::asio::serial_port_base::flow_control opt_flow=boost::asio::serial_port_base::flow_control(boost::asio::serial_port_base::flow_control::none), boost::asio::serial_port_base::stop_bits opt_stop=boost::asio::serial_port_base::stop_bits(boost::asio::serial_port_base::stop_bits::one))
- bool isOpen () const
- void close ()
- void setTimeout (const boost::posix_time::time_duration &t)
- void write (const char *data, size_t size)
- void write (const std::vector< char > &data)
- void writeString (const std::string &s)
- void read (char *data, size_t size)
- std::vector< char > read (size_t size)
- std::string readString (size_t size)
- std::string readStringUntil (const std::string &delim="\n")

Private Types

- enum ReadResult { **resultInProgress** , **resultSuccess** , **resultError** , **resultTimeoutExpired** }

Private Member Functions

- void performReadSetup (const ReadSetupParameters &param)
- void timeoutExpired (const boost::system::error_code &error)
- void readCompleted (const boost::system::error_code &error, const size_t bytesTransferred)

Private Attributes

- boost::asio::io_service io

  Io service object.
- boost::asio::serial_port port

  Serial port object.
- boost::asio::deadline_timer timer

  Timer for timeout.
- boost::posix_time::time_duration timeout

  Read/write timeout.
- boost::asio::streambuf readData

  Holds eventual read but not consumed.
- enum ReadResult result

  Used by read with timeout.
- size_t bytesTransferred

  Used by async read callback.
- ReadSetupParameters setupParameters

  Global because used in the OSX fix.

7.1.1 Detailed Description

Serial port class, with timeout on read operations.

7.1.2 Member Enumeration Documentation

7.1.2.1 ReadResult

```
enum Boost_serial::ReadResult  [private]
```

Possible outcome of a read. Set by callbacks, read from main code

7.1.3 Constructor & Destructor Documentation

7.1.3.1 Boost_serial()

```
Boost_serial::Boost_serial (
            const std::string & devname,
            unsigned int baud_rate,
            boost::asio::serial_port_base::parity opt_parity = boost::asio::serial_port_base↩
::parity(boost::asio::serial_port_base::parity::none),
            boost::asio::serial_port_base::character_size opt_csize = boost::asio::serial_↩
port_base::character_size(8),
            boost::asio::serial_port_base::flow_control opt_flow = boost::asio::serial_port_↩
base::flow_control(boost::asio::serial_port_base::flow_control::none),
            boost::asio::serial_port_base::stop_bits opt_stop = boost::asio::serial_port_↩
base::stop_bits(boost::asio::serial_port_base::stop_bits::one) )
```

Opens a serial device. By default timeout is disabled.

Parameters

devname	serial device name, example "/dev/ttyS0" or "COM1"
baud_rate	serial baud rate
opt_parity	serial parity, default none
opt_csize	serial character size, default 8bit
opt_flow	serial flow control, default none
opt_stop	serial stop bits, default 1

Exceptions

boost::system::system_error	if cannot open the serial device

7.1.4 Member Function Documentation

7.1.4.1 close()

```
void Boost_serial::close ( )
```

Close the serial device

Exceptions

boost::system::system_error	if any error

7.1.4.2 isOpen()

```
bool Boost_serial::isOpen ( ) const
```

Returns

true if serial device is open

7.1.4.3 open()

```
void Boost_serial::open (
            const std::string & devname,
            unsigned int baud_rate,
            boost::asio::serial_port_base::parity opt_parity = boost::asio::serial_port_base↩
::parity(                boost::asio::serial_port_base::parity::none),
            boost::asio::serial_port_base::character_size opt_csize = boost::asio::serial_↩
port_base::character_size(8),
            boost::asio::serial_port_base::flow_control opt_flow = boost::asio::serial_port_↩
base::flow_control(                boost::asio::serial_port_base::flow_control::none),
            boost::asio::serial_port_base::stop_bits opt_stop = boost::asio::serial_port_↩
base::stop_bits(                boost::asio::serial_port_base::stop_bits::one) )
```

Opens a serial device.

Parameters

devname	serial device name, example "/dev/ttyS0" or "COM1"
baud_rate	serial baud rate
opt_parity	serial parity, default none
opt_csize	serial character size, default 8bit
opt_flow	serial flow control, default none
opt_stop	serial stop bits, default 1

Exceptions

boost::system::system_error	if cannot open the serial device

7.1.4.4 performReadSetup()

```
void Boost_serial::performReadSetup (
            const ReadSetupParameters & param )  [private]
```

This member function sets up a read operation, both reading a specified number of characters and reading until a delimiter string.

7.1.4.5 read() [1/2]

```
void Boost_serial::read (
            char * data,
            size_t size )
```

Read some data, blocking

Parameters

data	array of char to be read through the serial device
size	array size

Returns

numbr of character actually read 0<=return<=size

Exceptions

boost::system::system_error	if any error
timeout_exception	in case of timeout

7.1.4.6 read() [2/2]

```
std::vector< char > Boost_serial::read (
            size_t size )
```

Read some data, blocking

Parameters

size	how much data to read

Returns

the receive buffer. It iempty if no data is available

Exceptions

boost::system::system_error	if any error
timeout_exception	in case of timeout

7.1.4.7 readCompleted()

```
void Boost_serial::readCompleted (
             const boost::system::error_code & error,
             const size_t bytesTransferred )  [private]
```

Callback called either if a read complete or read error occurs If called because of read complete, sets result to resultSuccess If called because read error, sets result to resultError

7.1.4.8 readString()

```
std::string Boost_serial::readString (
             size_t size )
```

Read a string, blocking Can only be used if the user is sure that the serial device will not send binary data. For binary data read, use read() The returned string is empty if no data has arrived

Parameters

size	hw much data to read

Returns

a string with the received data.

Exceptions

boost::system::system_error	if any error
timeout_exception	in case of timeout

7.1.4.9 readStringUntil()

```
std::string Boost_serial::readStringUntil (
            const std::string & delim = "\n" )
```

Read a line, blocking Can only be used if the user is sure that the serial device will not send binary data. For binary data read, use read() The returned string is empty if the line delimiter has not yet arrived.

Parameters

delimiter	line delimiter, default="\n"

Returns

a string with the received data. The delimiter is removed from the string.

Exceptions

boost::system::system_error	if any error
timeout_exception	in case of timeout

7.1.4.10 setTimeout()

```
void Boost_serial::setTimeout (
            const boost::posix_time::time_duration & t )
```

Set the timeout on read/write operations. To disable the timeout, call setTimeout(boost::posix_time::seconds(0));

7.1.4.11 timeoutExpired()

```
void Boost_serial::timeoutExpired (
            const boost::system::error_code & error )  [private]
```

Callack called either when the read timeout is expired or canceled. If called because timeout expired, sets result to resultTimeoutExpired

7.1.4.12 write() [1/2]

```
void Boost_serial::write (
            const char * data,
            size_t size )
```

Write data

Parameters

data	array of char to be sent through the serial device
size	array size

Exceptions

boost::system::system_error	if any error

7.1.4.13 write() [2/2]

```
void Boost_serial::write (
            const std::vector< char > & data )
```

Write data

Parameters

data	to be sent through the serial device

Exceptions

boost::system::system_error	if any error

7.1.4.14 writeString()

```
void Boost_serial::writeString (
            const std::string & s )
```

Write a string. Can be used to send ASCII data to the serial device. To send binary data, use write()

Parameters

s	string to send

Exceptions

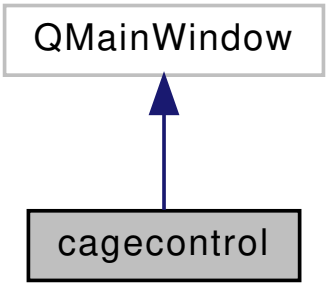
boost::system::system_error	if any error

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

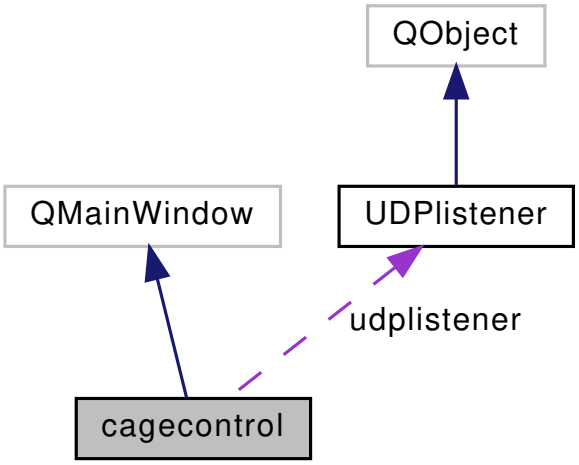
- /home/peter/Development/cagecontrol/src/boost_serial.h
- /home/peter/Development/cagecontrol/src/boost_serial.cpp

7.2 cagecontrol Class Reference

Inheritance diagram for cagecontrol:



Collaboration diagram for cagecontrol:



Public Slots

- void slot_changeWPangles (QVector< double > angles)

  slot_changeWPangles sets offsetangles for all waveplates
- void slot_changeoffsetusage (bool uo_in)

  slot_changeoffsetusage changes the usage of the waveplate offset
- void slot_movemotors (QString color, double HWPang, double QWPang, double QWP2ang=0)

  slot_movemotors
- void useinvertedbases (QString id, bool inv)

  useinvertedbases
- void moveHV (QString id)

  moveHV moves cage with colorcode id to H/V basis
- void movePM (QString id)

  movePM moves cage with colorcode id to P/M basis
- void moveRL (QString id)

  moveRL moves cage with colorcode id to R/L basis
- void moveANG (QString id)

  moveANG moves cage with colorcode id to angles set in GUI

Public Member Functions

- **cagecontrol** (QWidget *parent=nullptr)

Private Slots

- void updatesettings (double d)

  updatesettings fills variables with data from GUI
- void updatesettingsint (int i)

  updatesettingsint wrapper, just calls
- void updateUI ()

  updateUI updates UI with supposedly new numbers (loaded from conf file, e.g.)

Private Member Functions

- void setupUI (QGridLayout *layout)

  Puts together the GUI.
- void openmotors ()

  Opens serial connections to the PCB motor controlllers.
- void changebases ()

  changebases changes bases periodically
- void setbasesfile ()

  setbasesfile reads filename from dialog
- int readbasesfile ()

  readbasesfile reads file containing bases
- void updatestatus (QString msg)

  updatestatus writes message to statusbar and to a logfile
- void LoadConfig ()

  LoadConfig loads config from a file.
- void SaveConfig ()

  SaveConfig stores config to a file.
- void motorGB (QGroupBox *gb, QString id)

  motorGB fills an empty QGroupBox with motor controls
- void initconnections ()

  initconnections connects Qt Signals to slots
- void invertall ()

  invertall convenient way to tick all 'invert' boxes. does not rotate WPs.
- void movemotor (QString motor, double HWPang, double QWPang, double QWP2ang=0)

  movemotor moves both motors of a cage to certain angles

Private Attributes

- QString basesfname

  path and filename of file containing bases
- QTimer basestimer

  Runs out every.
- QDir basesdir

  Directory of basesfile.
- QFile basesf

  Bases file.
- int currentbasisidx

  index of current basis
- int basestime

  When reading bases from file: Number of seconds after which a basischange occurs.
- int udpport

  Hold the UDP port to listen to for commandds.
- bool pauseupdating

  Keep updateUI and updatesettings from interfering with each other.
- bool useoffset

  If true, the angles in the settings-tab will be used as '0'.
- QSettings * settings

  A QSettings object, used to store settings in a config file.
- UDPlistener * udplistener

  Listens to a UDP port, aquiires & checks commands send to it.
- QTabWidget * tabs

  GUI tab widget.
- QWidget * settingstab

  GUI tab containing settings.
- QWidget * motorstab

  GUI tab containing motor controls.
- QStatusBar * status

  Status bar.
- QVector< std::string > comports

  Vector containing available serial ports names ports.
- QVector< QStringList > bases

  vector holding all bases for automatic basis change
- QVector< bool > invert

  True: invert predefined bases (H/V -> V/H, P/M->M/P, L/R->R/L)
- QVector< bool > isthreewps

  True: cage has three waveplates. False: cage has two waveplates.
- QVector< motorwrapper * > motors

  List of serial connections to the cages.
- QVector< int > motorType

  0: PCBmotor, 1 Thorlabs Elliptec
- QVector< QString > motorName

  List of colorcodes of the cages.
- QVector< QDoubleSpinBox > HWP0sp

  List of QSpinBoxes to set the '0' of the HWPs.
- QVector< QDoubleSpinBox > QWP0sp

  List of QSpinBoxes to set the '0' of the QWPs.
- QVector< uint8_t > HWPmnum

Motornumber of controller the HWP is connected to.

- QVector< uint8_t > QWPmnum

  Motornumber of controller the first QWP is connected to.
- QVector< uint8_t > QWP2mnum

  Motornumber of controller the second QWP is connected to.
- QVector< double > HWP0

  '0' of HWPs
- QVector< double > QWP0

  '0' of first QWPs
- QVector< double > QWP20

  '0' of second QWPs
- QVector< double > HWPcust

  custum set angle to rotate HWP to
- QVector< double > QWPcust

  custom set angle to rotate first QWP to
- QVector< double > QWP2cust

  custom set angle to rotate second QWP to
- QVector< QGroupBox * > uiMotorGroupBoxes

  List of Groupboxes containing cage controls.

7.2.1 Member Function Documentation

7.2.1.1 initconnections()

```
void cagecontrol::initconnections ( )  [private]
```

initconnections connects Qt Signals to slots

Defines what happens when a button is clicked, a number is changet, et cetera

7.2.1.2 LoadConfig()

```
void cagecontrol::LoadConfig ( )  [private]
```

LoadConfig loads config from a file.

The dialog is set up with values already stored in the QSettings object. If a specific quantity does not exist there, it is set to a standard value.

7.2.1.3 motorGB()

```
void cagecontrol::motorGB (
            QGroupBox * gb,
            QString id )  [private]
```

motorGB fills an empty QGroupBox with motor controls

Parameters

gb	empty QGroupBox
id	colorcode of the cage

7.2.1.4 moveANG

```
void cagecontrol::moveANG (
            QString id )  [slot]
```

moveANG moves cage with colorcode id to angles set in GUI

Parameters

id	colorcode of cage, or "all"

7.2.1.5 moveHV

```
void cagecontrol::moveHV (
            QString id )  [slot]
```

moveHV moves cage with colorcode id to H/V basis

Parameters

id	colorcode of cage, or "all"

7.2.1.6 movemotor()

```
void cagecontrol::movemotor (
            QString motor,
            double HWPang,
            double QWPang,
            double QWP2ang = 0 )  [private]
```

movemotor moves both motors of a cage to certain angles

Parameters

motor	colorcode of the cage
HWPang	angle of the HWP in degrees
QWPang	angle of the 1st QWP in degrees
QWP2ang	angle of the 2nd QWP in degrees. defaults to 0 because of 2/3 WP cages

7.2.1.7 movePM

```
void cagecontrol::movePM (
            QString id )  [slot]
```

movePM moves cage with colorcode id to P/M basis

Parameters

id	colorcode of cage, or "all"

7.2.1.8 moveRL

```
void cagecontrol::moveRL (
            QString id )  [slot]
```

moveRL moves cage with colorcode id to R/L basis

Parameters

id	colorcode of cage, or "all"

7.2.1.9 readbasesfile()

```
int cagecontrol::readbasesfile ( )  [private]
```

readbasesfile reads file containing bases

Returns

0 on success

7.2.1.10 SaveConfig()

```
void cagecontrol::SaveConfig ( )  [private]
```

SaveConfig stores config to a file.

The QSettings object is updated with the values received from the dialog and saved immediately.

7.2.1.11 slot_changeoffsetusage

```
void cagecontrol::slot_changeoffsetusage (
            bool uo_in )  [slot]
```

slot_changeoffsetusage changes the usage of the waveplate offset

Parameters

useoffset	true if waveplate offset is to be used

The term 'offset' refers to the waveplate angles specified in the

See also

settingstab. E.g.: 'H' of HWP specified in he settingstab is 50° and one wants to rotate the waveplate to H+10°. If (useoffset==true), one needs to rotate the motor to 10°. If (useoffset==false), one needs to rotate the motor to 60°.

7.2.1.12 slot_changeWPangles

```
void cagecontrol::slot_changeWPangles (
            QVector< double > angles )  [slot]
```

slot_changeWPangles sets offsetangles for all waveplates

Parameters

angles	vector containing all angles. Ordering: HWP0,HWP1,...,HWPn,QWP0,QWP1,...,QWPn

7.2.1.13 slot_movemotors

```
void cagecontrol::slot_movemotors (
            QString color,
            double HWPang,
            double QWPang,
            double QWP2ang = 0 )  [slot]
```

slot_movemotors

Parameters

color	colorcode of the cage, or 'all'
HWPang	angle of the HWP in degrees
QWPang	angle of the 1st QWP in degrees
QWP2ang	angle of the 2nd QWP in degrees. defaults to 0 because of 2/3 WP cages

7.2.1.14 updatesettings

```
void cagecontrol::updatesettings (
            double d )  [private], [slot]
```

updatesettings fills variables with data from GUI

Parameters

d	unused

7.2.1.15 updatesettingsint

```
void cagecontrol::updatesettingsint (
            int i )  [private], [slot]
```

updatesettingsint wrapper, just calls

See also

updatesettings(double d)

Parameters

i	unused

7.2.1.16 updatestatus()

```
void cagecontrol::updatestatus (
            QString msg )  [private]
```

updatestatus writes message to statusbar and to a logfile

Parameters

msg	Message to write

7.2.1.17 useinvertedbases

```
void cagecontrol::useinvertedbases (
            QString id,
            bool inv )  [slot]
```

useinvertedbases

Parameters

id	colorcode of cage, or 'all'
inv	true if consecutive bases changes shall be inverted (using e.g. V/H instead of H/V)

7.2.2 Member Data Documentation

7.2.2.1 basestimer

```
QTimer cagecontrol::basestimer  [private]
```

Runs out every.

See also

basestime seconds to change bases

7.2.2.2 useoffset

```
bool cagecontrol::useoffset  [private]
```

If true, the angles in the settings-tab will be used as '0'.

See also

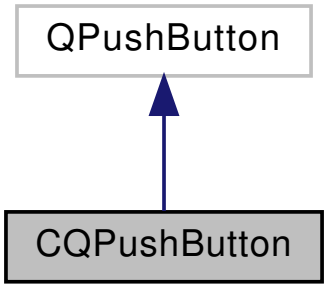
slot_changeoffsetusage

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

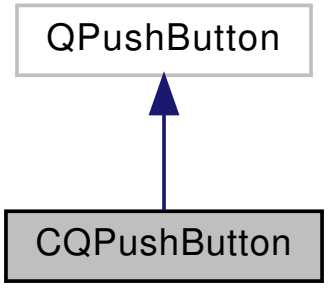
- /home/peter/Development/cagecontrol/src/cagecontrol.h
- /home/peter/Development/cagecontrol/src/cagecontrol.cpp

7.3 CQPushButton Class Reference

Inheritance diagram for CQPushButton:



Collaboration diagram for CQPushButton:



Public Slots

- void setid (QString id)

  Sets QString id which is sent when signal pressed_id is emitted.
- QString getid ()

  Returns QString id which is sent when signal pressed_id is emitted.
- void triggerOutput ()

  Function to capture Qt's standard &QAbstracButton::pressed(void) signal and emits pressed_id(QString) instead.

Signals

- void pressed_id (QString)

  pressed_id signal emitted when button is pressed. Carries QString id.
- void **rightClicked** ()

Public Member Functions

- **CQPushButton** (QString ButtonText, QString id="", QWidget ∗parent=nullptr)

Protected Attributes

- QString id

  QString to send when pressed_id is emitted.

Private Slots

- void **mousePressEvent** (QMouseEvent ∗e)

7.3.1 Member Function Documentation

7.3.1.1 getid

```
QString CQPushButton::getid ( )  [slot]
```

Returns QString id which is sent when signal pressed_id is emitted.

See also

setid

pressed_id

id

7.3.1.2 pressed_id

```
void CQPushButton::pressed_id (
            QString  )  [signal]
```

pressed_id signal emitted when button is pressed. Carries QString id.

See also

setid

getid

id

7.3.1.3 setid

```
void CQPushButton::setid (
            QString id )  [slot]
```

Sets QString id which is sent when signal pressed_id is emitted.

See also

getid
pressed_id
id

7.3.2 Member Data Documentation

7.3.2.1 id

```
QString CQPushButton::id  [protected]
```

QString to send when pressed_id is emitted.

See also

setid
getid
pressed_id

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

- /home/peter/Development/cagecontrol/src/cqpushbutton.h
- /home/peter/Development/cagecontrol/src/cqpushbutton.cpp

7.4 ell_device Struct Reference

Public Attributes

- std::string **address**
- uint16_t **type**
- uint64_t **serial**
- uint16_t **year**
- uint8_t **fw**
- uint8_t **hw**
- uint32_t **travel**
- uint64_t **pulses**

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

- /home/peter/Development/cagecontrol/src/elliptec.h

7.5 ell_response Struct Reference

Public Attributes

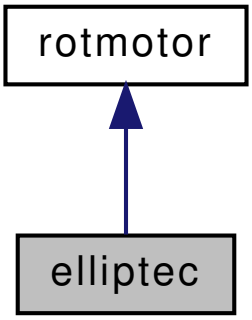
- uint8_t **address** = 0
- std::string **type** = ""
- std::string **data** = ""

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

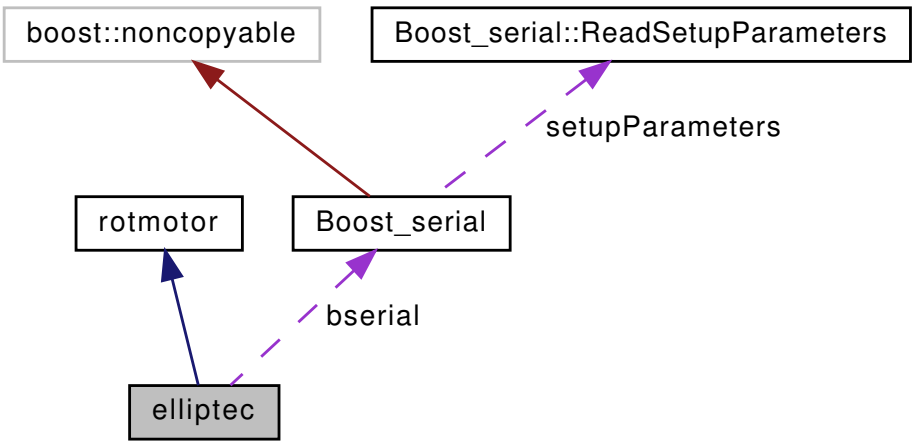
- /home/peter/Development/cagecontrol/src/elliptec.h

7.6 elliptec Class Reference

Inheritance diagram for elliptec:



Collaboration diagram for elliptec:



Public Member Functions

- elliptec (std::string devname="", std::vector< uint8_t > inmids=std::vector< uint8_t >(0), bool dohome=true, bool freqsearch=true)

  Motor the contructor initializes variables and establishes the serial connection.
- void **open** (std::string port, bool dohome, bool freqsearch)
- void close ()

  close closes the serialport connection
- bool isopen ()

  isopen returns the state of the serial connection
- void **home** (std::string addr, std::string dir="0")
- void **move_absolute** (std::string addr, double pos)
- void **move_relative** (std::string addr, double pos)
- void **get_position** (std::string addr)
- void **get_status** (std::string addr)
- void **get_velocity** (std::string addr)
- void **set_velocity** (std::string addr, uint8_t percent)
- void **save_userdata** (std::string addr)
- void **change_address** (std::string addr, std::string newaddr)
- void **command_moveboth** (int hwp_mnum, int qwp_mnum, double hwpang, double qwpang)
- void **command_movethree** (int hwp_mnum, int qwp_mnum, int qwp2_mnum, double hwpang, double qwpang, double qwp2ang)

Private Member Functions

- std::string **query** (const std::string &data)
- void **read** ()
- void **write** (const std::string &data)
- void **getinfo** (std::string addr)
- void **get_motor1_info** (std::string addr)
- void **get_motor2_info** (std::string addr)
- void **get_motor3_info** (std::string addr)
- void **search_motor1_freq** (std::string addr)
- void **search_motor2_freq** (std::string addr)
- void **search_motor3_freq** (std::string addr)
- void **search_freq** (std::string addr)
- void **optimize_motors** (std::string addr)
- void **clean_mechanics** (std::string addr)
- void **stop_clean** (std::string addr)
- ell_response **process_response** ()
- void **handle_devinfo** (ell_device dev)
- void **print_dev_info** (ell_device dev)
- bool **devintype** (std::string type, uint8_t id)
- std::optional< ell_device > **devinfo_at_addr** (std::string addr)
- int64_t **deg2step** (std::string addr, double deg)
- int64_t **mm2step** (std::string addr, double mm)
- double **step2deg** (std::string addr, int64_t step)
- double **step2mm** (std::string addr, int64_t step)
- std::string **step2hex** (int64_t step)
- int64_t **hex2step** (std::string hex)
- std::string **ll2hex** (int64_t i)
- std::string **int2addr** (uint8_t id)

Private Attributes

- const uint8_t **CW** = 0
- const uint8_t **CCW** = 1
- const double **DEGERR** = 0.1
- const double **MMERR** = 0.05
- Boost_serial * **bserial**
- std::string **response**
- uint16_t **_ser_timeout**
- std::unordered_map< std::string, std::vector< uint8_t > > **devtype**
- std::vector< std::string > mids

  motor ids
- std::vector< ell_device > **devices**

7.6.1 Member Function Documentation

7.6.1.1 isopen()

```
bool elliptec::isopen ( )  [virtual]
```

isopen returns the state of the serial connection

Returns

true if serial connection was established successfully, false otherwise

Implements rotmotor.

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

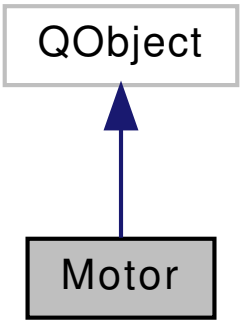
- /home/peter/Development/cagecontrol/src/elliptec.h
- /home/peter/Development/cagecontrol/src/elliptec.cpp

7.7 Motor Class Reference

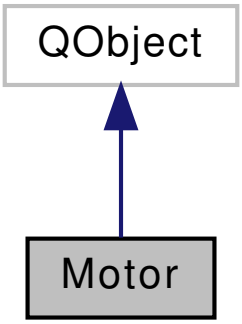
The Motor class operates the PCB-motor.

```
#include <motor.h>
```

Inheritance diagram for Motor:



Collaboration diagram for Motor:



Public Slots

- void open (QString port)

  open establishes a connection over a serial port
- void close ()

  close closes the serialport connection
- void read ()

  read reads from the serial port
- void write (const QByteArray &data)

  write writes to the serialport
- void handleError (QSerialPort::SerialPortError error)

  handleError prints an error message of the serialport connection and closes the connection
- void showStatusMessage (const QString &message)

  showStatusMessage fills the label in the GUI with text
- bool isopen ()

  isopen returns the state of the serial connection
- void command_park ()

  command_park moves the motor to the mechanical stop
- void command_home ()

  command_home sends commands to position at the mechanical stop and afterwards go to the offset starting position, but in an inaccurate way
- void command_info ()

  command_info sends the command to request the PCBMotor information
- void command_help ()

  command_help sends the command to print the PCBMotor help
- void command_frequency_sweep ()

  command_frequency_sweep sends the PCBMotor command for a frequency sweep
- void command_singlestep (QString dirstring)

  command_singlestep moves the motor a single step in a direction specified by dirstring
- void command_step (uint16_t numsteps, QString dirstring)

  command_step moves the motor numstep steps in a direction specified by dirstring
- void command_microstep (uint16_t nummsteps, QString dirstring)

  command_microstep aplies nummsteps micropulses to the motor
- void stop (bool stop)

  stop Tries to stop movenents if possible
- void command_moveboth (double ang1, double ang2)

  command_moveboth moves both motors connected to the controller
- void command_movethree (int idx1, int idx2, int idx3, double ang1, double ang2, double ang3)

  command_movethree moves three motors connected to the controller

Signals

- void motorstatusmessage (const QString &message)

  motorstatusmessage emitted when the status of the serial connection changes, with a string indicating the actual state.
- void ConnectionClosed ()

  emitted when serial connection is closed

Public Member Functions

- Motor ()

  Motor the contructor initializes variables and establishes the serial connection.
- bool sensordata ()

  sensordata returns the current PCBMotor optical encoder wheel sendor state

Public Attributes

- QString publicmotorstatusmessage

  A string containing the current state of the serial connection.
- QSerialPort * serial

  Qt serial connection interface.

Private Member Functions

- void moveboth ()

  command_moveboth moves both motors connected to the controller
- void movethree ()

  command_movethree moves three motors connected to the controller

Private Attributes

- QTimer hometimer

  Used to iterate through the steps of 'go to the starting position' - but in an inaccurate way.
- QTimer boththtimer

  Used to iterate through the steps of moving two motors of one controller.
- QTimer threetimer

  Used to iterate through the steps of moving two motors of one controller.
- int movebothstep

  Controls logic flow when two motors are to be moved consecutively.
- int movethreestep

  Controls logic flow when two motors are to be moved consecutively.
- bool serialconnectionok

  False if opening the serial connection failed.
- uint16_t motor1steps

  number of steps the 1st motor is to be moved
- uint16_t motor2steps

  number of steps the 2nd motor is to be moved
- uint16_t motor3steps

  number of steps the 3rd motor is to be moved
- uint16_t motor1idx

  controller index of motor 1
- uint16_t motor2idx

  controller index of motor 2
- uint16_t motor3idx

  controller index of motor 3

7.7.1 Detailed Description

The Motor class operates the PCB-motor.

Bug There are no known bugs.

The PCBMotor is controllable by sending ASCII commands over a serial connection. This class establishes such a connection and controls the movements of the motor.

7.7.2 Member Function Documentation

7.7.2.1 command_microstep

```
void Motor::command_microstep (
            uint16_t nummsteps,
            QString dirstring )  [slot]
```

command_microstep aplies nummsteps micropulses to the motor

Parameters

nummsteps	number of micropulses to apply
dirstring	string containing the desired direction

dirstring may either be "bw" of "fw" for backward/forward movement.

7.7.2.2 command_moveboth

```
void Motor::command_moveboth (
            double ang1,
            double ang2 )  [slot]
```

command_moveboth moves both motors connected to the controller

Parameters

ang1	angle motor 1 is to be moved to
ang2	angle motor 2 is to be moved to

7.7.2.3 command_movethree

```
void Motor::command_movethree (
            int idx1,
```

```
            int idx2,
            int idx3,
            double ang1,
            double ang2,
            double ang3 )  [slot]
```

command_movethree moves three motors connected to the controller

Parameters

idx1	index of motor 1
idx2	index of motor 2
idx3	index of motor 3
ang1	angle to rotate motor 1 to
ang2	angle to rotate motor 2 to
ang3	angle to rotate motor 3 to

7.7.2.4 command_singlestep

```
void Motor::command_singlestep (
            QString dirstring )  [slot]
```

command_singlestep moves the motor a single step in a direction specified by dirstring

Parameters

dirstring	a string containing the desired movenent direction

Dirstring may either be "bw" of "fw" for backward/forward movement.

7.7.2.5 command_step

```
void Motor::command_step (
            uint16_t numsteps,
            QString dirstring )  [slot]
```

command_step moves the motor numstep steps in a direction specified by dirstring

Parameters

numsteps	number of steps to go
dirstring	direction to go

Dirstring may either be "bw" of "fw" for backward/forward movement.

7.7.2.6 handleError

```
void Motor::handleError (
            QSerialPort::SerialPortError error )  [slot]
```

handleError prints an error message of the serialport connection and closes the connection

Parameters

error	

7.7.2.7 isopen

```
bool Motor::isopen ( )  [slot]
```

isopen returns the state of the serial connection

Returns

true if serial connection was established successfully, false otherwise

7.7.2.8 motorstatusmessage

```
void Motor::motorstatusmessage (
            const QString & message )  [signal]
```

motorstatusmessage emitted when the status of the serial connection changes, with a string indicating the actual state.

Parameters

message	the message

7.7.2.9 sensordata()

```
bool Motor::sensordata ( )
```

sensordata returns the current PCBMotor optical encoder wheel sendor state

Returns

the current PCBMotor optical encoder wheel sendor state

7.7.2.10 showStatusMessage

```
void Motor::showStatusMessage (
            const QString & message )  [slot]
```

showStatusMessage fills the label in the GUI with text

Parameters

message	the text to be shown in the label

7.7.2.11 stop

```
void Motor::stop (
            bool stop )  [slot]
```

stop Tries to stop movenents if possible

Parameters

stop	Input: True if movents shall be stopped if possible

7.7.2.12 write

```
void Motor::write (
            const QByteArray & data )  [slot]
```

write writes to the serialport

Parameters

data	data to be written to the serial port

7.7.3 Member Data Documentation

7.7.3.1 hometimer

```
QTimer Motor::hometimer  [private]
```

Used to iterate through the steps of 'go to the starting position' - but in an inaccurate way.

See also

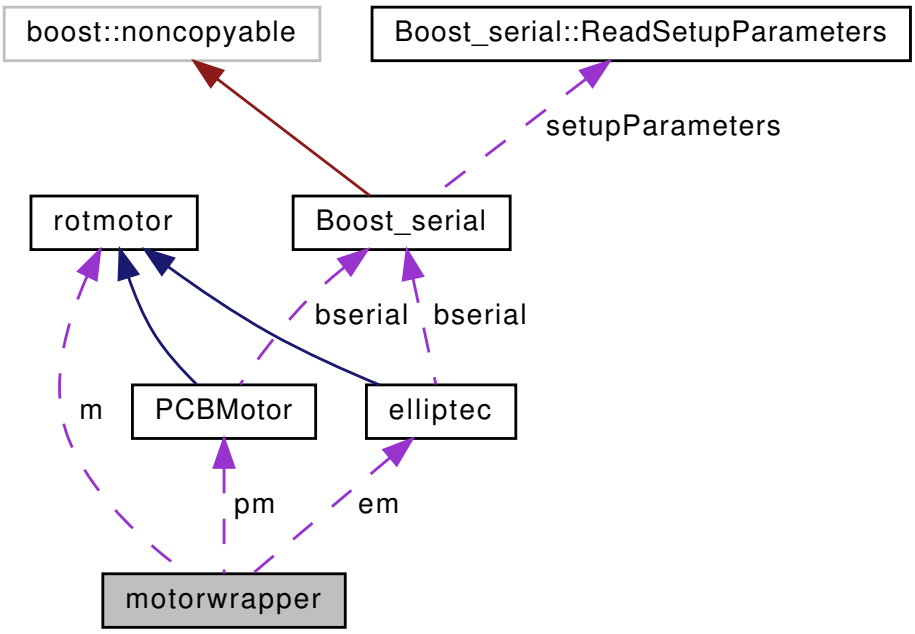
command_home()

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

- /home/peter/Development/cagecontrol/src/motor.h
- /home/peter/Development/cagecontrol/src/motor.cpp

7.8 motorwrapper Class Reference

Collaboration diagram for motorwrapper:



Public Member Functions

- **motorwrapper** (uint8_t intype, std::string devname, std::vector< uint8_t > mids)
- void **open** (std::string port)
- void **close** ()
- void **move_absolute** (int mnum, double ang)
- void **command_moveboth** (int hwp_mnum, int qwp_mnum, double hwpang, double qwpang)
- void **command_movethree** (int hwp_mnum, int qwp_mnum, int qwp2_mnum, double hwp_ang, double qwp_ang, double qwp2_ang)

Private Attributes

- uint8_t **_devtype**
- rotmotor * **m**
- elliptec * **em**
- PCBMotor * **pm**

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

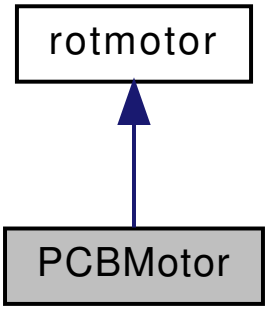
- /home/peter/Development/cagecontrol/src/motorwrapper.h
- /home/peter/Development/cagecontrol/src/motorwrapper.cpp

7.9 PCBMotor Class Reference

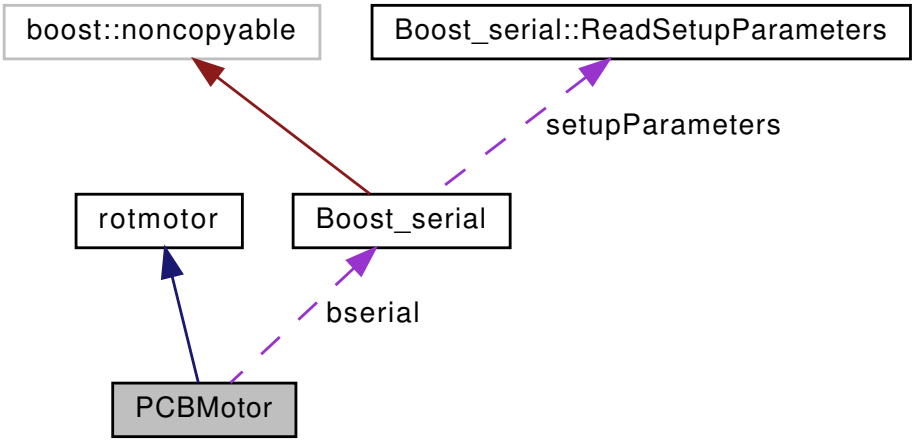
The Motor class operates the PCB-motor.

`#include <pcbmotor.h>`

Inheritance diagram for PCBMotor:



Collaboration diagram for PCBMotor:



Public Member Functions

- PCBMotor (std::string devname="", std::vector< uint8_t > mids=std::vector< uint8_t >(1))

  Motor the contructor initializes variables and establishes the serial connection.
- bool sensordata ()

  sensordata returns the current PCBMotor optical encoder wheel sendor state
- void command_park ()

  command_park moves the motor to the mechanical stop
- void command_home ()

  command_home sends commands to position at the mechanical stop and afterwards go to the offset starting position, but in an inaccurate way
- void command_info ()

  command_info sends the command to request the PCBMotor information
- void command_help ()

  command_help sends the command to print the PCBMotor help

- void command_frequency_sweep ()

  command_frequency_sweep sends the PCBMotor command for a frequency sweep
- void command_singlestep (std::string dirstring)

  command_singlestep moves the motor a single step in a direction specified by dirstring
- void command_step (uint16_t numsteps, std::string dirstring)

  command_step moves the motor numstep steps in a direction specified by dirstring
- void command_microstep (uint16_t nummsteps, std::string dirstring)

  command_microstep aplies nummsteps micropulses to the motor
- void stop (bool stop)

  stop Tries to stop movenents if possible
- void command_moveboth (int hwp_mnum, int qwp_mnum, double hwpang, double qwpang)

  command_moveboth moves both motors connected to the controller
- void command_move (int mnum, double ang)

  command_moveboth moves both motors connected to the controller
- void command_movethree (int hwp_mnum, int qwp_mnum, int qwp2_mnum, double hwp_ang, double qwp↩ _ang, double qwp2_ang)

  command_movethree moves three motors connected to the controller

Private Member Functions

- void move ()

  command_moveboth moves both motors connected to the controller
- void moveboth ()

  command_moveboth moves both motors connected to the controller
- void movethree ()

  command_movethree moves three motors connected to the controller
- void **open** (std::string port)
- void **read** ()
- std::string **query** (const std::string &data)
- void close ()

  close closes the serialport connection
- bool isopen ()

  isopen returns the state of the serial connection
- void write (const std::string &data)

  write writes to the serialport

Private Attributes

- int movebothstep

  Controls logic flow when two motors are to be moved consecutively.
- int movethreestep

  Controls logic flow when two motors are to be moved consecutively.
- bool serialconnectionok

  False if opening the serial connection failed.
- uint16_t motor1steps

  number of steps the 1st motor is to be moved
- uint16_t motor2steps

  number of steps the 2nd motor is to be moved
- uint16_t motor3steps

number of steps the 3rd motor is to be moved

- uint16_t motor1idx

  controller index of motor 1
- uint16_t motor2idx

  controller index of motor 2
- uint16_t motor3idx

  controller index of motor 3
- Boost_serial * **bserial**
- std::string **response**

7.9.1 Detailed Description

The Motor class operates the PCB-motor.

Bug There are no known bugs.

The PCBMotor is controllable by sending ASCII commands over a serial connection. This class establishes such a connection and controls the movements of the motor.

7.9.2 Member Function Documentation

7.9.2.1 command_microstep()

```
void PCBMotor::command_microstep (
            uint16_t nummsteps,
            std::string dirstring )
```

command_microstep aplies nummsteps micropulses to the motor

Parameters

nummsteps	number of micropulses to apply
dirstring	string containing the desired direction

dirstring may either be "bw" of "fw" for backward/forward movement.

7.9.2.2 command_move()

```
void PCBMotor::command_move (
            int mnum,
            double ang )
```

command_moveboth moves both motors connected to the controller

Parameters

hwp_mnum	motor number of HWP
hwp_mnum	motor number of QWP
hwpang	angle HWP is to be moved to
qwpang	angle QWP is to be moved to

7.9.2.3 command_moveboth()

```
void PCBMotor::command_moveboth (
            int hwp_mnum,
            int qwp_mnum,
            double hwpang,
            double qwpang )  [virtual]
```

command_moveboth moves both motors connected to the controller

Parameters

hwp_mnum	motor number of HWP
hwp_mnum	motor number of QWP
hwpang	angle HWP is to be moved to
qwpang	angle QWP is to be moved to

Implements rotmotor.

7.9.2.4 command_movethree()

```
void PCBMotor::command_movethree (
            int hwp_mnum,
            int qwp_mnum,
            int qwp2_mnum,
            double hwp_ang,
            double qwp_ang,
            double qwp2_ang )  [virtual]
```

command_movethree moves three motors connected to the controller

Parameters

hwp_mnum	index of HWP
qwp_mnum	index of QWP
qwp2_mnum	index of QWP2
hwp_ang	angle to rotate HWP to
qwp_ang	angle to rotate QWP to
qwp2_ang	angle to rotate QWP2 to

Implements rotmotor.

7.9.2.5 command_singlestep()

```
void PCBMotor::command_singlestep (
            std::string dirstring )
```

command_singlestep moves the motor a single step in a direction specified by dirstring

Parameters

dirstring	a string containing the desired movenent direction

Dirstring may either be "bw" of "fw" for backward/forward movement.

7.9.2.6 command_step()

```
void PCBMotor::command_step (
            uint16_t numsteps,
            std::string dirstring )
```

command_step moves the motor numstep steps in a direction specified by dirstring

Parameters

numsteps	number of steps to go
dirstring	direction to go

Dirstring may either be "bw" of "fw" for backward/forward movement.

7.9.2.7 isopen()

```
bool PCBMotor::isopen ( )  [private], [virtual]
```

isopen returns the state of the serial connection

Returns

true if serial connection was established successfully, false otherwise

Implements rotmotor.

7.9.2.8 sensordata()

```
bool PCBMotor::sensordata ( )
```

sensordata returns the current PCBMotor optical encoder wheel sendor state

Returns

the current PCBMotor optical encoder wheel sendor state

7.9.2.9 stop()

```
void PCBMotor::stop (
            bool stop )
```

stop Tries to stop movenents if possible

Parameters

stop	Input: True if movents shall be stopped if possible

7.9.2.10 write()

```
void PCBMotor::write (
            const std::string & data )  [private]
```

write writes to the serialport

Parameters

data	data to be written to the serial port

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

- /home/peter/Development/cagecontrol/src/pcbmotor.h
- /home/peter/Development/cagecontrol/src/pcbmotor.cpp

7.10 Boost_serial::ReadSetupParameters Class Reference

Public Member Functions

- **ReadSetupParameters** (const std::string &delim)
- **ReadSetupParameters** (char *data, size_t size)

Public Attributes

- bool fixedSize

  True if need to read a fixed number of parameters.
- std::string delim

  String end delimiter (valid if fixedSize=false)
- char * data

  Pointer to data array (valid if fixedSize=true)
- size_t size

  Array size (valid if fixedSize=true)

7.10.1 Detailed Description

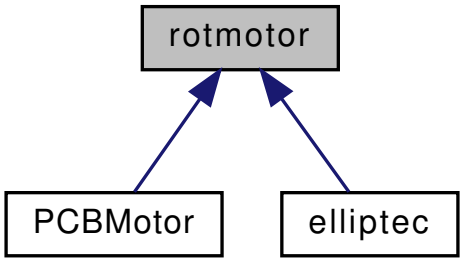
Parameters of performReadSetup. Just wrapper class, no encapsulation provided

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

- /home/peter/Development/cagecontrol/src/boost_serial.h

7.11 rotmotor Class Reference

Inheritance diagram for rotmotor:



Public Member Functions

- virtual void close ()=0

  close closes the serialport connection
- virtual bool isopen ()=0

  isopen returns the state of the serial connection
- virtual void **command_moveboth** (int a, int b, double c, double d)=0
- virtual void **command_movethree** (int a, int b, int c, double d, double e, double f)=0

7.11.1 Member Function Documentation

7.11.1.1 isopen()

```
virtual bool rotmotor::isopen ( )  [pure virtual]
```

isopen returns the state of the serial connection

Returns

true if serial connection was established successfully, false otherwise

Implemented in PCBMotor, and elliptec.

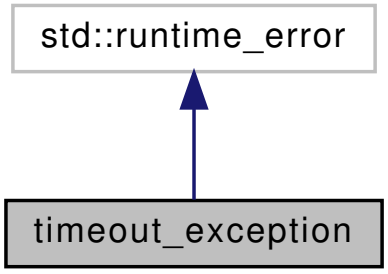
The documentation for this class was generated from the following files:

- /home/peter/Development/cagecontrol/src/rotmotor.h
- /home/peter/Development/cagecontrol/src/rotmotor.cpp

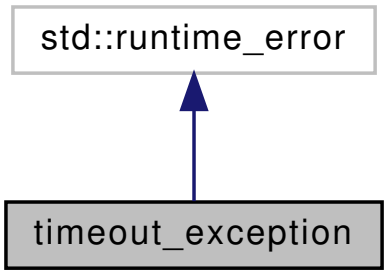
7.12 timeout_exception Class Reference

```
#include <boost_serial.h>
```

Inheritance diagram for timeout_exception:



Collaboration diagram for timeout_exception:



Public Member Functions

- **timeout_exception** (const std::string &arg)

7.12.1 Detailed Description

Thrown if timeout occurs

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

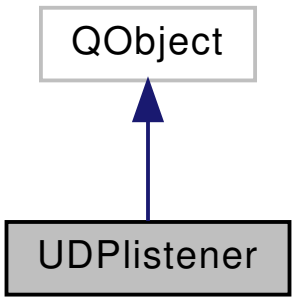
- /home/peter/Development/cagecontrol/src/boost_serial.h

7.13 UDPlistener Class Reference

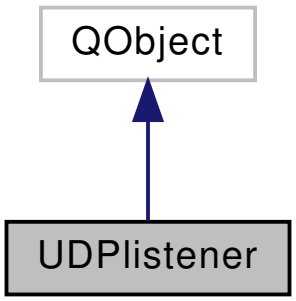
The UDPlistener class is used to control dinspect with UDP packages.

`#include <udplistener.h>`

Inheritance diagram for UDPlistener:



Collaboration diagram for UDPlistener:



Public Slots

- void bind ()

  bind Binds to a new UDP port

Signals

- void changeWPangles (QVector< double > angles)

  changeWPangles emitted when received message contains command to set waveplate characterization angles ('H')
- void changeoffsetusage (bool useoffset)

  changeoffsetusage emitted when massage to change usage of offset is received
- void Move (QString controller, double HWPang, double QWPang, double QWP2ang)

  Move emitted when massage to move the waveplates in a cage to certain angles is received.
- void MoveHV (QString controller)

  MoveHV emitted when massage to move cage to H/V basis is received.
- void MovePM (QString controller)

  MovePM emitted when massage to move cage to P/M basis is received.
- void MoveRL (QString controller)

  MoveLR emitted when massage to move cage to R/L basis is received.
- void invert (QString motorcolor, bool inv)

  invert emitted when next bases change shall be inverted (V/H instead of H/V)
- void showmsg (QString msg)

  showmsg submits QString to show in status bar

Public Member Functions

- UDPlistener (QSettings *settings, QObject *parent=0)

  UDPlistener listen for commands on a UDP port and execute them.

Private Slots

- void processPendingDatagrams ()

  processPendingDatagrams reads data from the UDP socket
- void processCommands (QString msg)

  processCommands extracts commands out of received data and executes them

Private Attributes

- QSettings * settings

  configuration
- QUdpSocket socket

  the UDP socket
- uint port

  port the listener listens to
- bool alreadybound

  true if the listener is already listening to a port

7.13.1 Detailed Description

The UDPlistener class is used to control dinspect with UDP packages.

Bug There are no known bugs

7.13.2 Constructor & Destructor Documentation

7.13.2.1 UDPlistener()

```
UDPlistener::UDPlistener (
            QSettings * settings,
            QObject * parent = 0 )  [explicit]
```

UDPlistener listen for commands on a UDP port and execute them.

Parameters

settings	a pointer to a qsettings instance, used to get the port to bind to and known commands

UDPlistener opens a UDP socket and binds to a port specified in qsettings. Incoming packages are analysed to check if they contain known commands. If they do, these commands are executed.

- move(QString, QString)
- move(QString, double, double)
- useoffset(int)
- setwpangles(many doubles)

7.13.3 Member Function Documentation

7.13.3.1 bind

```
void UDPlistener::bind ( )  [slot]
```

bind Binds to a new UDP port

This function checks whether the UDP socket is already bound to a a specific port. If so, it closes this connection and binds to the new port. If not, it binds to the port right away.

7.13.3.2 changeoffsetusage

```
void UDPlistener::changeoffsetusage (
            bool useoffset )  [signal]
```

changeoffsetusage emitted when massage to change usage of offset is received

Parameters

useoffset	true if waveplate offset is to be used

7.13.3.3 changeWPangles

```
void UDPlistener::changeWPangles (
            QVector< double > angles )  [signal]
```

changeWPangles emitted when received message contains command to set waveplate characterization angles ('H')

Parameters

angles	vector containing all angles. Ordering: HWP0,HWP1,...,HWPn,QWP0,QWP1,...,QWPn

7.13.3.4 invert

```
void UDPlistener::invert (
            QString motorcolor,
            bool inv )  [signal]
```

invert emitted when next bases change shall be inverted (V/H instead of H/V)

Parameters

motorcolor	either colorcode of stage, or 'all'
invert	use inverted bases if true; don't do so if false

7.13.3.5 Move

```
void UDPlistener::Move (
            QString controller,
            double HWPang,
            double QWPang,
            double QWP2ang )  [signal]
```

Move emitted when massage to move the waveplates in a cage to certain angles is received.

Parameters

controller	either colorcode of cage or 'all'
HWPang	angle of the HWP in degree
QWPang	angle of the 1st QWP in degree
QWP2ang	angle of the 2nd QWP in degree

7.13.3.6 MoveHV

```
void UDPlistener::MoveHV (
            QString controller )  [signal]
```

MoveHV emitted when massage to move cage to H/V basis is received.

Parameters

controller	either colorcode of stage, or 'all'

7.13.3.7 MovePM

```
void UDPlistener::MovePM (
            QString controller )  [signal]
```

MovePM emitted when massage to move cage to P/M basis is received.

Parameters

controller	either colorcode of stage, or 'all'

7.13.3.8 MoveRL

```
void UDPlistener::MoveRL (
            QString controller )  [signal]
```

MoveLR emitted when massage to move cage to R/L basis is received.

Parameters

controller	either colorcode of stage, or 'all'

7.13.3.9 processCommands

```
void UDPlistener::processCommands (
            QString msg )  [private], [slot]
```

processCommands extracts commands out of received data and executes them

Parameters

msg	input: the received message

7.13.3.10 showmsg

```
void UDPlistener::showmsg (
            QString msg )  [signal]
```

showmsg submits QString to show in status bar

Parameters

msg	message

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

- /home/peter/Development/cagecontrol/src/udplistener.h
- /home/peter/Development/cagecontrol/src/udplistener.cpp

Chapter 8

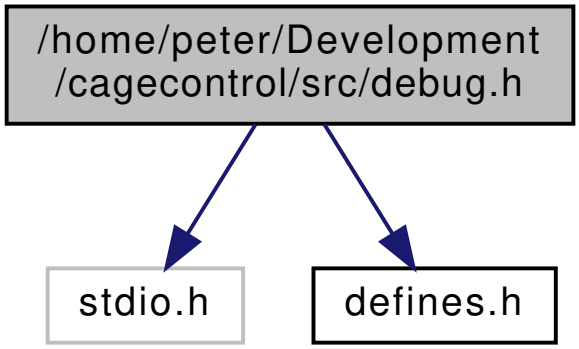
File Documentation

8.1 /home/peter/Development/cagecontrol/src/debug.h File Reference

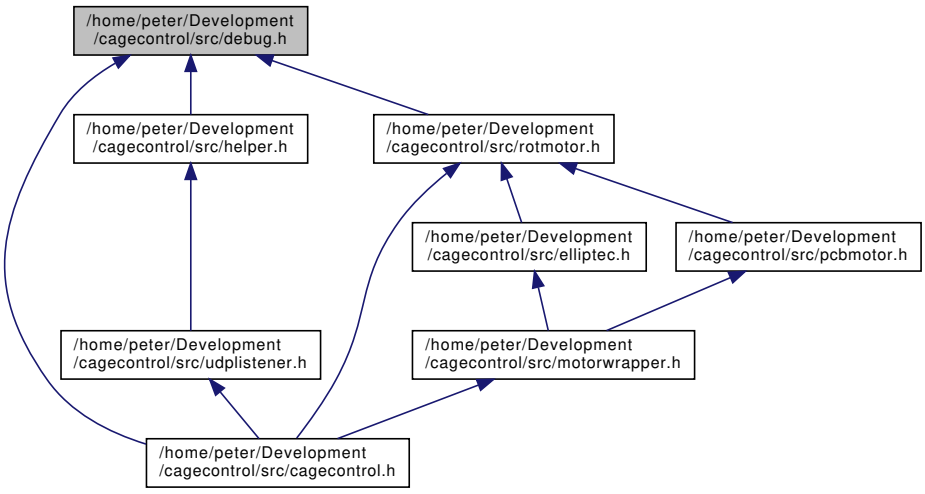
contains debug macros

```
#include "stdio.h"
#include "defines.h"
```

Include dependency graph for debug.h:



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



8.1.1 Detailed Description

contains debug macros

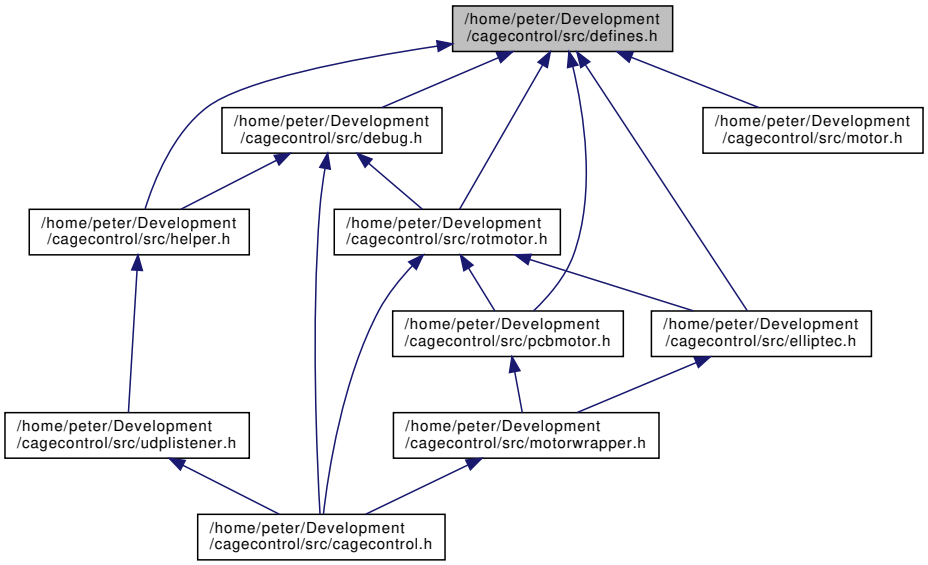
Bug Printing to console does not work on Windows. Workaround: Redirect stderr to stdout and redirect stdout to a file.

This file defines macros to style and simplify output to console.

8.2 /home/peter/Development/cagecontrol/src/defines.h File Reference

Various compile-time definitions.

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



Macros

- #define UNUSED(expr) do { (void)(expr); } while (0)
- #define DEBUG True
- #define DEBUGERROR 1
- #define DEBUGWARNING 1
- #define DEBUGINFO 1
- #define EPS 0.0000001
- #define PI 3.14159265358979323846
- #define DEGTORAD PI/180
- #define RADTODEG 180/PI

8.2.1 Detailed Description

Various compile-time definitions.

Bug There are no known bugs.

Contains definitions of various kind - mathematical, version constants, debug-variables, ...

8.2.2 Macro Definition Documentation

8.2.2.1 DEBUG

```
#define DEBUG True
```

Enables the execution of various debug-paths used during development.
default: FALSE.

8.2.2.2 DEBUGERROR

```
#define DEBUGERROR 1
```

If set to TRUE, enables the execution of the DEBUG_ERROR() makro which is used to write error messages (critical) to stdout.
default: true

8.2.2.3 DEBUGINFO

```
#define DEBUGINFO 1
```

If set to TRUE, enables the execution of the DEBUG_INFO() makro which is used to write usefull information to stdout.
default: true

8.2.2.4 DEBUGWARNING

```
#define DEBUGWARNING 1
```

If set to TRUE, enables the execution of the DEBUG_WARNING() makro which is used to write warnings about unexpected behaviour to stdout.
default: true

8.2.2.5 DEGTORAD

```
#define DEGTORAD PI/180
```

Conversion factor from degree to radians. PI/180

8.2.2.6 EPS

```
#define EPS 0.0000001
```

'epsilon' used to check floatingpoint variables in if-conditions.
default: 0.0000001

8.2.2.7 PI

```
#define PI 3.14159265358979323846
```

Pi.
default: 3.14159265358979323846

8.2.2.8 RADTODEG

```
#define RADTODEG 180/PI
```

Conversion factor from radians to degree. 180/PI

8.2.2.9 UNUSED

```
#define UNUSED(
              expr ) do { (void)(expr); } while (0)
```

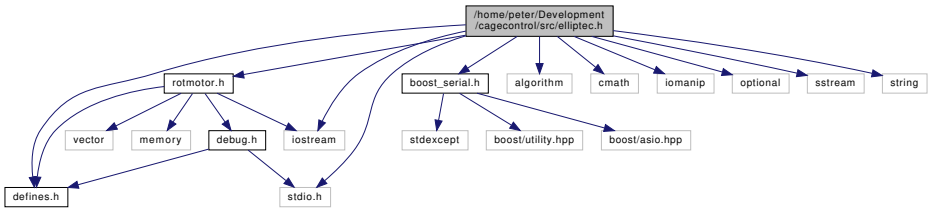
Use UNUSED(var) in function f(...,type var,...) to silence compiler warnings about unused parameter var.

8.3 /home/peter/Development/cagecontrol/src/elliptec.h File Reference

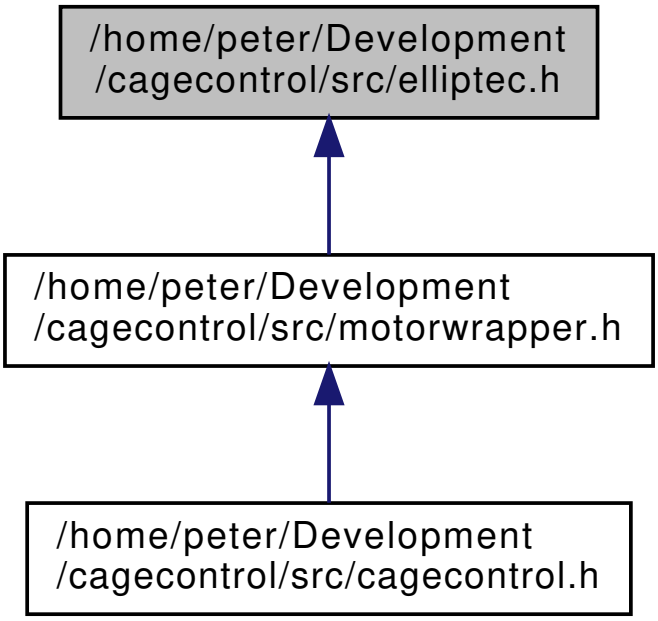
```
#include "defines.h"
#include "rotmotor.h"
#include "boost_serial.h"
#include <algorithm>
#include <cmath>
#include <iostream>
#include <iomanip>
#include <optional>
#include <sstream>
#include <stdio.h>
```

```
#include <string>
```

Include dependency graph for elliptec.h:



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



Classes

- struct ell_device
- struct ell_response
- class elliptec

Enumerations

- enum **ell_errors** {
  OK = 0 , **COMM_TIMEOUT** = 1 , **MECH_TIMEOUT** = 2 , **COMMAND_ERR** = 3 ,
  VAL_OUT_OF_RANGE = 4 , **MOD_ISOLATED** = 5 , **MOD_OUT_OF_ISOL** = 6 , **INIT_ERROR** = 7 ,
  THERMAL_ERROR = 8 , **BUSY** = 9 , **SENSOR_ERROR** = 10 , **MOTOR_ERROR** = 11 ,
  OUT_OF_RANGE = 12 , **OVER_CURRENT** = 13 , **GENERAL_ERROR** = 14 }

Variables

- const std::vector< std::string > **error_msgs**
- const std::unordered_map< std::string, std::vector< uint8_t > > **dt**

8.3.1 Variable Documentation

8.3.1.1 dt

```
const std::unordered_map<std::string, std::vector<uint8_t> > dt
```

Initial value:

```
= {
{"rotary", std::vector<uint8_t>({8, 14, 18})},
{"linear",  std::vector<uint8_t>({7, 10, 17, 20})},
{"linrot", std::vector<uint8_t>({7, 8, 10, 14, 17, 18, 20})},
{"indexed",  std::vector<uint8_t>({6, 9, 12})},
{"hasclean",  std::vector<uint8_t>({14, 17, 18, 20})}
}
```

8.3.1.2 error_msgs

```
const std::vector<std::string> error_msgs
```

Initial value:

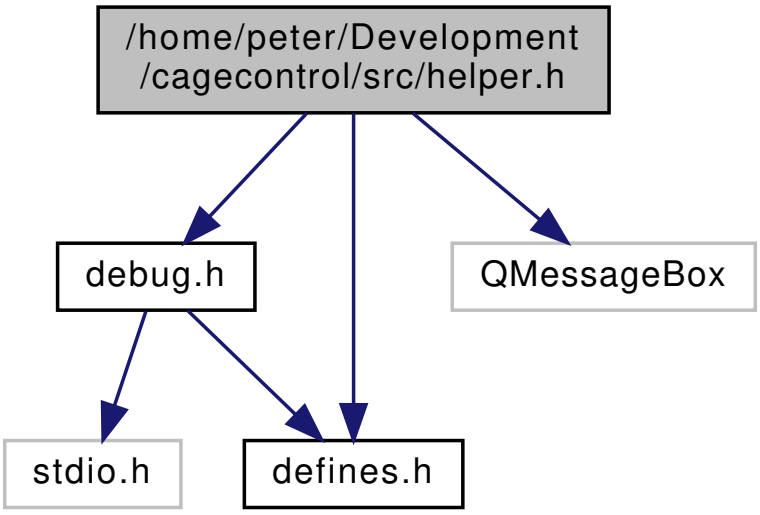
```
= {
    "OK, no error",
    "Communication time out",
    "Mechanical time out",
    "Command error or not supported",
    "Value out of range",
    "Module isolated",
    "Module out of isolation",
    "Initializing error",
    "Thermal error",
    "Busy",
    "Sensor Error (May appear during self test. If code persists there is an error)",
    "Motor Error (May appear during self test. If code persists there is an error)",
    "Out of Range (e.g. stage has been instructed to move beyond its travel range)",
    "Over Current error",

}
```

8.4 /home/peter/Development/cagecontrol/src/helper.h File Reference

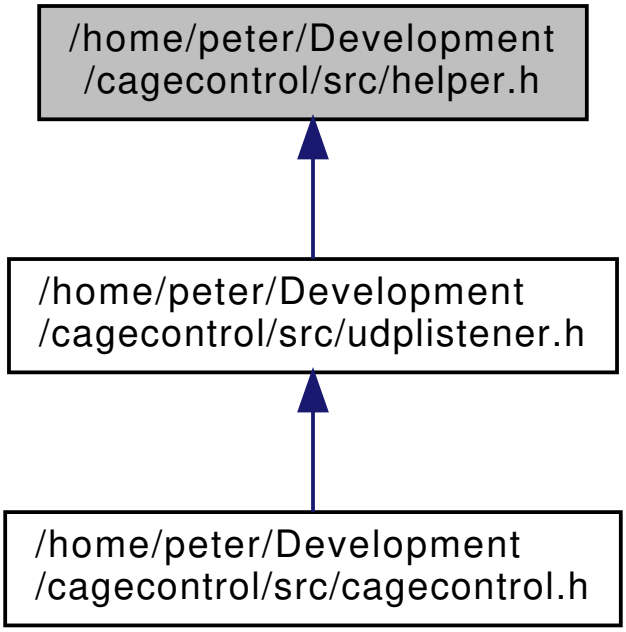
```
#include "debug.h"
#include "defines.h"
```

```
#include <QMessageBox>
```

Include dependency graph for helper.h:



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



Namespaces

- helper

  contains small functions to display messages

Functions

- void helper::message (QString msg)

message displays a message box

- void helper::error (QString msg)

  error displays an error-messagebox and writes a debug_error message to stdout

- void helper::warning (QString msg)

  warning displays warning-messagebox and writes a debug_warning message to stdout

- void helper::info (QString msg)

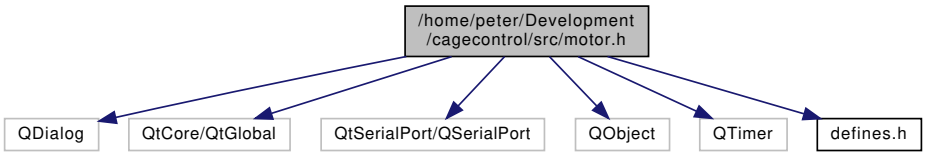
  info displays an info-messagebox and writes a debug_info message to stdout

- std::string **helper::str_tolower** (std::string s)

8.5 /home/peter/Development/cagecontrol/src/motor.h File Reference

```
#include <QDialog>
#include <QtCore/QtGlobal>
#include <QtSerialPort/QSerialPort>
#include <QObject>
#include <QTimer>
#include "defines.h"
```

Include dependency graph for motor.h:



Classes

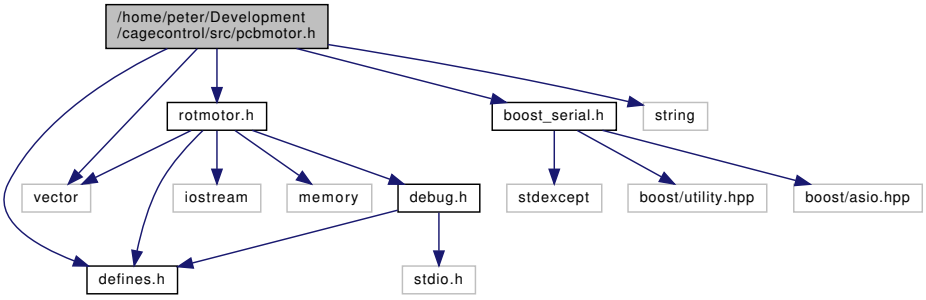
- class Motor

  The Motor class operates the PCB-motor.

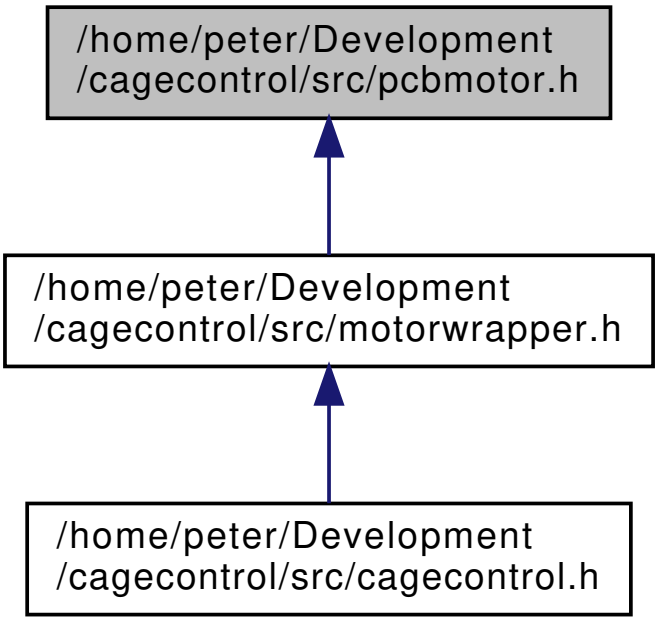
8.6 /home/peter/Development/cagecontrol/src/pcbmotor.h File Reference

```
#include "defines.h"
#include "rotmotor.h"
#include "boost_serial.h"
#include <string>
#include <vector>
```

Include dependency graph for pcbmotor.h:



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



Classes

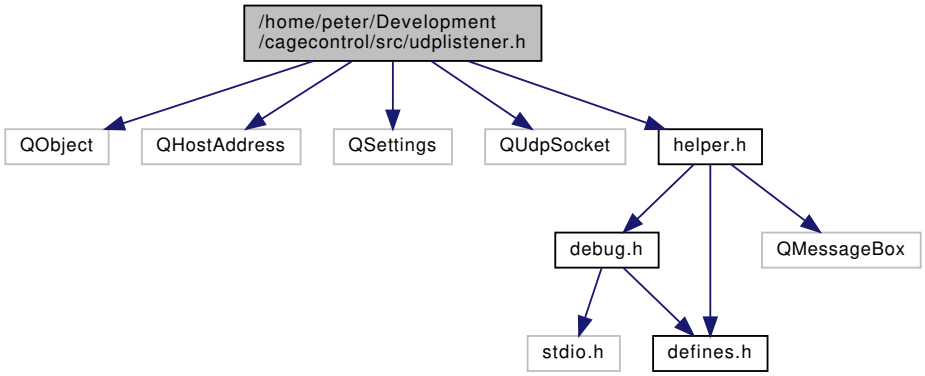
- class PCBMotor

  The Motor class operates the PCB-motor.

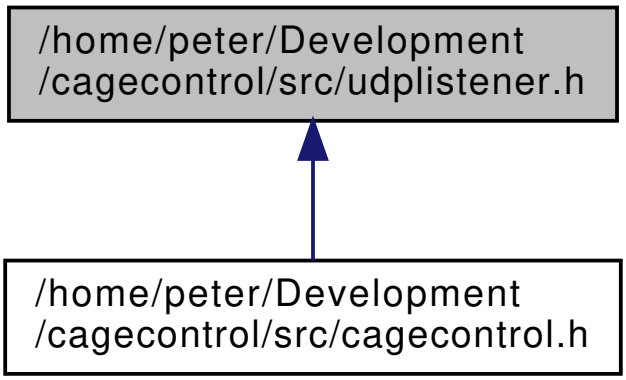
8.7 /home/peter/Development/cagecontrol/src/udplistener.h File Reference

```
#include <QObject>
#include <QHostAddress>
#include <QSettings>
#include <QUdpSocket>
#include "helper.h"
```

Include dependency graph for udplistener.h:



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



Classes

- class UDPlistener

  The UDPlistener class is used to control dinspect with UDP packages.

8.8 /home/peter/Development/cagecontrol/src/version.h File Reference

This file contains information about the code version.

Macros

- #define VERSION_GIT "v0.1-21-g6ad7527"

  The git commit description.
- #define VERSION_GIT_DATE 201812201517

  The date of the git commit.
- #define VERSION_BUILD_DATE 201812201519

  The builddate.

8.8.1 Detailed Description

This file contains information about the code version.

Author

Peter Schiansky

Bug There are no known bugs.

The definitions in this file are used to fill the about-dialog with information about the code: The git commit description, the commit date and the builddate.

Index