ohCaptain

V1.0

Generated by Doxygen 1.8.12

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 oCpt Namespace Reference

Namespaces

- components
- protocol
- vessels

Classes

- class Actuator
- class ActuatorTask
- class ARM
- class Boatswain
- class Captain
- class CommunicationTask
- class CoveragePathTask

  An object representing a coverage path task.
- class DredgeTask

  An Object representing a dredging task.
- class FollowTask

  An object representing a follow the target task.
- class iActuator
- class iBoatswain
- class iCaptain
- class iComm
- class iController
- class iSensor
- class iTask

  Task interface, all tasks need to adhere to this structure.
- class iVessel
- class LogTask

  An Object representing a data logging task.
- class LoRa
- class oCptException

- class PathTask

  An object representing a normal A to B type of path planning.
- class RouteTask
- class Sensor
- class SensorTask
- class Task
- class Vessel
- class WorkTask
- class World

5.2 oCpt::components Namespace Reference

Namespaces

- comm
- controller
- sensors

5.3 oCpt::components::comm Namespace Reference

Classes

- class LoRa_RN2483

5.4 oCpt::components::controller Namespace Reference

Classes

- class BBB

5.5 oCpt::components::sensors Namespace Reference

Classes

- class Gps
- class PT100
- class Razor

5.6 oCpt::protocol Namespace Reference

Classes

- class adc
- class gpio
- class Serial
- class userspace

5.7 oCpt::vessels Namespace Reference

Classes

- class Meetcatamaran

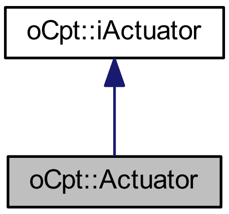
Chapter 6

Class Documentation

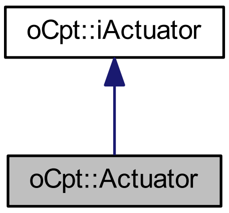
6.1 oCpt::Actuator Class Reference

```
#include <Actuator.h>
```

Inheritance diagram for oCpt::Actuator:



Collaboration diagram for oCpt::Actuator:



Public Member Functions

- Actuator ()
- virtual ∼Actuator () override
- virtual void setActuator () override
- virtual void run () override
- virtual void stop () override

Additional Inherited Members

6.1.1 Detailed Description

Definition at line 32 of file Actuator.h.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Actuator()

```
oCpt::Actuator::Actuator ( )
```

Definition at line 17 of file Actuator.cpp.

6.1.2.2 ∼Actuator()

```
oCpt::Actuator::∼Actuator ( )  [override], [virtual]
```

Definition at line 21 of file Actuator.cpp.

6.1.3 Member Function Documentation

6.1.3.1 run()

```
void oCpt::Actuator::run ( )  [override], [virtual]
```

Implements oCpt::iActuator.

Definition at line 29 of file Actuator.cpp.

6.1.3.2 setActuator()

```
void oCpt::Actuator::setActuator ( )  [override], [virtual]
```

Implements oCpt::iActuator.

Definition at line 25 of file Actuator.cpp.

6.1.3.3 stop()

```
void oCpt::Actuator::stop ( )  [override], [virtual]
```

Implements oCpt::iActuator.

Definition at line 33 of file Actuator.cpp.

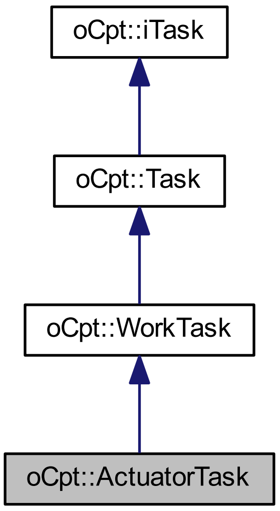
The documentation for this class was generated from the following files:

- include/Core/Actuator.h
- src/Core/Actuator.cpp

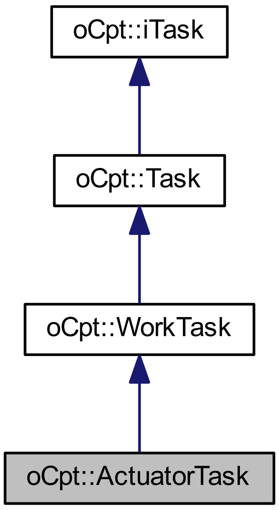
6.2 oCpt::ActuatorTask Class Reference

```
#include <Task.h>
```

Inheritance diagram for oCpt::ActuatorTask:



Collaboration diagram for oCpt::ActuatorTask:



Public Member Functions

- ActuatorTask (Vessel::ptr vessel, bool concurrent=true)
- virtual ∼ActuatorTask ()

Additional Inherited Members

6.2.1 Detailed Description

Definition at line 312 of file Task.h.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 ActuatorTask()

```
oCpt::ActuatorTask::ActuatorTask (
            Vessel::ptr vessel,
            bool concurrent = true )
```

Definition at line 77 of file Task.cpp.

6.2.2.2 ∼ActuatorTask()

```
oCpt::ActuatorTask::∼ActuatorTask ( )  [virtual]
```

Definition at line 79 of file Task.cpp.

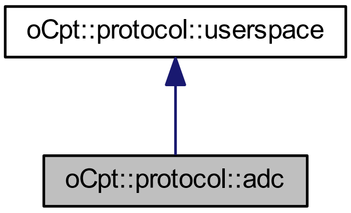
The documentation for this class was generated from the following files:

- include/Core/Task.h
- src/Core/Task.cpp

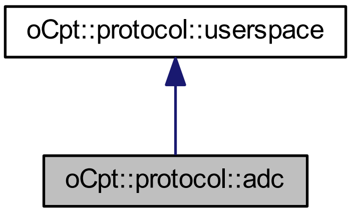
6.3 oCpt::protocol::adc Class Reference

```
#include <Controller.h>
```

Inheritance diagram for oCpt::protocol::adc:



Collaboration diagram for oCpt::protocol::adc:



Public Types

- typedef boost::shared_ptr< adc > ptr

Public Member Functions

- adc (uint8_t id, uint8_t device, std::string modName="")
- virtual ∼adc ()
- uint16_t & getValue ()
- bool operator== (const adc &rhs)
- bool compare (const uint8_t &id, const uint8_t &device=0)

Private Attributes

- uint8_t id_ = 0
- uint8_t device_ = 0
- std::string path_ = ""
- uint16_t value_ = 0

Additional Inherited Members

6.3.1 Detailed Description

The Analogue to Digital converter class. This class reads the voltage of an analogie pin, from user space.

Definition at line 89 of file Controller.h.

6.3.2 Member Typedef Documentation

6.3.2.1 ptr

```
typedef boost::shared_ptr<adc> oCpt::protocol::adc::ptr
```

Definition at line 91 of file Controller.h.

6.3.3 Constructor & Destructor Documentation

6.3.3.1 adc()

```
oCpt::protocol::adc::adc (
            uint8_t id,
            uint8_t device,
            std::string modName = "" )
```

The constructor of the adc class

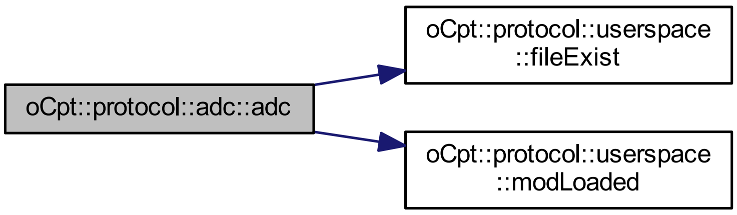
Parameters

id	the pin ID as an uint8_t value
device	the device or chip which handles the communication with the analogue pins
modName	the name of the modules which needs to be loaded TODO check if it is allways needed to load a module

Definition at line 61 of file Controller.cpp.

References ADC_IO_BASE_PATH, ADC_VOLTAGE_PATH, ADC_VOLTAGE_SUB_PATH, oCpt::protocol↩ ::userspace::fileExist(), and oCpt::protocol::userspace::modLoaded().

Here is the call graph for this function:


oCpt::protocol::adc::adc
oCpt::protocol::userspace ::fileExist
oCpt::protocol::userspace ::modLoaded

6.3.3.2 ∼adc()

```
oCpt::protocol::adc::∼adc ( )  [virtual]
```

The deconstrcutor

Definition at line 79 of file Controller.cpp.

6.3.4 Member Function Documentation

6.3.4.1 compare()

```
bool oCpt::protocol::adc::compare (
             const uint8_t & id,
             const uint8_t & device = 0 )
```

Compare function

Parameters

id	ID to be checked
device	Device name to be checked

Returns

either true or false

Definition at line 95 of file Controller.cpp.

6.3.4.2 getValue()

```
uint16_t & oCpt::protocol::adc::getValue ( )
```

gets the current raw voltage level as resolution

Returns

the raw voltage level as uint16_t

Definition at line 81 of file Controller.cpp.

6.3.4.3 operator==()

```
bool oCpt::protocol::adc::operator== (
             const adc & rhs )
```

Checks if adc object is the same

Parameters

rhs	other adc object, to be checked against

Returns

either true or false

Definition at line 91 of file Controller.cpp.

References path_.

6.3.5 Member Data Documentation

6.3.5.1 device_

```
uint8_t oCpt::protocol::adc::device_ = 0  [private]
```

Definition at line 130 of file Controller.h.

6.3.5.2 id_

```
uint8_t oCpt::protocol::adc::id_ = 0  [private]
```

Definition at line 129 of file Controller.h.

6.3.5.3 path_

```
std::string oCpt::protocol::adc::path_ = ""  [private]
```

Definition at line 131 of file Controller.h.

Referenced by operator==().

6.3.5.4 value_

```
uint16_t oCpt::protocol::adc::value_ = 0  [private]
```

Definition at line 132 of file Controller.h.

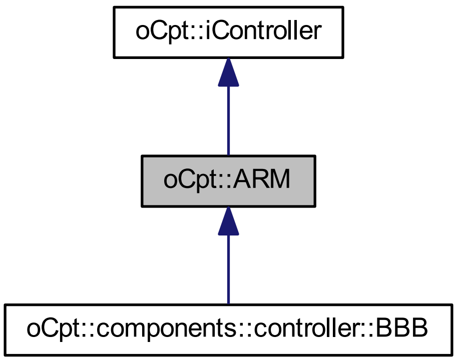
The documentation for this class was generated from the following files:

- include/Core/Controller.h
- src/Core/Controller.cpp

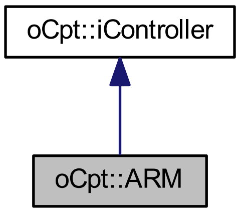
6.4 oCpt::ARM Class Reference

```
#include <Controller.h>
```

Inheritance diagram for oCpt::ARM:



Collaboration diagram for oCpt::ARM:



Public Member Functions

- ARM (World::ptr world)
- virtual ∼ARM ()
- virtual std::vector< protocol::adc::ptr > ∗ getAdcVector ()
- virtual protocol::adc::ptr getADC (uint8_t id, uint8_t device)

Additional Inherited Members

6.4.1 Detailed Description

An ARM like controller. Currently only ARM devices are implemented

Definition at line 597 of file Controller.h.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 ARM()

```
oCpt::ARM::ARM (
              World::ptr world )
```

The constructor of an ARM controller

Parameters

world	a shared_ptr to the World

Definition at line 469 of file Controller.cpp.

6.4.2.2 ∼ARM()

```
oCpt::ARM::∼ARM ( )  [virtual]
```

The deconstructor

Definition at line 472 of file Controller.cpp.

6.4.3 Member Function Documentation

6.4.3.1 getADC()

```
protocol::adc::ptr oCpt::ARM::getADC (
            uint8_t id,
            uint8_t device )  [virtual]
```

Get a specific shared_ptr to an ADC

Parameters

id	the pin ID
device	the device ID

Returns

returns the specified ADC

Implements oCpt::iController.

Definition at line 474 of file Controller.cpp.

References oCpt::iController::adcVector_.

6.4.3.2 getAdcVector()

```
std::vector< protocol::adc::ptr > * oCpt::ARM::getAdcVector ( )  [virtual]
```

Obtain a vector of available ADCs

Returns

Implements oCpt::iController.

Definition at line 481 of file Controller.cpp.

References oCpt::iController::adcVector_.

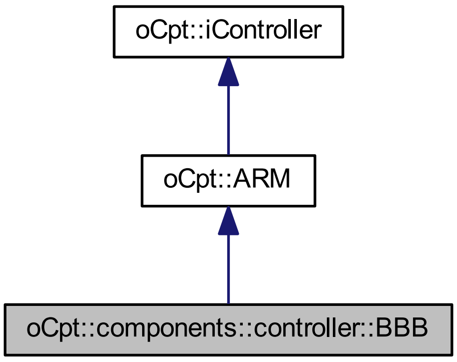
The documentation for this class was generated from the following files:

- include/Core/Controller.h
- src/Core/Controller.cpp

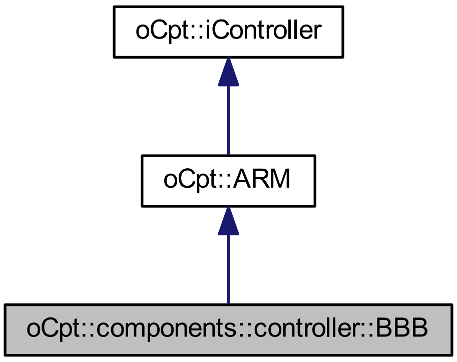
6.5 oCpt::components::controller::BBB Class Reference

```
#include <BeagleboneBlack.h>
```

Inheritance diagram for oCpt::components::controller::BBB:



Collaboration diagram for oCpt::components::controller::BBB:



Public Member Functions

- BBB (World::ptr world)
- virtual ∼BBB ()

Additional Inherited Members

6.5.1 Detailed Description

Definition at line 14 of file BeagleboneBlack.h.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 BBB()

```
oCpt::components::controller::BBB::BBB (
            World::ptr world )
```

Definition at line 11 of file BeagleboneBlack.cpp.

References oCpt::iController::adcVector_.

6.5.2.2 ∼BBB()

```
oCpt::components::controller::BBB::∼BBB ( )  [virtual]
```

Definition at line 20 of file BeagleboneBlack.cpp.

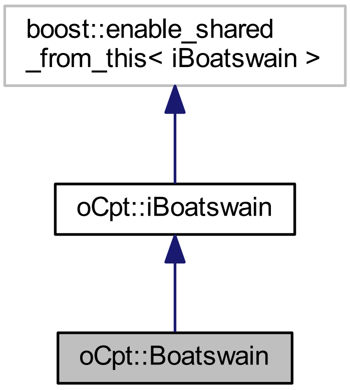
The documentation for this class was generated from the following files:

- include/Controllers/BeagleboneBlack.h
- src/Controllers/BeagleboneBlack.cpp

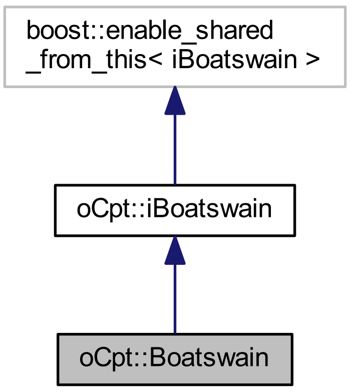
6.6 oCpt::Boatswain Class Reference

```
#include <Boatswain.h>
```

Inheritance diagram for oCpt::Boatswain:



Collaboration diagram for oCpt::Boatswain:



Public Member Functions

- Boatswain (iController::ptr controller)
- virtual ∼Boatswain () override
- virtual void run () override
- virtual void stop () override
- virtual void initialize () override
- virtual void registerSensor (iSensor::ptr sensor) override
- virtual void registerActuator (iActuator::ptr actuator) override
- virtual void registerComm (iComm::ptr comm) override

Protected Member Functions

- void resetTimer (iSensor::ptr sensor) override

Additional Inherited Members

6.6.1 Detailed Description

The Boatswain performs all the labours tasks, suchs updateing and interpretting sensor readings, setting actuators according to the Captain wishes, updating the state representation of the vessel in the World. Each Boatswain runs on its own thread. It is possible for a vessel to have multiple Boatswains, responsible for multiple tasks, such as communication, localization, steering. Each Boatswain has to adhere to the iBoatswain interface.

Definition at line 113 of file Boatswain.h.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 Boatswain()

```
oCpt::Boatswain::Boatswain (
            iController::ptr controller )
```

The constructor for a Boatswain

Parameters

controller	a shared_ptr to the controller with which teh Boatswain interacts

Definition at line 29 of file Boatswain.cpp.

6.6.2.2 ∼Boatswain()

```
oCpt::Boatswain::∼Boatswain ( )  [override], [virtual]
```

Deconstructor

Definition at line 33 of file Boatswain.cpp.

6.6.3 Member Function Documentation

6.6.3.1 initialize()

```
void oCpt::Boatswain::initialize ( )  [override], [virtual]
```

Initialize the Boatswain

Implements oCpt::iBoatswain.

Definition at line 48 of file Boatswain.cpp.

6.6.3.2 registerActuator()

```
void oCpt::Boatswain::registerActuator (
            iActuator::ptr actuator )  [override], [virtual]
```

Register a new Actuator with the Boatswain

Parameters

actuator	a shared_ptr to an Actuator

Implements oCpt::iBoatswain.

Definition at line 71 of file Boatswain.cpp.

6.6.3.3 registerComm()

```
void oCpt::Boatswain::registerComm (
            iComm::ptr comm )  [override], [virtual]
```

Register a new iComm device by setting a shared IO service

Parameters

comm	

Implements oCpt::iBoatswain.

Definition at line 100 of file Boatswain.cpp.

References oCpt::iBoatswain::ioservice_.

6.6.3.4 registerSensor()

```
void oCpt::Boatswain::registerSensor (
            iSensor::ptr sensor )  [override], [virtual]
```

Register a new Sensor with the Boatswain. If the Timer for the Sensor is set to a value greater the 0, the Sensor is registered with the timerSensors_ and a timer is set. Otherwise the Sensor is registered as a manualSensors_

Parameters

sensor	a shared_ptr to a Sensor

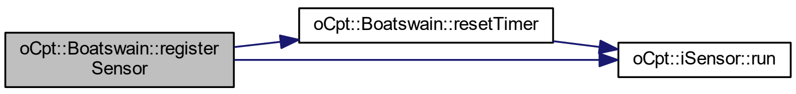
If the timer is set for the sensor, create a new timer service, register the sensor with the timer sensors, and set the callback functions to execute the Sensor::run function and the internall resetTimer function. If the timer is not set register the Sonsor with the manual sensor.

Implements oCpt::iBoatswain.

Definition at line 52 of file Boatswain.cpp.

References oCpt::iBoatswain::ioservice_, oCpt::iBoatswain::manualSensors_, resetTimer(), oCpt::iSensor::run(), oCpt::iBoatswain::timers_, and oCpt::iBoatswain::timerSensors_.

Here is the call graph for this function:



6.6.3.5 resetTimer()

```
void oCpt::Boatswain::resetTimer (
            iSensor::ptr sensor )  [override], [protected], [virtual]
```

reset the timer of a Sensor

Parameters

sensor	a shared_ptr to the Sensor

Don't excute if the thread is stopped

Find the current index of the sensor, this could maybe optimized by using a mappping list

Set the new timer. drift isn't taken into account at the current time.

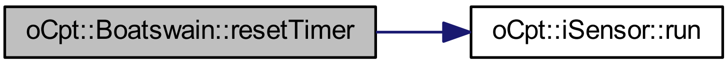
Implements oCpt::iBoatswain.

Definition at line 75 of file Boatswain.cpp.

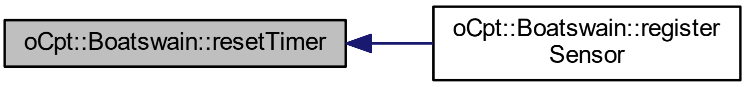
References oCpt::iBoatswain::ioservice_, oCpt::iBoatswain::localStopThread_, oCpt::iSensor::run(), oCpt::i↩
Boatswain::stopThread_, oCpt::iBoatswain::timers_, and oCpt::iBoatswain::timerSensors_.

Referenced by registerSensor().

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.3.6 run()

```
void oCpt::Boatswain::run ( )  [override], [virtual]
```

Make the Boatswain work and execute the actuators, sensors and communications

Implements oCpt::iBoatswain.

Definition at line 37 of file Boatswain.cpp.

References oCpt::iBoatswain::ioservice_, and oCpt::iBoatswain::manualSensors_.

6.6.3.7 stop()

```
void oCpt::Boatswain::stop ( )  [override], [virtual]
```

Stop the execution of the tasks

Implements oCpt::iBoatswain.

Definition at line 44 of file Boatswain.cpp.

References oCpt::iBoatswain::localStopThread_.

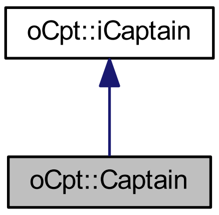
The documentation for this class was generated from the following files:

- include/Core/Boatswain.h
- src/Core/Boatswain.cpp

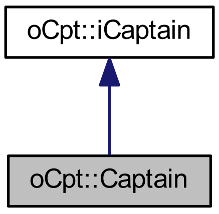
6.7 oCpt::Captain Class Reference

```
#include <Captain.h>
```

Inheritance diagram for oCpt::Captain:



Collaboration diagram for oCpt::Captain:



Public Member Functions

- Captain (World::ptr world)
- virtual ~Captain () override
- virtual void run () override
- virtual void stop () override
- virtual void initialize () override

Additional Inherited Members

6.7.1 Detailed Description

Definition at line 36 of file Captain.h.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 Captain()

```
oCpt::Captain::Captain (
            World::ptr world )
```

Definition at line 9 of file Captain.cpp.

References oCpt::iCaptain::localStopThread_.

6.7.2.2 ~Captain()

```
oCpt::Captain::~Captain ( )  [override], [virtual]
```

Definition at line 13 of file Captain.cpp.

6.7.3 Member Function Documentation

6.7.3.1 initialize()

```
void oCpt::Captain::initialize ( )  [override], [virtual]
```

Implements oCpt::iCaptain.

Definition at line 27 of file Captain.cpp.

6.7.3.2 run()

```
void oCpt::Captain::run ( )  [override], [virtual]
```

Implements oCpt::iCaptain.

Definition at line 17 of file Captain.cpp.

References oCpt::iCaptain::localStopThread_, and oCpt::iCaptain::stopThread_.

6.7.3.3 stop()

```
void oCpt::Captain::stop ( )  [override], [virtual]
```

Implements oCpt::iCaptain.

Definition at line 23 of file Captain.cpp.

References oCpt::iCaptain::localStopThread_.

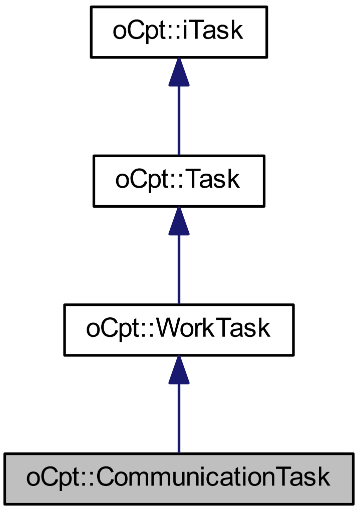
The documentation for this class was generated from the following files:

- include/Core/Captain.h
- src/Core/Captain.cpp

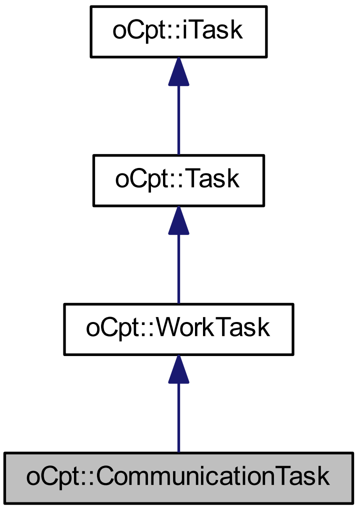
6.8 oCpt::CommunicationTask Class Reference

```
#include <Task.h>
```

Inheritance diagram for oCpt::CommunicationTask:



Collaboration diagram for oCpt::CommunicationTask:



Public Member Functions

- CommunicationTask (Vessel::ptr vessel, bool concurrent=true)
- virtual ∼CommunicationTask ()

Additional Inherited Members

6.8.1 Detailed Description

Definition at line 322 of file Task.h.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 CommunicationTask()

```
oCpt::CommunicationTask::CommunicationTask (
            Vessel::ptr vessel,
            bool concurrent = true )
```

Definition at line 81 of file Task.cpp.

6.8.2.2 ∼CommunicationTask()

```
oCpt::CommunicationTask::∼CommunicationTask ( )  [virtual]
```

Definition at line 83 of file Task.cpp.

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

- include/Core/Task.h
- src/Core/Task.cpp

6.9 oCpt::World::Location::coordinate Struct Reference

```
#include <World.h>
```

Public Attributes

- double value
- cardinal_direction direction

6.9.1 Detailed Description

Definition at line 121 of file World.h.

6.9.2 Member Data Documentation

6.9.2.1 direction

```
cardinal_direction oCpt::World::Location::coordinate::direction
```

Definition at line 123 of file World.h.

Referenced by oCpt::components::sensors::Gps::interpretMsg().

6.9.2.2 value

```
double oCpt::World::Location::coordinate::value
```

Definition at line 122 of file World.h.

Referenced by oCpt::components::sensors::Gps::interpretMsg().

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

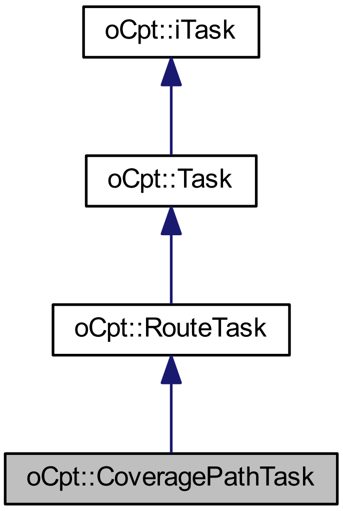
- include/Core/World.h

6.10 oCpt::CoveragePathTask Class Reference

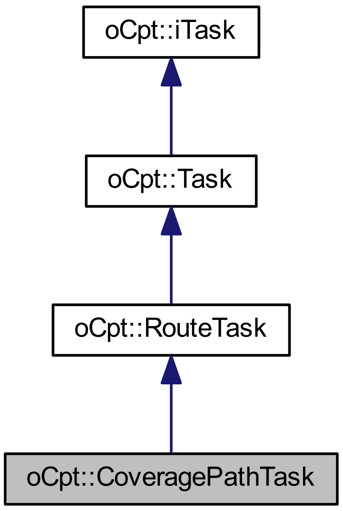
An object representing a coverage path task.

```
#include <Task.h>
```

Inheritance diagram for oCpt::CoveragePathTask:



Collaboration diagram for oCpt::CoveragePathTask:



Public Member Functions

- CoveragePathTask (Vessel::ptr vessel, bool concurrent=false)
- virtual ∼CoveragePathTask ()

Additional Inherited Members

6.10.1 Detailed Description

An object representing a coverage path task.

All these types of tasks need a robot to cover a complete region in order to perform their tasks. According to {cao_region_1988} such a mobile robot should use the following criteria, for a region filling operation:

1. The mobile robot must move through an entire area, i.e., the overall travel must cover a whole region.
2. The mobile robot must fill the region without overlapping paths.
3. Continuous and sequential operations without any repetition of paths is required of the robot.
4. The robot must avoid all obstacles in a region.
5. Simple motion trajectories (e.g., straight lines or circles) should be used for simplicity in control.
6. An "optimal" path is desired under the available conditions. It is not always possible to satisfy all these criteria for a complex environment. Sometimes a priority consideration is required.

Definition at line 201 of file Task.h.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 CoveragePathTask()

```
oCpt::CoveragePathTask::CoveragePathTask (
            Vessel::ptr vessel,
            bool concurrent = false )
```

Constructor of the interface

Returns

Definition at line 53 of file Task.cpp.

6.10.2.2 ∼CoveragePathTask()

```
oCpt::CoveragePathTask::∼CoveragePathTask ( )  [virtual]
```

The deconstructor

Definition at line 55 of file Task.cpp.

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

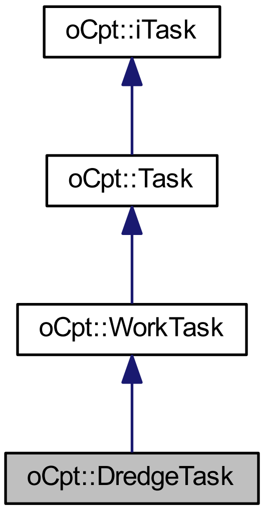
- include/Core/Task.h
- src/Core/Task.cpp

6.11 oCpt::DredgeTask Class Reference

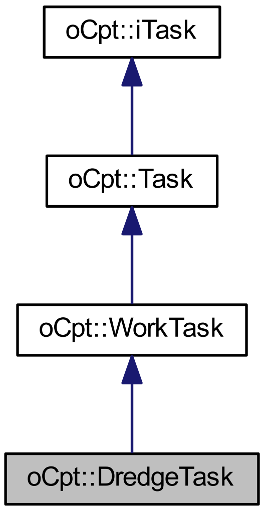
An Object representing a dredging task.

```
#include <Task.h>
```

Inheritance diagram for oCpt::DredgeTask:



Collaboration diagram for oCpt::DredgeTask:



Public Member Functions

- DredgeTask (Vessel::ptr vessel, bool concurrent=true)
- virtual ∼DredgeTask ()

Additional Inherited Members

6.11.1 Detailed Description

An Object representing a dredging task.

All these types tasks make use of an actuator and sensors to perform dredging tasks

Definition at line 287 of file Task.h.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 DredgeTask()

```
oCpt::DredgeTask::DredgeTask (
            Vessel::ptr vessel,
            bool concurrent = true )
```

Constructor of the interface

Returns

Definition at line 69 of file Task.cpp.

6.11.2.2 ∼DredgeTask()

```
oCpt::DredgeTask::∼DredgeTask ( )  [virtual]
```

The deconstructor

Definition at line 71 of file Task.cpp.

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

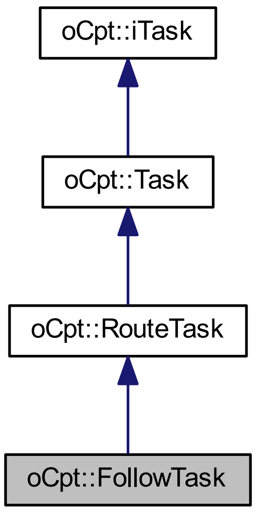
- include/Core/Task.h
- src/Core/Task.cpp

6.12 oCpt::FollowTask Class Reference

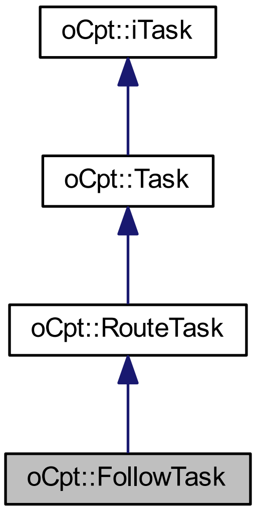
An object representing a follow the target task.

```
#include <Task.h>
```

Inheritance diagram for oCpt::FollowTask:



Collaboration diagram for oCpt::FollowTask:



Public Member Functions

- FollowTask (Vessel::ptr vessel, bool concurrent=false)
- virtual ∼FollowTask ()

Additional Inherited Members

6.12.1 Detailed Description

An object representing a follow the target task.

All these types of tasks need to follow a (moving) target

Definition at line 222 of file Task.h.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 FollowTask()

```
oCpt::FollowTask::FollowTask (
            Vessel::ptr vessel,
            bool concurrent = false )
```

Constructor of the interface

Returns

Definition at line 57 of file Task.cpp.

6.12.2.2 ∼FollowTask()

```
oCpt::FollowTask::∼FollowTask ( )  [virtual]
```

The deconstructor

Definition at line 59 of file Task.cpp.

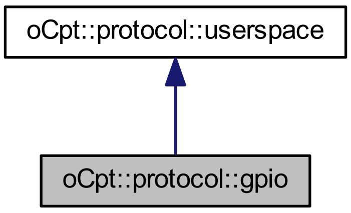
The documentation for this class was generated from the following files:

- include/Core/Task.h
- src/Core/Task.cpp

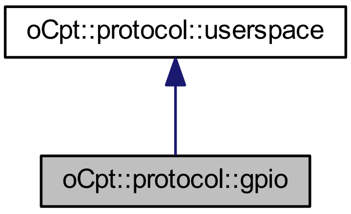
6.13 oCpt::protocol::gpio Class Reference

```
#include <Controller.h>
```

Inheritance diagram for oCpt::protocol::gpio:



Collaboration diagram for oCpt::protocol::gpio:



Public Types

- enum Direction { INPUT = 105, OUTPUT = 111 }
- enum Value { LOW = 48, HIGH = 49 }
- enum Edge { NONE = 110, RISING = 114, FALLING = 102, BOTH = 98 }
- typedef boost::shared_ptr< gpio > ptr
- typedef boost::signals2::signal< void()> signal_t
- typedef std::function< void()> cb_func

Public Member Functions

- gpio (int pinNumber, Direction direction=INPUT, Value value=LOW, Edge edge=NONE)
- ∼gpio ()
- int getPinNumber () const
- void setPinNumber (int pinNumber)

- Value getValue () const
- void setValue (Value value)
- Direction getDirection () const
- void setDirection (Direction direction)
- Edge getEdge () const
- void setEdge (Edge edge)
- void setCallbackFunction (cb_func cb)
- void waitForEdge ()
- void waitForEdgeAsync ()
- void toggle ()

Static Public Member Functions

- static std::vector< ptr > exportedGpios ()

Public Attributes

- signal_t signalChanged

Private Member Functions

- void internalCbFunc ()
- void exportPin (const int &number)
- void unexportPin (const int &number)
- template<typename T >
  void writePinValue (const int &number, const T &value)
- template<typename T >
  void writePinValue (std::string path, const T &value)

Static Private Member Functions

- template<typename T >
  static T readPinValue (const int &number)
- template<typename T >
  static T readPinValue (std::string path)

Private Attributes

- int pinNumber_
- Value value_
- Direction direction_
- Edge edge_
- std::string gpiopath_
- cb_func cb_
- bool threadRunning_

Additional Inherited Members

6.13.1 Detailed Description

A General Pin Input Output class. This is the class that handles gpio's in user space. Each pin can be set as either input or output, and have a High or a Low out-/input. When a pin is set as input, it can be polled on the edge, execute a function or send a signal on the rising, falling or changing edge of the signal

Definition at line 138 of file Controller.h.

6.13.2 Member Typedef Documentation

6.13.2.1 cb_func

```
typedef std::function<void()> oCpt::protocol::gpio::cb_func
```

Definition at line 142 of file Controller.h.

6.13.2.2 ptr

```
typedef boost::shared_ptr<gpio> oCpt::protocol::gpio::ptr
```

Definition at line 140 of file Controller.h.

6.13.2.3 signal_t

```
typedef boost::signals2::signal<void()> oCpt::protocol::gpio::signal_t
```

Definition at line 141 of file Controller.h.

6.13.3 Member Enumeration Documentation

6.13.3.1 Direction

```
enum oCpt::protocol::gpio::Direction
```

The Direction of the pin

Enumerator

INPUT	
OUTPUT	

Definition at line 147 of file Controller.h.

6.13.3.2 Edge

```
enum oCpt::protocol::gpio::Edge
```

Enumerator

NONE	
RISING	
FALLING	
BOTH	

Definition at line 157 of file Controller.h.

6.13.3.3 Value

```
enum oCpt::protocol::gpio::Value
```

Enumerator

LOW	
HIGH	

Definition at line 152 of file Controller.h.

6.13.4 Constructor & Destructor Documentation

6.13.4.1 gpio()

```
oCpt::protocol::gpio::gpio (
             int pinNumber,
             gpio::Direction direction = INPUT,
             gpio::Value value = LOW,
             gpio::Edge edge = NONE )
```

The constructor for the gpio class

Parameters

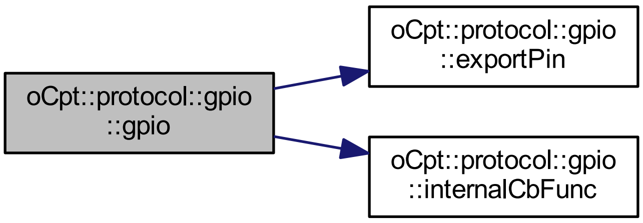
pinNumber	the pin pumber (in user-space mapping)
direction	the Direction of a pin, with a default value as Direction::INPUT
value	the start Value of a pin. With a default value of Value::LOW
edge	the Edge of a pin with the default value of Edge::NONE

Definition at line 337 of file Controller.cpp.

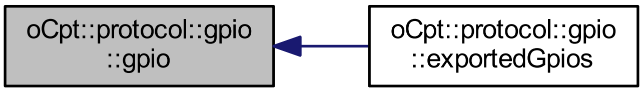
References cb_, direction_, edge_, exportPin(), GPIO_BASE_PATH, gpiopath_, internalCbFunc(), pinNumber_, and value_.

Referenced by exportedGpios().

Here is the call graph for this function:


oCpt::protocol::gpio ::gpio
oCpt::protocol::gpio ::exportPin
oCpt::protocol::gpio ::internalCbFunc

Here is the caller graph for this function:


oCpt::protocol::gpio ::gpio
oCpt::protocol::gpio ::exportedGpios

6.13.4.2 ∼gpio()

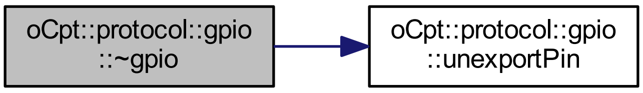
```
oCpt::protocol::gpio::∼gpio ( )
```

The deconstructor

Definition at line 353 of file Controller.cpp.

References pinNumber_, and unexportPin().

Here is the call graph for this function:


oCpt::protocol::gpio ::~gpio
oCpt::protocol::gpio ::unexportPin

6.13.5 Member Function Documentation

6.13.5.1 exportedGpios()

```
std::vector< gpio::ptr > oCpt::protocol::gpio::exportedGpios ( )  [static]
```

Static function which creates a vector containing new gpio shared_ptr for each pin that is currently exported in the user space.

Returns

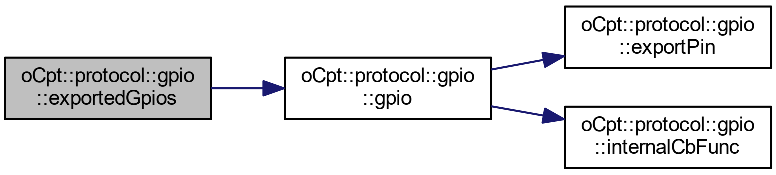
A vector with shared_ptr's of all exported gpio's in user-space

Iterate through all exported pins

Definition at line 357 of file Controller.cpp.

References gpio(), and GPIO_BASE_PATH.

Here is the call graph for this function:


oCpt::protocol::gpio ::exportedGpios
oCpt::protocol::gpio ::gpio
oCpt::protocol::gpio ::exportPin
oCpt::protocol::gpio ::internalCbFunc

6.13.5.2 exportPin()

```
void oCpt::protocol::gpio::exportPin (
             const int & number )  [private]
```

Export the pin in user-space

Parameters

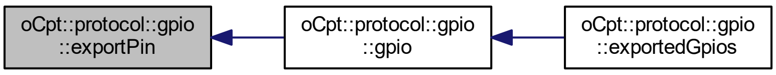
number	the pin number to be exported

Definition at line 381 of file Controller.cpp.

References GPIO_BASE_PATH.

Referenced by gpio().

Here is the caller graph for this function:


oCpt::protocol::gpio ::exportPin
oCpt::protocol::gpio ::gpio
oCpt::protocol::gpio ::exportedGpios

6.13.5.3 getDirection()

```
gpio::Direction oCpt::protocol::gpio::getDirection ( ) const
```

Get the current Direction. note this doesn't take into accoutn external changes done outside this library

Returns

either Direction::INPUT or Direction::Output

Definition at line 314 of file Controller.cpp.

6.13.5.4 getEdge()

```
gpio::Edge oCpt::protocol::gpio::getEdge ( ) const
```

Get the current Edge of the pin. If the Direction is set to Direction::INPUT the value is set in user-space, otherwise it is set in the object itself

Returns

Definition at line 323 of file Controller.cpp.

6.13.5.5 getPinNumber()

```
int oCpt::protocol::gpio::getPinNumber ( ) const
```

Get the current pin number

Returns

an int representing the pin number in user-space mapping

Definition at line 291 of file Controller.cpp.

6.13.5.6 getValue()

```
gpio::Value oCpt::protocol::gpio::getValue ( ) const
```

Get the current value of the pin, if the Direction is set to Direction::INPUT the value is obtained from the user space, otherwise the value is read from object itself

Returns

either Value::HIGH or Value::LOW

Definition at line 300 of file Controller.cpp.

6.13.5.7 internalCbFunc()

```
void oCpt::protocol::gpio::internalCbFunc ( )  [private]
```

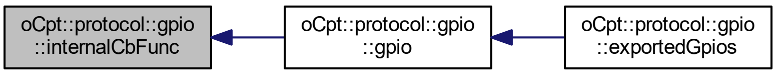
The internal callback function, which triggers the signalChanged signal

Definition at line 418 of file Controller.cpp.

References signalChanged.

Referenced by gpio().

Here is the caller graph for this function:


oCpt::protocol::gpio ::internalCbFunc
oCpt::protocol::gpio ::gpio
oCpt::protocol::gpio ::exportedGpios

6.13.5.8 readPinValue() [1/2]

```
template<typename T >
static T oCpt::protocol::gpio::readPinValue (
             const int & number )  [inline], [static], [private]
```

Static generic function returning the value in user-space of either Direction, Edge or Value. Depending on the typename

Template Parameters

T	The value to return either the Value, Edge or Direction

Parameters

number	the pin as number

Returns

The read value as either Value, Edge or Direction

Definition at line 291 of file Controller.h.

References GPIO_BASE_PATH.

6.13.5.9 readPinValue() [2/2]

```
template<typename T >
static T oCpt::protocol::gpio::readPinValue (
             std::string path )  [inline], [static], [private]
```

Static generic function returning the value in user-space of either Direction, Edge or Value. Depending on the typename. This function is quicker then the overload function taking the pin number as int. and is therefore preffered to obtain the Value.

Template Parameters

T	The value to return either the Value, Edge or Direction

Parameters

path	the pin as user-space path

Returns

the read value as either Value, Edge or Direction

Definition at line 304 of file Controller.h.

6.13.5.10 setCallbackFunction()

```
void oCpt::protocol::gpio::setCallbackFunction (
             gpio::cb_func cb )
```

Set a new Callbackfunction which is called on a certain Edge

Parameters

cb	the callback function

Definition at line 414 of file Controller.cpp.

References cb_.

6.13.5.11 setDirection()

```
void oCpt::protocol::gpio::setDirection (
             gpio::Direction direction )
```

Set the Direction of the pin

Parameters

direction	the Direction of the pin

Definition at line 318 of file Controller.cpp.

References direction_.

6.13.5.12 setEdge()

```
void oCpt::protocol::gpio::setEdge (
             gpio::Edge edge )
```

Set the Edge of the of the pin. if the Direction is set to Direction::INPUT, the value is set in user-space, otherwise it is set in the object itself

Parameters

edge	

Definition at line 330 of file Controller.cpp.

References edge_.

6.13.5.13 setPinNumber()

```
void oCpt::protocol::gpio::setPinNumber (
             int pinNumber )
```

Set the new pinbumber (don't use yet)

Parameters

pinNumber	the pinmuber to be set

Definition at line 295 of file Controller.cpp.

References pinNumber_.

6.13.5.14 setValue()

```
void oCpt::protocol::gpio::setValue (
            gpio::Value value )
```

Set the current value, if the Direction is set to Direction::OUTPUT the value is set ti userspace, either it is set to object itself

Parameters

value	

Definition at line 307 of file Controller.cpp.

References value_.

6.13.5.15 toggle()

```
void oCpt::protocol::gpio::toggle ( )
```

Toggle the value_ of the pin if Value::High then the value_ is set to Value::LOW

Definition at line 409 of file Controller.cpp.

References gpiopath_, and value_.

6.13.5.16 unexportPin()

```
void oCpt::protocol::gpio::unexportPin (
            const int & number )  [private]
```

Unexport the pin in user-space

Parameters

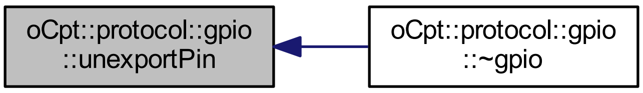
number	the pin number to be unexported

Definition at line 395 of file Controller.cpp.

References GPIO_BASE_PATH.

Referenced by ∼gpio().

Here is the caller graph for this function:


oCpt::protocol::gpio ::unexportPin
oCpt::protocol::gpio ::~gpio

6.13.5.17 waitForEdge()

```
void oCpt::protocol::gpio::waitForEdge ( )
```

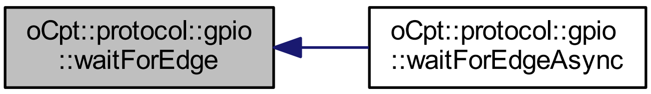
Wait for the occurance of a change in Edge, corresponding with the set value of Edge. When the change is detected the callbackfunction is called. This function blocks the current thread.

Definition at line 422 of file Controller.cpp.

References cb_, direction_, and gpiopath_.

Referenced by waitForEdgeAsync().

Here is the caller graph for this function:


oCpt::protocol::gpio ::waitForEdge
oCpt::protocol::gpio ::waitForEdgeAsync

6.13.5.18 waitForEdgeAsync()

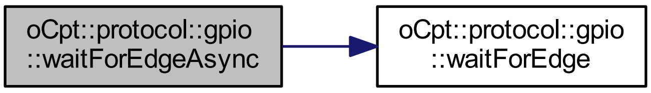
```
void oCpt::protocol::gpio::waitForEdgeAsync ( )
```

Wait for the ocurrance of a change in Edge, corresponding with the set value of Edge. When the chance is detected the callbackfunction is called. This function creates a new thread, allowing the current thread to run unhindered

Definition at line 457 of file Controller.cpp.

References waitForEdge().

Here is the call graph for this function:


oCpt::protocol::gpio ::waitForEdgeAsync
oCpt::protocol::gpio ::waitForEdge

6.13.5.19 writePinValue() [1/2]

```
template<typename T >
void oCpt::protocol::gpio::writePinValue (
             const int & number,
             const T & value )  [inline], [private]
```

Write the value to the pin. The T parameter determines which value to set

Template Parameters

T	the type could either be Value, Direction ro Direction

Parameters

number	the pin number as an integer
value	the Value to be set

Definition at line 331 of file Controller.h.

References GPIO_BASE_PATH.

6.13.5.20 writePinValue() [2/2]

```
template<typename T >
void oCpt::protocol::gpio::writePinValue (
             std::string path,
             const T & value )  [inline], [private]
```

Write the value to the pin, The T paramter determines which value to set. This overload is quicker then the one taking the integer and is therefore preferred

Template Parameters

T	the type could either be Value, Direction or Edge

Parameters

path	the pin as an user-space path
value	the Value to write

Definition at line 344 of file Controller.h.

6.13.6 Member Data Documentation

6.13.6.1 cb_

```
cb_func oCpt::protocol::gpio::cb_  [private]
```

Definition at line 264 of file Controller.h.

Referenced by gpio(), setCallbackFunction(), and waitForEdge().

6.13.6.2 direction_

```
Direction oCpt::protocol::gpio::direction_  [private]
```

Definition at line 261 of file Controller.h.

Referenced by gpio(), setDirection(), and waitForEdge().

6.13.6.3 edge_

```
Edge oCpt::protocol::gpio::edge_  [private]
```

Definition at line 262 of file Controller.h.

Referenced by gpio(), and setEdge().

6.13.6.4 gpiopath_

```
std::string oCpt::protocol::gpio::gpiopath_  [private]
```

Definition at line 263 of file Controller.h.

Referenced by gpio(), toggle(), and waitForEdge().

6.13.6.5 pinNumber_

```
int oCpt::protocol::gpio::pinNumber_  [private]
```

Definition at line 259 of file Controller.h.

Referenced by gpio(), setPinNumber(), and ∼gpio().

6.13.6.6 signalChanged

```
signal_t oCpt::protocol::gpio::signalChanged
```

The signal that is send if the internal callback fucntion is executed

Definition at line 256 of file Controller.h.

Referenced by internalCbFunc().

6.13.6.7 threadRunning_

```
bool oCpt::protocol::gpio::threadRunning_  [private]
```

Definition at line 265 of file Controller.h.

6.13.6.8 value_

```
Value oCpt::protocol::gpio::value_  [private]
```

Definition at line 260 of file Controller.h.

Referenced by gpio(), setValue(), and toggle().

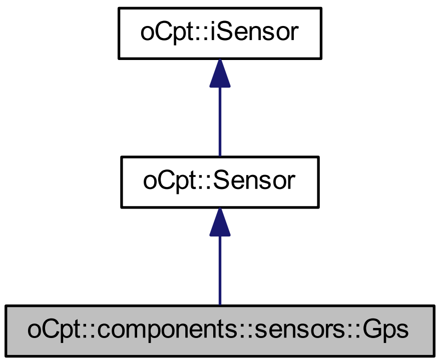
The documentation for this class was generated from the following files:

- include/Core/Controller.h
- src/Core/Controller.cpp

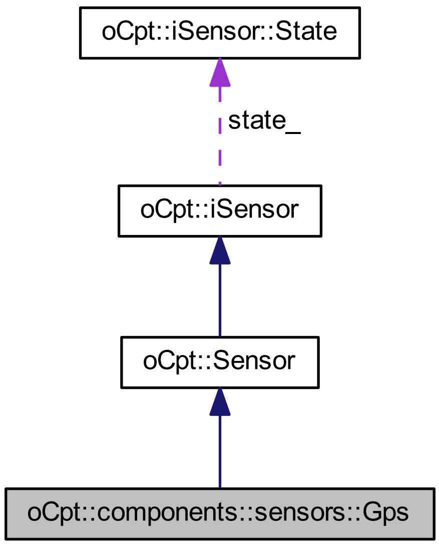
6.14 oCpt::components::sensors::Gps Class Reference

```
#include <Gps.h>
```

Inheritance diagram for oCpt::components::sensors::Gps:



Collaboration diagram for oCpt::components::sensors::Gps:



Public Types

- typedef oCpt::World::Location::gpsPoint_t ReturnValue_t

Public Member Functions

- Gps (iController::ptr controller, World::ptr world, std::string id, std::string device, unsigned int baudrate)
- ∼Gps ()
- void updateSensor ()
- void run ()
- void stop ()
- void setIOservice (boost::shared_ptr< boost::asio::io_service > ioservice)

Protected Member Functions

- void interpretMsg ()

Protected Attributes

- std::string device_
- protocol::Serial::ptr serial_

6.14.1 Detailed Description

Definition at line 13 of file Gps.h.

6.14.2 Member Typedef Documentation

6.14.2.1 ReturnValue_t

```
typedef oCpt::World::Location::gpsPoint_t oCpt::components::sensors::Gps::ReturnValue_t
```

Definition at line 15 of file Gps.h.

6.14.3 Constructor & Destructor Documentation

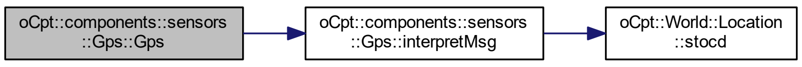
6.14.3.1 Gps()

```
oCpt::components::sensors::Gps::Gps (
            iController::ptr controller,
            World::ptr world,
            std::string id,
            std::string device,
            unsigned int baudrate )
```

Definition at line 13 of file Gps.cpp.

References interpretMsg(), and serial_.

Here is the call graph for this function:


oCpt::components::sensors ::Gps::Gps
oCpt::components::sensors ::Gps::interpretMsg
oCpt::World::Location ::stocd

6.14.3.2 ∼Gps()

```
oCpt::components::sensors::Gps::∼Gps ( )
```

Definition at line 28 of file Gps.cpp.

References serial_.

6.14.4 Member Function Documentation

6.14.4.1 interpretMsg()

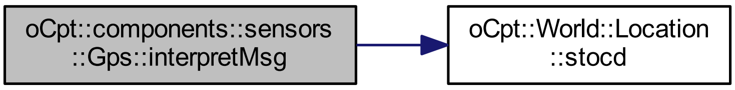
```
void oCpt::components::sensors::Gps::interpretMsg ( )  [protected]
```

Definition at line 59 of file Gps.cpp.

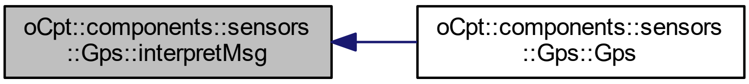
References oCpt::World::Location::coordinate::direction, oCpt::World::Location::gpsPoint::latitude, oCpt::World::↩ Location::gpsPoint::longitude, serial_, oCpt::iSensor::sig_, oCpt::iSensor::State::Stamp, oCpt::iSensor::state_, o↩ Cpt::World::Location::stocd(), oCpt::iSensor::State::Value, oCpt::World::Location::coordinate::value, and oCpt::i↩ Sensor::world_.

Referenced by Gps().

Here is the call graph for this function:


oCpt::components::sensors ::Gps::interpretMsg
oCpt::World::Location ::stocd

Here is the caller graph for this function:


oCpt::components::sensors ::Gps::interpretMsg
oCpt::components::sensors ::Gps::Gps

6.14.4.2 run()

```
void oCpt::components::sensors::Gps::run ( )  [virtual]
```

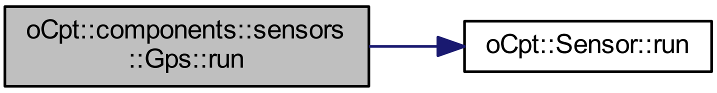
virtual function starting the run service for the IO

Reimplemented from oCpt::Sensor.

Definition at line 38 of file Gps.cpp.

References oCpt::Sensor::run(), oCpt::iSensor::sensorRunning_, and serial_.

Here is the call graph for this function:



6.14.4.3 setIOservice()

```
void oCpt::components::sensors::Gps::setIOservice (
             boost::shared_ptr< boost::asio::io_service > ioservice )  [virtual]
```

Setting the used Asynchronous Input Output service

Parameters

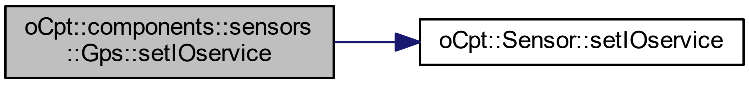
ioservice	ASIO IO service, which handles the async calls from multiple sensors

Reimplemented from oCpt::Sensor.

Definition at line 54 of file Gps.cpp.

References serial_, and oCpt::Sensor::setIOservice().

Here is the call graph for this function:


oCpt::components::sensors ::Gps::setIOservice
oCpt::Sensor::setIOservice

6.14.4.4 stop()

```
void oCpt::components::sensors::Gps::stop ( )  [virtual]
```

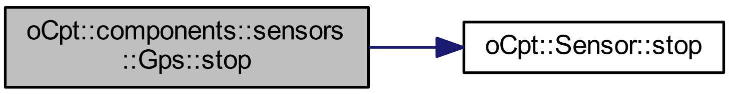
virtual function stopping the run

Reimplemented from oCpt::Sensor.

Definition at line 45 of file Gps.cpp.

References oCpt::iSensor::sensorRunning_, serial_, and oCpt::Sensor::stop().

Here is the call graph for this function:



6.14.4.5 updateSensor()

```
void oCpt::components::sensors::Gps::updateSensor ( )  [virtual]
```

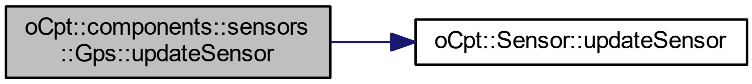
virtual function which performs a sensor update, obtaining a new value and sending a signal afterwards

Reimplemented from oCpt::Sensor.

Definition at line 34 of file Gps.cpp.

References oCpt::Sensor::updateSensor().

Here is the call graph for this function:


oCpt::components::sensors ::Gps::updateSensor
oCpt::Sensor::updateSensor

6.14.5 Member Data Documentation

6.14.5.1 device_

```
std::string oCpt::components::sensors::Gps::device_  [protected]
```

Definition at line 31 of file Gps.h.

6.14.5.2 serial_

```
protocol::Serial::ptr oCpt::components::sensors::Gps::serial_  [protected]
```

Definition at line 32 of file Gps.h.

Referenced by Gps(), interpretMsg(), run(), setIOservice(), stop(), and ∼Gps().

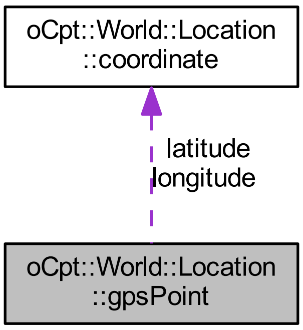
The documentation for this class was generated from the following files:

- include/Sensors/Gps.h
- src/Sensors/Gps.cpp

6.15 oCpt::World::Location::gpsPoint Struct Reference

```
#include <World.h>
```

Collaboration diagram for oCpt::World::Location::gpsPoint:


oCpt::World::Location ::coordinate
latitude
longitude
oCpt::World::Location ::gpsPoint

Public Member Functions

- std::string toString ()

Public Attributes

- coordinate_t longitude
- coordinate_t latitude
- double height

6.15.1 Detailed Description

Definition at line 126 of file World.h.

6.15.2 Member Function Documentation

6.15.2.1 toString()

```
std::string oCpt::World::Location::gpsPoint::toString ( )
```

Convert a gps coordinate to a text string

Returns

a text string eq. 5.000E,52.000N

Definition at line 72 of file World.cpp.

6.15.3 Member Data Documentation

6.15.3.1 height

```
double oCpt::World::Location::gpsPoint::height
```

Definition at line 129 of file World.h.

6.15.3.2 latitude

```
coordinate_t oCpt::World::Location::gpsPoint::latitude
```

Definition at line 128 of file World.h.

Referenced by oCpt::components::sensors::Gps::interpretMsg().

6.15.3.3 longitude

```
coordinate_t oCpt::World::Location::gpsPoint::longitude
```

Definition at line 127 of file World.h.

Referenced by oCpt::components::sensors::Gps::interpretMsg().

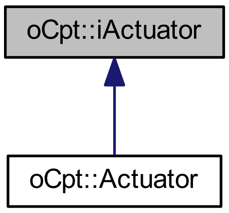
The documentation for this struct was generated from the following files:

- include/Core/World.h
- src/Core/World.cpp

6.16 oCpt::iActuator Class Reference

```
#include <Actuator.h>
```

Inheritance diagram for oCpt::iActuator:



Public Types

- typedef boost::shared_ptr< iActuator > ptr

Public Member Functions

- iActuator ()
- virtual ~iActuator ()
- virtual void setActuator ()=0
- virtual void run ()=0
- virtual void stop ()=0

6.16.1 Detailed Description

Definition at line 17 of file Actuator.h.

6.16.2 Member Typedef Documentation

6.16.2.1 ptr

```
typedef boost::shared_ptr<iActuator> oCpt::iActuator::ptr
```

Definition at line 19 of file Actuator.h.

6.16.3 Constructor & Destructor Documentation

6.16.3.1 iActuator()

```
oCpt::iActuator::iActuator ( )
```

Definition at line 9 of file Actuator.cpp.

6.16.3.2 ∼iActuator()

```
oCpt::iActuator::∼iActuator ( )  [virtual]
```

Definition at line 13 of file Actuator.cpp.

6.16.4 Member Function Documentation

6.16.4.1 run()

```
virtual void oCpt::iActuator::run ( )  [pure virtual]
```

Implemented in oCpt::Actuator.

6.16.4.2 setActuator()

```
virtual void oCpt::iActuator::setActuator ( )  [pure virtual]
```

Implemented in oCpt::Actuator.

6.16.4.3 stop()

```
virtual void oCpt::iActuator::stop ( )  [pure virtual]
```

Implemented in oCpt::Actuator.

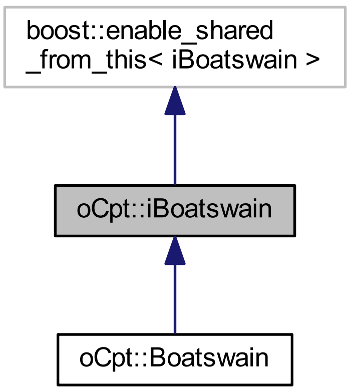
The documentation for this class was generated from the following files:

- include/Core/Actuator.h
- src/Core/Actuator.cpp

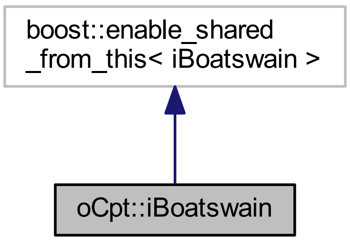
6.17 oCpt::iBoatswain Class Reference

```
#include <Boatswain.h>
```

Inheritance diagram for oCpt::iBoatswain:



Collaboration diagram for oCpt::iBoatswain:



Public Types

- typedef boost::shared_ptr< iBoatswain > ptr
- typedef boost::shared_ptr< boost::asio::deadline_timer > timerPtr

Public Member Functions

- iBoatswain (iController::ptr controller)
- virtual ∼iBoatswain ()
- virtual void run ()=0
- virtual void stop ()=0
- virtual void initialize ()=0
- virtual void registerSensor (iSensor::ptr sensor)=0
- virtual void registerActuator (iActuator::ptr actuator)=0
- virtual void registerComm (iComm::ptr comm)=0
- const boost::shared_ptr< bool > & getStopThread () const
- void setStopThread (const boost::shared_ptr< bool > &stopThread)
- boost::shared_ptr< boost::asio::io_service > & getIOservice ()

Protected Member Functions

- virtual void resetTimer (iSensor::ptr sensor)=0

Protected Attributes

- boost::shared_ptr< boost::asio::io_service > ioservice_
- iController::ptr controller_
- std::vector< timerPtr > timers_
- std::vector< iSensor::ptr > timerSensors_
- std::vector< iSensor::ptr > manualSensors_
- boost::shared_ptr< bool > stopThread_
- boost::shared_ptr< bool > localStopThread_

6.17.1 Detailed Description

The Boatswain performs all the labours tasks, suchs updateing and interpretting sensor readings, setting actuators according to the Captain wishes, updating the state representation of the vessel in the World. Each Boatswain runs on its own thread. It is possible for a vessel to have multiple Boatswains, responsible for multiple tasks, such as communication, localization, steering. Each Boatswain has to adhere to the iBoatswain interface.

Definition at line 27 of file Boatswain.h.

6.17.2 Member Typedef Documentation

6.17.2.1 ptr

```
typedef boost::shared_ptr<iBoatswain> oCpt::iBoatswain::ptr
```

Definition at line 29 of file Boatswain.h.

6.17.2.2 timerPtr

```
typedef boost::shared_ptr<boost::asio::deadline_timer> oCpt::iBoatswain::timerPtr
```

Definition at line 30 of file Boatswain.h.

6.17.3 Constructor & Destructor Documentation

6.17.3.1 iBoatswain()

```
oCpt::iBoatswain::iBoatswain (
            iController::ptr controller )
```

Constructor for a iBoatswain

Parameters

controller	a shared_ptr to the controller with which teh Boatswain interacts

Definition at line 7 of file Boatswain.cpp.

References ioservice_, and localStopThread_.

6.17.3.2 ∼iBoatswain()

```
oCpt::iBoatswain::∼iBoatswain ( )  [virtual]
```

Deconstructor for the iBoatswain

Definition at line 13 of file Boatswain.cpp.

6.17.4 Member Function Documentation

6.17.4.1 getIOservice()

```
boost::shared_ptr< boost::asio::io_service > & oCpt::iBoatswain::getIOservice ( )
```

get the used Input Output service

Returns

a shared_ptr to the ASIO io service

Definition at line 25 of file Boatswain.cpp.

References ioservice_.

6.17.4.2 getStopThread()

```
const boost::shared_ptr< bool > & oCpt::iBoatswain::getStopThread ( ) const
```

get if the thread is stopped

Returns

returns if the thread should stop

Definition at line 17 of file Boatswain.cpp.

References stopThread_.

6.17.4.3 initialize()

```
virtual void oCpt::iBoatswain::initialize ( )  [pure virtual]
```

pure virtual function of initialzing the Boatswain

Implemented in oCpt::Boatswain.

6.17.4.4 registerActuator()

```
virtual void oCpt::iBoatswain::registerActuator (
             iActuator::ptr actuator )  [pure virtual]
```

Pure virtual function for registering a new actuator with the Boatswain

Parameters

actuator	a shared_ptr to an Actuator which need to be maintained by the Boatswain

Implemented in oCpt::Boatswain.

6.17.4.5 registerComm()

```
virtual void oCpt::iBoatswain::registerComm (
             iComm::ptr comm )  [pure virtual]
```

Pure virtual function for registering a new communication device which

Parameters

comm	

Implemented in oCpt::Boatswain.

6.17.4.6 registerSensor()

```
virtual void oCpt::iBoatswain::registerSensor (
             iSensor::ptr sensor )  [pure virtual]
```

Pure virtual function for registering a new sensor with the Boatswain

Parameters

sensor	a shared_ptr to a Sensor which need to maintained by the Boatswain

Implemented in oCpt::Boatswain.

6.17.4.7 resetTimer()

```
virtual void oCpt::iBoatswain::resetTimer (
             iSensor::ptr sensor )  [protected], [pure virtual]
```

Pure virtual function for resetting the timer

Parameters

sensor	

Implemented in oCpt::Boatswain.

6.17.4.8 run()

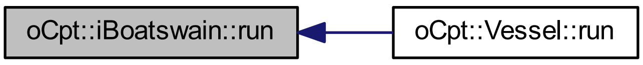
```
virtual void oCpt::iBoatswain::run ( ) [pure virtual]
```

pure virtual function for running the boatswain and his registered sensors

Implemented in oCpt::Boatswain.

Referenced by oCpt::Vessel::run().

Here is the caller graph for this function:



6.17.4.9 setStopThread()

```
void oCpt::iBoatswain::setStopThread (
            const boost::shared_ptr< bool > & stopThread )
```

set the value of the stopthread

Parameters

stopThread	//<! a shared_ptr to the boolean

Definition at line 21 of file Boatswain.cpp.

References stopThread_.

6.17.4.10 stop()

```
virtual void oCpt::iBoatswain::stop ( ) [pure virtual]
```

pure virtual function for stopping the run task

Implemented in oCpt::Boatswain.

6.17.5 Member Data Documentation

6.17.5.1 controller_

```
iController::ptr oCpt::iBoatswain::controller_ [protected]
```

Definition at line 96 of file Boatswain.h.

6.17.5.2 ioservice_

```
boost::shared_ptr<boost::asio::io_service> oCpt::iBoatswain::ioservice_  [protected]
```

Definition at line 95 of file Boatswain.h.

Referenced by getIOservice(), iBoatswain(), oCpt::Boatswain::registerComm(), oCpt::Boatswain::registerSensor(), oCpt::Boatswain::resetTimer(), and oCpt::Boatswain::run().

6.17.5.3 localStopThread_

```
boost::shared_ptr<bool> oCpt::iBoatswain::localStopThread_  [protected]
```

Definition at line 101 of file Boatswain.h.

Referenced by iBoatswain(), oCpt::Boatswain::resetTimer(), and oCpt::Boatswain::stop().

6.17.5.4 manualSensors_

```
std::vector<iSensor::ptr> oCpt::iBoatswain::manualSensors_  [protected]
```

Definition at line 99 of file Boatswain.h.

Referenced by oCpt::Boatswain::registerSensor(), and oCpt::Boatswain::run().

6.17.5.5 stopThread_

```
boost::shared_ptr<bool> oCpt::iBoatswain::stopThread_  [protected]
```

Definition at line 100 of file Boatswain.h.

Referenced by getStopThread(), oCpt::Boatswain::resetTimer(), and setStopThread().

6.17.5.6 timers_

```
std::vector<timerPtr> oCpt::iBoatswain::timers_  [protected]
```

Definition at line 97 of file Boatswain.h.

Referenced by oCpt::Boatswain::registerSensor(), and oCpt::Boatswain::resetTimer().

6.17.5.7 timerSensors_

```
std::vector<iSensor::ptr> oCpt::iBoatswain::timerSensors_  [protected]
```

Definition at line 98 of file Boatswain.h.

Referenced by oCpt::Boatswain::registerSensor(), and oCpt::Boatswain::resetTimer().

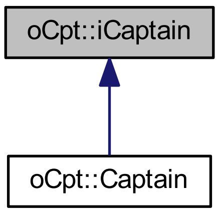
The documentation for this class was generated from the following files:

- include/Core/Boatswain.h
- src/Core/Boatswain.cpp

6.18 oCpt::iCaptain Class Reference

```
#include <Captain.h>
```

Inheritance diagram for oCpt::iCaptain:



Public Types

- typedef boost::shared_ptr< iCaptain > ptr

Public Member Functions

- iCaptain (World::ptr world)
- virtual ∼iCaptain ()
- virtual void run ()=0
- virtual void stop ()=0
- virtual void initialize ()=0
- const boost::shared_ptr< bool > & getStopThread_ () const
- void setStopThread_ (const boost::shared_ptr< bool > &stopThread_)

Protected Attributes

- boost::shared_ptr< bool > stopThread_
- boost::shared_ptr< bool > localStopThread_
- World::ptr world_

6.18.1 Detailed Description

Definition at line 12 of file Captain.h.

6.18.2 Member Typedef Documentation

6.18.2.1 ptr

```
typedef boost::shared_ptr<iCaptain> oCpt::iCaptain::ptr
```

Definition at line 14 of file Captain.h.

6.18.3 Constructor & Destructor Documentation

6.18.3.1 iCaptain()

```
oCpt::iCaptain::iCaptain (
            World::ptr world )
```

Definition at line 31 of file Captain.cpp.

6.18.3.2 ∼iCaptain()

```
oCpt::iCaptain::∼iCaptain ( )  [virtual]
```

Definition at line 35 of file Captain.cpp.

6.18.4 Member Function Documentation

6.18.4.1 getStopThread_()

```
const boost::shared_ptr< bool > & oCpt::iCaptain::getStopThread_ ( ) const
```

Definition at line 39 of file Captain.cpp.

References stopThread_.

6.18.4.2 initialize()

```
virtual void oCpt::iCaptain::initialize ( )  [pure virtual]
```

Implemented in oCpt::Captain.

6.18.4.3 run()

```
virtual void oCpt::iCaptain::run ( )  [pure virtual]
```

Implemented in oCpt::Captain.

6.18.4.4 setStopThread_()

```
void oCpt::iCaptain::setStopThread_ (
            const boost::shared_ptr< bool > & stopThread_ )
```

Definition at line 43 of file Captain.cpp.

References stopThread_.

6.18.4.5 stop()

```
virtual void oCpt::iCaptain::stop ( )  [pure virtual]
```

Implemented in oCpt::Captain.

6.18.5 Member Data Documentation

6.18.5.1 localStopThread_

```
boost::shared_ptr<bool> oCpt::iCaptain::localStopThread_  [protected]
```

Definition at line 32 of file Captain.h.

Referenced by oCpt::Captain::Captain(), oCpt::Captain::run(), and oCpt::Captain::stop().

6.18.5.2 stopThread_

```
boost::shared_ptr<bool> oCpt::iCaptain::stopThread_  [protected]
```

Definition at line 31 of file Captain.h.

Referenced by getStopThread_(), oCpt::Captain::run(), and setStopThread_().

6.18.5.3 world_

```
World::ptr oCpt::iCaptain::world_  [protected]
```

Definition at line 33 of file Captain.h.

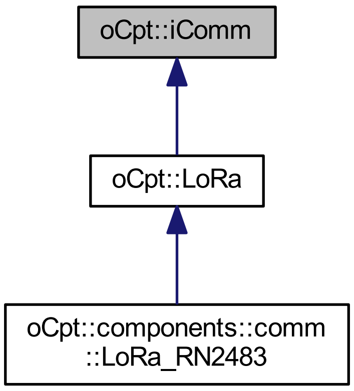
The documentation for this class was generated from the following files:

- include/Core/Captain.h
- src/Core/Captain.cpp

6.19 oCpt::iComm Class Reference

```
#include <Communication.h>
```

Inheritance diagram for oCpt::iComm:



Classes

- struct Message

Public Types

- typedef boost::shared_ptr< iComm > ptr
- typedef boost::signals2::signal< void()> signal_t

Public Member Functions

- iComm (const std::string &id, const std::string &device, World::ptr world=World::ptr(new World()), i↩
Controller::io_t ioservice=iController::io_t(new boost::asio::io_service()))
- virtual ∼iComm ()
- virtual void run ()=0
- virtual void stop ()=0
- virtual void initialize ()=0
- virtual void sendMessage (Message msg)=0
- virtual Message::ptr recieveMessage ()=0
- virtual void recieveAsyncMessage ()=0
- const std::string & getId () const
- void setId (const std::string &id)
- const std::string & getTypeOfComm () const
- void setTypeOfComm (const std::string &typeOfComm)
- Message::ptr readFiFoMsg ()
- std::deque< Message::ptr > ∗ getMsgQueue ()
- void setIoservice (const iController::io_t &ioservice)

Public Attributes

- signal_t msgRecievedSig

Protected Attributes

- std::string id_
- std::string typeOfComm_
- std::string device_
- boost::posix_time::milliseconds timer_
- std::deque< Message::ptr > msgQueue_
- iController::io_t ioservice_
- World::ptr world_

6.19.1 Detailed Description

The interface for communication devices

Definition at line 26 of file Communication.h.

6.19.2 Member Typedef Documentation

6.19.2.1 ptr

```
typedef boost::shared_ptr<iComm> oCpt::iComm::ptr
```

Definition at line 28 of file Communication.h.

6.19.2.2 signal_t

```
typedef boost::signals2::signal<void()> oCpt::iComm::signal_t
```

Definition at line 29 of file Communication.h.

6.19.3 Constructor & Destructor Documentation

6.19.3.1 iComm()

```
oCpt::iComm::iComm (
            const std::string & id,
            const std::string & device,
            World::ptr world = World::ptr(new World()),
            iController::io_t ioservice = iController::io_t(new boost::asio::io_service()) )
```

The constructor for the communication interface

Parameters

id	The ID of the communication device
device	The device path eq. /dev/ttyS0
world	A shared_ptr to the world with a default to a newly created one
ioservice	A shared_ptr to an ASIO Input Output service with a newly created one as default

Definition at line 12 of file Communication.cpp.

6.19.3.2 ∼iComm()

```
oCpt::iComm::∼iComm ( )  [virtual]
```

The deconstructor

Definition at line 21 of file Communication.cpp.

6.19.4 Member Function Documentation

6.19.4.1 getId()

```
const std::string & oCpt::iComm::getId ( ) const
```

Returns the ID of the communication device

Returns

a string with the ID

Definition at line 25 of file Communication.cpp.

References id_.

6.19.4.2 getMsgQueue()

```
std::deque< iComm::Message::ptr > * oCpt::iComm::getMsgQueue ( )
```

A que with received Message::ptr

Returns

a pointer to the Message que

Definition at line 54 of file Communication.cpp.

References msgQueue_.

6.19.4.3 getTypeOfComm()

```
const std::string & oCpt::iComm::getTypeOfComm ( ) const
```

Get the type of communication device

Returns

a string with type of device eq. modem, serial, LoRa, WiFi

Definition at line 33 of file Communication.cpp.

References typeOfComm_.

6.19.4.4 initialize()

```
virtual void oCpt::iComm::initialize ( ) [pure virtual]
```

A pure virtual function which initializes the communication device

Implemented in oCpt::LoRa.

6.19.4.5 readFiFoMsg()

```
iComm::Message::ptr oCpt::iComm::readFiFoMsg ( )
```

Get a pointer to the first message in Queu

Returns

Definition at line 41 of file Communication.cpp.

References msgQueue_.

6.19.4.6 recieveAsyncMessage()

```
virtual void oCpt::iComm::recieveAsyncMessage ( ) [pure virtual]
```

A pure virtual function which performs the polling for a new message on a seperate threads, so it won't block the current one, it needs to send a signal when the message is received

Implemented in oCpt::LoRa.

6.19.4.7 recieveMessage()

```
virtual Message::ptr oCpt::iComm::recieveMessage ( ) [pure virtual]
```

A pure virtual function with a shared_ptr to the first in queue received message, this function will hold the current thread

Returns

a shared_ptr pointing towards the queued Message

Implemented in oCpt::LoRa.

6.19.4.8 run()

```
virtual void oCpt::iComm::run ( ) [pure virtual]
```

a pure virtual function which runs the communication device

Implemented in oCpt::LoRa.

6.19.4.9 sendMessage()

```
virtual void oCpt::iComm::sendMessage (
             Message msg ) [pure virtual]
```

A pure virtual function which sends the message

Parameters

msg	the Message, consisting of a payload and a time stamp

Implemented in oCpt::LoRa.

6.19.4.10 setId()

```
void oCpt::iComm::setId (
             const std::string & id )
```

Set the ID of the communication device

Parameters

id	The ID of the communication device

Definition at line 29 of file Communication.cpp.

References id_.

6.19.4.11 setIoservice()

```
void oCpt::iComm::setIoservice (
             const iController::io_t & ioservice )
```

The ASIO Input Output service handling the messages

Parameters

ioservice	a shared_ptr to a IO service

Definition at line 50 of file Communication.cpp.

References ioservice_.

6.19.4.12 setTypeOfComm()

```
void oCpt::iComm::setTypeOfComm (
             const std::string & typeOfComm )
```

Set the type of communication device. eq. modem, serial, LoRa, WiFi

Parameters

typeOfComm	string representing the type of communication

Definition at line 37 of file Communication.cpp.

References typeOfComm_.

6.19.4.13 stop()

```
virtual void oCpt::iComm::stop ( )  [pure virtual]
```

A pure virtual function which stops the communication device

Implemented in oCpt::LoRa.

6.19.5 Member Data Documentation

6.19.5.1 device_

```
std::string oCpt::iComm::device_  [protected]
```

Definition at line 150 of file Communication.h.

6.19.5.2 id_

```
std::string oCpt::iComm::id_  [protected]
```

Definition at line 148 of file Communication.h.

Referenced by getId(), and setId().

6.19.5.3 ioservice_

```
iController::io_t oCpt::iComm::ioservice_  [protected]
```

Definition at line 153 of file Communication.h.

Referenced by setIoservice().

6.19.5.4 msgQueue_

```
std::deque<Message::ptr> oCpt::iComm::msgQueue_  [protected]
```

Definition at line 152 of file Communication.h.

Referenced by getMsgQueue(), oCpt::LoRa::messageRecieved(), and readFiFoMsg().

6.19.5.5 msgRecievedSig

```
signal_t oCpt::iComm::msgRecievedSig
```

A signal which is send when a new Message

Definition at line 146 of file Communication.h.

Referenced by oCpt::LoRa::messageRecieved().

6.19.5.6 timer_

```
boost::posix_time::milliseconds oCpt::iComm::timer_  [protected]
```

Definition at line 151 of file Communication.h.

6.19.5.7 typeOfComm_

```
std::string oCpt::iComm::typeOfComm_  [protected]
```

Definition at line 149 of file Communication.h.

Referenced by getTypeOfComm(), and setTypeOfComm().

6.19.5.8 world_

```
World::ptr oCpt::iComm::world_  [protected]
```

Definition at line 154 of file Communication.h.

Referenced by oCpt::LoRa::messageRecieved(), and oCpt::LoRa::sendMessage().

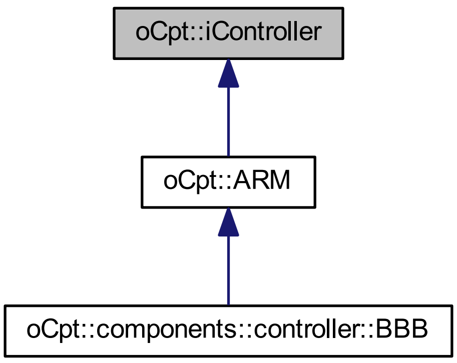
The documentation for this class was generated from the following files:

- include/Core/Communication.h
- src/Core/Communication.cpp

6.20 oCpt::iController Class Reference

```
#include <Controller.h>
```

Inheritance diagram for oCpt::iController:



Public Types

- typedef boost::shared_ptr< iController > ptr
- typedef boost::shared_ptr< boost::asio::io_service > io_t

Public Member Functions

- iController (World::ptr world)
- virtual ∼iController ()
- virtual std::vector< protocol::adc::ptr > ∗ getAdcVector ()=0
- virtual protocol::adc::ptr getADC (uint8_t id, uint8_t device)=0

Protected Attributes

- std::vector< protocol::adc::ptr > adcVector_
- World::ptr world_

6.20.1 Detailed Description

The interface for a controller. Each controller like for instance a Beaglebone black, Raspberry PI or x64 computer, should adhere to this interface

Definition at line 559 of file Controller.h.

6.20.2 Member Typedef Documentation

6.20.2.1 io_t

```
typedef boost::shared_ptr<boost::asio::io_service> oCpt::iController::io_t
```

Definition at line 562 of file Controller.h.

6.20.2.2 ptr

```
typedef boost::shared_ptr<iController> oCpt::iController::ptr
```

Definition at line 561 of file Controller.h.

6.20.3 Constructor & Destructor Documentation

6.20.3.1 iController()

```
oCpt::iController::iController (
            World::ptr world )
```

The constructor

Parameters

world	a pointer to the World

Definition at line 464 of file Controller.cpp.

6.20.3.2 ∼iController()

```
oCpt::iController::∼iController ( )  [virtual]
```

The deconstructor

Definition at line 467 of file Controller.cpp.

6.20.4 Member Function Documentation

6.20.4.1 getADC()

```
virtual protocol::adc::ptr oCpt::iController::getADC (
            uint8_t id,
            uint8_t device )  [pure virtual]
```

A pure virtual function which gets a Pointer to a specific ADC

Parameters

id	The pin ID
device	the device ID

Returns

a pointer to the requested ADC

Implemented in oCpt::ARM.

6.20.4.2 getAdcVector()

```
virtual std::vector<protocol::adc::ptr>* oCpt::iController::getAdcVector ( )  [pure virtual]
```

A pure virtual function which gets a pointer to all available ADC, if present. TODO check how it handles no ADC presents

Returns

a vector with pointers the all available ADCs

Implemented in oCpt::ARM.

6.20.5 Member Data Documentation

6.20.5.1 adcVector_

`std::vector<protocol::adc::ptr> oCpt::iController::adcVector_ [protected]`

Definition at line 590 of file Controller.h.

Referenced by oCpt::components::controller::BBB::BBB(), oCpt::ARM::getADC(), and oCpt::ARM::getAdcVector().

6.20.5.2 world_

`World::ptr oCpt::iController::world_ [protected]`

Definition at line 591 of file Controller.h.

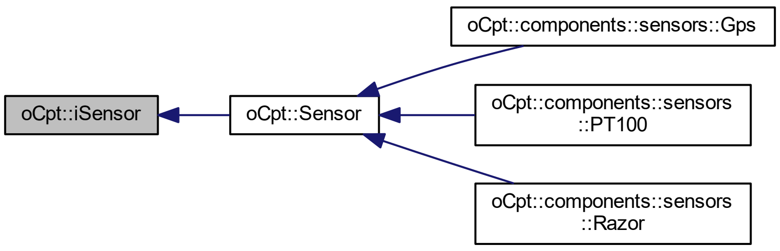
The documentation for this class was generated from the following files:

- include/Core/Controller.h
- src/Core/Controller.cpp

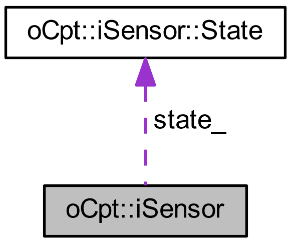
6.21 oCpt::iSensor Class Reference

`#include <Sensor.h>`

Inheritance diagram for oCpt::iSensor:


oCpt::components::sensors::Gps
oCpt::iSensor
oCpt::Sensor
oCpt::components::sensors ::PT100
oCpt::components::sensors ::Razor

Collaboration diagram for oCpt::iSensor:



Classes

- struct State

Public Types

- typedef boost::shared_ptr< iSensor > ptr
- typedef boost::signals2::signal< void()> signal_t
- typedef boost::any generic_t

Public Member Functions

- iSensor (iController::ptr controller, World::ptr world, std::string id, std::string typeOfSensor="")
- virtual ∼iSensor ()
- virtual void updateSensor ()=0
- virtual void run ()=0
- virtual void stop ()=0
- virtual void init ()=0
- virtual void setIOservice (boost::shared_ptr< boost::asio::io_service > ioservice)=0
- virtual bool operator== (iSensor::ptr rhs)
- const boost::posix_time::milliseconds & getTimer () const
- void setTimer (const boost::posix_time::milliseconds &timer)
- signal_t & getSig ()
- const State & getState () const
- const std::string & getID () const
- void setID (const std::string &id)
- const std::string & getTypeOfSensor () const
- void setTypeOfSensor (const std::string &typeOfSensor)

Protected Attributes

- std::string id_
- std::string typeOfSensor_
- iController::ptr controller_
- World::ptr world_
- boost::posix_time::milliseconds timer_
- signal_t sig_
- State state_
- bool sensorRunning_
- boost::shared_ptr< boost::asio::io_service > ioservice_

6.21.1 Detailed Description

Each sensor that is used should adhere to the sensor interface. A sensor consits of an connection to a controller, such as a ARM device and the world. The sensor needs to be initiated with the construct, where afterwards the init fucntion is called. The sensor should then be registerd by the Boatswain, using Boatswain::registerSensor(). This ensures that the boatswain can run the sensors. Some sensors are automatically update, whilst other need a manual action, such it is common practice to call the iSensor::updateSensor(). Once the value is update, a new Boost::Signal2 is fired, which allow for the main function to obtain the State of the sensor. Via iSensor::getState(). Since the return value of a sensor can vary, it is important to note that the final sensor should include a typedef with the return type named ReturnValue_t. After a sensor update is given or a signal is recieved, the return value can be CAST using the macro CAST(x,t)

Definition at line 29 of file Sensor.h.

6.21.2 Member Typedef Documentation

6.21.2.1 generic_t

```
typedef boost::any oCpt::iSensor::generic_t
```

Definition at line 33 of file Sensor.h.

6.21.2.2 ptr

```
typedef boost::shared_ptr<iSensor> oCpt::iSensor::ptr
```

Definition at line 31 of file Sensor.h.

6.21.2.3 signal_t

```
typedef boost::signals2::signal<void()> oCpt::iSensor::signal_t
```

Definition at line 32 of file Sensor.h.

6.21.3 Constructor & Destructor Documentation

6.21.3.1 iSensor()

```
oCpt::iSensor::iSensor (
            iController::ptr controller,
            World::ptr world,
            std::string id,
            std::string typeOfSensor = "" )
```

Constructor of iSensor

Parameters

controller	a shared_ptr of the controller where the sensor is hooked to
world	a shared_ptr of the world in which the vessel operates
id	a identifying name of the sensor
typeOfSensor	a identifying category for the sensor

Definition at line 10 of file Sensor.cpp.

References state_, and oCpt::iSensor::State::Value.

6.21.3.2 ∼iSensor()

```
oCpt::iSensor::∼iSensor ( ) [virtual]
```

Deconstructor of the sensor

Definition at line 18 of file Sensor.cpp.

6.21.4 Member Function Documentation

6.21.4.1 getID()

```
const std::string & oCpt::iSensor::getID ( ) const
```

get the current ID

Returns

returns the ID as string

Definition at line 41 of file Sensor.cpp.

References id_.

6.21.4.2 getSig()

```
iSensor::signal_t & oCpt::iSensor::getSig ( )
```

get the signal that is to be fired when the state is updated

Returns

the signal_t

Definition at line 28 of file Sensor.cpp.

References sig_.

6.21.4.3 getState()

```
const iSensor::State & oCpt::iSensor::getState ( ) const
```

gets the last State of the sensor

Returns

the State object. Remember to CAST the value like such <sensorClass>::ReturnValue_t ret = CA↩
ST(<sensorname>->getState().Value, <sensorClass>);

Definition at line 32 of file Sensor.cpp.

References state_.

6.21.4.4 getTimer()

```
const boost::posix_time::milliseconds & oCpt::iSensor::getTimer ( ) const
```

Get the number of milliseconds when this sensor should be updated

Returns

returns a boost::posix_time::milliseconds type

Definition at line 20 of file Sensor.cpp.

References timer_.

6.21.4.5 getTypeOfSensor()

```
const std::string & oCpt::iSensor::getTypeOfSensor ( ) const
```

get the type of sensor

Returns

category identifying string

Definition at line 49 of file Sensor.cpp.

References typeOfSensor_.

6.21.4.6 init()

```
virtual void oCpt::iSensor::init ( ) [pure virtual]
```

pure virtual function for initializing the sensor

Implemented in oCpt::Sensor, oCpt::components::sensors::Razor, and oCpt::components::sensors::PT100.

6.21.4.7 operator==()

```
bool oCpt::iSensor::operator== (
            iSensor::ptr rhs ) [virtual]
```

Equal operator determining if this sensor is equal with the pointer

Parameters

rhs	shared_ptr with the other sensor

Returns

returns either true or false

Definition at line 36 of file Sensor.cpp.

References controller_, id_, and typeOfSensor_.

6.21.4.8 run()

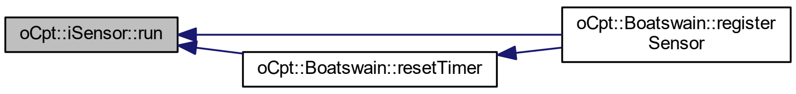
```
virtual void oCpt::iSensor::run ( )  [pure virtual]
```

pure virtual function for running of the sensor

Implemented in oCpt::Sensor, oCpt::components::sensors::Razor, oCpt::components::sensors::Gps, and oCpt↩
::components::sensors::PT100.

Referenced by oCpt::Boatswain::registerSensor(), and oCpt::Boatswain::resetTimer().

Here is the caller graph for this function:



6.21.4.9 setID()

```
void oCpt::iSensor::setID (
            const std::string & id )
```

sets the ID of the sensor

Parameters

id	identifying string

Definition at line 45 of file Sensor.cpp.

References id_.

6.21.4.10 setIOservice()

```
virtual void oCpt::iSensor::setIOservice (
            boost::shared_ptr< boost::asio::io_service > ioservice )  [pure virtual]
```

pure virtual function for registering the Input Output service

Parameters

ioservice	teh Input Output service used by Boost ASIO

Implemented in oCpt::Sensor, oCpt::components::sensors::Razor, and oCpt::components::sensors::Gps.

6.21.4.11 setTimer()

```
void oCpt::iSensor::setTimer (
             const boost::posix_time::milliseconds & timer )
```

set the number of milliseconds when this sensor should be updated

Parameters

timer	the number of milliseconds as an boost::posix_time::milliseconds type

Definition at line 24 of file Sensor.cpp.

References timer_.

6.21.4.12 setTypeOfSensor()

```
void oCpt::iSensor::setTypeOfSensor (
             const std::string & typeOfSensor )
```

sets the category of the sensor, suchs as GPS, temperature

Parameters

typeOfSensor	category identifying string

Definition at line 53 of file Sensor.cpp.

References typeOfSensor_.

6.21.4.13 stop()

```
virtual void oCpt::iSensor::stop ( )  [pure virtual]
```

pure virtual function for stopping the sensor

Implemented in oCpt::Sensor, oCpt::components::sensors::Razor, oCpt::components::sensors::Gps, and oCpt↩
::components::sensors::PT100.

6.21.4.14 updateSensor()

```
virtual void oCpt::iSensor::updateSensor ( )  [pure virtual]
```

pure virtual function for the updating of a sensor

Implemented in oCpt::Sensor, oCpt::components::sensors::Razor, oCpt::components::sensors::Gps, and oCpt↩
::components::sensors::PT100.

6.21.5 Member Data Documentation

6.21.5.1 controller_

```
iController::ptr oCpt::iSensor::controller_  [protected]
```

Definition at line 140 of file Sensor.h.

Referenced by operator==(), and oCpt::components::sensors::PT100::updateSensor().

6.21.5.2 id_

```
std::string oCpt::iSensor::id_  [protected]
```

Definition at line 112 of file Sensor.h.

Referenced by getID(), operator==(), and setID().

6.21.5.3 ioservice_

```
boost::shared_ptr<boost::asio::io_service> oCpt::iSensor::ioservice_  [protected]
```

Definition at line 146 of file Sensor.h.

Referenced by oCpt::Sensor::setIOservice().

6.21.5.4 sensorRunning_

```
bool oCpt::iSensor::sensorRunning_  [protected]
```

Definition at line 145 of file Sensor.h.

Referenced by oCpt::components::sensors::Gps::run(), oCpt::components::sensors::Razor::run(), oCpt::Sensor↩
::run(), oCpt::components::sensors::Gps::stop(), oCpt::components::sensors::Razor::stop(), and oCpt::Sensor↩
::stop().

6.21.5.5 sig_

```
signal_t oCpt::iSensor::sig_ [protected]
```

Definition at line 143 of file Sensor.h.

Referenced by getSig(), oCpt::components::sensors::Gps::interpretMsg(), oCpt::components::sensors::Razor↩
::msgHandler(), and oCpt::components::sensors::PT100::run().

6.21.5.6 state_

```
State oCpt::iSensor::state_ [protected]
```

Definition at line 144 of file Sensor.h.

Referenced by getState(), oCpt::components::sensors::Gps::interpretMsg(), iSensor(), oCpt::components↩
::sensors::Razor::msgHandler(), oCpt::components::sensors::Razor::Razor(), oCpt::components::sensors::P↩
T100::updateSensor(), and oCpt::components::sensors::Razor::updateSensor().

6.21.5.7 timer_

```
boost::posix_time::milliseconds oCpt::iSensor::timer_ [protected]
```

Definition at line 142 of file Sensor.h.

Referenced by getTimer(), and setTimer().

6.21.5.8 typeOfSensor_

```
std::string oCpt::iSensor::typeOfSensor_ [protected]
```

Definition at line 139 of file Sensor.h.

Referenced by getTypeOfSensor(), operator==(), and setTypeOfSensor().

6.21.5.9 world_

```
World::ptr oCpt::iSensor::world_ [protected]
```

Definition at line 141 of file Sensor.h.

Referenced by oCpt::components::sensors::Gps::interpretMsg(), oCpt::components::sensors::Razor::msg↩
Handler(), oCpt::components::sensors::Razor::Razor(), oCpt::components::sensors::PT100::updateSensor(), and
oCpt::components::sensors::Razor::updateSensor().

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

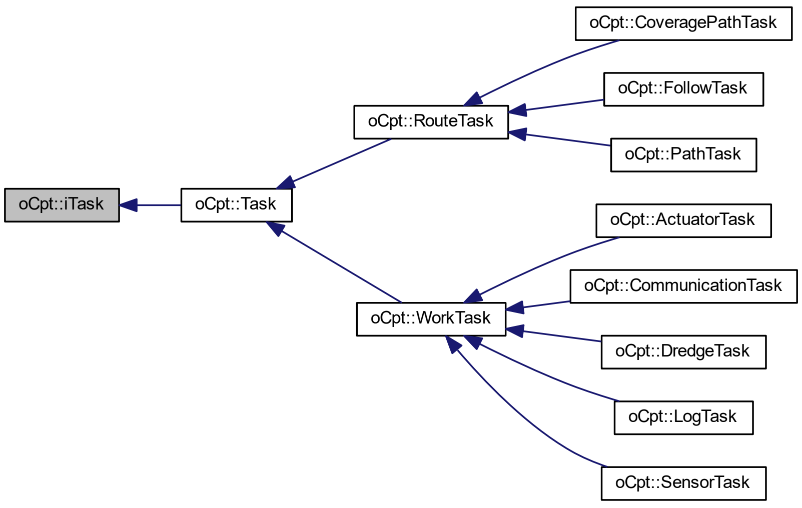
- include/Core/Sensor.h
- src/Core/Sensor.cpp

6.22 oCpt::iTask Class Reference

Task interface, all tasks need to adhere to this structure.

`#include <Task.h>`

Inheritance diagram for oCpt::iTask:



Classes

- class Status

Public Types

- enum TypeOf { ROUTE = 1, WORK = 2 }
- typedef boost::shared_ptr< iTask > ptr

    Boost shared_ptr to a task.
- typedef std::list< iTask::ptr > taskqueue

    A list of shared pointer tasks.

Public Member Functions

- iTask (iVessel::ptr vessel, bool concurrent=false)
- virtual ∼iTask ()
- virtual void start ()=0
- virtual iTask::Status::ptr status ()=0
- virtual void stop ()=0

Public Attributes

- taskqueue Work

Protected Attributes

- bool _concurrent = false

  Allowed to run as a seperate thread.
- Vessel::ptr _vessel = nullptr

  Pointer to the world.

6.22.1 Detailed Description

Task interface, all tasks need to adhere to this structure.

This interface make sure that all task adheres to the same runtime rules and enable run-time polymorphism

Definition at line 20 of file Task.h.

6.22.2 Member Typedef Documentation

6.22.2.1 ptr

```
typedef boost::shared_ptr<iTask> oCpt::iTask::ptr
```

Boost shared_ptr to a task.

Definition at line 22 of file Task.h.

6.22.2.2 taskqueue

```
typedef std::list<iTask::ptr> oCpt::iTask::taskqueue
```

A list of shared pointer tasks.

Definition at line 23 of file Task.h.

6.22.3 Member Enumeration Documentation

6.22.3.1 TypeOf

```
enum oCpt::iTask::TypeOf
```

Enumeration indicating which type of task the object is

Enumerator

ROUTE	
WORK	

Definition at line 72 of file Task.h.

6.22.4 Constructor & Destructor Documentation

6.22.4.1 iTask()

```
oCpt::iTask::iTask (
            iVessel::ptr vessel,
            bool concurrent = false )
```

Constructor of the interface

Returns

Definition at line 8 of file Task.cpp.

References _concurrent, and _vessel.

6.22.4.2 ∼iTask()

```
oCpt::iTask::∼iTask ( ) [virtual]
```

Deconstructor of the interface

Definition at line 13 of file Task.cpp.

6.22.5 Member Function Documentation

6.22.5.1 start()

```
virtual void oCpt::iTask::start ( ) [pure virtual]
```

The start command for a task

Implemented in oCpt::Task.

6.22.5.2 status()

```
virtual iTask::Status::ptr oCpt::iTask::status ( ) [pure virtual]
```

Retrieves the Status of a task

Returns

Boost shared_ptr of the task status

Implemented in oCpt::Task.

6.22.5.3 stop()

```
virtual void oCpt::iTask::stop ( )  [pure virtual]
```

The stop command for a task

Implemented in oCpt::Task.

6.22.6 Member Data Documentation

6.22.6.1 _concurrent

```
bool oCpt::iTask::_concurrent = false  [protected]
```

Allowed to run as a seperate thread.

Definition at line 104 of file Task.h.

Referenced by iTask().

6.22.6.2 _vessel

```
Vessel::ptr oCpt::iTask::_vessel = nullptr  [protected]
```

Pointer to the world.

Definition at line 105 of file Task.h.

Referenced by iTask().

6.22.6.3 Work

```
taskqueue oCpt::iTask::Work
```

Definition at line 25 of file Task.h.

Referenced by oCpt::Task::start().

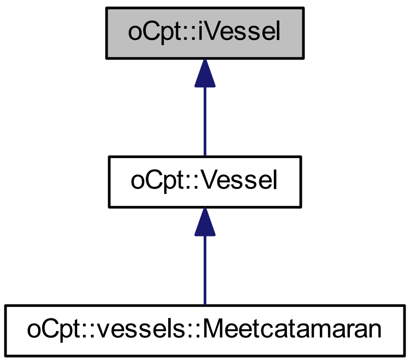
The documentation for this class was generated from the following files:

- include/Core/Task.h
- src/Core/Task.cpp

6.23 oCpt::iVessel Class Reference

```
#include <Vessel.h>
```

Inheritance diagram for oCpt::iVessel:



Public Types

- typedef boost::shared_ptr< iVessel > ptr

  Boost shared_ptr to a vessel.

Public Member Functions

- iVessel ()
- iVessel (iController::ptr controller)
- virtual ~iVessel ()
- virtual void initialize ()=0
- virtual void run ()=0
- virtual void stop ()=0
- const boost::shared_ptr< bool > & getStopThread () const
- void setStopThread (const boost::shared_ptr< bool > &stopThread)

Protected Attributes

- boost::shared_ptr< bool > stopThread_

  The global shared pointer for stopping all threads.

6.23.1 Detailed Description

The interface for each vessel

Definition at line 24 of file Vessel.h.

6.23.2 Member Typedef Documentation

6.23.2.1 ptr

```
typedef boost::shared_ptr<iVessel> oCpt::iVessel::ptr
```

Boost shared_ptr to a vessel.

Definition at line 26 of file Vessel.h.

6.23.3 Constructor & Destructor Documentation

6.23.3.1 iVessel() [1/2]

```
oCpt::iVessel::iVessel ( )
```

Constructor of the vessel interface

Returns

Definition at line 9 of file Vessel.cpp.

6.23.3.2 iVessel() [2/2]

```
oCpt::iVessel::iVessel (
              iController::ptr controller )
```

Constructor of the vessel interface

Parameters

controller	shared_ptr to the controller

Returns

Definition at line 11 of file Vessel.cpp.

6.23.3.3 ∼iVessel()

```
oCpt::iVessel::∼iVessel ( )  [virtual]
```

Deconstructor

Definition at line 13 of file Vessel.cpp.

6.23.4 Member Function Documentation

6.23.4.1 getStopThread()

```
const boost::shared_ptr< bool > & oCpt::iVessel::getStopThread ( ) const
```

Get the stop thread variable

Returns

shared_ptr for each each thread;

Definition at line 15 of file Vessel.cpp.

References stopThread_.

6.23.4.2 initialize()

```
virtual void oCpt::iVessel::initialize ( )  [pure virtual]
```

Initialize the vessel

Implemented in oCpt::Vessel.

6.23.4.3 run()

```
virtual void oCpt::iVessel::run ( )  [pure virtual]
```

Run the vessel normal operations

Implemented in oCpt::Vessel.

6.23.4.4 setStopThread()

```
void oCpt::iVessel::setStopThread (
             const boost::shared_ptr< bool > & stopThread )
```

Set the stop thread variable

Parameters

stopThread	a shared_ptr for all threads

Definition at line 19 of file Vessel.cpp.

References stopThread_.

6.23.4.5 stop()

```
virtual void oCpt::iVessel::stop ( )  [pure virtual]
```

Stop the vessel, everything except critical parts, which are needed to survive

Implemented in oCpt::Vessel.

6.23.5 Member Data Documentation

6.23.5.1 stopThread_

```
boost::shared_ptr<bool> oCpt::iVessel::stopThread_  [protected]
```

The global shared pointer for stopping all threads.

Definition at line 74 of file Vessel.h.

Referenced by getStopThread(), setStopThread(), oCpt::Vessel::stop(), and oCpt::Vessel::Vessel().

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

- include/Core/Vessel.h
- src/Core/Vessel.cpp

6.24 oCpt::World::Location Class Reference

```
#include <World.h>
```

Classes

- struct coordinate
- struct gpsPoint
- struct RoutePoint

Public Types

- enum cardinal_direction { North = 110, South = 115, East = 101, West = 119 }
- typedef struct oCpt::World::Location::coordinate coordinate_t
- typedef struct oCpt::World::Location::gpsPoint gpsPoint_t
- typedef boost::shared_ptr< Location > ptr

Public Member Functions

- Location ()
- virtual ∼Location ()
- RoutePoint::ptr getCurrentLocation (bool newMeasurement=false)
- void push_back (RoutePoint::ptr routePoint)
- std::vector< RoutePoint::ptr > getLocationHistory ()

Static Public Member Functions

- static cardinal_direction stocd (std::string str)

Private Attributes

- RoutePoint::ptr currentLocation_
- std::vector< RoutePoint::ptr > LocationHistory

6.24.1 Detailed Description

A location in the World

Definition at line 112 of file World.h.

6.24.2 Member Typedef Documentation

6.24.2.1 coordinate_t

```
typedef struct oCpt::World::Location::coordinate oCpt::World::Location::coordinate_t
```

6.24.2.2 gpsPoint_t

```
typedef struct oCpt::World::Location::gpsPoint oCpt::World::Location::gpsPoint_t
```

6.24.2.3 ptr

```
typedef boost::shared_ptr<Location> oCpt::World::Location::ptr
```

Definition at line 138 of file World.h.

6.24.3 Member Enumeration Documentation

6.24.3.1 cardinal_direction

```
enum oCpt::World::Location::cardinal_direction
```

Enumerator

North	enum value North
South	enum value South
East	enum value East
West	enum value West

Definition at line 114 of file World.h.

6.24.4 Constructor & Destructor Documentation

6.24.4.1 Location()

```
oCpt::World::Location::Location ( )
```

Constructor for the Location

Definition at line 37 of file World.cpp.

6.24.4.2 ∼Location()

```
oCpt::World::Location::∼Location ( )  [virtual]
```

Deconstruction for the Location

Definition at line 41 of file World.cpp.

6.24.5 Member Function Documentation

6.24.5.1 getCurrentLocation()

```
World::Location::RoutePoint::ptr oCpt::World::Location::getCurrentLocation (
            bool newMeasurement = false )
```

get the current Location

Parameters

newMeasurement	should a new measurement be executed? or is the lattest log sufficient

Returns

returns the last Way point

Definition at line 45 of file World.cpp.

6.24.5.2 getLocationHistory()

```
std::vector< World::Location::RoutePoint::ptr > oCpt::World::Location::getLocationHistory ( )
```

Get the complete location history

Returns

returns a vector with shared_ptr of all waypoints reached

Definition at line 53 of file World.cpp.

6.24.5.3 push_back()

```
void oCpt::World::Location::push_back (
            RoutePoint::ptr routePoint )
```

Add a new waypoint to the history log

Parameters

routePoint	a waypoint

Definition at line 49 of file World.cpp.

6.24.5.4 stocd()

```
World::Location::cardinal_direction oCpt::World::Location::stocd (
            std::string str )  [static]
```

Convert a string to a cardinal direction

Parameters

str	North/north/N/n / West,west,W,w / South,south,S,s / East,east,E,e are taken as argument

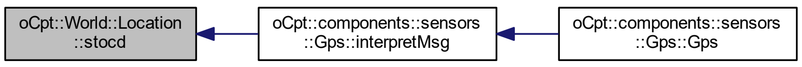
Returns

a cardinal direction

Definition at line 57 of file World.cpp.

Referenced by oCpt::components::sensors::Gps::interpretMsg().

Here is the caller graph for this function:


oCpt::World::Location ::stocd
oCpt::components::sensors ::Gps::interpretMsg
oCpt::components::sensors ::Gps::Gps

6.24.6 Member Data Documentation

6.24.6.1 currentLocation_

```
RoutePoint::ptr oCpt::World::Location::currentLocation_  [private]
```

Definition at line 183 of file World.h.

6.24.6.2 LocationHistory

```
std::vector<RoutePoint::ptr> oCpt::World::Location::LocationHistory  [private]
```

Definition at line 184 of file World.h.

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

- include/Core/World.h
- src/Core/World.cpp

6.25 oCpt::World::Time::Log< T > Class Template Reference

```
#include <World.h>
```

Public Types

- typedef boost::shared_ptr< Log > ptr

Public Member Functions

- Log ()
- Log (const T &value, const timepoint_t &epoch=clock_t::now())
- virtual ∼Log ()
- const timepoint_t & getEpoch () const
- const T & getValue () const

Private Attributes

- timepoint_t _epoch
- T _value

6.25.1 Detailed Description

template<typename T>
class oCpt::World::Time::Log< T >

A template class to Log generic values at an certain epoch in time

Template Parameters

T	Type of value to log

Definition at line 48 of file World.h.

6.25.2 Member Typedef Documentation

6.25.2.1 ptr

```
template<typename T >
typedef boost::shared_ptr<Log> oCpt::World::Time::Log< T >::ptr
```

Definition at line 50 of file World.h.

6.25.3 Constructor & Destructor Documentation

6.25.3.1 Log() [1/2]

```
template<typename T >
oCpt::World::Time::Log< T >::Log ( )  [inline]
```

Constructor of the Log class

Definition at line 59 of file World.h.

6.25.3.2 Log() [2/2]

```
template<typename T >
oCpt::World::Time::Log< T >::Log (
             const T & value,
             const timepoint_t & epoch = clock_t::now() )  [inline]
```

Constructor of the Log class

Parameters

value	The Value to store
epoch	the Time point, with a default to the now moment

Definition at line 66 of file World.h.

6.25.3.3 ∼Log()

```
template<typename T >
virtual oCpt::World::Time::Log< T >::∼Log ( )  [inline], [virtual]
```

Deconstructor of the Log class

Definition at line 74 of file World.h.

6.25.4 Member Function Documentation

6.25.4.1 getEpoch()

```
template<typename T >
const timepoint_t& oCpt::World::Time::Log< T >::getEpoch ( ) const  [inline]
```

Get the current Epoch

Returns

returns a time point when the Log has taken place

Definition at line 80 of file World.h.

References oCpt::World::Time::Log< T >::_epoch.

6.25.4.2 getValue()

```
template<typename T >
const T& oCpt::World::Time::Log< T >::getValue ( ) const  [inline]
```

Gets the current value

Returns

returns the value at an certain time

Definition at line 88 of file World.h.

References oCpt::World::Time::Log< T >::_value.

6.25.5 Member Data Documentation

6.25.5.1 _epoch

```
template<typename T >
timepoint_t oCpt::World::Time::Log< T >::_epoch  [private]
```

Definition at line 52 of file World.h.

Referenced by oCpt::World::Time::Log< T >::getEpoch().

6.25.5.2 _value

```
template<typename T >
T oCpt::World::Time::Log< T >::_value  [private]
```

Definition at line 53 of file World.h.

Referenced by oCpt::World::Time::Log< T >::getValue().

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

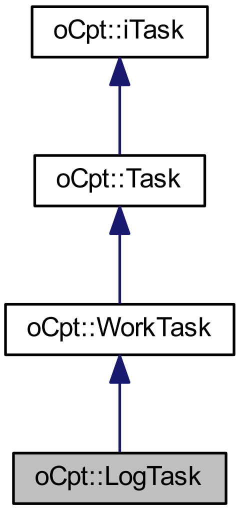
- include/Core/World.h

6.26 oCpt::LogTask Class Reference

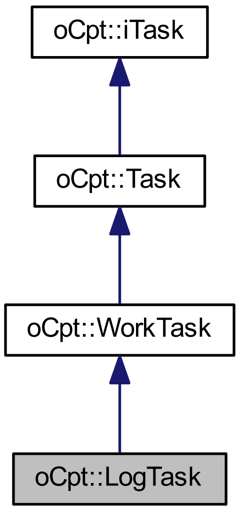
An Object representing a data logging task.

```
#include <Task.h>
```

Inheritance diagram for oCpt::LogTask:



Collaboration diagram for oCpt::LogTask:



Public Member Functions

- LogTask (Vessel::ptr vessel, bool concurrent=true)
- virtual ∼LogTask ()

Additional Inherited Members

6.26.1 Detailed Description

An Object representing a data logging task.

All these types of tasks make use of a sensor to record and log

Definition at line 265 of file Task.h.

6.26.2 Constructor & Destructor Documentation

6.26.2.1 LogTask()

```
oCpt::LogTask::LogTask (
            Vessel::ptr vessel,
            bool concurrent = true )
```

Constructor of the interface

Returns

Definition at line 65 of file Task.cpp.

6.26.2.2 ∼LogTask()

```
oCpt::LogTask::∼LogTask ( )  [virtual]
```

The deconstructor

Definition at line 67 of file Task.cpp.

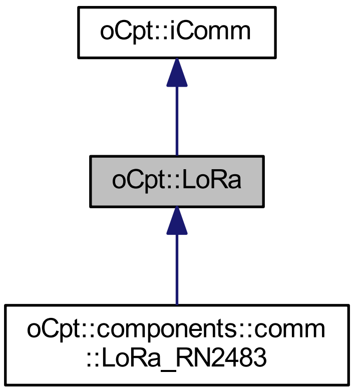
The documentation for this class was generated from the following files:

- include/Core/Task.h
- src/Core/Task.cpp

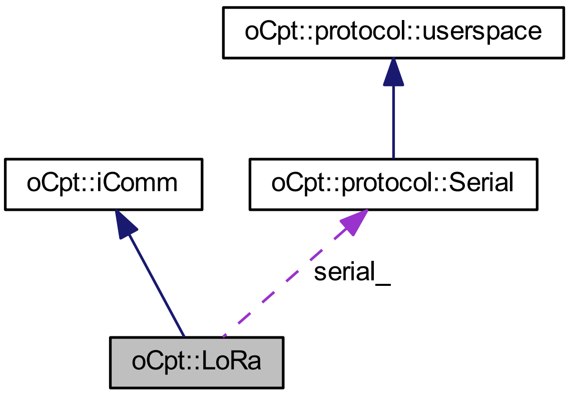
6.27 oCpt::LoRa Class Reference

```
#include <Communication.h>
```

Inheritance diagram for oCpt::LoRa:



Collaboration diagram for oCpt::LoRa:



Public Types

- enum ModulationMode { LORA = 1, FSK = 2 }
- enum SpreadingFactor {
  SF7 = 7, SF8 = 8, SF9 = 9, SF10 = 10,
  SF11 = 11, SF12 = 12 }

- enum BandWidth {
  BW250 = 1, BW200 = 2, BW166_7 = 3, BW125 = 4,
  BW100 = 5, BW83_3 = 6, BW62_5 = 7, BW50 = 8,
  BW41_7 = 9, BW31_3 = 10, BW25 = 11, BW20_8 = 12,
  BW15_6 = 13, BW12_5 = 14, BW10_4 = 15, BW7_8 = 16,
  BW6_3 = 17, BW5_2 = 18, BW3_9 = 19, BW3_1 = 20,
  BW2_6 = 21 }
- enum CodingRate { CR4_5 = 4, CR4_6 = 3, CR4_7 = 2, CR4_8 = 1 }
- enum RadioBandWidth { RBW500 = 500, RBW250 = 250, RBW125 = 125 }
- enum GetSet { GET, SET, NONE }
- enum RadioCommand {
  MOD, FREQ, PWR, SF,
  AFCBW, RXBW, FSKBITRATE, FDEV,
  PRLEN, CRC, CR, WDT,
  SYNC, BW, rRX, rTX }
- enum MacCommand {
  PAUSE, RESET, mTX, JOIN,
  SAVE, FORCEENABLE, RESUME }

Public Member Functions

- LoRa (const std::string &id, const std::string &device, World::ptr world=World::ptr(new World()), iController↩
  ::io_t ioservice=iController::io_t(new boost::asio::io_service()))
- virtual ∼LoRa ()
- virtual void run () override
- virtual void stop () override
- virtual void initialize () override
- virtual void sendMessage (Message msg) override
- virtual Message::ptr recieveMessage () override
- virtual void recieveAsyncMessage () override

Protected Member Functions

- void messageRecieved ()
- std::string bandWidthToString (const BandWidth &value)
- std::string codingRateToString (const CodingRate &value)
- void stringToHex (const std::string str, std::string &hexStr, const bool capital=true)
- void hexToString (const std::string hexStr, std::string &str)
- template<typename T >
  std::string encodeTypeToHex (T value, bool capital=true)
- template<typename T >
  std::string buildMacCmdString (MacCommand cmd, T value=0, GetSet prop=NONE)
- template<typename T >
  std::string buildRadioCmdString (RadioCommand cmd, T value=0, GetSet prop=SET)
- std::string buildRadioCmdString (RadioCommand cmd, std::string value, GetSet prop=SET)
- unsigned long calculateDownTime (unsigned int payload)
- void write (const std::string &value)
- void rx ()
- void macpause ()

Protected Attributes

- bool proceed_
- bool ignoreWarn_
- bool listen_
- protocol::Serial serial_

  SerialPort for UART communication with the chip.
- unsigned int baudrate_
- ModulationMode mod_

  Modulation mode.
- unsigned long freq_

  Frequency between 433050000..4347900000 or 863000000...870000000.
- int8_t pwr_

  Power of transmission between -3...15.
- SpreadingFactor sf_

  Spreading factor of the signal.
- BandWidth afcbw_

  Automatic frequency correction in kHz.
- BandWidth rxbw_

  Signal bandwidth in kHz.
- uint fskBitRate_

  FSK bitrate between 1...300000.
- uint fdev_

  Frequency deviation between 0...200000.
- uint prlen_

  Preamble length between 0...65535.
- bool crc_

  CRC Header on or off.
- CodingRate cr_

  The coding rate.
- unsigned long wdt_

  WatchDog 0...4294967295. Set to 0 to disable.
- unsigned int sync_

  Sync word.
- RadioBandWidth bw_

  RadioBandWidth in kHz.
- bool sendAllowed_

Additional Inherited Members

6.27.1 Detailed Description

Communication class for the LoRa protocol. The current class is mostly based on node 2 node communication TODO rewrite sho it will allow mesh network communication. Most of the commands are taken from `http↩://ww1.microchip.com/downloads/en/DeviceDoc/40001784B.pdf`

Definition at line 160 of file Communication.h.

6.27.2 Member Enumeration Documentation

6.27.2.1 BandWidth

`enum oCpt::LoRa::BandWidth`

The bandwidth

Enumerator

BW250	
BW200	
BW166↩ _7	
BW125	
BW100	
BW83_3	
BW62_5	
BW50	
BW41_7	
BW31_3	
BW25	
BW20_8	
BW15_6	
BW12_5	
BW10_4	
BW7_8	
BW6_3	
BW5_2	
BW3_9	
BW3_1	
BW2_6	

Definition at line 186 of file Communication.h.

6.27.2.2 CodingRate

`enum oCpt::LoRa::CodingRate`

The Coding rate of the signal

Enumerator

CR4↩ _5	
CR4↩ _6	
CR4↩ _7	
CR4↩ _8	

Definition at line 213 of file Communication.h.

6.27.2.3 GetSet

`enum oCpt::LoRa::GetSet`

Perform a get or a set command or otherwise none

Enumerator

GET	
SET	
NONE	

Definition at line 232 of file Communication.h.

6.27.2.4 MacCommand

`enum oCpt::LoRa::MacCommand`

Type to control the MAC layer, currently only pause is used, because node 2 node communication doesn't use MAC

Enumerator

PAUSE	
RESET	
mTX	
JOIN	
SAVE	
FORCEENABLE	
RESUME	

Definition at line 263 of file Communication.h.

6.27.2.5 ModulationMode

`enum oCpt::LoRa::ModulationMode`

The modulation mode of the LoRa module

Enumerator

LORA	
FSK	

Definition at line 166 of file Communication.h.

6.27.2.6 RadioBandWidth

`enum oCpt::LoRa::RadioBandWidth`

The radio bandwidth

Enumerator

RBW500	
RBW250	
RBW125	

Definition at line 223 of file Communication.h.

6.27.2.7 RadioCommand

```
enum oCpt::LoRa::RadioCommand
```

Types of radio commands

Enumerator

MOD	
FREQ	
PWR	
SF	
AFCBW	
RXBW	
FSKBITRATE	
FDEV	
PRLEN	
CRC	
CR	
WDT	
SYNC	
BW	
rRX	
rTX	

Definition at line 241 of file Communication.h.

6.27.2.8 SpreadingFactor

```
enum oCpt::LoRa::SpreadingFactor
```

The Spreading factor

Enumerator

SF7	
SF8	
SF9	
SF10	
SF11	
SF12	

Definition at line 174 of file Communication.h.

6.27.3 Constructor & Destructor Documentation

6.27.3.1 LoRa()

```
oCpt::LoRa::LoRa (
            const std::string & id,
            const std::string & device,
            World::ptr world = World::ptr(new World()),
            iController::io_t ioservice = iController::io_t(new boost::asio::io_service()) )
```

LoRa device constructor

Parameters

id	the ID of the device as an string
device	the device path eq. /dev/ttyS0
world	shared_ptr to the World default = a newly created World
ioservice	shared_ptr to an IO service, default is a newly created IO service

Definition at line 58 of file Communication.cpp.

6.27.3.2 ∼LoRa()

```
oCpt::LoRa::∼LoRa ( )  [virtual]
```

Definition at line 81 of file Communication.cpp.

6.27.4 Member Function Documentation

6.27.4.1 bandWidthToString()

```
std::string oCpt::LoRa::bandWidthToString (
            const BandWidth & value )  [protected]
```

Definition at line 101 of file Communication.cpp.

References BW100, BW10_4, BW125, BW12_5, BW15_6, BW166_7, BW200, BW20_8, BW25, BW250, BW31_3, BW3_1, BW3_9, BW41_7, BW50, BW5_2, BW62_5, BW6_3, BW7_8, and BW83_3.

6.27.4.2 buildMacCmdString()

```
template<typename T >
std::string oCpt::LoRa::buildMacCmdString (
            MacCommand cmd,
            T value = 0,
            GetSet prop = NONE )  [inline], [protected]
```

A command string builder for MAC commands currently onlu PAUSE implemented

Template Parameters

T	the type of MAC command eq. MacCommand::PAUSE

Parameters

cmd	the MacCommand to be performed
value	the Value to be send
prop	Additional properties

Returns

a string which can be send to the LoRa module eq. "mac set pause"

Definition at line 339 of file Communication.h.

6.27.4.3 buildRadioCmdString() [1/2]

```
template<typename T >
std::string oCpt::LoRa::buildRadioCmdString (
            RadioCommand cmd,
            T value = 0,
            GetSet prop = SET )  [inline], [protected]
```

A command string builder for radio commands

Template Parameters

T	

Parameters

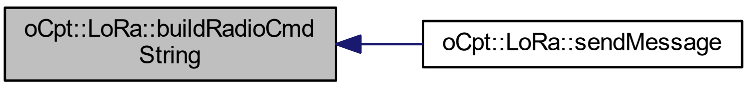
cmd	
value	
prop	

Returns

Definition at line 367 of file Communication.h.

Referenced by sendMessage().

Here is the caller graph for this function:



6.27.4.4 buildRadioCmdString() [2/2]

```
std::string oCpt::LoRa::buildRadioCmdString (
             RadioCommand cmd,
             std::string value,
             GetSet prop = SET )  [inline], [protected]
```

Definition at line 437 of file Communication.h.

6.27.4.5 calculateDownTime()

```
unsigned long oCpt::LoRa::calculateDownTime (
             unsigned int payload )  [protected]
```

Definition at line 85 of file Communication.cpp.

References bw_, cr_, crc_, prlen_, and sf_.

6.27.4.6 codingRateToString()

```
std::string oCpt::LoRa::codingRateToString (
             const CodingRate & value )  [protected]
```

Definition at line 148 of file Communication.cpp.

References CR4_5, CR4_6, and CR4_7.

6.27.4.7 encodeTypeToHex()

```
template<typename T >
std::string oCpt::LoRa::encodeTypeToHex (
             T value,
             bool capital = true )  [inline], [protected]
```

Convert the value of a Type T to a hexidecimal string, which can be send to a LoRa device, such that it can be transmitted

Template Parameters

T	the type of value, to be converted

Parameters

value	the to be converted value
capital	boolean indicating if the hexidecimal string should consist of capital letters

Returns

a string with the value as hexidecimal values

Definition at line 316 of file Communication.h.

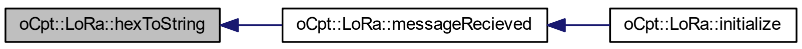
6.27.4.8 hexToString()

```
void oCpt::LoRa::hexToString (
             const std::string hexStr,
             std::string & str )  [protected]
```

Definition at line 280 of file Communication.cpp.

Referenced by messageRecieved().

Here is the caller graph for this function:



6.27.4.9 initialize()

```
void oCpt::LoRa::initialize ( )  [override], [virtual]
```

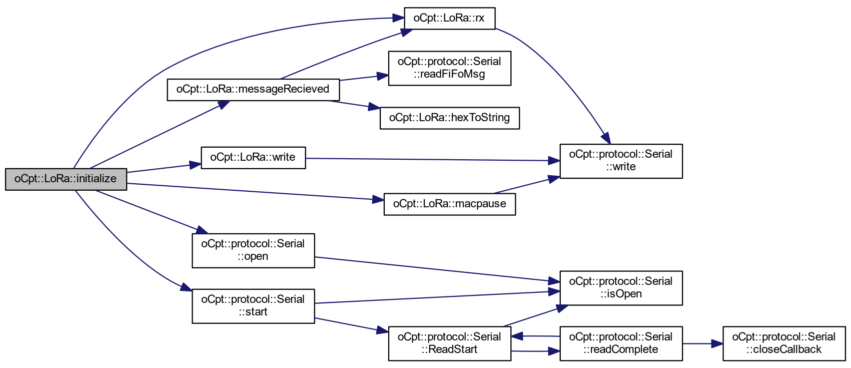
A pure virtual function which initializes the communication device

Implements oCpt::iComm.

Definition at line 189 of file Communication.cpp.

References AFCBW, afcbw_, BW, bw_, CR, cr_, CRC, crc_, FDEV, fdev_, FREQ, freq_, ignoreWarn_, listen_↩
, macpause(), messageRecieved(), MOD, mod_, oCpt::protocol::Serial::msgRecievedSig, oCpt::protocol::Serial↩
::open(), PRLEN, prlen_, PWR, pwr_, rx(), RXBW, rxbw_, serial_, SF, sf_, oCpt::protocol::Serial::start(), SYNC, sync_, WDT, wdt_, and write().

Here is the call graph for this function:


oCpt::LoRa::rx
oCpt::protocol::Serial ::readFiFoMsg
oCpt::LoRa::messageRecieved
oCpt::LoRa::hexToString
oCpt::LoRa::write
oCpt::protocol::Serial ::write
oCpt::LoRa::initialize
oCpt::LoRa::macpause
oCpt::protocol::Serial ::open
oCpt::protocol::Serial ::isOpen
oCpt::protocol::Serial ::start
oCpt::protocol::Serial ::ReadStart
oCpt::protocol::Serial ::readComplete
oCpt::protocol::Serial ::closeCallback

6.27.4.10 macpause()

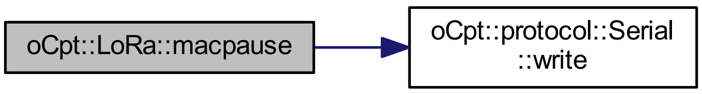
```
void oCpt::LoRa::macpause ( )  [protected]
```

Definition at line 275 of file Communication.cpp.

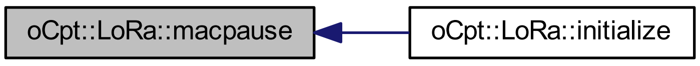
References PAUSE, serial_, and oCpt::protocol::Serial::write().

Referenced by initialize().

Here is the call graph for this function:



Here is the caller graph for this function:



6.27.4.11 messageRecieved()

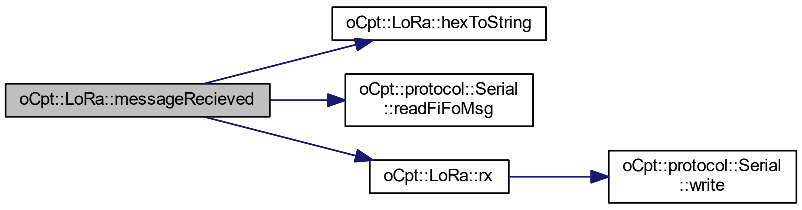
```
void oCpt::LoRa::messageRecieved ( )  [protected]
```

Definition at line 161 of file Communication.cpp.

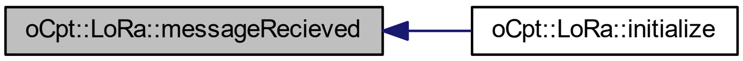
References hexToString(), ignoreWarn_, listen_, oCpt::iComm::msgQueue_, oCpt::iComm::msgRecievedSig, proceed_, oCpt::protocol::Serial::readFiFoMsg(), rx(), serial_, and oCpt::iComm::world_.

Referenced by initialize().

Here is the call graph for this function:


oCpt::LoRa::hexToString
oCpt::LoRa::messageRecieved
oCpt::protocol::Serial ::readFiFoMsg
oCpt::LoRa::rx
oCpt::protocol::Serial ::write

Here is the caller graph for this function:



6.27.4.12 recieveAsyncMessage()

```
void oCpt::LoRa::recieveAsyncMessage ( )  [override], [virtual]
```

A pure virtual function which performs the polling for a new message on a seperate threads, so it won't block the current one, it needs to send a signal when the message is received

Implements oCpt::iComm.

Definition at line 241 of file Communication.cpp.

6.27.4.13 recieveMessage()

`iComm::Message::ptr oCpt::LoRa::recieveMessage ( ) [override], [virtual]`

A pure virtual function with a shared_ptr to the first in queue received message, this function will hold the current thread

Returns

a shared_ptr pointing towards the queued Message

Implements oCpt::iComm.

Definition at line 237 of file Communication.cpp.

6.27.4.14 run()

`void oCpt::LoRa::run ( ) [override], [virtual]`

a pure virtual function which runs the communication device

Implements oCpt::iComm.

Definition at line 181 of file Communication.cpp.

6.27.4.15 rx()

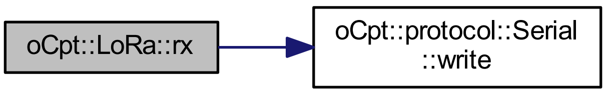
`void oCpt::LoRa::rx ( ) [protected]`

Definition at line 270 of file Communication.cpp.

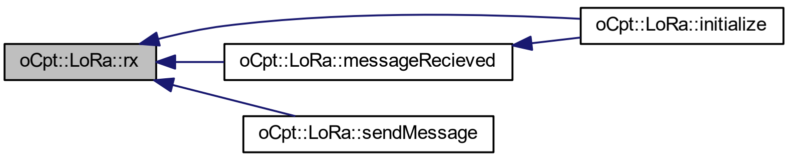
References NONE, rRX, serial_, and oCpt::protocol::Serial::write().

Referenced by initialize(), messageRecieved(), and sendMessage().

Here is the call graph for this function:



Here is the caller graph for this function:



6.27.4.16 sendMessage()

```
void oCpt::LoRa::sendMessage (
            Message msg )  [override], [virtual]
```

A pure virtual function which sends the message

Parameters

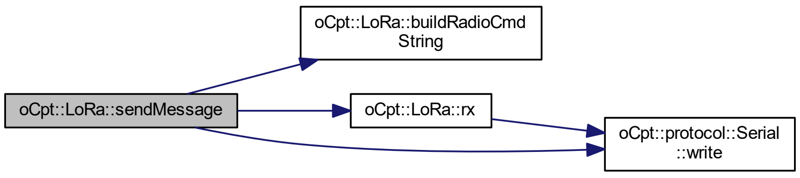
msg	the Message, consisting of a payload and a time stamp

Implements oCpt::iComm.

Definition at line 222 of file Communication.cpp.

References buildRadioCmdString(), NONE, PAUSE, oCpt::iComm::Message::Payload, rTX, rx(), sendAllowed_↩
, serial_, oCpt::iComm::Message::Stamp, oCpt::iComm::world_, and oCpt::protocol::Serial::write().

Here is the call graph for this function:


oCpt::LoRa::buildRadioCmd String
oCpt::LoRa::sendMessage
oCpt::LoRa::rx
oCpt::protocol::Serial ::write

6.27.4.17 stop()

```
void oCpt::LoRa::stop ( )  [override], [virtual]
```

A pure virtual function which stops the communication device

Implements oCpt::iComm.

Definition at line 185 of file Communication.cpp.

6.27.4.18 stringToHex()

```
void oCpt::LoRa::stringToHex (
            const std::string str,
            std::string & hexStr,
            const bool capital = true )  [protected]
```

Definition at line 245 of file Communication.cpp.

6.27.4.19 write()

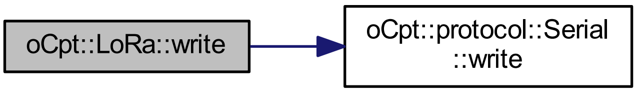
```
void oCpt::LoRa::write (
             const std::string & value )  [protected]
```

Definition at line 262 of file Communication.cpp.

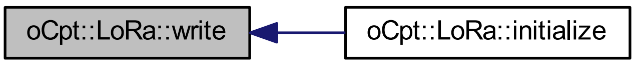
References proceed_, serial_, and oCpt::protocol::Serial::write().

Referenced by initialize().

Here is the call graph for this function:



Here is the caller graph for this function:



6.27.5 Member Data Documentation

6.27.5.1 afcbw_

```
BandWidth oCpt::LoRa::afcbw_  [protected]
```

Automatic frequency correction in kHz.

Definition at line 470 of file Communication.h.

Referenced by initialize().

6.27.5.2 baudrate_

```
unsigned int oCpt::LoRa::baudrate_  [protected]
```

Definition at line 465 of file Communication.h.

6.27.5.3 bw_

```
RadioBandWidth oCpt::LoRa::bw_  [protected]
```

RadioBandWidth in kHz.

Definition at line 479 of file Communication.h.

Referenced by calculateDownTime(), and initialize().

6.27.5.4 cr_

```
CodingRate oCpt::LoRa::cr_  [protected]
```

The coding rate.

Definition at line 476 of file Communication.h.

Referenced by calculateDownTime(), and initialize().

6.27.5.5 crc_

```
bool oCpt::LoRa::crc_  [protected]
```

CRC Header on or off.

Definition at line 475 of file Communication.h.

Referenced by calculateDownTime(), and initialize().

6.27.5.6 fdev_

```
uint oCpt::LoRa::fdev_  [protected]
```

Frequency deviation between 0...200000.

Definition at line 473 of file Communication.h.

Referenced by initialize().

6.27.5.7 freq_

```
unsigned long oCpt::LoRa::freq_  [protected]
```

Frequency between 433050000..4347900000 or 863000000...870000000.

Definition at line 467 of file Communication.h.

Referenced by initialize().

6.27.5.8 fskBitRate_

```
uint oCpt::LoRa::fskBitRate_ [protected]
```

FSK bitrate between 1...300000.

Definition at line 472 of file Communication.h.

6.27.5.9 ignoreWarn_

```
bool oCpt::LoRa::ignoreWarn_ [protected]
```

Definition at line 462 of file Communication.h.

Referenced by initialize(), and messageRecieved().

6.27.5.10 listen_

```
bool oCpt::LoRa::listen_ [protected]
```

Definition at line 463 of file Communication.h.

Referenced by initialize(), and messageRecieved().

6.27.5.11 mod_

```
ModulationMode oCpt::LoRa::mod_ [protected]
```

Modulation mode.

Definition at line 466 of file Communication.h.

Referenced by initialize().

6.27.5.12 prlen_

```
uint oCpt::LoRa::prlen_ [protected]
```

Preamble length between 0...65535.

Definition at line 474 of file Communication.h.

Referenced by calculateDownTime(), and initialize().

6.27.5.13 proceed_

```
bool oCpt::LoRa::proceed_ [protected]
```

Definition at line 461 of file Communication.h.

Referenced by messageRecieved(), and write().

6.27.5.14 pwr_

```
int8_t oCpt::LoRa::pwr_ [protected]
```

Power of transmission between -3...15.

Definition at line 468 of file Communication.h.

Referenced by initialize().

6.27.5.15 rxbw_

```
BandWidth oCpt::LoRa::rxbw_ [protected]
```

Signal bandwidth in kHz.

Definition at line 471 of file Communication.h.

Referenced by initialize().

6.27.5.16 sendAllowed_

```
bool oCpt::LoRa::sendAllowed_ [protected]
```

Definition at line 480 of file Communication.h.

Referenced by sendMessage().

6.27.5.17 serial_

```
protocol::Serial oCpt::LoRa::serial_ [protected]
```

SerialPort for UART communication with the chip.

Definition at line 464 of file Communication.h.

Referenced by initialize(), macpause(), messageRecieved(), rx(), sendMessage(), and write().

6.27.5.18 sf_

```
SpreadingFactor oCpt::LoRa::sf_ [protected]
```

Spreading factor of the signal.

Definition at line 469 of file Communication.h.

Referenced by calculateDownTime(), and initialize().

6.27.5.19 sync_

```
unsigned int oCpt::LoRa::sync_  [protected]
```

Sync word.

Definition at line 478 of file Communication.h.

Referenced by initialize().

6.27.5.20 wdt_

```
unsigned long oCpt::LoRa::wdt_  [protected]
```

WatchDog 0...4294967295. Set to 0 to disable.

Definition at line 477 of file Communication.h.

Referenced by initialize().

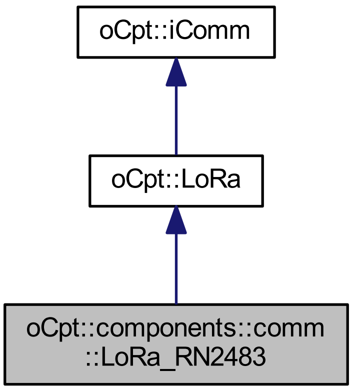
The documentation for this class was generated from the following files:

- include/Core/Communication.h
- src/Core/Communication.cpp

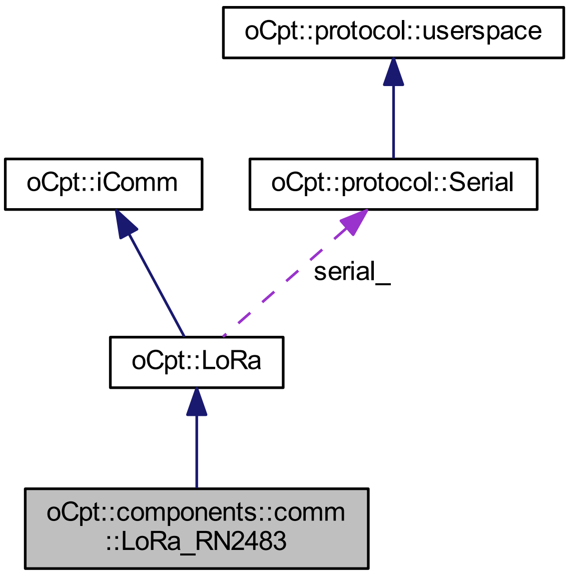
6.28 oCpt::components::comm::LoRa_RN2483 Class Reference

```
#include <LoRa_RN2483.h>
```

Inheritance diagram for oCpt::components::comm::LoRa_RN2483:



Collaboration diagram for oCpt::components::comm::LoRa_RN2483:


oCpt::protocol::userspace
oCpt::iComm
oCpt::protocol::Serial
serial_
oCpt::LoRa
oCpt::components::comm ::LoRa_RN2483

Public Member Functions

- LoRa_RN2483 (const std::string &id, const std::string &device, World::ptr world=World::ptr(new World()), i↩ Controller::io_t ioservice=iController::io_t(new boost::asio::io_service()))
- virtual ∼LoRa_RN2483 ()

Additional Inherited Members

6.28.1 Detailed Description

Definition at line 35 of file LoRa_RN2483.h.

6.28.2 Constructor & Destructor Documentation

6.28.2.1 LoRa_RN2483()

```
oCpt::components::comm::LoRa_RN2483::LoRa_RN2483 (
            const std::string & id,
            const std::string & device,
            World::ptr world = World::ptr(new World()),
            iController::io_t ioservice = iController::io_t(new boost::asio::io_service()) )
```

Definition at line 12 of file LoRa_RN2483.cpp.

6.28.2.2 ∼LoRa_RN2483()

```
oCpt::components::comm::LoRa_RN2483::∼LoRa_RN2483 ( )  [virtual]
```

Definition at line 18 of file LoRa_RN2483.cpp.

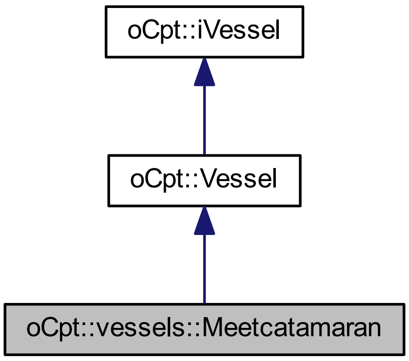
The documentation for this class was generated from the following files:

- include/Communication/LoRa_RN2483.h
- src/Communication/LoRa_RN2483.cpp

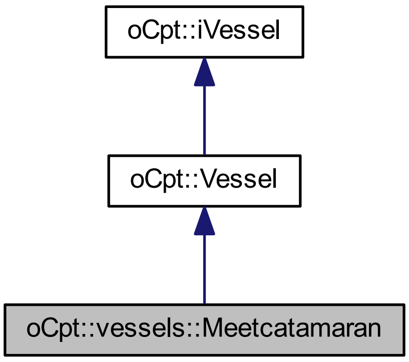
6.29 oCpt::vessels::Meetcatamaran Class Reference

```
#include <Meetcatamaran.h>
```

Inheritance diagram for oCpt::vessels::Meetcatamaran:



Collaboration diagram for oCpt::vessels::Meetcatamaran:



Public Member Functions

- Meetcatamaran ()
- virtual ∼Meetcatamaran ()

Additional Inherited Members

6.29.1 Detailed Description

Definition at line 11 of file Meetcatamaran.h.

6.29.2 Constructor & Destructor Documentation

6.29.2.1 Meetcatamaran()

```
oCpt::vessels::Meetcatamaran::Meetcatamaran ( )
```

Definition at line 17 of file Meetcatamaran.cpp.

6.29.2.2 ∼Meetcatamaran()

```
oCpt::vessels::Meetcatamaran::∼Meetcatamaran ( )  [virtual]
```

Definition at line 31 of file Meetcatamaran.cpp.

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

- include/Vessels/Meetcatamaran.h
- src/Vessels/Meetcatamaran.cpp

6.30 oCpt::iComm::Message Struct Reference

```
#include <Communication.h>
```

Public Types

- typedef boost::shared_ptr< Message > ptr

Public Member Functions

- Message (const std::string &payload, const World::Time::timepoint_t &stamp)
- ∼Message ()

Public Attributes

- std::string Payload
- World::Time::timepoint_t Stamp

6.30.1 Detailed Description

A Message struct. Each device should receive and send messages with this type, consisting of Payload in the format of a string and a time when it was send or received

Definition at line 34 of file Communication.h.

6.30.2 Member Typedef Documentation

6.30.2.1 ptr

```
typedef boost::shared_ptr<Message> oCpt::iComm::Message::ptr
```

Definition at line 35 of file Communication.h.

6.30.3 Constructor & Destructor Documentation

6.30.3.1 Message()

```
oCpt::iComm::Message::Message (
            const std::string & payload,
            const World::Time::timepoint_t & stamp )  [inline]
```

A Message constructor taking the payload and the time

Parameters

payload	A string containing the payload TODO make generic with template
stamp	A time stamp, when the message was received, or is send

Definition at line 42 of file Communication.h.

6.30.3.2 ∼Message()

```
oCpt::iComm::Message::∼Message ( )  [inline]
```

The deconstructor

Definition at line 49 of file Communication.h.

References Payload.

6.30.4 Member Data Documentation

6.30.4.1 Payload

`std::string oCpt::iComm::Message::Payload`

Definition at line 49 of file Communication.h.

Referenced by oCpt::LoRa::sendMessage(), and ∼Message().

6.30.4.2 Stamp

`World::Time::timepoint_t oCpt::iComm::Message::Stamp`

Definition at line 51 of file Communication.h.

Referenced by oCpt::LoRa::sendMessage().

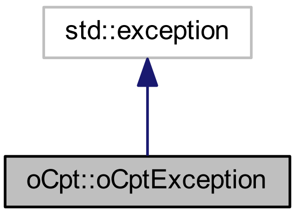
The documentation for this struct was generated from the following file:

- include/Core/Communication.h

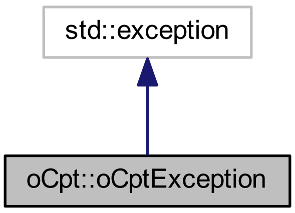
6.31 oCpt::oCptException Class Reference

`#include <Exception.h>`

Inheritance diagram for oCpt::oCptException:



Collaboration diagram for oCpt::oCptException:



Public Member Functions

- oCptException (std::string msg="exception!", int id=-1)
- ∼oCptException () throw ()
- const char ∗ what () const throw ()

Private Attributes

- std::string _msg
- int _id

6.31.1 Detailed Description

Definition at line 13 of file Exception.h.

6.31.2 Constructor & Destructor Documentation

6.31.2.1 oCptException()

```
oCpt::oCptException::oCptException (
             std::string msg = "exception!",
             int id = -1 )  [inline]
```

Definition at line 15 of file Exception.h.

6.31.2.2 ∼oCptException()

```
oCpt::oCptException::∼oCptException ( ) throw )   [inline]
```

Definition at line 17 of file Exception.h.

6.31.3 Member Function Documentation

6.31.3.1 what()

```
const char∗ oCpt::oCptException::what ( ) const throw )   [inline]
```

Definition at line 19 of file Exception.h.

References _msg.

6.31.4 Member Data Documentation

6.31.4.1 _id

```
int oCpt::oCptException::_id  [private]
```

Definition at line 23 of file Exception.h.

6.31.4.2 _msg

```
std::string oCpt::oCptException::_msg [private]
```

Definition at line 22 of file Exception.h.

Referenced by what().

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

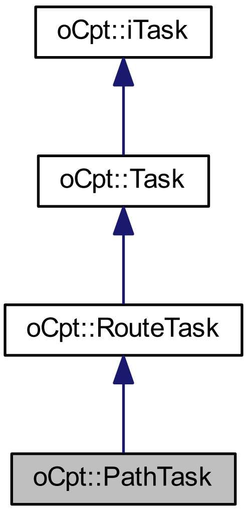
- include/Core/Exception.h

6.32 oCpt::PathTask Class Reference

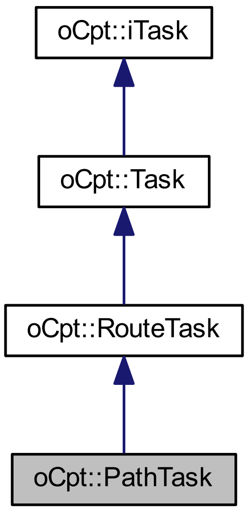
An object representing a normal A to B type of path planning.

```
#include <Task.h>
```

Inheritance diagram for oCpt::PathTask:



Collaboration diagram for oCpt::PathTask:



Public Member Functions

- PathTask (Vessel::ptr vessel, bool concurrent=false)
- virtual ∼PathTask ()

Additional Inherited Members

6.32.1 Detailed Description

An object representing a normal A to B type of path planning.

All these types of tasks need to plann an optimum route between A and B, either in time, energy consumption or

Definition at line 244 of file Task.h.

6.32.2 Constructor & Destructor Documentation

6.32.2.1 PathTask()

```
oCpt::PathTask::PathTask (
            Vessel::ptr vessel,
            bool concurrent = false )
```

Constructor of the interface

Returns

Definition at line 61 of file Task.cpp.

6.32.2.2 ∼PathTask()

```
oCpt::PathTask::∼PathTask ( )  [virtual]
```

The deconstructor

Definition at line 63 of file Task.cpp.

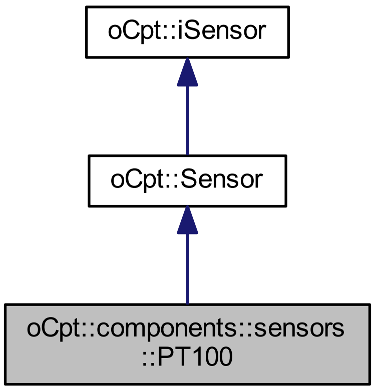
The documentation for this class was generated from the following files:

- include/Core/Task.h
- src/Core/Task.cpp

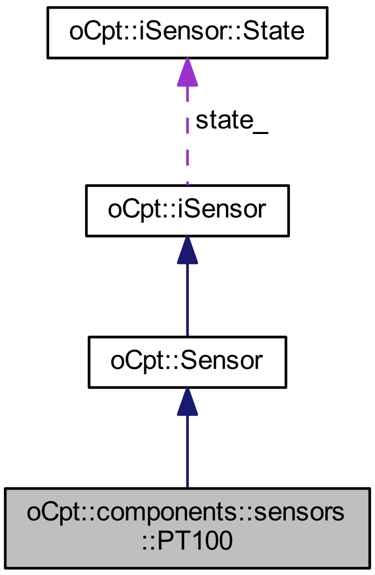
6.33 oCpt::components::sensors::PT100 Class Reference

```
#include <PT100.h>
```

Inheritance diagram for oCpt::components::sensors::PT100:



Collaboration diagram for oCpt::components::sensors::PT100:



Public Types

- typedef double ReturnValue_t

Public Member Functions

- PT100 (iController::ptr controller, World::ptr world, std::string id, uint8_t pinid, uint8_t device)
- ∼PT100 ()
- void updateSensor ()
- void run ()
- void stop ()
- void init ()
- void setCalibrationTemperature (std::pair< double, double > temparature, std::pair< uint16_t, uint16_t > analogeValue)

Private Attributes

- uint16_t _analogeValue
- uint8_t _device = 0
- uint8_t _pinid = 0
- double _dy_dx = 1.0
- double _constant = 0.0

Additional Inherited Members

6.33.1 Detailed Description

Definition at line 14 of file PT100.h.

6.33.2 Member Typedef Documentation

6.33.2.1 ReturnValue_t

```
typedef double oCpt::components::sensors::PT100::ReturnValue_t
```

Definition at line 16 of file PT100.h.

6.33.3 Constructor & Destructor Documentation

6.33.3.1 PT100()

```
oCpt::components::sensors::PT100::PT100 (
            iController::ptr controller,
            World::ptr world,
            std::string id,
            uint8_t pinid,
            uint8_t device )
```

Definition at line 12 of file PT100.cpp.

6.33.3.2 ∼PT100()

```
oCpt::components::sensors::PT100::∼PT100 ( )
```

Definition at line 19 of file PT100.cpp.

6.33.4 Member Function Documentation

6.33.4.1 init()

```
void oCpt::components::sensors::PT100::init ( ) [virtual]
```

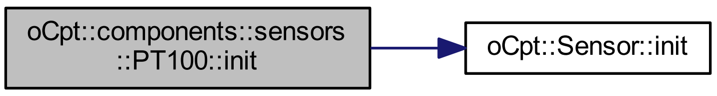
Initialize the sensor

Reimplemented from oCpt::Sensor.

Definition at line 45 of file PT100.cpp.

References oCpt::Sensor::init().

Here is the call graph for this function:



6.33.4.2 run()

```
void oCpt::components::sensors::PT100::run ( )  [virtual]
```

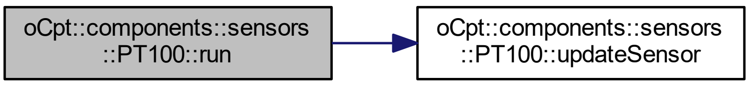
virtual function starting the run service for the IO

Reimplemented from oCpt::Sensor.

Definition at line 36 of file PT100.cpp.

References oCpt::iSensor::sig_, and updateSensor().

Here is the call graph for this function:


oCpt::components::sensors ::PT100::run
oCpt::components::sensors ::PT100::updateSensor

6.33.4.3 setCalibrationTemperature()

```
void oCpt::components::sensors::PT100::setCalibrationTemperature (
            std::pair< double, double > temparature,
            std::pair< uint16_t, uint16_t > analogeValue )
```

Definition at line 30 of file PT100.cpp.

References _constant, and _dy_dx.

6.33.4.4 stop()

```
void oCpt::components::sensors::PT100::stop ( )  [virtual]
```

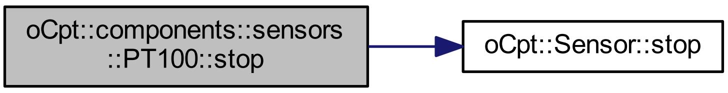
virtual function stopping the run

Reimplemented from oCpt::Sensor.

Definition at line 41 of file PT100.cpp.

References oCpt::Sensor::stop().

Here is the call graph for this function:



6.33.4.5 updateSensor()

```
void oCpt::components::sensors::PT100::updateSensor ( )  [virtual]
```

virtual function which performs a sensor update, obtaining a new value and sending a signal afterwards

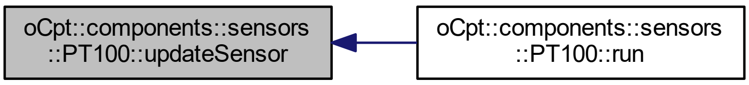
Reimplemented from oCpt::Sensor.

Definition at line 23 of file PT100.cpp.

References _analogeValue, _constant, _dy_dx, _pinid, oCpt::iSensor::controller_, oCpt::iSensor::State::Stamp, o↩
Cpt::iSensor::state_, oCpt::iSensor::State::Value, and oCpt::iSensor::world_.

Referenced by run().

Here is the caller graph for this function:


oCpt::components::sensors ::PT100::updateSensor
oCpt::components::sensors ::PT100::run

6.33.5 Member Data Documentation

6.33.5.1 _analogeValue

```
uint16_t oCpt::components::sensors::PT100::_analogeValue  [private]
```

Definition at line 34 of file PT100.h.

Referenced by updateSensor().

6.33.5.2 _constant

```
double oCpt::components::sensors::PT100::_constant = 0.0  [private]
```

Definition at line 38 of file PT100.h.

Referenced by setCalibrationTemperature(), and updateSensor().

6.33.5.3 _device

```
uint8_t oCpt::components::sensors::PT100::_device = 0  [private]
```

Definition at line 35 of file PT100.h.

6.33.5.4 _dy_dx

```
double oCpt::components::sensors::PT100::_dy_dx = 1.0  [private]
```

Definition at line 37 of file PT100.h.

Referenced by setCalibrationTemperature(), and updateSensor().

6.33.5.5 _pinid

```
uint8_t oCpt::components::sensors::PT100::_pinid = 0  [private]
```

Definition at line 36 of file PT100.h.

Referenced by updateSensor().

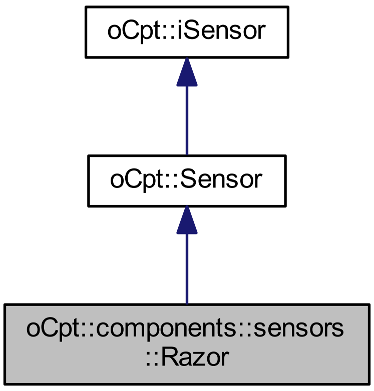
The documentation for this class was generated from the following files:

- include/Sensors/PT100.h
- src/Sensors/PT100.cpp

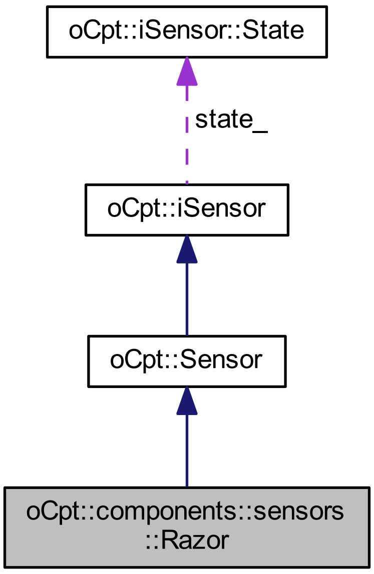
6.34 oCpt::components::sensors::Razor Class Reference

```
#include <Razor.h>
```

Inheritance diagram for oCpt::components::sensors::Razor:



Collaboration diagram for oCpt::components::sensors::Razor:



Classes

- struct ReturnValue

Public Types

- enum Mode { CONT = 0, REQ = 1 }
- typedef struct oCpt::components::sensors::Razor::ReturnValue ReturnValue_t

Public Member Functions

- Razor (iController::ptr controller, World::ptr world, std::string id, std::string device, unsigned int baudrate, Mode mode=Mode::REQ, uint8_t freq=50)
- ∼Razor ()
- void updateSensor ()
- void run ()
- void stop ()
- void init ()
- void setIOservice (boost::shared_ptr< boost::asio::io_service > ioservice)
- Mode getMode () const
- void setMode (Mode mode)
- uint8_t getFreq () const
- void setFreq (uint8_t freq)

Private Member Functions

- void fillReturnValue (ReturnValue_t &retVal, float ∗values)
- void msgHandler (const unsigned char ∗data, size_t size)
- bool checkLRC (std::vector< char ∗> data)

Private Attributes

- std::string device_
- protocol::Serial::ptr serial_
- protocol::Serial::cb_func cb
- Mode mode_
- uint8_t freq_ = 50

Additional Inherited Members

6.34.1 Detailed Description

Definition at line 11 of file Razor.h.

6.34.2 Member Typedef Documentation

6.34.2.1 ReturnValue_t

```
typedef struct oCpt::components::sensors::Razor::ReturnValue oCpt::components::sensors::↩
Razor::ReturnValue_t
```

6.34.3 Member Enumeration Documentation

6.34.3.1 Mode

```
enum oCpt::components::sensors::Razor::Mode
```

Enumerator

CONT	
REQ	

Definition at line 19 of file Razor.h.

6.34.4 Constructor & Destructor Documentation

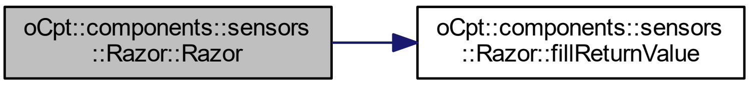
6.34.4.1 Razor()

```
oCpt::components::sensors::Razor::Razor (
            iController::ptr controller,
            World::ptr world,
            std::string id,
            std::string device,
            unsigned int baudrate,
            Mode mode = Mode::REQ,
            uint8_t freq = 50 )
```

Definition at line 13 of file Razor.cpp.

References fillReturnValue(), serial_, oCpt::iSensor::State::Stamp, oCpt::iSensor::state_, oCpt::iSensor::State::↩
Value, and oCpt::iSensor::world_.

Here is the call graph for this function:


oCpt::components::sensors ::Razor::Razor
oCpt::components::sensors ::Razor::fillReturnValue

6.34.4.2 ∼Razor()

```
oCpt::components::sensors::Razor::∼Razor ( )
```

Definition at line 35 of file Razor.cpp.

References serial_.

6.34.5 Member Function Documentation

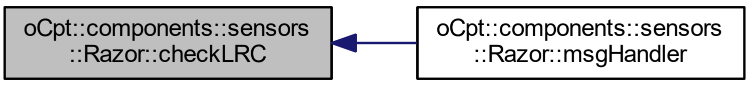
6.34.5.1 checkLRC()

```
bool oCpt::components::sensors::Razor::checkLRC (
            std::vector< char *> data )  [private]
```

Definition at line 120 of file Razor.cpp.

Referenced by msgHandler().

Here is the caller graph for this function:


oCpt::components::sensors ::Razor::checkLRC
oCpt::components::sensors ::Razor::msgHandler

6.34.5.2 fillReturnValue()

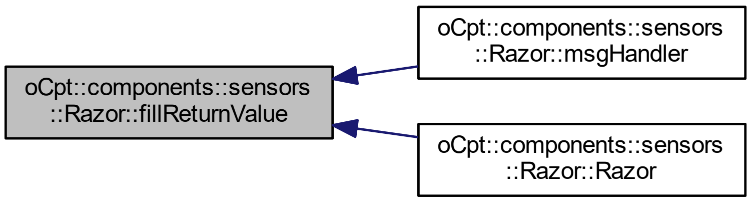
```
void oCpt::components::sensors::Razor::fillReturnValue (
            Razor::ReturnValue_t & retVal,
            float * values ) [private]
```

Definition at line 84 of file Razor.cpp.

References oCpt::components::sensors::Razor::ReturnValue::acc, oCpt::components::sensors::Razor::Return←Value::gyro, and oCpt::components::sensors::Razor::ReturnValue::mag.

Referenced by msgHandler(), and Razor().

Here is the caller graph for this function:


oCpt::components::sensors ::Razor::fillReturnValue
oCpt::components::sensors ::Razor::msgHandler
oCpt::components::sensors ::Razor::Razor

6.34.5.3 getFreq()

```
uint8_t oCpt::components::sensors::Razor::getFreq ( ) const
```

Definition at line 144 of file Razor.cpp.

References freq_.

6.34.5.4 getMode()

```
Razor::Mode oCpt::components::sensors::Razor::getMode ( ) const
```

Definition at line 131 of file Razor.cpp.

References mode_.

6.34.5.5 init()

```
void oCpt::components::sensors::Razor::init ( ) [virtual]
```

Initialize the sensor

Reimplemented from oCpt::Sensor.

Definition at line 75 of file Razor.cpp.

References cb, and serial_.

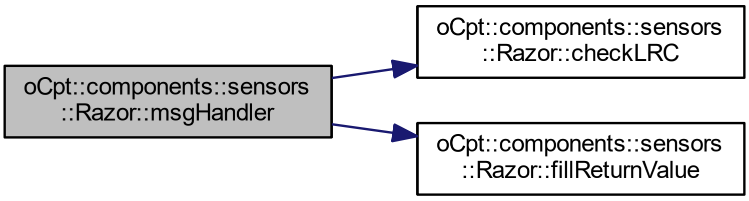
6.34.5.6 msgHandler()

```
void oCpt::components::sensors::Razor::msgHandler (
             const unsigned char * data,
             size_t size )  [private]
```

Definition at line 97 of file Razor.cpp.

References checkLRC(), fillReturnValue(), oCpt::iSensor::sig_, oCpt::iSensor::State::Stamp, oCpt::iSensor::state←
, oCpt::iSensor::State::Value, and oCpt::iSensor::world.

Here is the call graph for this function:


oCpt::components::sensors ::Razor::msgHandler
oCpt::components::sensors ::Razor::checkLRC
oCpt::components::sensors ::Razor::fillReturnValue

6.34.5.7 run()

```
void oCpt::components::sensors::Razor::run ( )  [virtual]
```

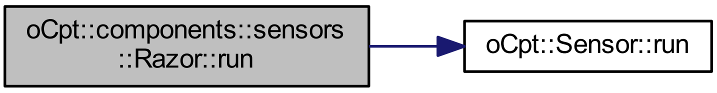
virtual function starting the run service for the IO

Reimplemented from oCpt::Sensor.

Definition at line 53 of file Razor.cpp.

References oCpt::Sensor::run(), oCpt::iSensor::sensorRunning_, and serial_.

Here is the call graph for this function:



6.34.5.8 setFreq()

```
void oCpt::components::sensors::Razor::setFreq (
            uint8_t freq )
```

Definition at line 148 of file Razor.cpp.

References freq_, and serial_.

6.34.5.9 setIOservice()

```
void oCpt::components::sensors::Razor::setIOservice (
            boost::shared_ptr< boost::asio::io_service > ioservice )  [virtual]
```

Setting the used Asynchronous Input Output service

Parameters

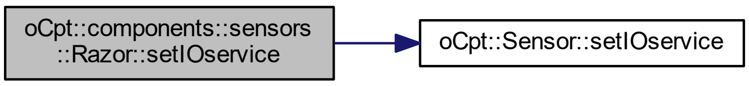
ioservice	ASIO IO service, which handles the async calls from multiple sensors

Reimplemented from oCpt::Sensor.

Definition at line 70 of file Razor.cpp.

References serial_, and oCpt::Sensor::setIOservice().

Here is the call graph for this function:


oCpt::components::sensors ::Razor::setIOservice
oCpt::Sensor::setIOservice

6.34.5.10 setMode()

```
void oCpt::components::sensors::Razor::setMode (
            Razor::Mode mode )
```

Definition at line 135 of file Razor.cpp.

References mode_, and serial_.

6.34.5.11 stop()

```
void oCpt::components::sensors::Razor::stop ( )  [virtual]
```

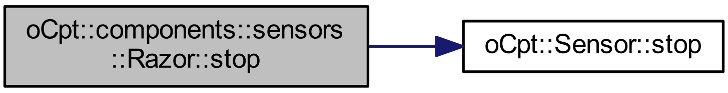
virtual function stopping the run

Reimplemented from oCpt::Sensor.

Definition at line 60 of file Razor.cpp.

References oCpt::iSensor::sensorRunning_, serial_, and oCpt::Sensor::stop().

Here is the call graph for this function:



6.34.5.12 updateSensor()

```
void oCpt::components::sensors::Razor::updateSensor ( )  [virtual]
```

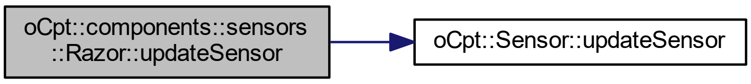
virtual function which performs a sensor update, obtaining a new value and sending a signal afterwards

Reimplemented from oCpt::Sensor.

Definition at line 41 of file Razor.cpp.

References mode_, serial_, oCpt::iSensor::State::Stamp, oCpt::iSensor::state_, oCpt::Sensor::updateSensor(), and oCpt::iSensor::world_.

Here is the call graph for this function:


oCpt::components::sensors ::Razor::updateSensor
oCpt::Sensor::updateSensor

6.34.6 Member Data Documentation

6.34.6.1 cb

```
protocol::Serial::cb_func oCpt::components::sensors::Razor::cb  [private]
```

Definition at line 41 of file Razor.h.

Referenced by init().

6.34.6.2 device_

```
std::string oCpt::components::sensors::Razor::device_  [private]
```

Definition at line 39 of file Razor.h.

6.34.6.3 freq_

```
uint8_t oCpt::components::sensors::Razor::freq_ = 50  [private]
```

Definition at line 53 of file Razor.h.

Referenced by getFreq(), and setFreq().

6.34.6.4 mode_

```
Mode oCpt::components::sensors::Razor::mode_  [private]
```

Definition at line 42 of file Razor.h.

Referenced by getMode(), setMode(), and updateSensor().

6.34.6.5 serial_

```
protocol::Serial::ptr oCpt::components::sensors::Razor::serial_  [private]
```

Definition at line 40 of file Razor.h.

Referenced by init(), Razor(), run(), setFreq(), setIOservice(), setMode(), stop(), updateSensor(), and ∼Razor().

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

- include/Sensors/Razor.h
- src/Sensors/Razor.cpp

6.35 oCpt::components::sensors::Razor::ReturnValue Struct Reference

```
#include <Razor.h>
```

Public Attributes

- float gyro [3]
- float mag [3]
- float acc [3]

6.35.1 Detailed Description

Definition at line 13 of file Razor.h.

6.35.2 Member Data Documentation

6.35.2.1 acc

```
float oCpt::components::sensors::Razor::ReturnValue::acc[3]
```

Definition at line 16 of file Razor.h.

Referenced by oCpt::components::sensors::Razor::fillReturnValue().

6.35.2.2 gyro

```
float oCpt::components::sensors::Razor::ReturnValue::gyro[3]
```

Definition at line 14 of file Razor.h.

Referenced by oCpt::components::sensors::Razor::fillReturnValue().

6.35.2.3 mag

```
float oCpt::components::sensors::Razor::ReturnValue::mag[3]
```

Definition at line 15 of file Razor.h.

Referenced by oCpt::components::sensors::Razor::fillReturnValue().

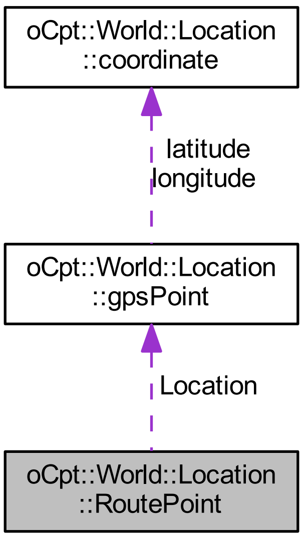
The documentation for this struct was generated from the following file:

- include/Sensors/Razor.h

6.36 oCpt::World::Location::RoutePoint Struct Reference

```
#include <World.h>
```

Collaboration diagram for oCpt::World::Location::RoutePoint:


oCpt::World::Location ::coordinate
latitude
longitude
oCpt::World::Location ::gpsPoint
Location
oCpt::World::Location ::RoutePoint

Public Types

- typedef boost::shared_ptr< RoutePoint > ptr

Public Attributes

- Time::timepoint_t TimePoint
- gpsPoint_t Location

6.36.1 Detailed Description

Definition at line 140 of file World.h.

6.36.2 Member Typedef Documentation

6.36.2.1 ptr

```
typedef boost::shared_ptr<RoutePoint> oCpt::World::Location::RoutePoint::ptr
```

Definition at line 141 of file World.h.

6.36.3 Member Data Documentation

6.36.3.1 Location

```
gpsPoint_t oCpt::World::Location::RoutePoint::Location
```

Definition at line 143 of file World.h.

6.36.3.2 TimePoint

```
Time::timepoint_t oCpt::World::Location::RoutePoint::TimePoint
```

Definition at line 142 of file World.h.

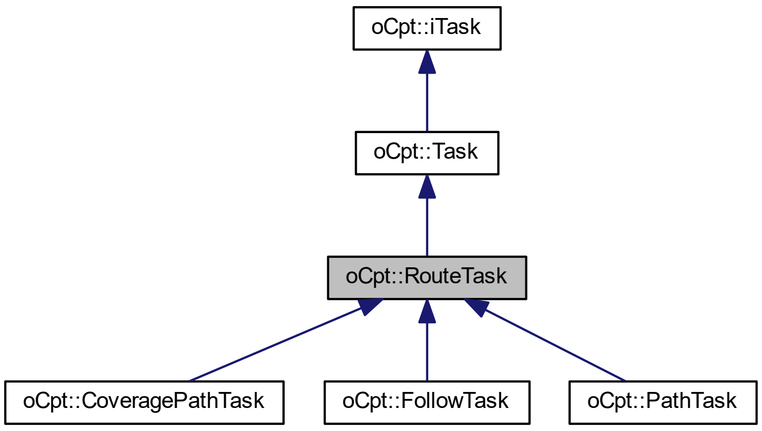
The documentation for this struct was generated from the following file:

- include/Core/World.h

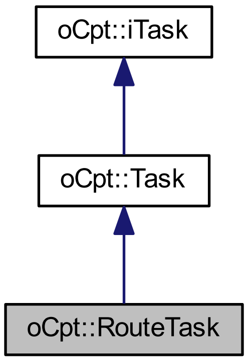
6.37 oCpt::RouteTask Class Reference

```
#include <Task.h>
```

Inheritance diagram for oCpt::RouteTask:



Collaboration diagram for oCpt::RouteTask:



Public Member Functions

- RouteTask (Vessel::ptr vessel, bool concurrent=false)
- virtual ∼RouteTask ()

Additional Inherited Members

6.37.1 Detailed Description

An object repsressenting route related tasks

Definition at line 148 of file Task.h.

6.37.2 Constructor & Destructor Documentation

6.37.2.1 RouteTask()

```
oCpt::RouteTask::RouteTask (
            Vessel::ptr vessel,
            bool concurrent = false )
```

Constructor of the interface

Returns

Definition at line 41 of file Task.cpp.

References oCpt::Task::_typeof.

6.37.2.2 ∼RouteTask()

`oCpt::RouteTask::~RouteTask ( ) [virtual]`

The deconstructor

Definition at line 45 of file Task.cpp.

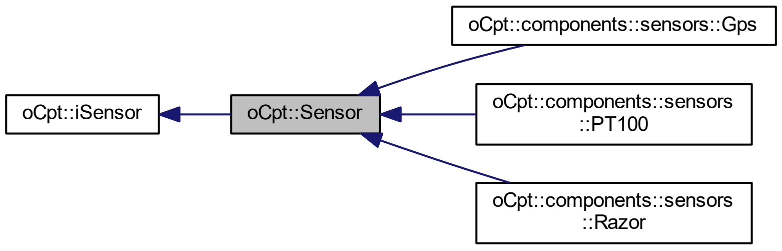
The documentation for this class was generated from the following files:

- include/Core/Task.h
- src/Core/Task.cpp

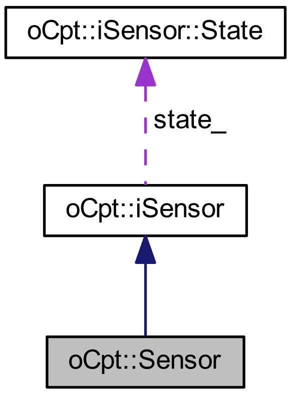
6.38 oCpt::Sensor Class Reference

`#include <Sensor.h>`

Inheritance diagram for oCpt::Sensor:


oCpt::components::sensors::Gps
oCpt::iSensor
oCpt::Sensor
oCpt::components::sensors ::PT100
oCpt::components::sensors ::Razor

Collaboration diagram for oCpt::Sensor:



Public Member Functions

- Sensor (iController::ptr controller, World::ptr world, std::string id, std::string typeOfSensor)
- virtual ∼Sensor () override
- virtual void updateSensor () override
- virtual void run () override
- virtual void stop () override
- virtual void init () override
- virtual void setIOservice (boost::shared_ptr< boost::asio::io_service > ioservice) override

Additional Inherited Members

6.38.1 Detailed Description

Implementation of the iSensor interface

Definition at line 152 of file Sensor.h.

6.38.2 Constructor & Destructor Documentation

6.38.2.1 Sensor()

```
oCpt::Sensor::Sensor (
            iController::ptr controller,
            World::ptr world,
            std::string id,
            std::string typeOfSensor )
```

Constructor of Sensor

Parameters

controller	a shared_ptr of the controller where the sensor is hooked to
world	a shared_ptr of the world in which the vessel operates
id	a identifying name of the sensor
typeOfSensor	a identifying category for the sensor

Definition at line 57 of file Sensor.cpp.

6.38.2.2 ∼Sensor()

```
oCpt::Sensor::∼Sensor ( )  [override], [virtual]
```

Deconstructor of the Sensor class

Definition at line 62 of file Sensor.cpp.

6.38.3 Member Function Documentation

6.38.3.1 init()

```
void oCpt::Sensor::init ( )  [override], [virtual]
```

Initialize the sensor

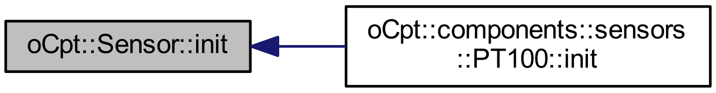
Implements oCpt::iSensor.

Reimplemented in oCpt::components::sensors::Razor, and oCpt::components::sensors::PT100.

Definition at line 78 of file Sensor.cpp.

Referenced by oCpt::components::sensors::PT100::init().

Here is the caller graph for this function:



6.38.3.2 run()

```
void oCpt::Sensor::run ( )  [override], [virtual]
```

virtual function starting the run service for the IO

Implements oCpt::iSensor.

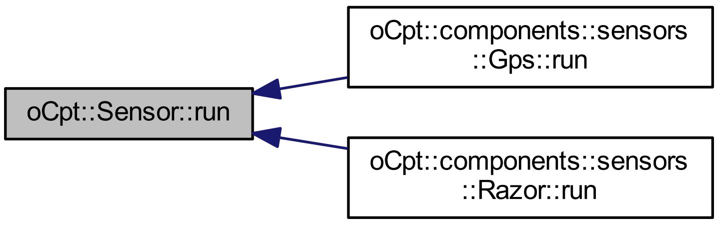
Reimplemented in oCpt::components::sensors::Razor, oCpt::components::sensors::Gps, and oCpt::components↩ ::sensors::PT100.

Definition at line 70 of file Sensor.cpp.

References oCpt::iSensor::sensorRunning_.

Referenced by oCpt::components::sensors::Gps::run(), and oCpt::components::sensors::Razor::run().

Here is the caller graph for this function:


oCpt::components::sensors ::Gps::run
oCpt::Sensor::run
oCpt::components::sensors ::Razor::run

6.38.3.3 setIOservice()

```
void oCpt::Sensor::setIOservice (
            boost::shared_ptr< boost::asio::io_service > ioservice )  [override], [virtual]
```

Setting the used Asynchronous Input Output service

Parameters

ioservice	ASIO IO service, which handles the async calls from multiple sensors

Implements oCpt::iSensor.

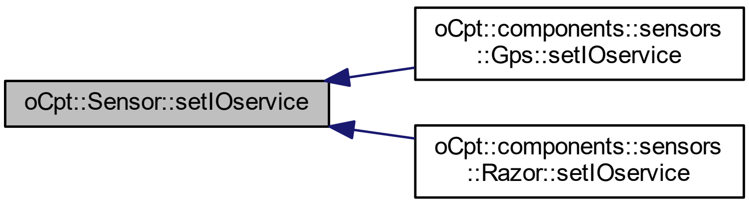
Reimplemented in oCpt::components::sensors::Razor, and oCpt::components::sensors::Gps.

Definition at line 82 of file Sensor.cpp.

References oCpt::iSensor::ioservice_.

Referenced by oCpt::components::sensors::Gps::setIOservice(), and oCpt::components::sensors::Razor::setI↩
Oservice().

Here is the caller graph for this function:


oCpt::Sensor::setIOservice
oCpt::components::sensors ::Gps::setIOservice
oCpt::components::sensors ::Razor::setIOservice

6.38.3.4 stop()

```
void oCpt::Sensor::stop ( )  [override], [virtual]
```

virtual function stopping the run

Implements oCpt::iSensor.

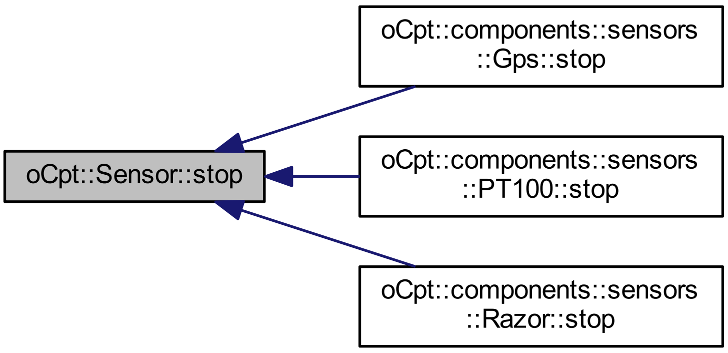
Reimplemented in oCpt::components::sensors::Razor, oCpt::components::sensors::Gps, and oCpt::components↩
::sensors::PT100.

Definition at line 74 of file Sensor.cpp.

References oCpt::iSensor::sensorRunning_.

Referenced by oCpt::components::sensors::Gps::stop(), oCpt::components::sensors::PT100::stop(), and oCpt↩ ::components::sensors::Razor::stop().

Here is the caller graph for this function:


oCpt::Sensor::stop
oCpt::components::sensors ::Gps::stop
oCpt::components::sensors ::PT100::stop
oCpt::components::sensors ::Razor::stop

6.38.3.5 updateSensor()

```
void oCpt::Sensor::updateSensor ( )  [override], [virtual]
```

virtual function which performs a sensor update, obtaining a new value and sending a signal afterwards

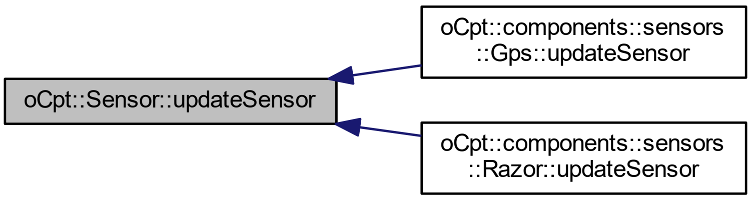
Implements oCpt::iSensor.

Reimplemented in oCpt::components::sensors::Razor, oCpt::components::sensors::Gps, and oCpt::components↩ ::sensors::PT100.

Definition at line 66 of file Sensor.cpp.

Referenced by oCpt::components::sensors::Gps::updateSensor(), and oCpt::components::sensors::Razor↩ ::updateSensor().

Here is the caller graph for this function:


oCpt::Sensor::updateSensor
oCpt::components::sensors ::Gps::updateSensor
oCpt::components::sensors ::Razor::updateSensor

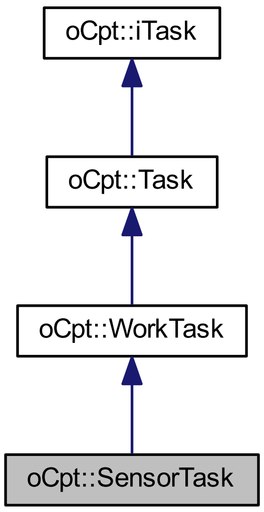
The documentation for this class was generated from the following files:

- include/Core/Sensor.h
- src/Core/Sensor.cpp

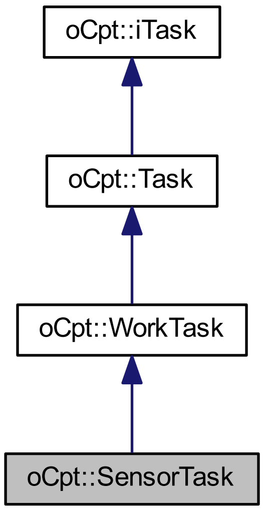
6.39 oCpt::SensorTask Class Reference

```
#include <Task.h>
```

Inheritance diagram for oCpt::SensorTask:



Collaboration diagram for oCpt::SensorTask:



Public Member Functions

- SensorTask (Vessel::ptr vessel, bool concurrent=true)
- virtual ∼SensorTask ()

Additional Inherited Members

6.39.1 Detailed Description

Definition at line 303 of file Task.h.

6.39.2 Constructor & Destructor Documentation

6.39.2.1 SensorTask()

```
oCpt::SensorTask::SensorTask (
            Vessel::ptr vessel,
            bool concurrent = true )
```

Definition at line 73 of file Task.cpp.

6.39.2.2 ∼SensorTask()

```
oCpt::SensorTask::∼SensorTask ( )  [virtual]
```

Definition at line 75 of file Task.cpp.

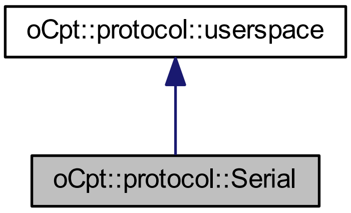
The documentation for this class was generated from the following files:

- include/Core/Task.h
- src/Core/Task.cpp

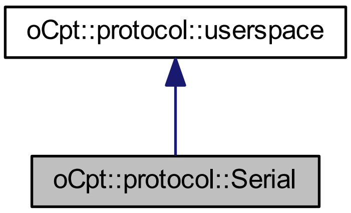
6.40 oCpt::protocol::Serial Class Reference

```
#include <Controller.h>
```

Inheritance diagram for oCpt::protocol::Serial:



Collaboration diagram for oCpt::protocol::Serial:



Public Types

- typedef boost::shared_ptr< Serial > ptr
- typedef std::function< void(const unsigned char ∗, size_t)> cb_func
- typedef boost::asio::serial_port_base::parity parity_t
- typedef boost::asio::serial_port_base::character_size character_size_t
- typedef boost::asio::serial_port_base::flow_control flow_control_t
- typedef boost::asio::serial_port_base::stop_bits stop_bits_t
- typedef boost::shared_ptr< boost::asio::io_service > io_service_t
- typedef boost::asio::serial_port serialport_t
- typedef boost::signals2::signal< void()> signal_t

Public Member Functions

- Serial (const std::string &device, unsigned int baudrate, io_service_t ioservice=io_service_t(new boost↵ ::asio::io_service()), parity_t parity=parity_t(parity_t::none), character_size_t csize=character_size_t(8), flow_control_t flow=flow_control_t(flow_control_t::none), stop_bits_t stop=stop_bits_t(stop_bits_t::one), unsigned int maxreadlentgh=4096)
- void open ()
- void start ()
- bool isOpen ()
- void close ()
- bool write (const std::string &msg)
- bool write (const std::vector< unsigned char > &data)
- void setReadCallback (cb_func cb_function)
- void setIOservice (boost::shared_ptr< boost::asio::io_service > io_ptr)
- std::deque< std::string > ∗ getReturnMsgQueue ()
- std::string readFiFoMsg ()
- void setMaxReadLength (unsigned int maxReadLength)

Public Attributes

- signal_t msgRecievedSig

Protected Member Functions

- void internalCallback (const unsigned char ∗data, size_t size)
- void closeCallback (const boost::system::error_code &error)
- void readComplete (const boost::system::error_code &error, size_t bytes_transferred)
- void writeCallback (const std::vector< unsigned char > &msg)
- void writeStart ()
- void writeComplete (const boost::system::error_code &error)
- void ReadStart ()

Protected Attributes

- unsigned int maxReadLength_
- std::deque< std::vector< unsigned char > > msgQueue_
- std::deque< std::string > returnMsgQueue_
- std::string msg_
- std::string receivedMsg_
- unsigned char read_msg [4096]
- cb_func callback_
- io_service_t ioservice_
- std::string device_
- unsigned int baudrate_
- parity_t parity_
- character_size_t csize_
- flow_control_t flow_
- stop_bits_t stop_
- serialport_t serialport_
- bool firstMsg = true

6.40.1 Detailed Description

Communication via the serial port, using an Asynchrounous Input Output setup, provided by Boost. All communication is handled on the background via a io_service. When data is recieved a callback function is called. This can either be an external function or an internal one, which sends a signal for each new line. The lines can then be read using a FiFo function.

Definition at line 390 of file Controller.h.

6.40.2 Member Typedef Documentation

6.40.2.1 cb_func

```
typedef std::function<void(const unsigned char *, size_t)> oCpt::protocol::Serial::cb_func
```

Definition at line 394 of file Controller.h.

6.40.2.2 character_size_t

```
typedef boost::asio::serial_port_base::character_size oCpt::protocol::Serial::character_size↩
_t
```

Definition at line 396 of file Controller.h.

6.40.2.3 flow_control_t

```
typedef boost::asio::serial_port_base::flow_control oCpt::protocol::Serial::flow_control_t
```

Definition at line 397 of file Controller.h.

6.40.2.4 io_service_t

```
typedef boost::shared_ptr<boost::asio::io_service> oCpt::protocol::Serial::io_service_t
```

Definition at line 399 of file Controller.h.

6.40.2.5 parity_t

```
typedef boost::asio::serial_port_base::parity oCpt::protocol::Serial::parity_t
```

Definition at line 395 of file Controller.h.

6.40.2.6 ptr

```
typedef boost::shared_ptr<Serial> oCpt::protocol::Serial::ptr
```

Definition at line 392 of file Controller.h.

6.40.2.7 serialport_t

```
typedef boost::asio::serial_port oCpt::protocol::Serial::serialport_t
```

Definition at line 400 of file Controller.h.

6.40.2.8 signal_t

```
typedef boost::signals2::signal<void()> oCpt::protocol::Serial::signal_t
```

Definition at line 401 of file Controller.h.

6.40.2.9 stop_bits_t

```
typedef boost::asio::serial_port_base::stop_bits oCpt::protocol::Serial::stop_bits_t
```

Definition at line 398 of file Controller.h.

6.40.3 Constructor & Destructor Documentation

6.40.3.1 Serial()

```
oCpt::protocol::Serial::Serial (
             const std::string & device,
             unsigned int baudrate,
             io_service_t ioservice = io_service_t(new boost::asio::io_service()),
             Serial::parity_t parity = parity_t(parity_t::none),
             Serial::character_size_t csize = character_size_t(8),
             Serial::flow_control_t flow = flow_control_t(flow_control_t::none),
             Serial::stop_bits_t stop = stop_bits_t(stop_bits_t::one),
             unsigned int maxreadlentgh = 4096 )
```

Constructor of the Serial class

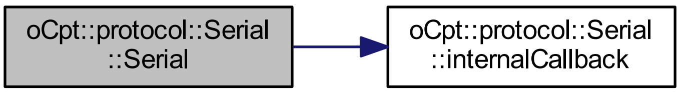
Parameters

device	a string representing the device path eq. /dev/tty0
baudrate	the baudrate of the device eq. 9600, 57600, 115200
ioservice	the io service to be used standard it's a new service
parity	the parity of the Serial port with a standard parity of Parity_t::none
csize	The character_size with a standard value of 8
flow	The flow control of thye device with a standard value of flow_control_t::none
stop	The stop bit of the device with a standard value of stop_bits_t::none
maxreadlentgh	the Maximum buffer with a standard value of 4096

Definition at line 99 of file Controller.cpp.

References callback_, and internalCallback().

Here is the call graph for this function:


oCpt::protocol::Serial ::Serial
oCpt::protocol::Serial ::internalCallback

6.40.4 Member Function Documentation

6.40.4.1 close()

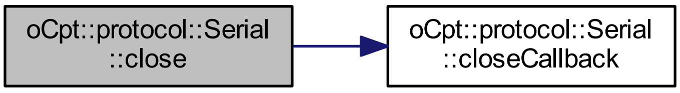
```
void oCpt::protocol::Serial::close ( )
```

Closes the port when it is open

Definition at line 142 of file Controller.cpp.

References closeCallback(), ioservice_, and serialport_.

Here is the call graph for this function:


oCpt::protocol::Serial ::close
oCpt::protocol::Serial ::closeCallback

6.40.4.2 closeCallback()

```
void oCpt::protocol::Serial::closeCallback (
             const boost::system::error_code & error )  [protected]
```

The callback function which is called after the port is closed

Parameters

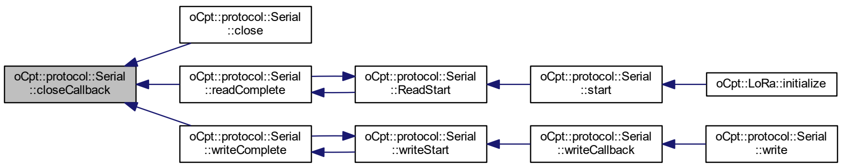
error	an past trhough boost::system::error_code

Definition at line 159 of file Controller.cpp.

References serialport_.

Referenced by close(), readComplete(), and writeComplete().

Here is the caller graph for this function:


oCpt::protocol::Serial ::closeCallback
oCpt::protocol::Serial ::close
oCpt::protocol::Serial ::readComplete
oCpt::protocol::Serial ::ReadStart
oCpt::protocol::Serial ::start
oCpt::LoRa::initialize
oCpt::protocol::Serial ::writeComplete
oCpt::protocol::Serial ::writeStart
oCpt::protocol::Serial ::writeCallback
oCpt::protocol::Serial ::write

6.40.4.3 getReturnMsgQueue()

```
std::deque< std::string > * oCpt::protocol::Serial::getReturnMsgQueue ( )
```

Get the complete returnMsg que

Returns

a deque with all the return lines

Definition at line 197 of file Controller.cpp.

References returnMsgQueue_.

6.40.4.4 internalCallback()

```
void oCpt::protocol::Serial::internalCallback (
            const unsigned char * data,
            size_t size )  [protected]
```

The internal callback function, which handles messages longer then maxreadlentgh and splits the message with

Parameters

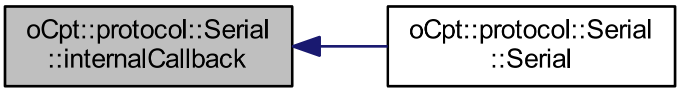
data	the buffer obtained by the serial port
size	the size obtained

Definition at line 208 of file Controller.cpp.

References msgRecievedSig, receivedMsg_, and returnMsgQueue_.

Referenced by Serial().

Here is the caller graph for this function:


oCpt::protocol::Serial ::internalCallback
oCpt::protocol::Serial ::Serial

6.40.4.5 isOpen()

```
bool oCpt::protocol::Serial::isOpen ( )
```

Checks if the port is open

Returns

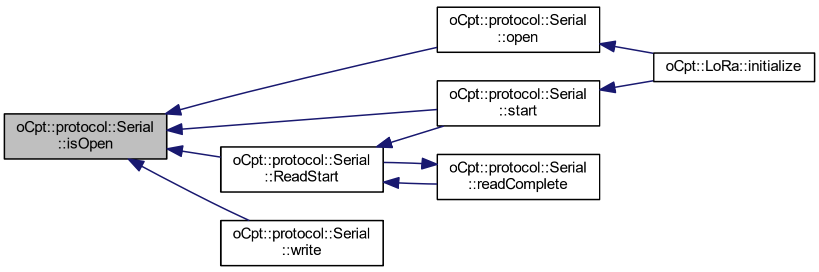
either true or false

Definition at line 138 of file Controller.cpp.

References serialport_.

Referenced by open(), ReadStart(), start(), and write().

Here is the caller graph for this function:


oCpt::protocol::Serial ::isOpen
oCpt::protocol::Serial ::open
oCpt::protocol::Serial ::start
oCpt::protocol::Serial ::ReadStart
oCpt::protocol::Serial ::readComplete
oCpt::protocol::Serial ::write
oCpt::LoRa::initialize

6.40.4.6 open()

```
void oCpt::protocol::Serial::open ( )
```

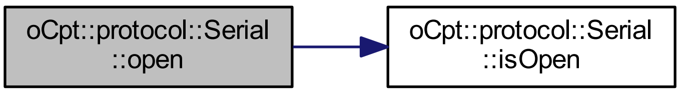
Open teh serial port

Definition at line 118 of file Controller.cpp.

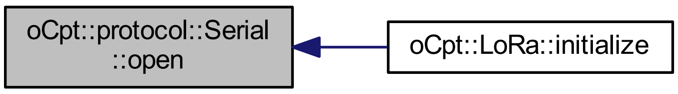
References baudrate_, csize_, device_, flow_, ioservice_, isOpen(), parity_, serialport_, and stop_.

Referenced by oCpt::LoRa::initialize().

Here is the call graph for this function:


oCpt::protocol::Serial ::open
oCpt::protocol::Serial ::isOpen

Here is the caller graph for this function:



6.40.4.7 readComplete()

```
void oCpt::protocol::Serial::readComplete (
             const boost::system::error_code & error,
             size_t bytes_transferred )  [protected]
```

The callbackfunction to be performed when reading is complete

Parameters

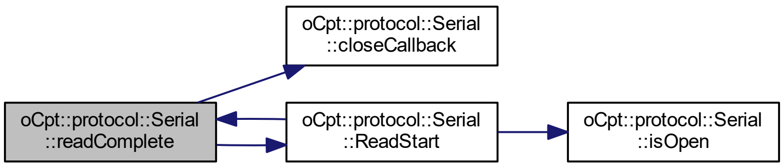
error	boost::system::error_code if an error is presented
bytes_transferred	number of bytes that are transfered

Definition at line 188 of file Controller.cpp.

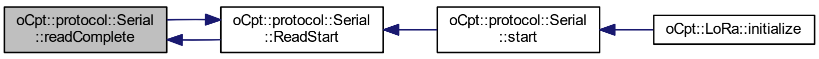
References callback_, closeCallback(), read_msg, and ReadStart().

Referenced by ReadStart().

Here is the call graph for this function:


oCpt::protocol::Serial ::closeCallback
oCpt::protocol::Serial ::readComplete
oCpt::protocol::Serial ::ReadStart
oCpt::protocol::Serial ::isOpen

Here is the caller graph for this function:


oCpt::protocol::Serial ::readComplete
oCpt::protocol::Serial ::ReadStart
oCpt::protocol::Serial ::start
oCpt::LoRa::initialize

6.40.4.8 readFiFoMsg()

```
std::string oCpt::protocol::Serial::readFiFoMsg ( )
```

Gets the first recieved message, which is then removed from the queue

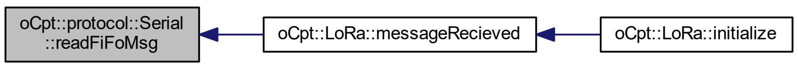
Returns

Definition at line 201 of file Controller.cpp.

References returnMsgQueue_.

Referenced by oCpt::LoRa::messageRecieved().

Here is the caller graph for this function:


oCpt::protocol::Serial ::readFiFoMsg
oCpt::LoRa::messageRecieved
oCpt::LoRa::initialize

6.40.4.9 ReadStart()

```
void oCpt::protocol::Serial::ReadStart ( )  [protected]
```

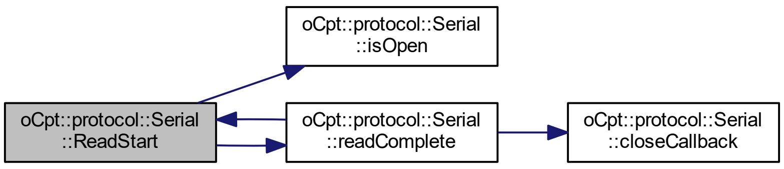
Start the reading process

Definition at line 178 of file Controller.cpp.

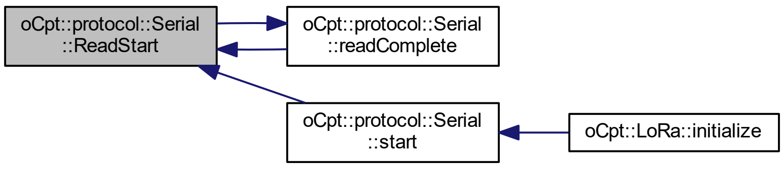
References isOpen(), maxReadLength_, read_msg, readComplete(), and serialport_.

Referenced by readComplete(), and start().

Here is the call graph for this function:


oCpt::protocol::Serial ::isOpen
oCpt::protocol::Serial ::ReadStart
oCpt::protocol::Serial ::readComplete
oCpt::protocol::Serial ::closeCallback

Here is the caller graph for this function:


oCpt::protocol::Serial ::ReadStart
oCpt::protocol::Serial ::readComplete
oCpt::protocol::Serial ::start
oCpt::LoRa::initialize

6.40.4.10 setIOservice()

```
void oCpt::protocol::Serial::setIOservice (
            boost::shared_ptr< boost::asio::io_service > io_ptr )
```

set a new IO service

Parameters

io_ptr	a shared_ptr to the new IO service

Definition at line 155 of file Controller.cpp.

References ioservice_.

6.40.4.11 setMaxReadLength()

```
void oCpt::protocol::Serial::setMaxReadLength (
             unsigned int maxReadLength )
```

Set the maximum buffer of the Serial class

Parameters

maxReadLength	the number of bytes

Definition at line 287 of file Controller.cpp.

References maxReadLength_.

6.40.4.12 setReadCallback()

```
void oCpt::protocol::Serial::setReadCallback (
             cb_func cb_function )
```

Set a new callback function

Parameters

cb_function	the callback function

Definition at line 151 of file Controller.cpp.

References callback_.

6.40.4.13 start()

```
void oCpt::protocol::Serial::start ( )
```

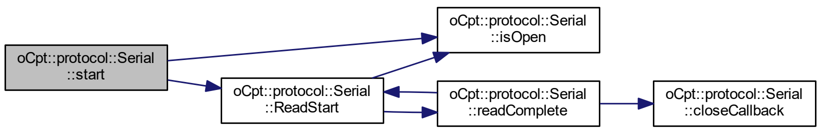
Start the io_service on a sepearte thread

Definition at line 166 of file Controller.cpp.

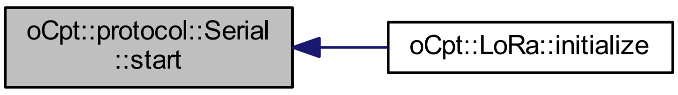
References ioservice_, isOpen(), and ReadStart().

Referenced by oCpt::LoRa::initialize().

Here is the call graph for this function:


oCpt::protocol::Serial ::start
oCpt::protocol::Serial ::isOpen
oCpt::protocol::Serial ::ReadStart
oCpt::protocol::Serial ::readComplete
oCpt::protocol::Serial ::closeCallback

Here is the caller graph for this function:



6.40.4.14 write() [1/2]

```
bool oCpt::protocol::Serial::write (
            const std::string & msg )
```

Write a message to the port

Parameters

msg	a string with the payload

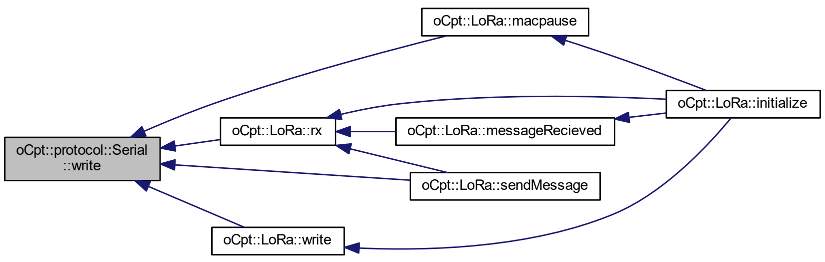
Returns

either true or false depending if writting was sucsesfull or not

Definition at line 238 of file Controller.cpp.

Referenced by oCpt::LoRa::macpause(), oCpt::LoRa::rx(), oCpt::LoRa::sendMessage(), and oCpt::LoRa::write().

Here is the caller graph for this function:



6.40.4.15 write() [2/2]

```
bool oCpt::protocol::Serial::write (
            const std::vector< unsigned char > & data )
```

Write a message as a vector of unsigned chars

Parameters

data	the message to be send

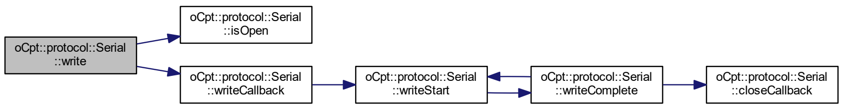
Returns

either true or false depending if writting was sucsesfull or not

Definition at line 244 of file Controller.cpp.

References ioservice_, isOpen(), and writeCallback().

Here is the call graph for this function:


oCpt::protocol::Serial ::write
oCpt::protocol::Serial ::isOpen
oCpt::protocol::Serial ::writeCallback
oCpt::protocol::Serial ::writeStart
oCpt::protocol::Serial ::writeComplete
oCpt::protocol::Serial ::closeCallback

6.40.4.16 writeCallback()

```
void oCpt::protocol::Serial::writeCallback (
            const std::vector< unsigned char > & msg ) [protected]
```

When the writing is finished call this function, which will write the next message if present

Parameters

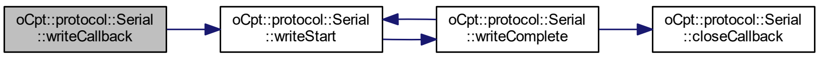
msg	the message to write as an vector of unsigned char

Definition at line 256 of file Controller.cpp.

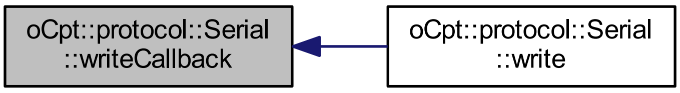
References msgQueue_, and writeStart().

Referenced by write().

Here is the call graph for this function:


oCpt::protocol::Serial ::writeCallback
oCpt::protocol::Serial ::writeStart
oCpt::protocol::Serial ::writeComplete
oCpt::protocol::Serial ::closeCallback

Here is the caller graph for this function:


oCpt::protocol::Serial ::writeCallback
oCpt::protocol::Serial ::write

6.40.4.17 writeComplete()

```
void oCpt::protocol::Serial::writeComplete (
             const boost::system::error_code & error )  [protected]
```

restart the write process when the previous write is finished

Parameters

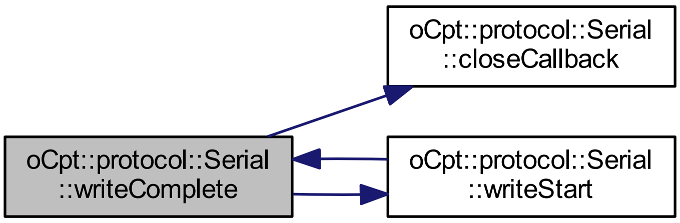
error	

Definition at line 275 of file Controller.cpp.

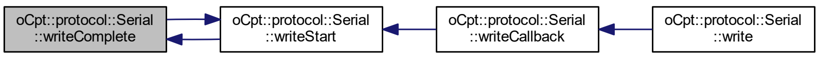
References closeCallback(), msgQueue_, and writeStart().

Referenced by writeStart().

Here is the call graph for this function:


oCpt::protocol::Serial ::closeCallback
oCpt::protocol::Serial ::writeComplete
oCpt::protocol::Serial ::writeStart

Here is the caller graph for this function:


oCpt::protocol::Serial ::writeComplete
oCpt::protocol::Serial ::writeStart
oCpt::protocol::Serial ::writeCallback
oCpt::protocol::Serial ::write

6.40.4.18 writeStart()

```
void oCpt::protocol::Serial::writeStart ( )  [protected]
```

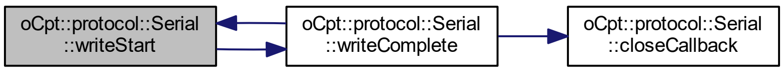
Start with the write sequence

Definition at line 264 of file Controller.cpp.

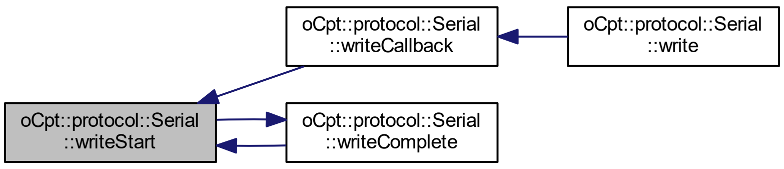
References msgQueue_, serialport_, and writeComplete().

Referenced by writeCallback(), and writeComplete().

Here is the call graph for this function:


oCpt::protocol::Serial ::writeStart
oCpt::protocol::Serial ::writeComplete
oCpt::protocol::Serial ::closeCallback

Here is the caller graph for this function:


oCpt::protocol::Serial ::writeCallback
oCpt::protocol::Serial ::write
oCpt::protocol::Serial ::writeStart
oCpt::protocol::Serial ::writeComplete

6.40.5 Member Data Documentation

6.40.5.1 baudrate_

```
unsigned int oCpt::protocol::Serial::baudrate_  [protected]
```

Definition at line 546 of file Controller.h.

Referenced by open().

6.40.5.2 callback_

```
cb_func oCpt::protocol::Serial::callback_  [protected]
```

Definition at line 543 of file Controller.h.

Referenced by readComplete(), Serial(), and setReadCallback().

6.40.5.3 csize_

```
character_size_t oCpt::protocol::Serial::csize_  [protected]
```

Definition at line 548 of file Controller.h.

Referenced by open().

6.40.5.4 device_

```
std::string oCpt::protocol::Serial::device_  [protected]
```

Definition at line 545 of file Controller.h.

Referenced by open().

6.40.5.5 firstMsg

```
bool oCpt::protocol::Serial::firstMsg = true  [protected]
```

Definition at line 552 of file Controller.h.

6.40.5.6 flow_

```
flow_control_t oCpt::protocol::Serial::flow_  [protected]
```

Definition at line 549 of file Controller.h.

Referenced by open().

6.40.5.7 ioservice_

```
io_service_t oCpt::protocol::Serial::ioservice_  [protected]
```

Definition at line 544 of file Controller.h.

Referenced by close(), open(), setIOservice(), start(), and write().

6.40.5.8 maxReadLength_

```
unsigned int oCpt::protocol::Serial::maxReadLength_  [protected]
```

Definition at line 529 of file Controller.h.

Referenced by ReadStart(), and setMaxReadLength().

6.40.5.9 msg_

```
std::string oCpt::protocol::Serial::msg_  [protected]
```

Definition at line 540 of file Controller.h.

6.40.5.10 msgQueue_

```
std::deque<std::vector<unsigned char> > oCpt::protocol::Serial::msgQueue_  [protected]
```

Definition at line 538 of file Controller.h.

Referenced by writeCallback(), writeComplete(), and writeStart().

6.40.5.11 msgRecievedSig

```
signal_t oCpt::protocol::Serial::msgRecievedSig
```

The signal which is send when a new line has been recieved or the buffer is full

Definition at line 484 of file Controller.h.

Referenced by oCpt::LoRa::initialize(), and internalCallback().

6.40.5.12 parity_

```
parity_t oCpt::protocol::Serial::parity_  [protected]
```

Definition at line 547 of file Controller.h.

Referenced by open().

6.40.5.13 read_msg

```
unsigned char oCpt::protocol::Serial::read_msg[4096]  [protected]
```

Definition at line 542 of file Controller.h.

Referenced by readComplete(), and ReadStart().

6.40.5.14 receivedMsg_

```
std::string oCpt::protocol::Serial::receivedMsg_  [protected]
```

Definition at line 541 of file Controller.h.

Referenced by internalCallback().

6.40.5.15 returnMsgQueue_

```
std::deque<std::string> oCpt::protocol::Serial::returnMsgQueue_ [protected]
```

Definition at line 539 of file Controller.h.

Referenced by getReturnMsgQueue(), internalCallback(), and readFiFoMsg().

6.40.5.16 serialport_

```
serialport_t oCpt::protocol::Serial::serialport_ [protected]
```

Definition at line 551 of file Controller.h.

Referenced by close(), closeCallback(), isOpen(), open(), ReadStart(), and writeStart().

6.40.5.17 stop_

```
stop_bits_t oCpt::protocol::Serial::stop_ [protected]
```

Definition at line 550 of file Controller.h.

Referenced by open().

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

- include/Core/Controller.h
- src/Core/Controller.cpp

6.41 oCpt::iSensor::State Struct Reference

```
#include <Sensor.h>
```

Public Attributes

- generic_t Value
- World::Time::timepoint_t Stamp

6.41.1 Detailed Description

Definition at line 35 of file Sensor.h.

6.41.2 Member Data Documentation

6.41.2.1 Stamp

```
World::Time::timepoint_t oCpt::iSensor::State::Stamp
```

Definition at line 37 of file Sensor.h.

Referenced by oCpt::components::sensors::Gps::interpretMsg(), oCpt::components::sensors::Razor::msg↩
Handler(), oCpt::components::sensors::Razor::Razor(), oCpt::components::sensors::PT100::updateSensor(), and oCpt::components::sensors::Razor::updateSensor().

6.41.2.2 Value

```
generic_t oCpt::iSensor::State::Value
```

Definition at line 36 of file Sensor.h.

Referenced by oCpt::components::sensors::Gps::interpretMsg(), oCpt::iSensor::iSensor(), oCpt::components↩
::sensors::Razor::msgHandler(), oCpt::components::sensors::Razor::Razor(), and oCpt::components::sensors::P↩
T100::updateSensor().

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

- include/Core/Sensor.h

6.42 oCpt::iTask::Status Class Reference

```
#include <Task.h>
```

Public Types

- typedef boost::shared_ptr< iTask::Status > ptr

  Boost shared_ptr to the task status.

Public Member Functions

- Status ()
- virtual ∼Status ()
- double progress ()
- bool running ()
- bool successful ()

Private Attributes

- double _progress = 0.0
- bool _running = false
- bool _successful

6.42.1 Detailed Description

Definition at line 27 of file Task.h.

6.42.2 Member Typedef Documentation

6.42.2.1 ptr

```
typedef boost::shared_ptr<iTask::Status> oCpt::iTask::Status::ptr
```

Boost shared_ptr to the task status.

Definition at line 30 of file Task.h.

6.42.3 Constructor & Destructor Documentation

6.42.3.1 Status()

```
oCpt::iTask::Status::Status ( )
```

Constructor of the iTask

Returns

Definition at line 15 of file Task.cpp.

6.42.3.2 ∼Status()

```
oCpt::iTask::Status::∼Status ( ) [virtual]
```

Deconstructor

Definition at line 17 of file Task.cpp.

6.42.4 Member Function Documentation

6.42.4.1 progress()

```
double oCpt::iTask::Status::progress ( )
```

Show the progress of the task

Returns

double between 0..1

Definition at line 19 of file Task.cpp.

6.42.4.2 running()

```
bool oCpt::iTask::Status::running ( )
```

Returns the running state of the task

Returns

bool where running is true

Definition at line 21 of file Task.cpp.

6.42.4.3 successful()

```
bool oCpt::iTask::Status::successful ( )
```

Returns if the task was completed succesfully

Returns

bool where a succesfully completed task is true, task in progress or failed are false

Definition at line 23 of file Task.cpp.

6.42.5 Member Data Documentation

6.42.5.1 _progress

```
double oCpt::iTask::Status::_progress = 0.0  [private]
```

Definition at line 63 of file Task.h.

6.42.5.2 _running

```
bool oCpt::iTask::Status::_running = false  [private]
```

Definition at line 64 of file Task.h.

6.42.5.3 _successful

```
bool oCpt::iTask::Status::_successful  [private]
```

Initial value:

```
=
                  false
```

Definition at line 65 of file Task.h.

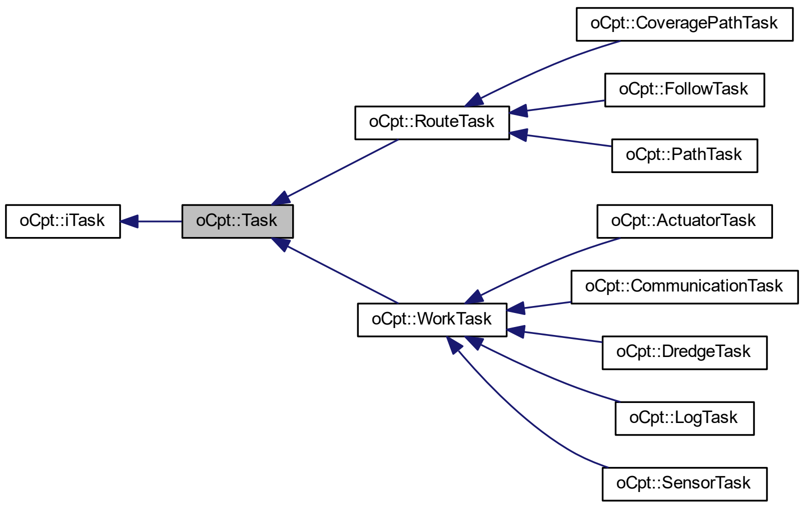
The documentation for this class was generated from the following files:

- include/Core/Task.h
- src/Core/Task.cpp

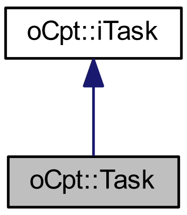
6.43 oCpt::Task Class Reference

```
#include <Task.h>
```

Inheritance diagram for oCpt::Task:



Collaboration diagram for oCpt::Task:



Public Member Functions

- Task (Vessel::ptr vessel, bool concurrent=false)
- virtual ∼Task ()
- virtual void start ()
- virtual iTask::Status::ptr status ()
- virtual void stop ()

Protected Attributes

- iTask::Status::ptr _status

    a boost share_ptr to the status of a task
- TypeOf _typeof

    Indicating the type of a task.

Additional Inherited Members

6.43.1 Detailed Description

The Base Task class

Definition at line 111 of file Task.h.

6.43.2 Constructor & Destructor Documentation

6.43.2.1 Task()

```
oCpt::Task::Task (
            Vessel::ptr vessel,
            bool concurrent = false )
```

The contructor

Returns

Definition at line 25 of file Task.cpp.

References _status.

6.43.2.2 ∼Task()

```
oCpt::Task::∼Task ( )  [virtual]
```

The deconstructor

Definition at line 29 of file Task.cpp.

6.43.3 Member Function Documentation

6.43.3.1 start()

```
void oCpt::Task::start ( )  [virtual]
```

The start command for a task

Implements oCpt::iTask.

Definition at line 31 of file Task.cpp.

References oCpt::iTask::Work.

6.43.3.2 status()

```
iTask::Status::ptr oCpt::Task::status ( ) [virtual]
```

Retrieves the Status of a task

Returns

Boost shared_ptr of the task status

Implements oCpt::iTask.

Definition at line 37 of file Task.cpp.

References _status.

6.43.3.3 stop()

```
void oCpt::Task::stop ( ) [virtual]
```

The stop command for a task

Implements oCpt::iTask.

Definition at line 39 of file Task.cpp.

6.43.4 Member Data Documentation

6.43.4.1 _status

```
iTask::Status::ptr oCpt::Task::_status [protected]
```

a boost share_ptr to the status of a task

Definition at line 141 of file Task.h.

Referenced by status(), and Task().

6.43.4.2 _typeof

```
TypeOf oCpt::Task::_typeof [protected]
```

Indicating the type of a task.

Definition at line 142 of file Task.h.

Referenced by oCpt::RouteTask::RouteTask(), and oCpt::WorkTask::WorkTask().

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

- include/Core/Task.h
- src/Core/Task.cpp

6.44 oCpt::World::Time Class Reference

```
#include <World.h>
```

Classes

- class Log

Public Types

- typedef boost::shared_ptr< Time > ptr

  Boost shared_ptr to a Time class.
- typedef boost::chrono::steady_clock::period tick_period

  a tick period for a steady clock
- typedef boost::chrono::steady_clock clock_t
- typedef boost::chrono::time_point< clock_t > timepoint_t
- template<typename T >
  using History = std::vector< boost::shared_ptr< Log< T >>>

Public Member Functions

- Time ()
- virtual ~Time ()
- clock_t & getTimeClock ()
- timepoint_t now ()

Private Attributes

- clock_t timeClock_

6.44.1 Detailed Description

The Time class all things time related, which allow for easy consite time manupulation trhough out the classes

Definition at line 24 of file World.h.

6.44.2 Member Typedef Documentation

6.44.2.1 clock_t

```
typedef boost::chrono::steady_clock oCpt::World::Time::clock_t
```

Definition at line 28 of file World.h.

6.44.2.2 History

```
template<typename T >
using oCpt::World::Time::History = std::vector<boost::shared_ptr<Log<T>>>
```

Definition at line 94 of file World.h.

6.44.2.3 ptr

```
typedef boost::shared_ptr<Time> oCpt::World::Time::ptr
```

Boost shared_ptr to a Time class.

Definition at line 26 of file World.h.

6.44.2.4 tick_period

```
typedef boost::chrono::steady_clock::period oCpt::World::Time::tick_period
```

a tick period for a steady clock

Definition at line 27 of file World.h.

6.44.2.5 timepoint_t

```
typedef boost::chrono::time_point<clock_t> oCpt::World::Time::timepoint_t
```

Definition at line 29 of file World.h.

6.44.3 Constructor & Destructor Documentation

6.44.3.1 Time()

```
oCpt::World::Time::Time ( )
```

Constructor of the Time class

Definition at line 11 of file World.cpp.

6.44.3.2 ∼Time()

```
oCpt::World::Time::∼Time ( ) [virtual]
```

Deconstructor of the Time class

Definition at line 13 of file World.cpp.

6.44.4 Member Function Documentation

6.44.4.1 getTimeClock()

`World::Time::clock_t & oCpt::World::Time::getTimeClock ( )`

get the current TimeClock

Returns

returns the time clock

Definition at line 15 of file World.cpp.

References timeClock_.

6.44.4.2 now()

`World::Time::timepoint_t oCpt::World::Time::now ( )`

Get the current time, as in now

Returns

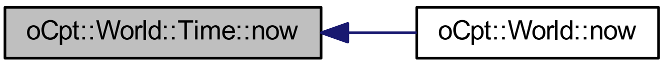
returns a timepoint_t which is now

Definition at line 19 of file World.cpp.

References timeClock_.

Referenced by oCpt::World::now().

Here is the caller graph for this function:



6.44.5 Member Data Documentation

6.44.5.1 timeClock_

`clock_t oCpt::World::Time::timeClock_ [private]`

Definition at line 31 of file World.h.

Referenced by getTimeClock(), and now().

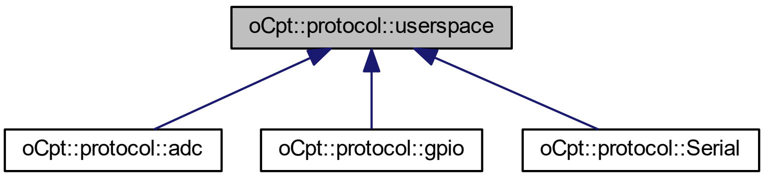
The documentation for this class was generated from the following files:

- include/Core/World.h
- src/Core/World.cpp

6.45 oCpt::protocol::userspace Class Reference

```
#include <Controller.h>
```

Inheritance diagram for oCpt::protocol::userspace:



Public Member Functions

- userspace ()
- virtual ~userspace ()

Protected Member Functions

- bool modLoaded (std::string modName)
- bool fileExist (std::string fileName)
- bool dtboLoaded (std::string dtboName)

Protected Attributes

- std::mutex usMutex

6.45.1 Detailed Description

Functions and routines related to the Linux userspace. Checking if a file exist, if capes or modules are loaded etc.

Definition at line 45 of file Controller.h.

6.45.2 Constructor & Destructor Documentation

6.45.2.1 userspace()

```
oCpt::protocol::userspace::userspace ( )
```

The constructor

Definition at line 16 of file Controller.cpp.

6.45.2.2 ∼userspace()

```
oCpt::protocol::userspace::∼userspace ( )  [virtual]
```

The deconstructor

Definition at line 18 of file Controller.cpp.

6.45.3 Member Function Documentation

6.45.3.1 dtboLoaded()

```
bool oCpt::protocol::userspace::dtboLoaded (
             std::string dtboName )  [protected]
```

Checks if a Device Tree overlay is loaded

Parameters

dtboName	The devicetree overlay as a string

Returns

either true or false

Definition at line 45 of file Controller.cpp.

References BBB_CAPE_MNGR.

6.45.3.2 fileExist()

```
bool oCpt::protocol::userspace::fileExist (
             std::string fileName )  [protected]
```

Checks if a file exist

Parameters

fileName	the filename as string

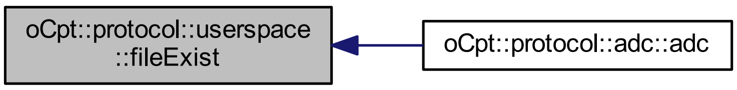
Returns

either true or false

Definition at line 39 of file Controller.cpp.

Referenced by oCpt::protocol::adc::adc().

Here is the caller graph for this function:



6.45.3.3 modLoaded()

```
bool oCpt::protocol::userspace::modLoaded (
             std::string modName )  [protected]
```

Checks if a Linux module is loaded

Parameters

modName	the name of the module as string

Returns

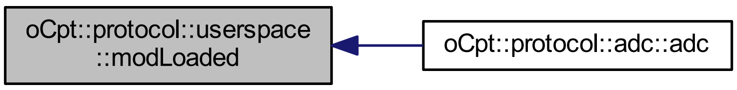
either true or false

Definition at line 20 of file Controller.cpp.

References MODULE_PATH.

Referenced by oCpt::protocol::adc::adc().

Here is the caller graph for this function:



6.45.4 Member Data Documentation

6.45.4.1 usMutex

```
std::mutex oCpt::protocol::userspace::usMutex  [protected]
```

The standard Mutex TODO check if this is really needed for the current setup

Definition at line 83 of file Controller.h.

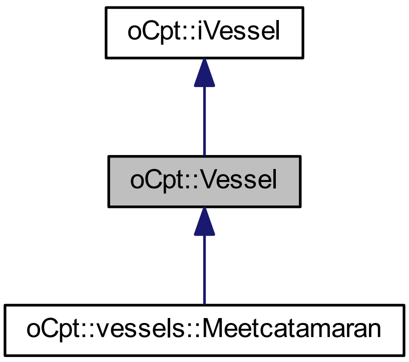
The documentation for this class was generated from the following files:

- include/Core/Controller.h
- src/Core/Controller.cpp

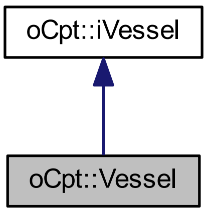
6.46 oCpt::Vessel Class Reference

```
#include <Vessel.h>
```

Inheritance diagram for oCpt::Vessel:



Collaboration diagram for oCpt::Vessel:



Public Member Functions

- Vessel ()
- Vessel (iController::ptr controller)
- virtual ∼Vessel ()
- virtual void initialize () override
- virtual void run () override
- virtual void stop () override

Protected Attributes

- World::ptr world_

  a shared_ptr to the world needed for time and location keeping
- iController::ptr controller_

  a shared_ptr to the controller needed for sensors, actuators and coommunication
- iCaptain::ptr captain_

  The captain for strategical planning.
- iBoatswain::ptr boatswain_

  The boatswain, the worker asynchronize operations for actuators, sensors, and communication.
- std::vector< iSensor::ptr > sensors_
- std::vector< iActuator::ptr > actuators_
- std::vector< iComm::ptr > comm_

Additional Inherited Members

6.46.1 Detailed Description

The vessel base class

Definition at line 80 of file Vessel.h.

6.46.2 Constructor & Destructor Documentation

6.46.2.1 Vessel() [1/2]

```
oCpt::Vessel::Vessel ( )
```

The constructor for a vessel

Returns

Definition at line 23 of file Vessel.cpp.

References oCpt::iVessel::stopThread_.

6.46.2.2 Vessel() [2/2]

```
oCpt::Vessel::Vessel (
            iController::ptr controller )
```

The constructor for a vessel

Parameters

controller	shared_ptr to the controller

Returns

Definition at line 33 of file Vessel.cpp.

References boatswain_, captain_, controller_, and world_.

6.46.2.3 ∼Vessel()

```
oCpt::Vessel::∼Vessel ( )  [virtual]
```

The deconstructor

Definition at line 40 of file Vessel.cpp.

6.46.3 Member Function Documentation

6.46.3.1 initialize()

```
void oCpt::Vessel::initialize ( )  [override], [virtual]
```

Initialize the vessel

Implements oCpt::iVessel.

Definition at line 42 of file Vessel.cpp.

References boatswain_, and captain_.

6.46.3.2 run()

```
void oCpt::Vessel::run ( )  [override], [virtual]
```

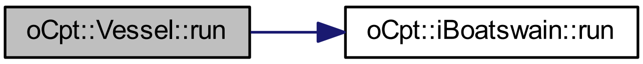
Run the vessel normal operations

Implements oCpt::iVessel.

Definition at line 47 of file Vessel.cpp.

References boatswain_, captain_, and oCpt::iBoatswain::run().

Here is the call graph for this function:



6.46.3.3 stop()

```
void oCpt::Vessel::stop ( )  [override], [virtual]
```

Stop the vessel, everything except critical parts, which are needed to survive

Implements oCpt::iVessel.

Definition at line 54 of file Vessel.cpp.

References oCpt::iVessel::stopThread_.

6.46.4 Member Data Documentation

6.46.4.1 actuators_

```
std::vector<iActuator::ptr> oCpt::Vessel::actuators_  [protected]
```

Definition at line 121 of file Vessel.h.

6.46.4.2 boatswain_

```
iBoatswain::ptr oCpt::Vessel::boatswain_  [protected]
```

The boatswain, the worker asynchronize operations for actuators, sensors, and communication.

Definition at line 119 of file Vessel.h.

Referenced by initialize(), run(), and Vessel().

6.46.4.3 captain_

```
iCaptain::ptr oCpt::Vessel::captain_  [protected]
```

The captain for strategical planning.

Definition at line 118 of file Vessel.h.

Referenced by initialize(), run(), and Vessel().

6.46.4.4 comm_

```
std::vector<iComm::ptr> oCpt::Vessel::comm_  [protected]
```

Definition at line 122 of file Vessel.h.

6.46.4.5 controller_

```
iController::ptr oCpt::Vessel::controller_  [protected]
```

a shared_ptr to the controller needed for sensors, actuators and coommunication

Definition at line 117 of file Vessel.h.

Referenced by Vessel().

6.46.4.6 sensors_

```
std::vector<iSensor::ptr> oCpt::Vessel::sensors_  [protected]
```

Definition at line 120 of file Vessel.h.

6.46.4.7 world_

```
World::ptr oCpt::Vessel::world_  [protected]
```

a shared_ptr to the world needed for time and location keeping

Definition at line 116 of file Vessel.h.

Referenced by Vessel().

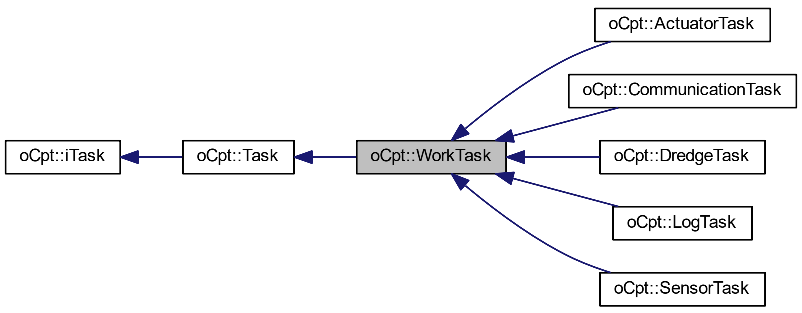
The documentation for this class was generated from the following files:

- include/Core/Vessel.h
- src/Core/Vessel.cpp

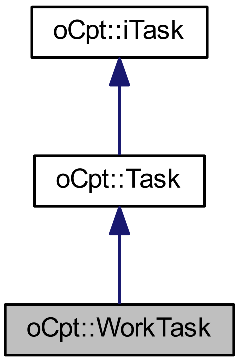
6.47 oCpt::WorkTask Class Reference

```
#include <Task.h>
```

Inheritance diagram for oCpt::WorkTask:



Collaboration diagram for oCpt::WorkTask:



Public Member Functions

- WorkTask (Vessel::ptr vessel, bool concurrent=false)
- virtual ∼WorkTask ()

Additional Inherited Members

6.47.1 Detailed Description

An object representing work related tasks

Definition at line 167 of file Task.h.

6.47.2 Constructor & Destructor Documentation

6.47.2.1 WorkTask()

```
oCpt::WorkTask::WorkTask (
            Vessel::ptr vessel,
            bool concurrent = false )
```

Constructor of the interface

Returns

Definition at line 47 of file Task.cpp.

References oCpt::Task::_typeof.

6.47.2.2 ∼WorkTask()

```
oCpt::WorkTask::∼WorkTask ( )  [virtual]
```

The deconstructor

Definition at line 51 of file Task.cpp.

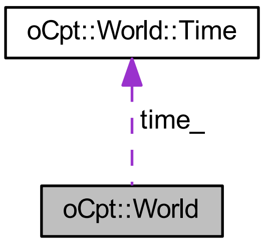
The documentation for this class was generated from the following files:

- include/Core/Task.h
- src/Core/Task.cpp

6.48 oCpt::World Class Reference

```
#include <World.h>
```

Collaboration diagram for oCpt::World:



Classes

- class Location
- class Time

Public Types

- typedef boost::shared_ptr< World > ptr

  Boost shared_ptr to a World class.

Public Member Functions

- World ()
- virtual ∼World ()
- Time & getTime ()
- Time::timepoint_t now ()

Protected Attributes

- Time time_

6.48.1 Detailed Description

The World class, this class is an shared pointer where the boatswain can place the state representation of the vessel at a certain time, which allows the captain to plan the strategic decisions

Definition at line 17 of file World.h.

6.48.2 Member Typedef Documentation

6.48.2.1 ptr

```
typedef boost::shared_ptr<World> oCpt::World::ptr
```

Boost shared_ptr to a World class.

Definition at line 19 of file World.h.

6.48.3 Constructor & Destructor Documentation

6.48.3.1 World()

```
oCpt::World::World ( )
```

Constructor for a World

Definition at line 23 of file World.cpp.

6.48.3.2 ∼World()

```
oCpt::World::∼World ( )  [virtual]
```

Deconstuctor for a World

Definition at line 27 of file World.cpp.

6.48.4 Member Function Documentation

6.48.4.1 getTime()

```
World::Time & oCpt::World::getTime ( )
```

get the current time object

Returns

returns Time

Definition at line 29 of file World.cpp.

References time_.

6.48.4.2 now()

```
World::Time::timepoint_t oCpt::World::now ( )
```

Get the current Epoch

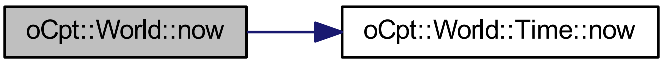
Returns

returns a timepoint representing now

Definition at line 33 of file World.cpp.

References oCpt::World::Time::now(), and time_.

Here is the call graph for this function:



6.48.5 Member Data Documentation

6.48.5.1 time_

```
Time oCpt::World::time_ [protected]
```

Definition at line 211 of file World.h.

Referenced by getTime(), and now().

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

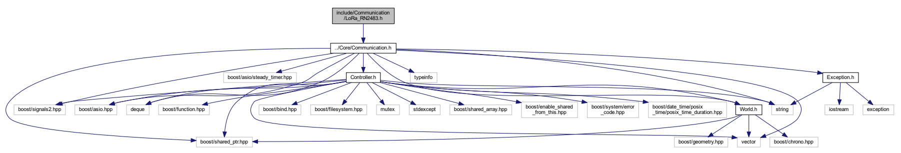
- include/Core/World.h
- src/Core/World.cpp

Chapter 7

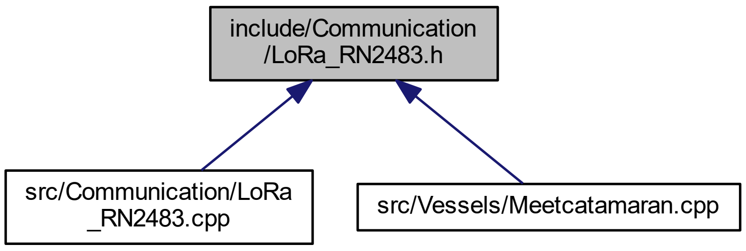
File Documentation

7.1 include/Communication/LoRa_RN2483.h File Reference

`#include "../Core/Communication.h"`
Include dependency graph for LoRa_RN2483.h:



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



Classes

- class oCpt::components::comm::LoRa_RN2483

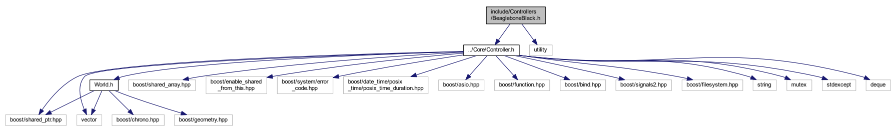
Namespaces

- oCpt
- oCpt::components
- oCpt::components::comm

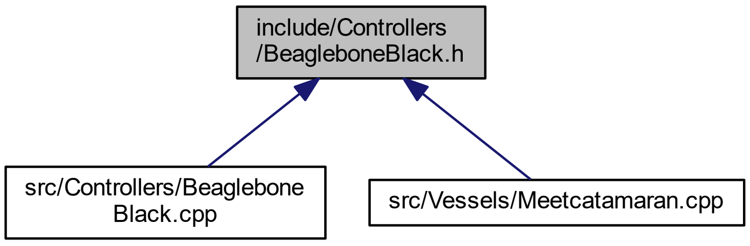
7.2 include/Controllers/BeagleboneBlack.h File Reference

```
#include "../Core/Controller.h"
#include <utility>
```

Include dependency graph for BeagleboneBlack.h:



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



Classes

- class oCpt::components::controller::BBB

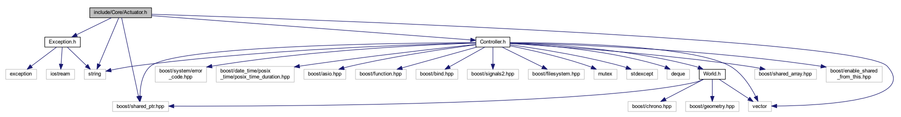
Namespaces

- oCpt
- oCpt::components
- oCpt::components::controller

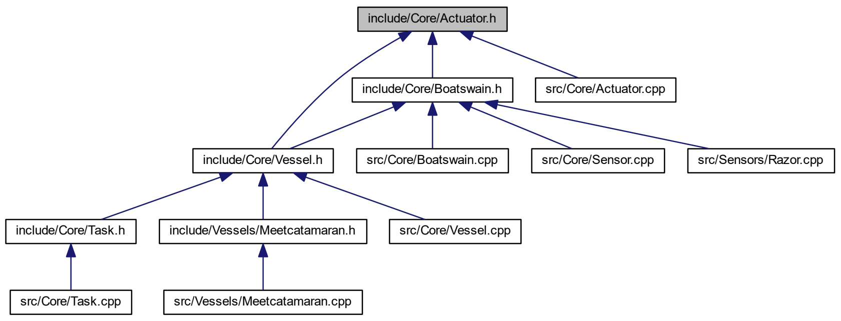
7.3 include/Core/Actuator.h File Reference

```
#include <boost/shared_ptr.hpp>
#include <string>
#include <vector>
#include "Controller.h"
#include "Exception.h"
```

Include dependency graph for Actuator.h:



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



Classes

- class oCpt::iActuator
- class oCpt::Actuator

Namespaces

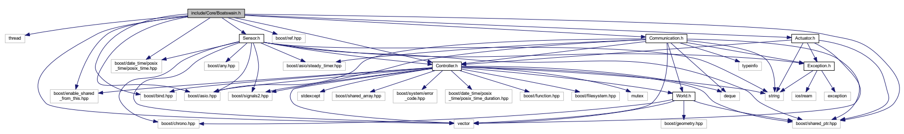
- oCpt

7.4 include/Core/Boatswain.h File Reference

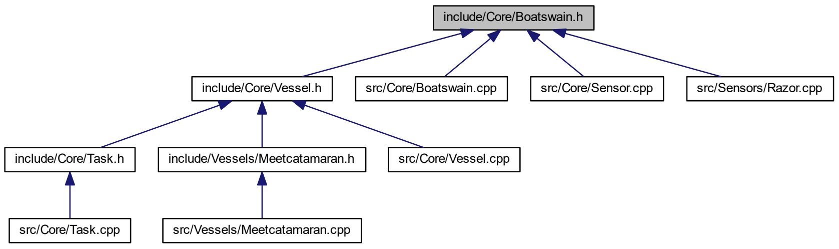
```
#include <thread>
#include <vector>
#include <boost/asio.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/shared_ptr.hpp>
#include <boost/enable_shared_from_this.hpp>
#include <boost/bind.hpp>
#include <boost/ref.hpp>
```

```
#include "Controller.h"
#include "Sensor.h"
#include "Actuator.h"
#include "Communication.h"
```

Include dependency graph for Boatswain.h:



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



Classes

- class oCpt::iBoatswain
- class oCpt::Boatswain

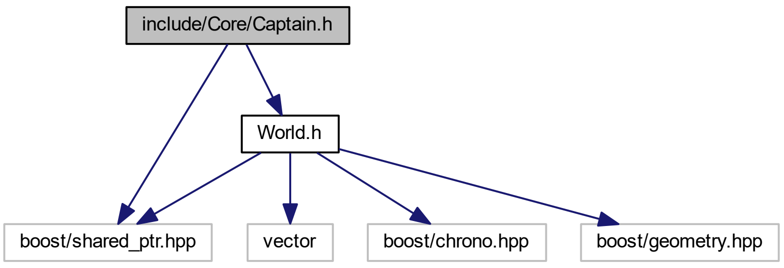
Namespaces

- oCpt

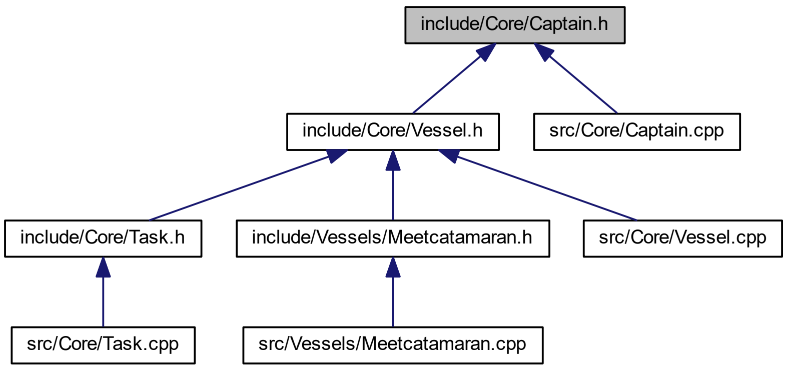
7.5 include/Core/Captain.h File Reference

```
#include <boost/shared_ptr.hpp>
#include "World.h"
```

Include dependency graph for Captain.h:



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



Classes

- class oCpt::iCaptain
- class oCpt::Captain

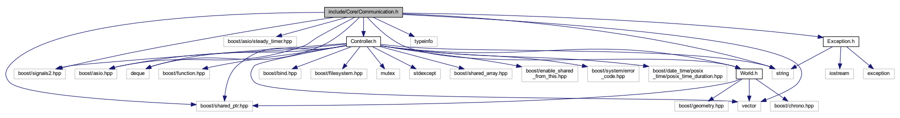
Namespaces

- oCpt

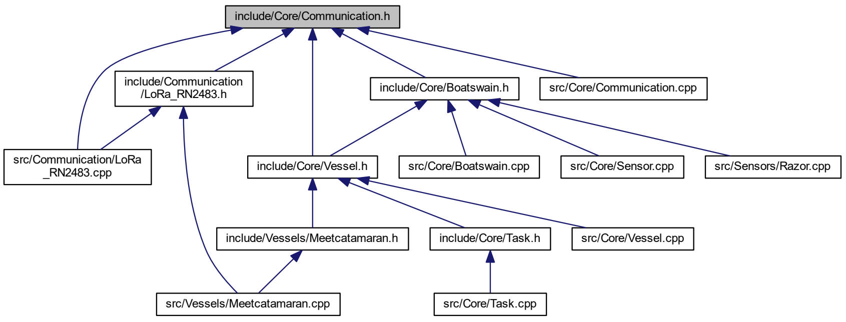
7.6 include/Core/Communication.h File Reference

```
#include <boost/shared_ptr.hpp>
#include <boost/signals2.hpp>
#include <boost/asio/steady_timer.hpp>
#include <boost/asio.hpp>
#include <string>
#include <vector>
#include <deque>
#include <typeinfo>
#include "Controller.h"
#include "World.h"
#include "Exception.h"
```

Include dependency graph for Communication.h:



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


include/Core/Communication.h
include/Communication/LoRa_RN2483.h
include/Core/Boatswain.h
src/Core/Communication.cpp
src/Communication/LoRa_RN2483.cpp
include/Core/Vessel.h
src/Core/Boatswain.cpp
src/Core/Sensor.cpp
src/Sensors/Razor.cpp
include/Vessels/Meetcatamaran.h
include/Core/Task.h
src/Core/Vessel.cpp
src/Vessels/Meetcatamaran.cpp
src/Core/Task.cpp

Classes

- class oCpt::iComm
- struct oCpt::iComm::Message
- class oCpt::LoRa

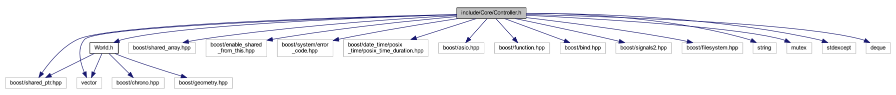
Namespaces

- oCpt

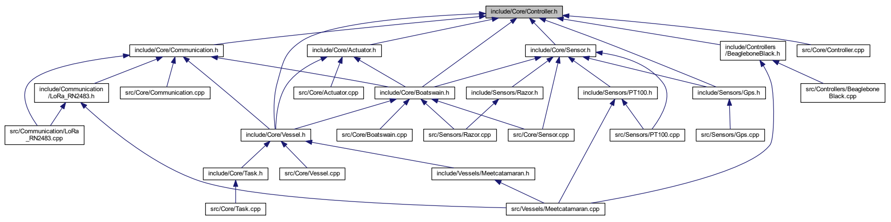
7.7 include/Core/Controller.h File Reference

```
#include <boost/shared_ptr.hpp>
#include <boost/shared_array.hpp>
#include <boost/enable_shared_from_this.hpp>
#include <boost/system/error_code.hpp>
#include <boost/date_time/posix_time/posix_time_duration.hpp>
#include <boost/asio.hpp>
#include <boost/function.hpp>
#include <boost/bind.hpp>
#include <boost/signals2.hpp>
#include <boost/filesystem.hpp>
#include <string>
#include <vector>
#include <mutex>
#include <stdexcept>
#include <deque>
#include "World.h"
```

Include dependency graph for Controller.h:



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



Classes

- class oCpt::protocol::userspace
- class oCpt::protocol::adc
- class oCpt::protocol::gpio
- class oCpt::protocol::Serial
- class oCpt::iController
- class oCpt::ARM

Namespaces

- oCpt
- oCpt::protocol

Macros

- #define MAX_READ_LENGTH 4096
- #define BBB_CAPE_MNGR "/sys/devices/platform/bone_capemgr/slots"
- #define GPIO_BASE_PATH "/sys/class/gpio/"
- #define ADC_IO_BASE_PATH "/sys/bus/iio/devices/iio:device"
- #define ADC_VOLTAGE_PATH "/in_voltage"
- #define ADC_VOLTAGE_SUB_PATH "_raw"
- #define MODULE_PATH "/proc/modules"

7.7.1 Macro Definition Documentation

7.7.1.1 ADC_IO_BASE_PATH

```
#define ADC_IO_BASE_PATH "/sys/bus/iio/devices/iio:device"
```

Definition at line 32 of file Controller.h.

Referenced by oCpt::protocol::adc::adc().

7.7.1.2 ADC_VOLTAGE_PATH

```
#define ADC_VOLTAGE_PATH "/in_voltage"
```

Definition at line 33 of file Controller.h.

Referenced by oCpt::protocol::adc::adc().

7.7.1.3 ADC_VOLTAGE_SUB_PATH

```
#define ADC_VOLTAGE_SUB_PATH "_raw"
```

Definition at line 34 of file Controller.h.

Referenced by oCpt::protocol::adc::adc().

7.7.1.4 BBB_CAPE_MNGR

```
#define BBB_CAPE_MNGR "/sys/devices/platform/bone_capemgr/slots"
```

Definition at line 28 of file Controller.h.

Referenced by oCpt::protocol::userspace::dtboLoaded().

7.7.1.5 GPIO_BASE_PATH

```
#define GPIO_BASE_PATH "/sys/class/gpio/"
```

Definition at line 30 of file Controller.h.

Referenced by oCpt::protocol::gpio::exportedGpios(), oCpt::protocol::gpio::exportPin(), oCpt::protocol::gpio::gpio(), oCpt::protocol::gpio::readPinValue(), oCpt::protocol::gpio::unexportPin(), and oCpt::protocol::gpio::writePinValue().

7.7.1.6 MAX_READ_LENGTH

```
#define MAX_READ_LENGTH 4096
```

Definition at line 26 of file Controller.h.

7.7.1.7 MODULE_PATH

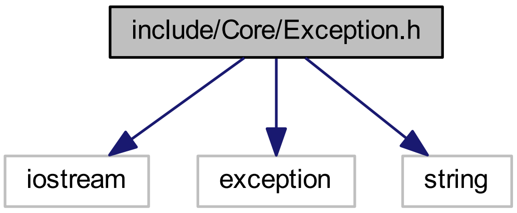
```
#define MODULE_PATH "/proc/modules"
```

Definition at line 36 of file Controller.h.

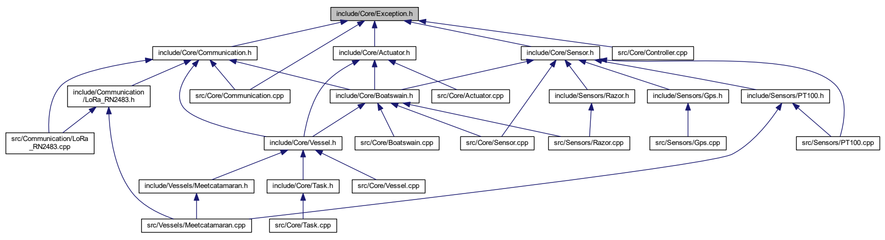
Referenced by oCpt::protocol::userspace::modLoaded().

7.8 include/Core/Exception.h File Reference

```
#include <iostream>
#include <exception>
#include <string>
```
Include dependency graph for Exception.h:



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



Classes

- class oCpt::oCptException

Namespaces

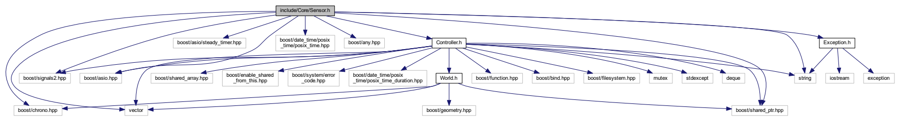
- oCpt

7.9 include/Core/Sensor.h File Reference

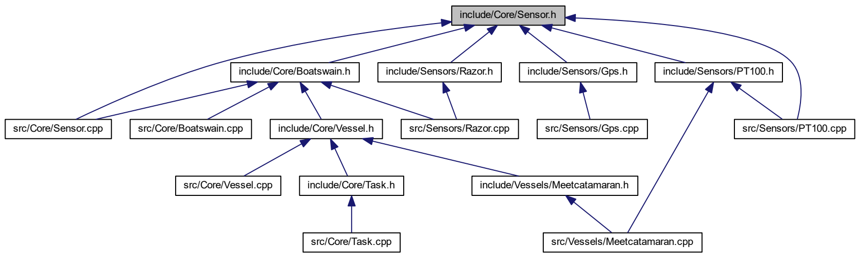
```
#include <boost/shared_ptr.hpp>
#include <boost/signals2.hpp>
#include <boost/chrono.hpp>
#include <boost/asio/steady_timer.hpp>
#include <boost/asio.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>
```

```
#include <boost/any.hpp>
#include <string>
#include <vector>
#include "Controller.h"
#include "Exception.h"
```

Include dependency graph for Sensor.h:



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



Classes

- class oCpt::iSensor
- struct oCpt::iSensor::State
- class oCpt::Sensor

Namespaces

- oCpt

Macros

- #define CAST(x, t)

7.9.1 Macro Definition Documentation

7.9.1.1 CAST

```
#define CAST(
                x,
                t )
```

Value:

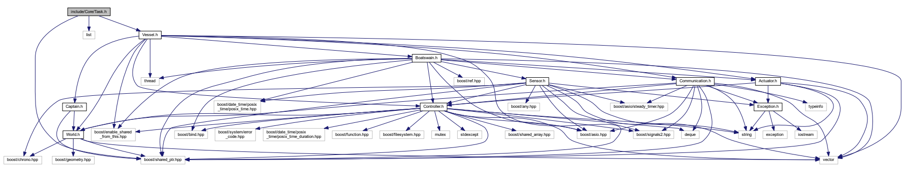
```
boost::any_cast<t::ReturnValue_t>(x) /*<! CAST the return value of a generic boost::any object, which can
        change for each sensor to a the proper return value. where the first parameter is the getState().Value and
        the second is the Sensor Class.
 */
```

Definition at line 21 of file Sensor.h.

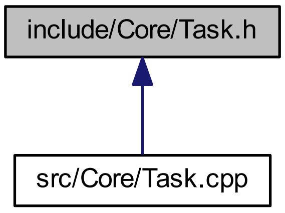
7.10 include/Core/Task.h File Reference

```
#include <boost/shared_ptr.hpp>
#include <list>
#include "Vessel.h"
```

Include dependency graph for Task.h:



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



Classes

- class oCpt::iTask

  Task interface, all tasks need to adhere to this structure.
- class oCpt::iTask::Status
- class oCpt::Task
- class oCpt::RouteTask
- class oCpt::WorkTask
- class oCpt::CoveragePathTask

  An object representing a coverage path task.
- class oCpt::FollowTask

  An object representing a follow the target task.
- class oCpt::PathTask

  An object representing a normal A to B type of path planning.
- class oCpt::LogTask

  An Object representing a data logging task.
- class oCpt::DredgeTask

  An Object representing a dredging task.
- class oCpt::SensorTask
- class oCpt::ActuatorTask
- class oCpt::CommunicationTask

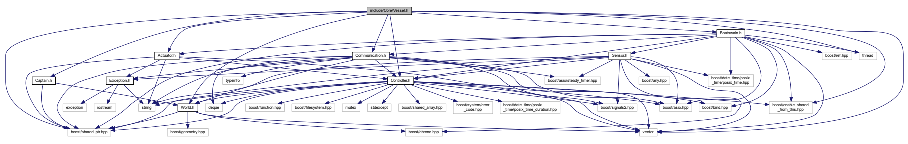
Namespaces

- oCpt

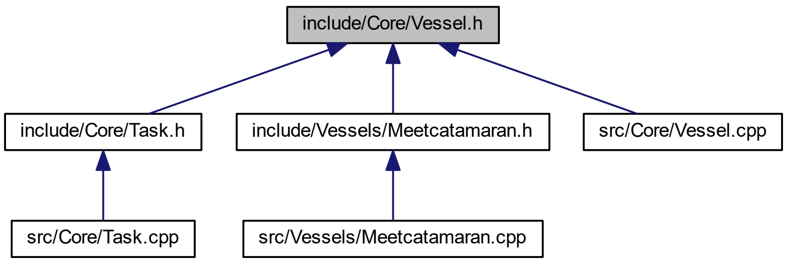
7.11 include/Core/Vessel.h File Reference

```
#include <boost/shared_ptr.hpp>
#include <boost/enable_shared_from_this.hpp>
#include <vector>
#include <thread>
#include "World.h"
#include "Controller.h"
#include "Captain.h"
#include "Boatswain.h"
#include "Actuator.h"
#include "Communication.h"
```

Include dependency graph for Vessel.h:



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



Classes

- class oCpt::iVessel
- class oCpt::Vessel

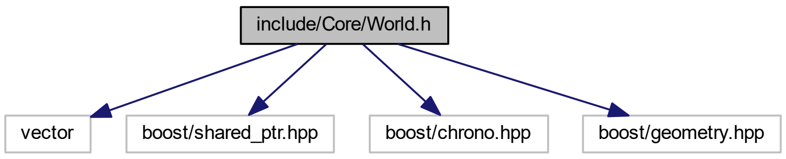
Namespaces

- oCpt

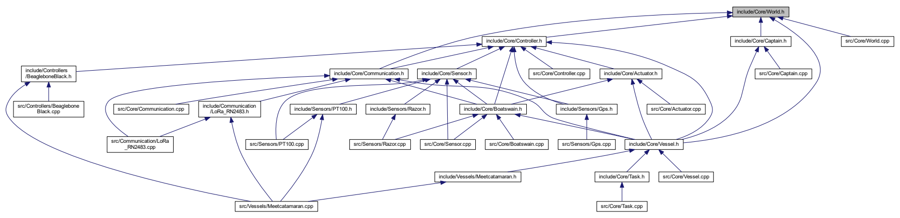
7.12 include/Core/World.h File Reference

```
#include <vector>
#include <boost/shared_ptr.hpp>
#include <boost/chrono.hpp>
#include <boost/geometry.hpp>
```

Include dependency graph for World.h:



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



Classes

- class oCpt::World
- class oCpt::World::Time
- class oCpt::World::Time::Log< T >
- class oCpt::World::Location
- struct oCpt::World::Location::coordinate
- struct oCpt::World::Location::gpsPoint
- struct oCpt::World::Location::RoutePoint

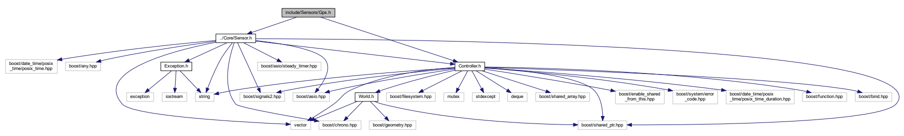
Namespaces

- oCpt

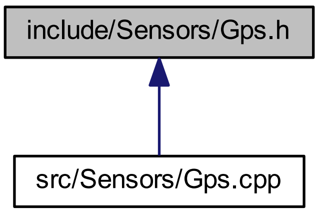
7.13 include/Sensors/Gps.h File Reference

```
#include "../Core/Sensor.h"
#include "../Core/Controller.h"
```

Include dependency graph for Gps.h:



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



Classes

- class oCpt::components::sensors::Gps

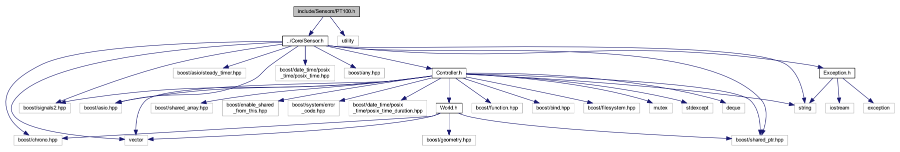
Namespaces

- oCpt
- oCpt::components
- oCpt::components::sensors

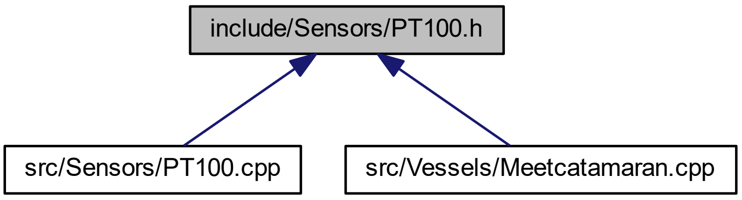
7.14 include/Sensors/PT100.h File Reference

```
#include "../Core/Sensor.h"
#include <utility>
```

Include dependency graph for PT100.h:



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



Classes

- class oCpt::components::sensors::PT100

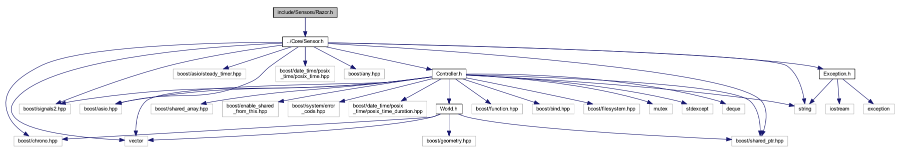
Namespaces

- oCpt
- oCpt::components
- oCpt::components::sensors

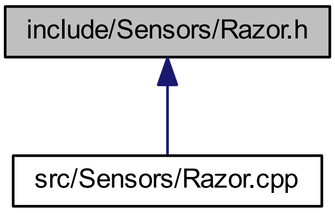
7.15 include/Sensors/Razor.h File Reference

```
#include "../Core/Sensor.h"
```

Include dependency graph for Razor.h:



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



Classes

- class oCpt::components::sensors::Razor
- struct oCpt::components::sensors::Razor::ReturnValue

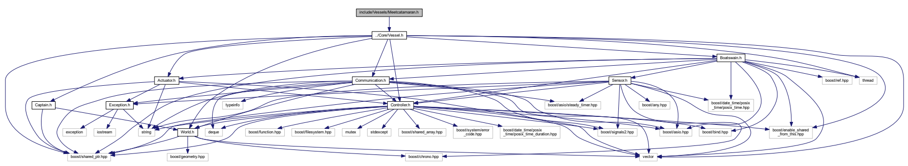
Namespaces

- oCpt
- oCpt::components
- oCpt::components::sensors

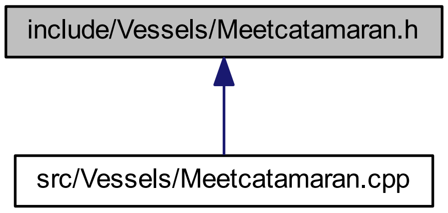
7.16 include/Vessels/Meetcatamaran.h File Reference

```
#include "../Core/Vessel.h"
```

Include dependency graph for Meetcatamaran.h:



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


include/Vessels/Meetcatamaran.h
src/Vessels/Meetcatamaran.cpp

Classes

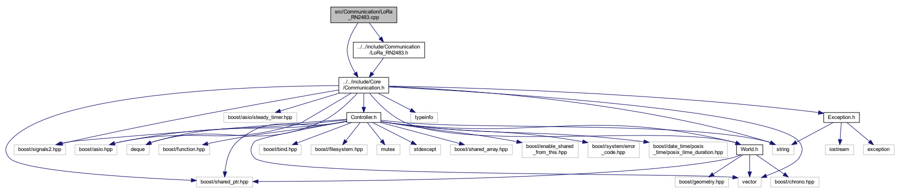
- class oCpt::vessels::Meetcatamaran

Namespaces

- oCpt
- oCpt::vessels

7.17 src/Communication/LoRa_RN2483.cpp File Reference

```
#include "../../include/Core/Communication.h"
#include "../../include/Communication/LoRa_RN2483.h"
```
Include dependency graph for LoRa_RN2483.cpp:

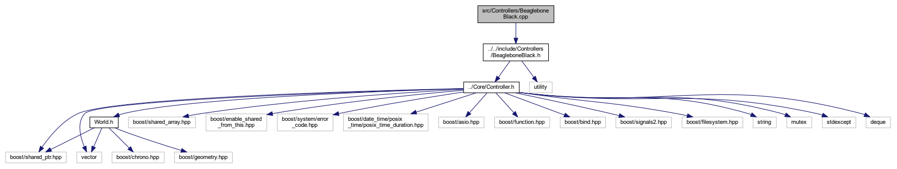


Namespaces

- oCpt
- oCpt::components
- oCpt::components::comm

7.18 src/Controllers/BeagleboneBlack.cpp File Reference

```
#include "../../include/Controllers/BeagleboneBlack.h"
```
Include dependency graph for BeagleboneBlack.cpp:

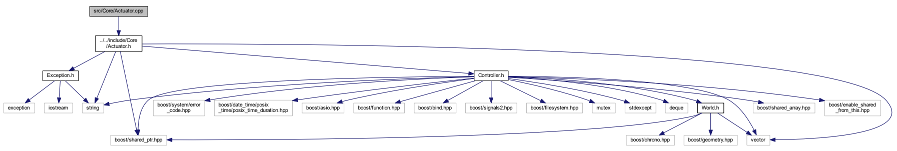


Namespaces

- oCpt
- oCpt::components
- oCpt::components::controller

7.19 src/Core/Actuator.cpp File Reference

```
#include "../../include/Core/Actuator.h"
```
Include dependency graph for Actuator.cpp:



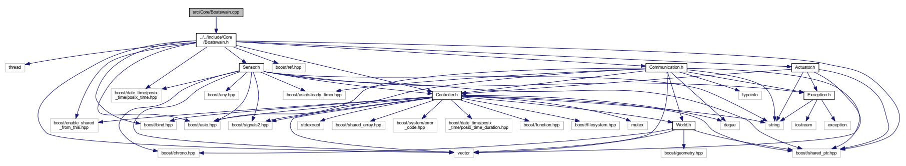
Namespaces

- oCpt

7.20 src/Core/Boatswain.cpp File Reference

```
#include "../../include/Core/Boatswain.h"
```

Include dependency graph for Boatswain.cpp:



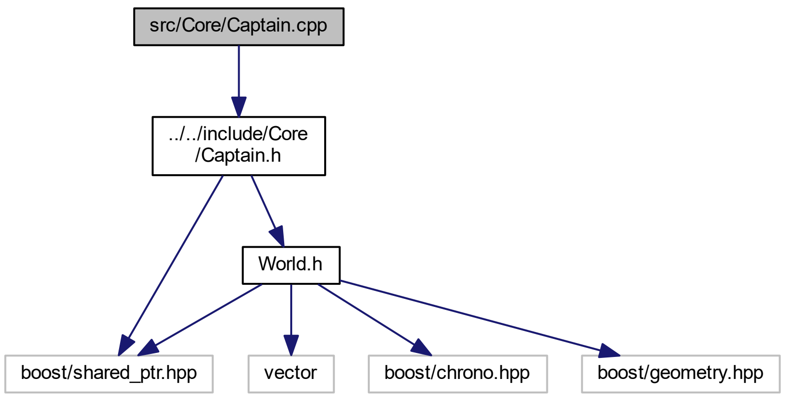
Namespaces

- oCpt

7.21 src/Core/Captain.cpp File Reference

```
#include "../../include/Core/Captain.h"
```

Include dependency graph for Captain.cpp:



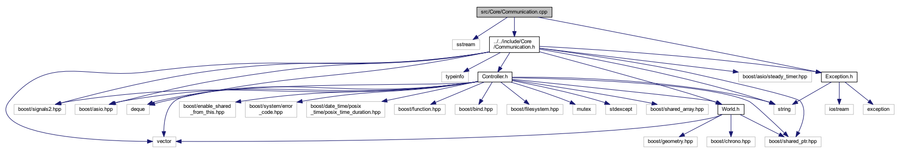
Namespaces

- oCpt

7.22 src/Core/Communication.cpp File Reference

```
#include <sstream>
#include "../../include/Core/Communication.h"
#include "../../include/Core/Exception.h"
```

Include dependency graph for Communication.cpp:



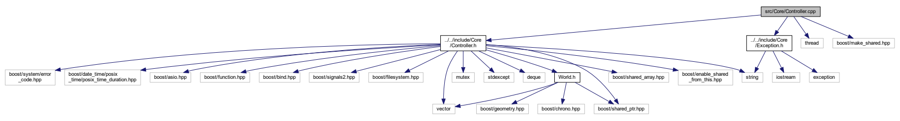
Namespaces

- oCpt

7.23 src/Core/Controller.cpp File Reference

```
#include "../../include/Core/Controller.h"
#include "../../include/Core/Exception.h"
#include <thread>
#include <boost/make_shared.hpp>
```

Include dependency graph for Controller.cpp:



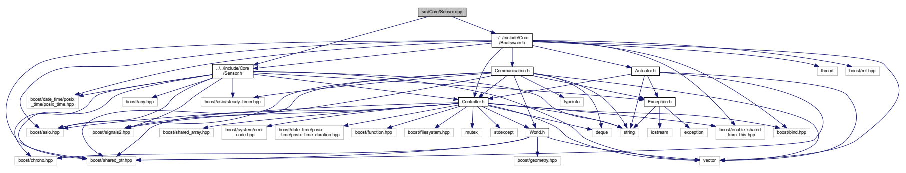
Namespaces

- oCpt
- oCpt::protocol

7.24 src/Core/Sensor.cpp File Reference

```
#include "../../include/Core/Sensor.h"
#include "../../include/Core/Boatswain.h"
```

Include dependency graph for Sensor.cpp:



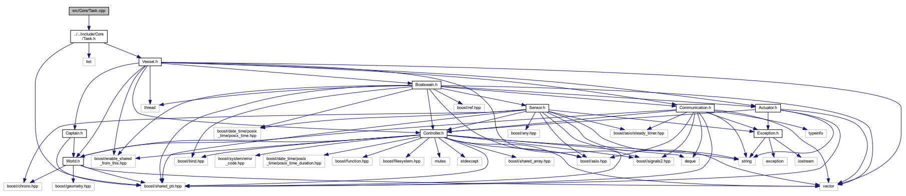
Namespaces

- oCpt

7.25 src/Core/Task.cpp File Reference

```
#include "../../include/Core/Task.h"
```

Include dependency graph for Task.cpp:



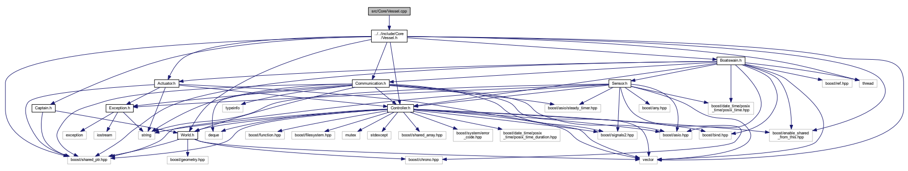
Namespaces

- oCpt

7.26 src/Core/Vessel.cpp File Reference

```
#include "../../include/Core/Vessel.h"
```

Include dependency graph for Vessel.cpp:



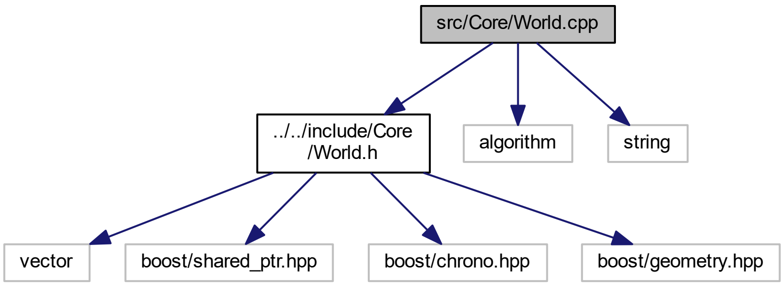
Namespaces

- oCpt

7.27 src/Core/World.cpp File Reference

```
#include "../../include/Core/World.h"
#include <algorithm>
#include <string>
```

Include dependency graph for World.cpp:



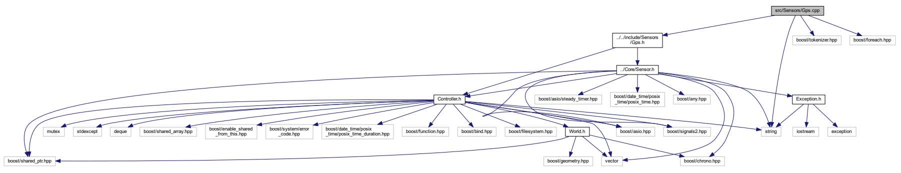
Namespaces

- oCpt

7.28 src/Sensors/Gps.cpp File Reference

```
#include "../../include/Sensors/Gps.h"
#include <boost/tokenizer.hpp>
#include <boost/foreach.hpp>
#include <string>
```

Include dependency graph for Gps.cpp:



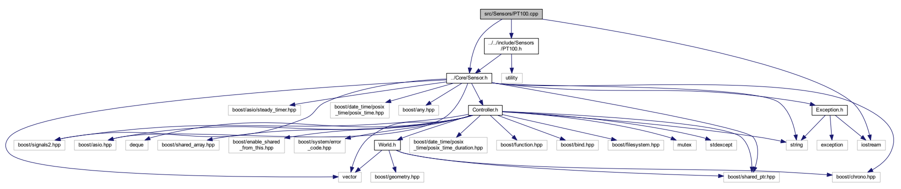
Namespaces

- oCpt
- oCpt::components
- oCpt::components::sensors

7.29 src/Sensors/PT100.cpp File Reference

```
#include "../../include/Sensors/PT100.h"
#include "../../include/Core/Sensor.h"
#include <iostream>
```

Include dependency graph for PT100.cpp:



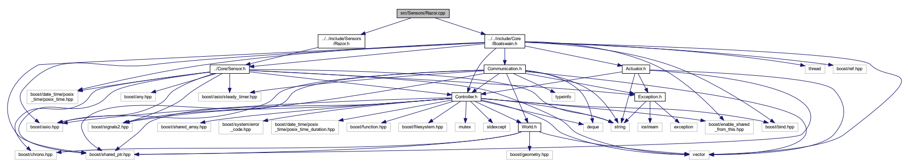
Namespaces

- oCpt
- oCpt::components
- oCpt::components::sensors

7.30 src/Sensors/Razor.cpp File Reference

```
#include "../../include/Sensors/Razor.h"
#include "../../include/Core/Boatswain.h"
```

Include dependency graph for Razor.cpp:



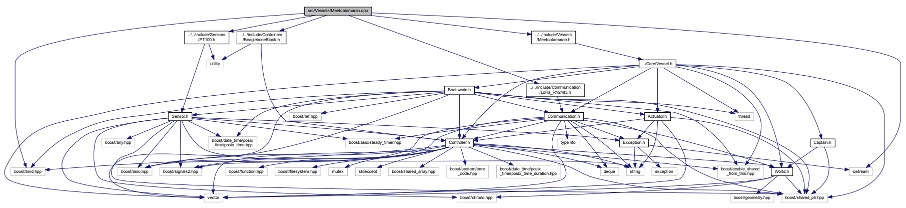
Namespaces

- oCpt
- oCpt::components
- oCpt::components::sensors

7.31 src/Vessels/Meetcatamaran.cpp File Reference

```
#include "../../include/Vessels/Meetcatamaran.h"
#include "../../include/Controllers/BeagleboneBlack.h"
#include "../../include/Sensors/PT100.h"
#include "../../include/Communication/LoRa_RN2483.h"
#include <boost/bind.hpp>
#include <iostream>
```

Include dependency graph for Meetcatamaran.cpp:



Namespaces

- oCpt
- oCpt::vessels