RESpecTa

1.0

Generated by Doxygen 1.7.3

Tue Aug 30 2011 23:37:12

Contents

Chapter 1

Class Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

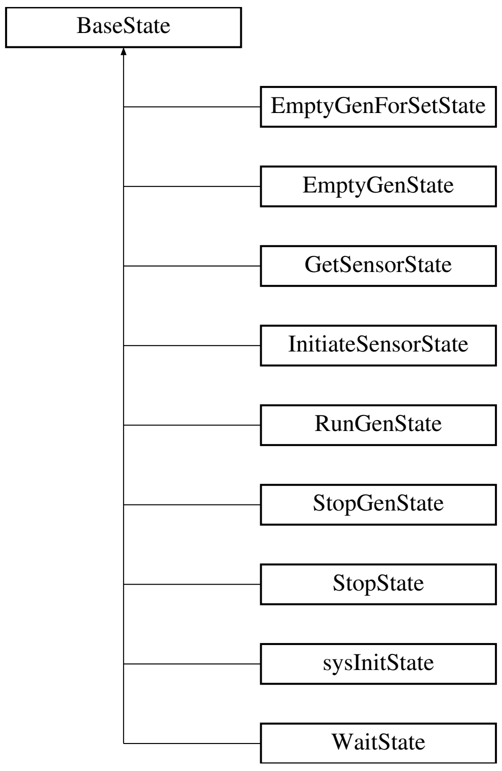
Chapter 3

Class Documentation

3.1 BaseState Class Reference

```
#include <baseState.h>
```

Inheritance diagram for BaseState:



Public Types

- enum { **Type** = UserType + 15 }

Public Member Functions

- void setSelected (bool selected)
- StateType getType ()
- void setType (StateType newType)
- QString getName ()
- void setName (QString newName)

- QString getArgument ()
- void setArgument (QString newArg)
- QString getParameters ()
- void setParameters (QString newParams)
- QGraphicsTextItem ∗ getNameTextItem ()
- QList< Transition ∗ > getSubtaskTransitions ()
- void updateSize ()
- void updateTextPositions ()
- virtual QStringList LoadFromXML (QXmlStreamReader ∗reader)
- virtual int itemCount ()
- virtual TreeItem ∗ getChild (int i, TreeItem ∗parent)
- virtual bool equals (BaseState ∗other)
- BaseState & operator= (const BaseState &)
- BaseState ()
- BaseState (BaseState &old)
- virtual ∼BaseState ()
- void removeSubtaskTrans (Transition ∗tr)
- void addSubtaskTrans (Transition ∗tr)
- void setMenu (QMenu ∗contextMenu)
- void removeTransition (Transition ∗transition)
- void removeTransitions ()
- QPolygonF polygon () const
- void addTransition (Transition ∗transition)
- QPixmap image () const
- int type () const
- QList< Transition ∗ > getTransitions ()
- virtual void Print (QXmlStreamWriter ∗writer)
- virtual std::string Print ()
- int outTransitionsCount ()

Protected Member Functions

- void contextMenuEvent (QGraphicsSceneContextMenuEvent ∗event)
- QVariant itemChange (GraphicsItemChange change, const QVariant &value)

Protected Attributes

- QString stateName
- StateType stateType
- QString argument
- QString parameters
- QGraphicsTextItem ∗ nameTextItem
- QList< Transition ∗ > Transitions
- QList< Transition ∗ > subtaskTransitions

3.1.1 Detailed Description

Class being a Base to all the state classes, having only the base attributes of the states.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 BaseState::BaseState ( )

Creates a state with empty nameTextItem and with no stateType.

3.1.2.2 BaseState::BaseState (BaseState & *old*)

Creates a state, which is a copy of state 'old'.

3.1.2.3 virtual BaseState::∼BaseState ( ) `[virtual]`

Deletes nameTextItem, and removes this state from all transitions, which pointed to it as to a subtask.

3.1.3 Member Function Documentation

3.1.3.1 void BaseState::addSubtaskTrans (Transition ∗ *tr*)

Adds the Transition tr to the subtaskTransitions list.

3.1.3.2 void BaseState::addTransition (Transition ∗ *transition*)

Adds transition to the state.

3.1.3.3 void BaseState::contextMenuEvent (QGraphicsSceneContextMenuEvent ∗ *event*) `[protected]`

Opens the context menu for the state.

3.1.3.4 virtual bool BaseState::equals (BaseState ∗ *other*) `[inline, virtual]`

Function comparing 2 states. Checks only the basic elements of the state(name, type, arguments, parameters).

Returns

true if states are equal.

Reimplemented in StopState, EmptyGenForSetState, EmptyGenState, GetSensorState, InitiateSensorState, RunGenState, StopGenState, sysInitState, and WaitState.

3.1.3.5 QString BaseState::getArgument ( ) `[inline]`

Getter function to argument.

Reimplemented in EmptyGenState.

3.1.3.6 virtual TreeItem∗ BaseState::getChild (int *i*, TreeItem ∗ *parent*) `[inline, virtual]`

Creates and returns the child of this state located at the index i.

Returns

Child of this state at index i.

Reimplemented in StopState, EmptyGenForSetState, EmptyGenState, GetSensorState, InitiateSensorState, RunGenState, StopGenState, sysInitState, and WaitState.

3.1.3.7 QString BaseState::getName ( ) `[inline]`

Getter function to stateName.

3.1.3.8 QGraphicsTextItem∗ BaseState::getNameTextItem ( ) `[inline]`

Getter function for nameTextItem.

3.1.3.9 QString BaseState::getParameters ( ) `[inline]`

Getter function to parameters.

3.1.3.10 QList<Transition*> BaseState::getSubtaskTransitions ( ) `[inline]`

Getter function for subtaskTransitions.

3.1.3.11 QList<Transition *> BaseState::getTransitions ( ) `[inline]`

Getter function for Transitions.

3.1.3.12 StateType BaseState::getType ( ) `[inline]`

Getter function to stateType.

3.1.3.13 QPixmap BaseState::image ( ) const

Returns the image representing the state.

3.1.3.14 QVariant BaseState::itemChange (GraphicsItemChange *change*, const QVariant & *value*) `[protected]`

If the change is change of position then updateposition() function is called for allt he transitions of this state.

3.1.3.15 virtual int BaseState::itemCount ( ) `[inline, virtual]`

Counts how many children does the object have in the TreeView.

Returns

Number of children which will be visible in the TreeView

Reimplemented in StopState, EmptyGenForSetState, EmptyGenState, GetSensorState, InitiateSensorState, RunGenState, StopGenState, sysInitState, and WaitState.

3.1.3.16 virtual QStringList BaseState::LoadFromXML (QXmlStreamReader * *reader*) `[inline, virtual]`

Loads from XML Stream the data and passes it to the subclasses(if any).

Parameters

reader	Stream from which the data is read

Returns

List of errors, which occured while loading

Reimplemented in StopState, EmptyGenForSetState, EmptyGenState, GetSensorState, InitiateSensorState, RunGenState, StopGenState, sysInitState, and WaitState.

3.1.3.17 BaseState& BaseState::operator= (const BaseState &)

Copy operator for BaseState.

3.1.3.18 int BaseState::outTransitionsCount ( )

Counts the number of transitions which are going out from this state.

Returns

Number of transitions starting at this state

3.1.3.19 QPolygonF BaseState::polygon ( ) const `[inline]`

Retruns the polygon representing the state.

3.1.3.20 virtual void BaseState::Print (QXmlStreamWriter $*$ *writer*) `[inline, virtual]`

Writes the data of the state to the XML stream.

Parameters

writer	Stream to which the data is written

Reimplemented in StopState, EmptyGenForSetState, EmptyGenState, GetSensorState, InitiateSensorState, RunGenState, Stop-GenState, sysInitState, and WaitState.

3.1.3.21 virtual std::string BaseState::Print ( ) `[inline, virtual]`

Creates a string describing the state attributes.

Returns

String with the description of the State

Reimplemented in StopState, EmptyGenForSetState, EmptyGenState, GetSensorState, InitiateSensorState, RunGenState, Stop-GenState, sysInitState, and WaitState.

3.1.3.22 void BaseState::removeSubtaskTrans (Transition $*$ *tr*)

Removes the tr Transition from the subtaskTransitions list.

3.1.3.23 void BaseState::removeTransition (Transition $*$ *transition*)

Removes given transition.

Parameters

transition	Transition to be removed

3.1.3.24 void BaseState::removeTransitions ( )

Removes from both states and deletes all transitions of the given state.

3.1.3.25 void BaseState::setArgument (QString *newArg*) `[inline]`

Setter function for argument.

Reimplemented in EmptyGenState.

3.1.3.26 void BaseState::setMenu (QMenu ∗ *contextMenu*)

Sets context menu for the state.

3.1.3.27 void BaseState::setName (QString *newName*)

Settes function for stateName.

3.1.3.28 void BaseState::setParameters (QString *newParams*) `[inline]`

Setter function for parameters.

3.1.3.29 void BaseState::setSelected (bool *selected*) `[inline]`

Sets selected and updates graphics view of this object.

3.1.3.30 void BaseState::setType (StateType *newType*) `[inline]`

Setter function for stateType.

3.1.3.31 int BaseState::type ( ) const `[inline]`

Getter function for Type.

3.1.3.32 void BaseState::updateSize ( )

Resizes the text in nameTextItem to fit the boundingRect of the state.

3.1.3.33 void BaseState::updateTextPositions ( ) `[inline]`

Moves the text to fit the left top corner with the state corner.

3.1.4 Member Data Documentation

3.1.4.1 QString **BaseState::argument** `[protected]`

Argument(optional) of the state.

3.1.4.2 QGraphicsTextItem∗ **BaseState::nameTextItem** `[protected]`

TextItem showing name and type of the state.

3.1.4.3 QString **BaseState::parameters** `[protected]`

Parameters(optional) of the state.

3.1.4.4 QString **BaseState::stateName** `[protected]`

Name of the state.

3.1.4.5 StateType BaseState::stateType [protected]

Type of the State.

3.1.4.6 QList< Transition * > BaseState::subtaskTransitions [protected]

List of transitions, which point to this item as to a subtask.

3.1.4.7 QList< Transition * > BaseState::Transitions [protected]

List of transitions of this state.

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

- baseState.h

3.2 CoordDialog Class Reference

`#include <StateTypeWidgets.h>`

Signals

- void InsertCoords (Coordinates *newCoords)
- void reportError (QString msgString)

Public Member Functions

- CoordDialog (QWidget *parent)
- Coordinates * getCoords ()
- void setCoords (Coordinates *newCoords)
- void coordsUpdated ()

3.2.1 Detailed Description

Dialogbox allowing to specify coordinates for runGenState.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 CoordDialog::CoordDialog (QWidget * *parent*)

Constructor creating this DialogBox and all it's sub-widgets.

3.2.3 Member Function Documentation

3.2.3.1 void CoordDialog::coordsUpdated ( )

Refreshes the information in the Dialogbox.

3.2.3.2 Coordinates* CoordDialog::getCoords ( ) [inline]

Getter function for coords.

3.2.3.3 void CoordDialog::InsertCoords (Coordinates ∗ *newCoords*) `[signal]`

Signal to RunGenWidget, that user has accepted Coordinates.

3.2.3.4 void CoordDialog::reportError (QString *msgString*) `[signal]`

Signal to parent, that an error has ocured.

3.2.3.5 void CoordDialog::setCoords (Coordinates ∗ *newCoords*) `[inline]`

Setter function for coords.

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

- StateTypeWidgets.h

3.3 Coordinates Class Reference

`#include <Coordinates.h>`

Public Member Functions

- Coordinates ()
- Coordinates (Coordinates &old)
- ∼Coordinates ()
- bool equals (Coordinates ∗other)
- CoordType getCoordType ()
- void setCoordType (CoordType newCoordType)
- MotionType getMotionType ()
- void setMotionType (MotionType newMotionType)
- std::vector< Pose ∗ > getPoses ()
- void setPoses (std::vector< Pose ∗ > newPoses)
- std::string Print ()
- void Print (QXmlStreamWriter ∗writer)
- QStringList LoadFromXML (QXmlStreamReader ∗reader)

3.3.1 Detailed Description

Class representing the Coordinates in the RunGenerator state.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Coordinates::Coordinates ( ) `[inline]`

Empty constructor.

3.3.2.2 Coordinates::Coordinates (Coordinates & *old*) `[inline]`

Copy constructor creating copies of poses.

3.3.2.3 Coordinates::∼Coordinates ( ) `[inline]`

Destructor, which deletes poses in this object.

3.3.3 Member Function Documentation

3.3.3.1 bool Coordinates::equals (Coordinates ∗ *other*) `[inline]`

Function, which checks if the othe rinstance of this class is equal to this instance.

Returns

true if objectshave same data.

3.3.3.2 CoordType Coordinates::getCoordType ( ) `[inline]`

Getter function for coordType.

3.3.3.3 MotionType Coordinates::getMotionType ( ) `[inline]`

Getter function for motionType.

3.3.3.4 std::vector<Pose ∗> Coordinates::getPoses ( ) `[inline]`

Getter function for poses.

3.3.3.5 QStringList Coordinates::LoadFromXML (QXmlStreamReader ∗ *reader*) `[inline]`

Loads from XML Stream the data and passes it to the Poses(if any).

Parameters

reader	Stream from which the data is read

Returns

List of errors, which occured while loading

3.3.3.6 std::string Coordinates::Print ( ) `[inline]`

Creates a string describing the coordinates attributes.

Returns

String with the description of the Coordinates

3.3.3.7 void Coordinates::Print (QXmlStreamWriter ∗ *writer*) `[inline]`

Writes the data of the state to the XML stream.

Parameters

writer	Stream to which the data is written

3.3.3.8 void Coordinates::setCoordType (CoordType *newCoordType*) `[inline]`

Setter function for coordType.

3.3.3.9 void Coordinates::setMotionType (MotionType *newMotionType*) `[inline]`

Setter function for motionType.

3.3.3.10 void Coordinates::setPoses (std::vector< Pose * > *newPoses*) `[inline]`

Setter function for poses.

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

- Coordinates.h

3.4 DiagramScene Class Reference

`#include <diagramscene.h>`

Public Slots

- void setMode (SceneMode mode)

Signals

- void LineCanceled ()
- void modeChanged (SceneMode mode)
- void itemInserted (BaseState *item)
- void lineInserted (Transition *line)
- void itemSelected (QGraphicsItem *item)
- void reportError (QString)

Public Member Functions

- DiagramScene (QMenu *itemMenu, QObject *parent, Model *newmod)
- void setToInsertState (BaseState *newState)
- void setTransitionAttributes (std::pair< QString, QString > thePair)
- void setItemParams (BaseState *toInsert)
- void checkIfFits (BaseState *state)

Protected Member Functions

- void mousePressEvent (QGraphicsSceneMouseEvent *mouseEvent)
- void mouseMoveEvent (QGraphicsSceneMouseEvent *mouseEvent)
- void mouseReleaseEvent (QGraphicsSceneMouseEvent *mouseEvent)

3.4.1 Detailed Description

Class representing a scene, on which the items representing a graph are shown.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 DiagramScene::DiagramScene (QMenu * *itemMenu*, QObject * *parent*, Model * *newmod*)

Constructor creating the DiagramScene.

Parameters

itemMenu	- menu, which will be added to items on the scene.
parent	- Parent widget.
newmod	- data model.

3.4.3 Member Function Documentation

3.4.3.1 void DiagramScene::checkIfFits (BaseState * *state*)

Checks if the state is inside the scene, and if not it resizes the scene.

3.4.3.2 void DiagramScene::itemInserted (BaseState * *item*) [signal]

Signals, that a state has been inserted into the scene.

3.4.3.3 void DiagramScene::itemSelected (QGraphicsItem * *item*) [signal]

Signals, that an item has been selected on the scene.

3.4.3.4 void DiagramScene::LineCanceled ( ) [signal]

Signals, that the line has been canceled, and the button representing transitions should be no longer pushed.

3.4.3.5 void DiagramScene::lineInserted (Transition * *line*) [signal]

Signals, that a transition has been inserted into the scene.

3.4.3.6 void DiagramScene::modeChanged (SceneMode *mode*) [signal]

Signals, that the mode has been changed and the tip should be changed.

3.4.3.7 void DiagramScene::mouseMoveEvent (QGraphicsSceneMouseEvent * *mouseEvent*) [protected]

Reacts to mouse move action. Moves items on the scene, or repaints the line representing future transition.

3.4.3.8 void DiagramScene::mousePressEvent (QGraphicsSceneMouseEvent * *mouseEvent*) [protected]

Reacts to mouse clicked action. Inserts a state, or sets startpoint for the line.

3.4.3.9 void DiagramScene::mouseReleaseEvent (QGraphicsSceneMouseEvent * *mouseEvent*) [protected]

Reacts to mouse released action. Inserts a transition, or checks if moved items fits after move.

3.4.3.10 void DiagramScene::reportError (QString) [signal]

Reports an error to the main window

3.4.3.11 void DiagramScene::setItemParams (BaseState * *toInsert*)

Sets contextMenu, graphicsItem and colour for the inserted item (used while loading from XML).

3.4.3.12 void DiagramScene::setMode (SceneMode *mode*) `[slot]`

Changes mode and emits modeChanged() signal.

3.4.3.13 void DiagramScene::setToInsertState (BaseState ∗ *newState*) `[inline]`

Sets State, which will be inserted next into the scene.

3.4.3.14 void DiagramScene::setTransitionAttributes (std::pair< QString, QString > *thePair*) `[inline]`

Sets attributes(condition, subtask) of the transition, which will be next inserted into the scene.

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

- diagramscene.h

3.5 ECPDialog Class Reference

```
#include <StateTypeWidgets.h>
```

Signals

- void InsertECP (robotInit newInit)
- void reportError (QString msgString)

Public Member Functions

- ECPDialog (QWidget ∗parent)
- void openForECP (robotInit robotIni)

3.5.1 Detailed Description

Widget allowing to create new robotInit instances for sysIniState.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 ECPDialog::ECPDialog (QWidget ∗ *parent*)

Constructor creating this DialogBox and all it's sub-widgets.

3.5.3 Member Function Documentation

3.5.3.1 void ECPDialog::InsertECP (robotInit *newInit*) `[signal]`

Signal to parent, that a new robotInit instance has been added.

3.5.3.2 void ECPDialog::openForECP (robotInit *robotIni*)

Loads robotIni into this window.

3.5.3.3 void ECPDialog::reportError (QString *msgString*) `[signal]`

Signal to parent, that an error has ocured.

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

- StateTypeWidgets.h

3.6 EditWidget Class Reference

`#include <editWidget.h>`

Signals

- void reportError (QString msgString)

Public Member Functions

- EditWidget (RESpecTa ∗parent, Model ∗mod)

3.6.1 Detailed Description

Class containing edit widgets.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 EditWidget::EditWidget (**RESpecTa** ∗ *parent,* **Model** ∗ *mod*)

Constructor creating all edit widgets (stateWidget, subtaskWidget and transWidget).

3.6.3 Member Function Documentation

3.6.3.1 void EditWidget::reportError (QString *msgString*) `[signal]`

Signal to the main window reporting error.

Parameters

msgString	Error to display

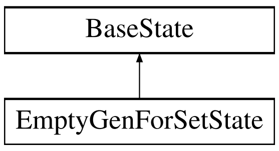
The documentation for this class was generated from the following file:

- editWidget.h

3.7 EmptyGenForSetState Class Reference

`#include <States.h>`

Inheritance diagram for EmptyGenForSetState:



Public Member Functions

- EmptyGenForSetState ()
- EmptyGenForSetState (EmptyGenForSetState &old)
- ~EmptyGenForSetState ()
- bool equals (BaseState *other)
- RobotSet getSet ()
- void setSet (RobotSet newSet)
- void Print (QXmlStreamWriter *writer)
- std::string Print ()
- QStringList LoadFromXML (QXmlStreamReader *reader)
- int itemCount ()
- TreeItem * getChild (int i, TreeItem *parent)

3.7.1 Detailed Description

State representing sets of robots, of which the first one waits for the second.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 EmptyGenForSetState::EmptyGenForSetState ( ) `[inline]`

Empty constructor setting stateType.

3.7.2.2 EmptyGenForSetState::EmptyGenForSetState (EmptyGenForSetState & *old*) `[inline]`

Copy constructor copying all date from state old.

3.7.2.3 EmptyGenForSetState::~EmptyGenForSetState ( ) `[inline]`

Empty destructor.

3.7.3 Member Function Documentation

3.7.3.1 bool EmptyGenForSetState::equals (BaseState * *other*) `[virtual]`

Function checking if the other object is equal to this.

Returns

true if objects data is the same.

Reimplemented from BaseState.

3.7.3.2 TreeItem* EmptyGenForSetState::getChild (int *i*, TreeItem * *parent*) `[virtual]`

Returns the i'th children of this state.

Reimplemented from BaseState.

3.7.3.3 RobotSet EmptyGenForSetState::getSet ( ) `[inline]`

Getter function for set.

3.7.3.4 int EmptyGenForSetState::itemCount ( ) `[inline, virtual]`

Function used to count how many children should there be for state's TreeView item.

Reimplemented from BaseState.

3.7.3.5 QStringList EmptyGenForSetState::LoadFromXML (QXmlStreamReader ∗ *reader*) `[virtual]`

Function loading a State from XML reader Stream.

Reimplemented from BaseState.

3.7.3.6 void EmptyGenForSetState::Print (QXmlStreamWriter ∗ *writer*) `[virtual]`

Function printing the state to XML writer stream.

Reimplemented from BaseState.

3.7.3.7 std::string EmptyGenForSetState::Print ( ) `[virtual]`

Function printing the attributes of the state into a String.

Reimplemented from BaseState.

3.7.3.8 void EmptyGenForSetState::setSet (RobotSet *newSet*) `[inline]`

Setter function for set.

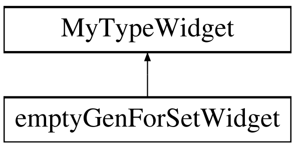
The documentation for this class was generated from the following file:

- States.h

3.8 emptyGenForSetWidget Class Reference

`#include <StateTypeWidgets.h>`

Inheritance diagram for emptyGenForSetWidget:



Signals

- void reportError (QString msgString)

Public Member Functions

- emptyGenForSetWidget (QWidget ∗parent, Model ∗newmod)
- BaseState ∗ getStateObject ()
- void setState (BaseState ∗state)

3.8.1 Detailed Description

Widget allowing to edit emptyGenForSetState.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 emptyGenForSetWidget::emptyGenForSetWidget (QWidget ∗ *parent*, Model ∗ *newmod*)

Constructor creating this widget and all it's sub-widgets.

3.8.3 Member Function Documentation

3.8.3.1 BaseState∗ emptyGenForSetWidget::getStateObject ( ) `[virtual]`

Returns a State with proper type and all the data from the widget.

Implements MyTypeWidget.

3.8.3.2 void emptyGenForSetWidget::reportError (QString *msgString*) `[signal]`

Signal to parent, that an error has ocured.

3.8.3.3 void emptyGenForSetWidget::setState (BaseState ∗ *state*) `[virtual]`

Function opening state for edition in the widget.

Implements MyTypeWidget.

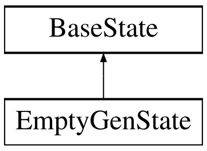
The documentation for this class was generated from the following file:

- StateTypeWidgets.h

3.9 EmptyGenState Class Reference

`#include <States.h>`

Inheritance diagram for EmptyGenState:



Public Member Functions

- EmptyGenState ()
- EmptyGenState (EmptyGenState &old)
- ∼EmptyGenState ()
- bool equals (BaseState ∗other)
- Robot getRobot ()
- void setRobot (Robot newRobot)
- QString getArgument ()
- void setArgument (QString newArg)
- void Print (QXmlStreamWriter ∗writer)

- std::string Print ()
- QStringList LoadFromXML (QXmlStreamReader ∗reader)
- int itemCount ()
- TreeItem ∗ getChild (int i, TreeItem ∗parent)

3.9.1 Detailed Description

State representing empty generator for one robot.

3.9.2 Constructor & Destructor Documentation

3.9.2.1 EmptyGenState::EmptyGenState ( ) `[inline]`

Empty constructor setting stateType.

3.9.2.2 EmptyGenState::EmptyGenState (EmptyGenState & *old*) `[inline]`

Copy constructor copying all date from state old.

3.9.2.3 EmptyGenState::∼EmptyGenState ( ) `[inline]`

Empty destructor.

3.9.3 Member Function Documentation

3.9.3.1 bool EmptyGenState::equals (BaseState ∗ *other*) `[virtual]`

Function checking if the other object is equal to this.

Returns

true if objects data is the same.

Reimplemented from BaseState.

3.9.3.2 QString EmptyGenState::getArgument ( ) `[inline]`

Getter function for argument.

Reimplemented from BaseState.

3.9.3.3 TreeItem∗ EmptyGenState::getChild (int *i*, TreeItem ∗ *parent*) `[virtual]`

Returns the i'th children of this state.

Reimplemented from BaseState.

3.9.3.4 Robot EmptyGenState::getRobot ( ) `[inline]`

Getter function for robot.

3.9.3.5 int EmptyGenState::itemCount ( ) `[inline, virtual]`

Function used to count how many children should there be for state's TreeView item.

Reimplemented from BaseState.

3.9.3.6 QStringList EmptyGenState::LoadFromXML (QXmlStreamReader $*$ *reader*) `[virtual]`

Function loading a State from XML reader Stream.

Reimplemented from BaseState.

3.9.3.7 std::string EmptyGenState::Print ( ) `[virtual]`

Function printing the attributes of the state into a String.

Reimplemented from BaseState.

3.9.3.8 void EmptyGenState::Print (QXmlStreamWriter $*$ *writer*) `[virtual]`

Function printing the state to XML writer stream.

Reimplemented from BaseState.

3.9.3.9 void EmptyGenState::setArgument (QString *newArg*) `[inline]`

Setter function for argument.

Reimplemented from BaseState.

3.9.3.10 void EmptyGenState::setRobot (Robot *newRobot*) `[inline]`

Setter function for robot.

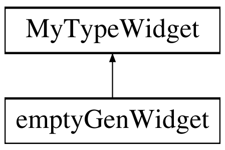
The documentation for this class was generated from the following file:

- States.h

3.10 emptyGenWidget Class Reference

`#include <StateTypeWidgets.h>`

Inheritance diagram for emptyGenWidget:



Public Member Functions

- emptyGenWidget (QWidget $*$parent, Model $*$newmod)
- BaseState $*$ getStateObject ()
- void setState (BaseState $*$state)

3.10.1 Detailed Description

Widget allowing to edit EmptyGenState.

3.10.2 Constructor & Destructor Documentation

3.10.2.1 emptyGenWidget::emptyGenWidget (QWidget ∗ *parent*, Model ∗ *newmod*)

Constructor creating this widget and all it's sub-widgets.

3.10.3 Member Function Documentation

3.10.3.1 BaseState∗ emptyGenWidget::getStateObject ( ) `[virtual]`

Returns a State with proper type and all the data from the widget.

Implements MyTypeWidget.

3.10.3.2 void emptyGenWidget::setState (BaseState ∗ *state*) `[virtual]`

Function opening state for edition in the widget.

Implements MyTypeWidget.

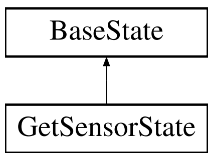
The documentation for this class was generated from the following file:

- StateTypeWidgets.h

3.11 GetSensorState Class Reference

`#include <States.h>`

Inheritance diagram for GetSensorState:



Public Member Functions

- GetSensorState ()
- GetSensorState (GetSensorState &old)
- ∼GetSensorState ()
- bool equals (BaseState ∗other)
- Sensor getSensor ()
- void setSensor (Sensor newSensor)
- void Print (QXmlStreamWriter ∗writer)
- std::string Print ()
- QStringList LoadFromXML (QXmlStreamReader ∗reader)
- int itemCount ()
- TreeItem ∗ getChild (int i, TreeItem ∗parent)

3.11.1 Detailed Description

State representing downloading data from a sensor.

3.11.2 Constructor & Destructor Documentation

3.11.2.1 GetSensorState::GetSensorState ( ) `[inline]`

Empty constructor setting stateType.

3.11.2.2 GetSensorState::GetSensorState (GetSensorState & *old*) `[inline]`

Copy constructor copying all date from state old.

3.11.2.3 GetSensorState::∼GetSensorState ( ) `[inline]`

Empty destructor.

3.11.3 Member Function Documentation

3.11.3.1 bool GetSensorState::equals (BaseState ∗ *other*) `[virtual]`

Function checking if the other object is equal to this.

Returns

true if objects data is the same.

Reimplemented from BaseState.

3.11.3.2 TreeItem∗ GetSensorState::getChild (int *i,* TreeItem ∗ *parent*) `[virtual]`

Returns the i'th children of this state.

Reimplemented from BaseState.

3.11.3.3 Sensor GetSensorState::getSensor ( ) `[inline]`

Getter function for sensor.

3.11.3.4 int GetSensorState::itemCount ( ) `[inline, virtual]`

Function used to count how many children should there be for state's TreeView item.

Reimplemented from BaseState.

3.11.3.5 QStringList GetSensorState::LoadFromXML (QXmlStreamReader ∗ *reader*) `[virtual]`

Function loading a State from XML reader Stream.

Reimplemented from BaseState.

3.11.3.6 void GetSensorState::Print (QXmlStreamWriter ∗ *writer*) `[virtual]`

Function printing the state to XML writer stream.

Reimplemented from BaseState.

3.11.3.7 std::string GetSensorState::Print ( ) `[virtual]`

Function printing the attributes of the state into a String.

Reimplemented from BaseState.

3.11.3.8 void GetSensorState::setSensor (Sensor *newSensor*) `[inline]`

Setter function for sensor.

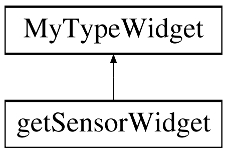
The documentation for this class was generated from the following file:

- States.h

3.12 getSensorWidget Class Reference

`#include <StateTypeWidgets.h>`

Inheritance diagram for getSensorWidget:



Public Member Functions

- getSensorWidget (QWidget ∗parent, Model ∗newmod)
- BaseState ∗ getStateObject ()
- void setState (BaseState ∗state)

3.12.1 Detailed Description

Widget allowing to edit getSensorState.

3.12.2 Constructor & Destructor Documentation

3.12.2.1 getSensorWidget::getSensorWidget (QWidget ∗ *parent*, Model ∗ *newmod*)

Constructor creating this widget and all it's sub-widgets.

3.12.3 Member Function Documentation

3.12.3.1 BaseState∗ getSensorWidget::getStateObject ( ) `[virtual]`

Returns a State with proper type and all the data from the widget.

Implements MyTypeWidget.

3.12.3.2 void getSensorWidget::setState (BaseState ∗ *state*) `[virtual]`

Function opening state for edition in the widget.

Implements MyTypeWidget.

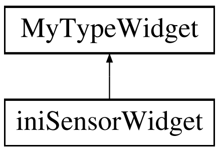
The documentation for this class was generated from the following file:

- StateTypeWidgets.h

3.13 iniSensorWidget Class Reference

```
#include <StateTypeWidgets.h>
```

Inheritance diagram for iniSensorWidget:



Public Member Functions

- iniSensorWidget (QWidget ∗parent, Model ∗newmod)
- BaseState ∗ getStateObject ()
- void setState (BaseState ∗state)

3.13.1 Detailed Description

State allowing to edit iniSensorState.

3.13.2 Constructor & Destructor Documentation

3.13.2.1 iniSensorWidget::iniSensorWidget (QWidget ∗ *parent*, Model ∗ *newmod*)

Constructor creating this widget and all it's sub-widgets.

3.13.3 Member Function Documentation

3.13.3.1 BaseState∗ iniSensorWidget::getStateObject ( ) `[virtual]`

Returns a State with proper type and all the data from the widget.

Implements MyTypeWidget.

3.13.3.2 void iniSensorWidget::setState (BaseState ∗ *state*) `[virtual]`

Function opening state for edition in the widget.

Implements MyTypeWidget.

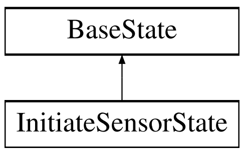
The documentation for this class was generated from the following file:

- StateTypeWidgets.h

3.14 InitiateSensorState Class Reference

```
#include <States.h>
```

Inheritance diagram for InitiateSensorState:



Public Member Functions

- InitiateSensorState ()
- InitiateSensorState (InitiateSensorState &old)
- ∼InitiateSensorState ()
- bool equals (BaseState ∗other)
- Sensor getSensor ()
- void setSensor (Sensor newSensor)
- void Print (QXmlStreamWriter ∗writer)
- std::string Print ()
- QStringList LoadFromXML (QXmlStreamReader ∗reader)
- int itemCount ()
- TreeItem ∗ getChild (int i, TreeItem ∗parent)

3.14.1 Detailed Description

State representing initiating sensor.

3.14.2 Constructor & Destructor Documentation

3.14.2.1 InitiateSensorState::InitiateSensorState ( ) `[inline]`

Empty constructor setting stateType.

3.14.2.2 InitiateSensorState::InitiateSensorState (InitiateSensorState & *old*) `[inline]`

Copy constructor copying all date from state old.

3.14.2.3 InitiateSensorState::∼InitiateSensorState ( ) `[inline]`

Empty destructor.

3.14.3 Member Function Documentation

3.14.3.1 bool InitiateSensorState::equals (BaseState ∗ *other*) `[virtual]`

Function checking if the other object is equal to this.

Returns

true if objects data is the same.

Reimplemented from BaseState.

3.14.3.2 TreeItem∗ InitiateSensorState::getChild (int *i*, TreeItem ∗ *parent*) `[virtual]`

Returns the i'th children of this state.

Reimplemented from BaseState.

3.14.3.3 Sensor InitiateSensorState::getSensor ( ) `[inline]`

Getter function for sensor.

3.14.3.4 int InitiateSensorState::itemCount ( ) `[inline, virtual]`

Function used to count how many children should there be for state's TreeView item.

Reimplemented from BaseState.

3.14.3.5 QStringList InitiateSensorState::LoadFromXML (QXmlStreamReader ∗ *reader*) `[virtual]`

Function loading a State from XML reader Stream.

Reimplemented from BaseState.

3.14.3.6 void InitiateSensorState::Print (QXmlStreamWriter ∗ *writer*) `[virtual]`

Function printing the state to XML writer stream.

Reimplemented from BaseState.

3.14.3.7 std::string InitiateSensorState::Print ( ) `[virtual]`

Function printing the attributes of the state into a String.

Reimplemented from BaseState.

3.14.3.8 void InitiateSensorState::setSensor (Sensor *newSensor*) `[inline]`

Setter function for sensor.

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

- States.h

3.15 Model Class Reference

`#include <Model.h>`

Signals

- void reportError (QString)
- void reportMsg (QString)

Public Member Functions

- Model ()
- int vertNum (QString SubtaskName)
- void CleanTask (QString name)
- void deleteAll ()
- void deleteTransition (Transition ∗transition)
- void deleteState (BaseState ∗state)
- void DeleteTask (QString SubtaskName)
- void addSubtask (QString name)
- bool addState (BaseState ∗item, QString subtaskName)

- bool tryInsertTransition (Transition ∗trans)
- void changeSubtaskName (QString oldName, QString NewName)
- bool ReplaceState (BaseState ∗oldState, BaseState ∗newState)
- void MoveTransitionUp (BaseState ∗st, int _index)
- void MoveTransitionDown (BaseState ∗st, int _index)
- Transition ∗ getTransition (BaseState ∗st, int i)
- QStringList getAllStartStateNames (QString sub)
- QStringList getAllEndStateNames (QString sub)
- BaseState ∗ getState (QString name)
- BaseState ∗ getState (QString name, QString subtaskName)
- BaseState ∗ getState (MyGraphType ∗graph, int i)
- std::vector< Transition ∗ > getTransitions (BaseState ∗state)
- MyGraphType ∗ getGraph (QString Name)
- QString getSubtaskName (QString StateName)
- QString getSubtaskName (BaseState ∗State)
- QString getMainName ()
- QString getSubNameOfTrans (Transition ∗transition)
- QStringList getStateNames (MyGraphType G)
- QStringList getTasksNameLists ()
- QStringList getTasksNameListsWithoutMain ()
- QStringList getAllStateNames (QString sub)
- boost::graph_traits< MyGraphType >::edge_iterator findEdge (MyGraphType ∗graph, Transition ∗toFind)
- boost::graph_traits< MyGraphType >::vertex_iterator findVertex (MyGraphType ∗graph, BaseState ∗toFind)
- void setMainName (QString newName)
- void setView (RESpecTa ∗newres)
- void setSaveFolder (QString newSaveFolder)
- bool checkTransCondAvailabe (BaseState ∗source, ConditionType condType, QString cond)
- bool checkSubtaskExists (QString Name)
- bool checkNameAvailable (QString Name, MyGraphType ∗G)
- bool checkNameAvailable (QString Name)
- bool checkTransitonExists (Transition ∗trans)
- bool checkTransCondAvailabe (Transition ∗tr, ConditionType condType, QString cond)
- void save (QString filename)
- void printStates (MyGraphType ∗G, std::string FileName, bool ifMain)
- QStringList checkIfOK ()
- void setChanged (bool newChanged)
- bool wasChanged ()
- void setBlock (bool block)

3.15.1 Detailed Description

Class responsible for keeping the data and allowing operation on the data.

3.15.2 Constructor & Destructor Documentation

3.15.2.1 Model::Model ( )

Constructor creating subtask called "MAIN".

3.15.3 Member Function Documentation

3.15.3.1 bool Model::addState (BaseState ∗ *item,* QString *subtaskName*)

Adds a state to a subtask.

Parameters

item	State, which will be inserted.
subtas	Q_OBJECTkName Name of subtask, to which the state will be inserted.

Returns

True if operation successful.

3.15.3.2 void Model::addSubtask (QString *name*)

Adds a subtask to the model.

Parameters

name	Name of subtask which will be created.

3.15.3.3 void Model::changeSubtaskName (QString *oldName,* QString *NewName*)

Changes the name of subtask from oldName to NewName.

3.15.3.4 QStringList Model::checkIfOK ( )

Checks for errors in the Model.

Returns

List of errors.

3.15.3.5 bool Model::checkNameAvailable (QString *Name,* MyGraphType ∗ *G*)

Checks if name Name is available for graph G.

Returns

True if no state with name Name exists in graph G.

3.15.3.6 bool Model::checkNameAvailable (QString *Name*)

Checks if state with name Name is present in the model.

Returns

True if subtask containing state with name Name exists.

3.15.3.7 bool Model::checkSubtaskExists (QString *Name*)

Checks if subtask with name Name exists.

Returns

True if subtask with name Name exists.

3.15.3.8 bool Model::checkTransCondAvailabe (BaseState ∗ *source,* ConditionType *condType,* QString *cond*)

Checks if condition cond is available for the state source.

Returns

True if the condition cond hasn't been used as a out-condition from the state source.

3.15.3.9 bool Model::checkTransCondAvailabe (Transition $*$ *tr,* ConditionType *condType,* QString *cond*)

Checks if condition of transition tr can be changed to cond.

Returns

True if no transition from the source state of tr has condition cond.

3.15.3.10 bool Model::checkTransitonExists (Transition $*$ *trans*)

Checks if Trnsition trans is present in the model.

Returns

True if subtask containing Transition trans exists.

3.15.3.11 void Model::CleanTask (QString *name*)

Function removing all vertices and edges from a task with name=name.

3.15.3.12 void Model::deleteAll ( )

Clears the model leaving only empty main state.

3.15.3.13 void Model::deleteState (BaseState $*$ *state*)

Removes the given state from the model.

3.15.3.14 void Model::DeleteTask (QString *SubtaskName*)

Deletes all states and transitions of the task from model, if it's not the main task erases the given Subtask from the model.

3.15.3.15 void Model::deleteTransition (Transition $*$ *transition*)

Removes the given transition from the model.

3.15.3.16 boost::graph_traits<MyGraphType>::edge_iterator Model::findEdge (MyGraphType $*$ *graph,* Transition $*$ *toFind*)

Returns edge_iterator of the Edge containing transition toFind in the graph.

3.15.3.17 boost::graph_traits<MyGraphType>::vertex_iterator Model::findVertex (MyGraphType $*$ *graph,* BaseState $*$ *toFind*)

Returns vertex_iterator of the Vertex containing state toFind in the graph.

3.15.3.18 QStringList Model::getAllEndStateNames (QString *sub*) `[inline]`

Returns all names of states in task sub without init state.

3.15.3.19 QStringList Model::getAllStartStateNames (QString *sub*) `[inline]`

Returns names of all States in the task sub without the end state.

3.15.3.20 QStringList Model::getAllStateNames (QString *sub*)

Returns List with names of all states from the subtask with name sub.

3.15.3.21 MyGraphType∗ Model::getGraph (QString *Name*)

Returns Graph with given Name.

3.15.3.22 QString Model::getMainName ( ) `[inline]`

Returns Name of the main task.

3.15.3.23 BaseState∗ Model::getState (QString *name*)

Returns the State with given name. Not for use for _END_ or _STOP_ states.

3.15.3.24 BaseState∗ Model::getState (QString *name,* QString *subtaskName*)

Returns the State with given name from the subtask with given subtaskName.

3.15.3.25 BaseState∗ Model::getState (MyGraphType ∗ *graph,* int *i*)

Returns the State at index i in Graph graph.

3.15.3.26 QStringList Model::getStateNames (MyGraphType *G*)

Returns List with names of all states from the graph G.

3.15.3.27 QString Model::getSubNameOfTrans (Transition ∗ *transition*)

Returns Name of subtask containing given transition.

3.15.3.28 QString Model::getSubtaskName (QString *StateName*)

Returns Name of subtask containing a state with given StateName.

3.15.3.29 QString Model::getSubtaskName (BaseState ∗ *State*)

Returns Name of subtask containing a state State.

3.15.3.30 QStringList Model::getTasksNameLists ( )

Returns List with names of all tasks from the model.

3.15.3.31 QStringList Model::getTasksNameListsWithoutMain ( )

Returns List with names of all tasks from the model without the main task..

3.15.3.32 Transition∗ Model::getTransition (BaseState ∗ *st,* int *i*)

Function returning the i'th transition going out of state st.

3.15.3.33 std::vector<Transition *> Model::getTransitions (BaseState * *state*)

Returns all transitions of a state from the model.

3.15.3.34 void Model::MoveTransitionDown (BaseState * *st*, int *_index*)

Moves the transition at index _index one position Down in the Transition List in the graph representing task.

3.15.3.35 void Model::MoveTransitionUp (BaseState * *st*, int *_index*)

Moves the transition at index _index one position Up in the Transition List in the graph representing task.

3.15.3.36 void Model::printStates (MyGraphType * *G*, std::string *FileName*, bool *ifMain*)

Saves the Task G to file FileName, if ifmain==1 saves the paths to other files.

3.15.3.37 bool Model::ReplaceState (BaseState * *oldState*, BaseState * *newState*)

Changes the oldState to newState in the model.

Returns

True if operation successful.

3.15.3.38 void Model::reportError (QString) `[signal]`

Reports error to the main window.

3.15.3.39 void Model::reportMsg (QString) `[signal]`

Reports a message to main view window.

3.15.3.40 void Model::save (QString *filename*)

Saves the Model to the file filename.

3.15.3.41 void Model::setBlock (bool *block*) `[inline]`

Setter function to block.

3.15.3.42 void Model::setChanged (bool *newChanged*)

Sets changed parameter and refreshes the TreeView in the main window.

3.15.3.43 void Model::setMainName (QString *newName*)

Changes name of the main task to newName.

3.15.3.44 void Model::setSaveFolder (QString *newSaveFolder*) `[inline]`

Sets save folder to newSaveFolder.

3.15.3.45 void Model::setView (**RESpecTa** ∗ *newres*) `[inline]`

Sets main window to newres.

3.15.3.46 bool Model::tryInsertTransition (**Transition** ∗ *trans*)

Adds a subtask to the model.

Parameters

trans	Transition which will be inserted.

Returns

True if operation successful.

3.15.3.47 int Model::vertNum (QString *SubtaskName*)

Returns Number of Vertices in the subtask defined by the SubtaskName.

3.15.3.48 bool Model::wasChanged ( ) `[inline]`

Getter function to changed.

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

- Model.h

3.16 MPDialog Class Reference

`#include <StateTypeWidgets.h>`

Signals

- void InsertMP (std::vector< Sensor > sensors, Transmitter trans)
- void reportError (QString msgString)

Public Member Functions

- MPDialog (QWidget ∗parent)
- void setSensTrans (std::vector< Sensor > sens, Transmitter tran)

3.16.1 Detailed Description

DialogBox allowing to edit MP section of the sysIniState.

3.16.2 Constructor & Destructor Documentation

3.16.2.1 MPDialog::MPDialog (QWidget ∗ *parent*)

Constructor creating this DialogBox and all it’s sub-widgets.

3.16.3 Member Function Documentation

3.16.3.1 void MPDialog::InsertMP (std::vector< Sensor > *sensors*, Transmitter *trans*) `[signal]`

Signals parent, that user has accepted changes.

3.16.3.2 void MPDialog::reportError (QString *msgString*) `[signal]`

Signal to parent, that an error has ocured.

3.16.3.3 void MPDialog::setSensTrans (std::vector< Sensor > *sens*, Transmitter *tran*)

Loads data of sensors and transmitter into the Dialog.

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

- StateTypeWidgets.h

3.17 myTreeView Class Reference

`#include <myTreeView.h>`

Public Member Functions

- **myTreeView** (QWidget ∗parent, RESpecTa ∗_res)

3.17.1 Detailed Description

Class responsible for treeview of the task.

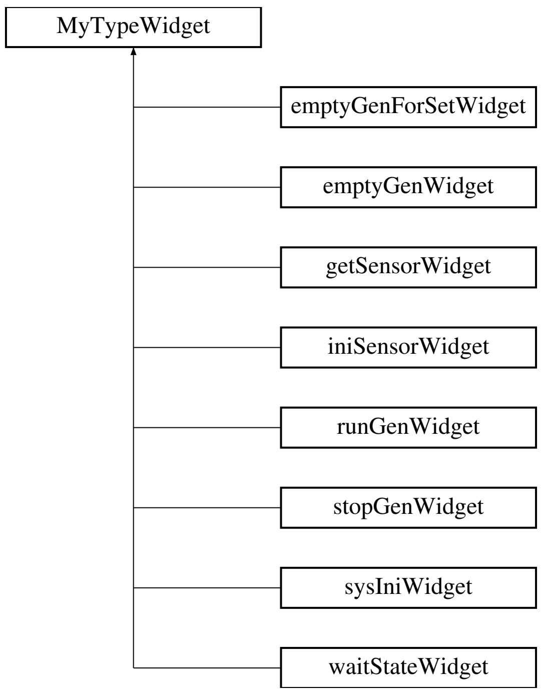
The documentation for this class was generated from the following file:

- myTreeView.h

3.18 MyTypeWidget Class Reference

`#include <StateTypeWidgets.h>`

Inheritance diagram for MyTypeWidget:



Public Member Functions

- **MyTypeWidget** (QWidget ∗parent, Model ∗newmod)
- virtual BaseState ∗ getStateObject ()=0
- virtual void setState (BaseState ∗State)=0

3.18.1 Detailed Description

Base class to all widgets, which allow editing different state types.

3.18.2 Member Function Documentation

3.18.2.1 virtual **BaseState**∗ MyTypeWidget::getStateObject ( ) `[pure virtual]`

Function returning object of the class, which is represented by the child-class of this class.

Implemented in sysIniWidget, runGenWidget, emptyGenForSetWidget, emptyGenWidget, waitStateWidget, stopGenWidget, iniSensorWidget, and getSensorWidget.

3.18.2.2 virtual void MyTypeWidget::setState (**BaseState** ∗ *State*) `[pure virtual]`

Function opening state State(of a child-class of BaseState) for edition in the widget.

Implemented in sysIniWidget, runGenWidget, emptyGenForSetWidget, emptyGenWidget, waitStateWidget, stopGenWidget, iniSensorWidget, and getSensorWidget.

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

- StateTypeWidgets.h

3.19 Pose Class Reference

```
#include <Pose.h>
```

Public Member Functions

- Pose ()
- Pose (const Pose &other)
- bool equals (Pose *other)
- std::vector< double > getA ()
- void setA (std::vector< double > newA)
- std::vector< double > getV ()
- void setV (std::vector< double > newV)
- std::vector< double > getC ()
- void setC (std::vector< double > newC)
- std::string Print ()
- void Print (QXmlStreamWriter *writer)
- QStringList LoadFromXML (QXmlStreamReader *reader)

3.19.1 Detailed Description

Class holding parameters of one Pose in the movement by robot.

3.19.2 Constructor & Destructor Documentation

3.19.2.1 Pose::Pose ( ) `[inline]`

Empty constructor.

3.19.2.2 Pose::Pose (const Pose & *other*) `[inline]`

Copy constructor.

3.19.3 Member Function Documentation

3.19.3.1 bool Pose::equals (Pose * *other*) `[inline]`

Function checking if the other object has same data as this one.

Returns

true if objects equal.

3.19.3.2 std::vector<double> Pose::getA ( ) `[inline]`

Getter function for accelerations vector.

3.19.3.3 std::vector<double> Pose::getC ( ) `[inline]`

Getter function for coordinates vector.

3.19.3.4 std::vector<double> Pose::getV ( ) `[inline]`

Getter function for velocity vector.

3.19.3.5 QStringList Pose::LoadFromXML (QXmlStreamReader $*$ *reader*) `[inline]`

Loads from XML Stream the data and passes it to the Poses(if any).

Parameters

reader	Stream from which the data is read

Returns

List of errors, which occured while loading

3.19.3.6 std::string Pose::Print ( ) `[inline]`

Creates a string describing the coordinates attributes.

Returns

String with the description of the Pose

3.19.3.7 void Pose::Print (QXmlStreamWriter $*$ *writer*) `[inline]`

Writes the data of the state to the XML stream.

Parameters

writer	Stream to which the data is written

3.19.3.8 void Pose::setA (std::vector$<$ double $>$ *newA*) `[inline]`

Setter function for accelerations vector.

3.19.3.9 void Pose::setC (std::vector$<$ double $>$ *newC*) `[inline]`

Setter function for coordinates vector.

3.19.3.10 void Pose::setV (std::vector$<$ double $>$ *newV*) `[inline]`

Setter function for velocity vector.

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

- Pose.h

3.20 RESpecTa Class Reference

`#include <respecta.h>`

Signals

- void EditTasksSig ()
- void itemSelectedSig (QGraphicsItem $*$item)
- void refreshWidgets ()
- void SignalDeleted ()

Public Member Functions

- RESpecTa (Model ∗newmod)
- ∼RESpecTa ()
- void getWarning (QString msg)
- void CenterOn (QString _name)
- void HideSubtask ()
- void WasChanged ()
- void setCurrentSubtask (QString newSubtask)
- void getError (QString error)
- void reportMsg (QString msgString)
- void clearSaveName ()
- void deleteState (BaseState ∗state)
- void deleteTrans (Transition ∗trans)
- void SaveGraphicsAttributes (QXmlStreamWriter ∗writer, QString SubName)
- void listSelectionChanged (QModelIndexList list)

3.20.1 Detailed Description

Main window class. It is a parent to all the widgets in the project.

3.20.2 Constructor & Destructor Documentation

3.20.2.1 RESpecTa::RESpecTa (Model ∗ *newmod*)

Constructor creating all sub-elements of the view.

3.20.2.2 RESpecTa::∼RESpecTa ( ) `[inline]`

Destructor, which closes the log file.

3.20.3 Member Function Documentation

3.20.3.1 void RESpecTa::CenterOn (QString *_name*)

Function centering the view on a State with name _name.

3.20.3.2 void RESpecTa::clearSaveName ( ) `[inline]`

Clears name of the Save file causing browse file window to open next time while saving.

3.20.3.3 void RESpecTa::deleteState (BaseState ∗ *state*)

Removes a state from the model, calling remove functions to all transitions connected.

3.20.3.4 void RESpecTa::deleteTrans (Transition ∗ *trans*)

Removes transition from the model and from all states, to which it is connected.

3.20.3.5 void RESpecTa::EditTasksSig ( ) `[signal]`

Signal, which informs editwidget, that the Task widget should be opened.

3.20.3.6 void RESpecTa::getError (QString *error*) [inline]

Calls reportError function.

Parameters

error	Error to report.

3.20.3.7 void RESpecTa::getWarning (QString *msg*) [inline]

Function which displays a warning in the terminal.

3.20.3.8 void RESpecTa::HideSubtask ( )

Unchecks the tasksaction button.

3.20.3.9 void RESpecTa::itemSelectedSig (QGraphicsItem * *item*) [signal]

Signal from Scene, which is forwarded to EditWidget and then to transWidget or stateWidget to display item for edition.

Parameters

item	Item, which has been selected on the scene.

3.20.3.10 void RESpecTa::listSelectionChanged (QModelIndexList *list*)

Reacts on the change on the treeview list with concentrating on the item selected.

3.20.3.11 void RESpecTa::refreshWidgets ( ) [signal]

Refreshes stateWidget and transWidget.

3.20.3.12 void RESpecTa::reportMsg (QString *msgString*)

Displays a dialog showing the message.

Parameters

msgString	Message, which will be displayed.

3.20.3.13 void RESpecTa::SaveGraphicsAttributes (QXmlStreamWriter * *writer*, QString *SubName*)

Save graphic attributes of scene to XML.

Parameters

writer	XML Strem, to which the attributes are saved.
SubName	Name of the subtask which parameters are saved.

3.20.3.14 void RESpecTa::setCurrentSubtask (QString *newSubtask*) [inline]

Sets curentSubtask to newSubtask.

Parameters

newSubtask	Name of the new subtask to show.

3.20.3.15 void RESpecTa::SignalDeleted ( ) `[signal]`

Signals that a deletion has occured ont he scene, and all editions should be canceled.

3.20.3.16 void RESpecTa::WasChanged ( )

Function refreshing the TreeView, called when change in the model has occured.

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

- respecta.h

3.21 robotInit Class Reference

```
#include <robotInit.h>
```

Public Member Functions

- robotInit ()
- robotInit (const robotInit &other)
- bool equals (robotInit other)
- std::string Print ()
- void Print (QXmlStreamWriter ∗writer)
- QStringList LoadFromXML (QXmlStreamReader ∗reader)

Public Attributes

- Robot robot
- std::vector< std::pair< GeneratorType, int > > init_values

3.21.1 Detailed Description

Class representing the initiation of one robot.

3.21.2 Constructor & Destructor Documentation

3.21.2.1 robotInit::robotInit ( ) `[inline]`

Empty construcot creating new instance of robotInit.

3.21.2.2 robotInit::robotInit (const **robotInit** & *other*) `[inline]`

Copy constructor creating an instance of robotInit with same data os the other.

3.21.3 Member Function Documentation

3.21.3.1 bool robotInit::equals (robotInit *other*) [inline]

Function checking if this instance equals the other.

Returns

true if objects have the same data.

3.21.3.2 QStringList robotInit::LoadFromXML (QXmlStreamReader ∗ *reader*) [inline]

Loads from XML Stream the data.

Parameters

reader	Stream from which the data is read

Returns

List of errors, which occured while loading

3.21.3.3 void robotInit::Print (QXmlStreamWriter ∗ *writer*) [inline]

Writes the data of the state to the XML stream.

Parameters

writer	Stream to which the data is written

3.21.3.4 std::string robotInit::Print ( ) [inline]

Creates a string describing the coordinates attributes.

Returns

String with the description of the Robot initialization

3.21.4 Member Data Documentation

3.21.4.1 std::vector< std::pair<GeneratorType, int> > robotInit::init_values

Vector of generators, and their init arguments.

3.21.4.2 Robot robotInit::robot

Robot, which is being initialized

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

- robotInit.h

3.22 RobotSet Class Reference

```
#include <RobotSet.h>
```

Public Member Functions

- std::string Print ()
- bool equals (RobotSet other)
- void Print (QXmlStreamWriter ∗writer)
- QStringList LoadFromXML (QXmlStreamReader ∗reader)

Public Attributes

- std::vector< Robot > first

3.22.1 Detailed Description

Class representing XML element of RobotSet.

3.22.2 Member Function Documentation

3.22.2.1 bool RobotSet::equals (RobotSet *other*) `[inline]`

Function checks if this instance equals other.

Returns

true if objects are equal.

3.22.2.2 QStringList RobotSet::LoadFromXML (QXmlStreamReader ∗ *reader*) `[inline]`

Loads from XML Stream the data and passes it to the Poses(if any).

Parameters

reader	Stream from which the data is read

Returns

List of errors, which occured while loading

3.22.2.3 void RobotSet::Print (QXmlStreamWriter ∗ *writer*) `[inline]`

Writes the data of the state to the XML stream.

Parameters

writer	Stream to which the data is written

3.22.2.4 std::string RobotSet::Print ( ) `[inline]`

Creates a string describing the coordinates attributes.

Returns

String with the description of the RobotSet

3.22.3 Member Data Documentation

3.22.3.1 std::vector<Robot> **RobotSet::first**

First set of Robots.

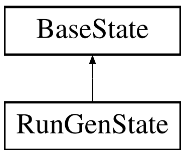
The documentation for this class was generated from the following file:

- RobotSet.h

3.23 RunGenState Class Reference

`#include <States.h>`

Inheritance diagram for RunGenState:



Public Member Functions

- RunGenState ()
- RunGenState (RunGenState &old)
- ~RunGenState ()
- bool equals (BaseState ∗other)
- Robot getRobot ()
- void setRobot (Robot newRobot)
- GeneratorType getGenType ()
- void setGenType (GeneratorType newGenType)
- Coordinates ∗ getCoords ()
- void setCoords (Coordinates ∗newCoords)
- QString getSpeech ()
- void setSpeech (QString newSpeech)
- QString getArgs ()
- void setArgs (QString newGenArgs)
- QString getFilePath ()
- void setFilePath (QString newPath)
- void Print (QXmlStreamWriter ∗writer)
- std::string Print ()
- QStringList LoadFromXML (QXmlStreamReader ∗reader)
- int itemCount ()
- TreeItem ∗ getChild (int i, TreeItem ∗parent)

3.23.1 Detailed Description

State representing movement of one robot.

3.23.2 Constructor & Destructor Documentation

3.23.2.1 RunGenState::RunGenState ( ) `[inline]`

Empty constructor setting stateType.

3.23.2.2 RunGenState::RunGenState (RunGenState & *old*) `[inline]`

Copy constructor copying all date from state old.

3.23.2.3 RunGenState::∼RunGenState ( ) `[inline]`

Destructor, which deletes coords.

3.23.3 Member Function Documentation

3.23.3.1 bool RunGenState::equals (BaseState ∗ *other*) `[virtual]`

Function checking if the other object is equal to this.

Returns

true if objects data is the same.

Reimplemented from BaseState.

3.23.3.2 QString RunGenState::getArgs ( ) `[inline]`

Getter function for genArgs.

3.23.3.3 TreeItem∗ RunGenState::getChild (int *i*, TreeItem ∗ *parent*) `[virtual]`

Returns the i'th children of this state.

Reimplemented from BaseState.

3.23.3.4 Coordinates∗ RunGenState::getCoords ( ) `[inline]`

Getter function for coords.

3.23.3.5 QString RunGenState::getFilePath ( ) `[inline]`

Getter function for filePath.

3.23.3.6 GeneratorType RunGenState::getGenType ( ) `[inline]`

Getter function for genType.

3.23.3.7 Robot RunGenState::getRobot ( ) `[inline]`

Getter function for robot.

3.23.3.8 QString RunGenState::getSpeech ( ) `[inline]`

Getter function for speech.

3.23.3.9 int RunGenState::itemCount ( ) `[inline, virtual]`

Function used to count how many children should there be for state's TreeView item.

Reimplemented from BaseState.

3.23.3.10 QStringList RunGenState::LoadFromXML (QXmlStreamReader ∗ *reader*) `[virtual]`

Function loading a State from XML reader Stream.

Reimplemented from BaseState.

3.23.3.11 std::string RunGenState::Print ( ) `[virtual]`

Function printing the attributes of the state into a String.

Reimplemented from BaseState.

3.23.3.12 void RunGenState::Print (QXmlStreamWriter ∗ *writer*) `[virtual]`

Function printing the state to XML writer stream.

Reimplemented from BaseState.

3.23.3.13 void RunGenState::setArgs (QString *newGenArgs*) `[inline]`

Setter function for genArgs.

3.23.3.14 void RunGenState::setCoords (Coordinates ∗ *newCoords*) `[inline]`

Setter function for coords.

3.23.3.15 void RunGenState::setFilePath (QString *newPath*) `[inline]`

Setter function for filePath.

3.23.3.16 void RunGenState::setGenType (GeneratorType *newGenType*) `[inline]`

Setter function for genType.

3.23.3.17 void RunGenState::setRobot (Robot *newRobot*) `[inline]`

Setter function for robot.

3.23.3.18 void RunGenState::setSpeech (QString *newSpeech*) `[inline]`

Setter function for speech.

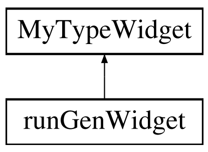
The documentation for this class was generated from the following file:

- States.h

3.24 runGenWidget Class Reference

`#include <StateTypeWidgets.h>`

Inheritance diagram for runGenWidget:



Signals

- void reportError (QString msgString)

Public Member Functions

- runGenWidget (QWidget ∗parent, Model ∗newmod)
- BaseState ∗ getStateObject ()
- void setState (BaseState ∗state)

3.24.1 Detailed Description

Widget allowing to edit runGenState.

3.24.2 Constructor & Destructor Documentation

3.24.2.1 runGenWidget::runGenWidget (QWidget ∗ *parent*, Model ∗ *newmod*)

Constructor creating this widget and all it's sub-widgets.

3.24.3 Member Function Documentation

3.24.3.1 BaseState∗ runGenWidget::getStateObject ( ) `[virtual]`

Returns a State with proper type and all the data from the widget.

Implements MyTypeWidget.

3.24.3.2 void runGenWidget::reportError (QString *msgString*) `[signal]`

Signal to parent, that an error has ocured.

3.24.3.3 void runGenWidget::setState (BaseState ∗ *state*) `[virtual]`

Function opening state for edition in the widget.

Implements MyTypeWidget.

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

- StateTypeWidgets.h

3.25 state_t Struct Reference

Public Types

- typedef vertex_property_tag **kind**

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

- Graph.h

3.26 StateWidget Class Reference

```
#include <stateWidget.h>
```

Signals

- void reportError (QString msgString)
- void ReplaceState (BaseState ∗oldState, BaseState ∗newState)

Public Member Functions

- **StateWidget** (QWidget ∗w, Model ∗newmod)
- void refreshData ()
- void StateSelected (BaseState ∗ToLoadState)
- void setOKButtonDisabled ()

3.26.1 Detailed Description

Widget allowing to edit states, containing widgets for all child-classes of BaseState.

3.26.2 Member Function Documentation

3.26.2.1 void StateWidget::refreshData ( )

Created for future needs, does nothing.

3.26.2.2 void StateWidget::ReplaceState (**BaseState** ∗ *oldState*, **BaseState** ∗ *newState*) `[signal]`

Signals, that user requests to change oldState to newState.

3.26.2.3 void StateWidget::reportError (QString *msgString*) `[signal]`

Signals, that newState will be inserted. Signals to a parent widget, that an error has occured.

3.26.2.4 void StateWidget::setOKButtonDisabled ( ) `[inline]`

Disables OK button and notes, that no state is currently edited.

3.26.2.5 void StateWidget::StateSelected (**BaseState** ∗ *ToLoadState*)

opens a MyTypeWidget connected to the state of ToLoadState, and loads all the data of ToLoadState to relevant fields.

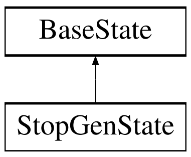
The documentation for this class was generated from the following file:

- stateWidget.h

3.27 StopGenState Class Reference

`#include <States.h>`

Inheritance diagram for StopGenState:



Public Member Functions

- StopGenState ()
- StopGenState (StopGenState &old)
- ∼StopGenState ()
- bool equals (BaseState ∗other)
- RobotSet getSet ()
- void setSet (RobotSet newSet)
- void Print (QXmlStreamWriter ∗writer)
- std::string Print ()
- QStringList LoadFromXML (QXmlStreamReader ∗reader)
- int itemCount ()
- TreeItem ∗ getChild (int i, TreeItem ∗parent)

3.27.1 Detailed Description

State representing stopping movement of a set of robots.

3.27.2 Constructor & Destructor Documentation

3.27.2.1 StopGenState::StopGenState ( ) `[inline]`

Empty constructor setting stateType.

3.27.2.2 StopGenState::StopGenState (StopGenState & *old*) `[inline]`

Copy constructor copying all date from state old.

3.27.2.3 StopGenState::∼StopGenState ( ) `[inline]`

Empty destructor.

3.27.3 Member Function Documentation

3.27.3.1 bool StopGenState::equals (BaseState ∗ *other*) `[virtual]`

Function checking if the other object is equal to this.

Returns

true if objects data is the same.

Reimplemented from BaseState.

3.27.3.2 TreeItem* StopGenState::getChild (int *i*, TreeItem * *parent*) [virtual]

Returns the i'th children of this state.

Reimplemented from BaseState.

3.27.3.3 RobotSet StopGenState::getSet ( ) [inline]

Getter function for set.

3.27.3.4 int StopGenState::itemCount ( ) [inline, virtual]

Function used to count how many children should there be for state's TreeView item.

Reimplemented from BaseState.

3.27.3.5 QStringList StopGenState::LoadFromXML (QXmlStreamReader * *reader*) [virtual]

Function loading a State from XML reader Stream.

Reimplemented from BaseState.

3.27.3.6 void StopGenState::Print (QXmlStreamWriter * *writer*) [virtual]

Function printing the state to XML writer stream.

Reimplemented from BaseState.

3.27.3.7 std::string StopGenState::Print ( ) [virtual]

Function printing the attributes of the state into a String.

Reimplemented from BaseState.

3.27.3.8 void StopGenState::setSet (RobotSet *newSet*) [inline]

Setter function for set.

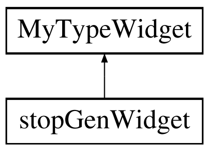
The documentation for this class was generated from the following file:

- States.h

3.28 stopGenWidget Class Reference

`#include <StateTypeWidgets.h>`

Inheritance diagram for stopGenWidget:



Signals

- void reportError (QString msgString)

Public Member Functions

- stopGenWidget (QWidget ∗parent, Model ∗newmod)
- BaseState ∗ getStateObject ()
- void setState (BaseState ∗state)

3.28.1 Detailed Description

Widget allowing to edit stopGenState.

3.28.2 Constructor & Destructor Documentation

3.28.2.1 stopGenWidget::stopGenWidget (QWidget ∗ *parent*, Model ∗ *newmod*)

Constructor creating this widget and all it's sub-widgets.

3.28.3 Member Function Documentation

3.28.3.1 BaseState∗ stopGenWidget::getStateObject ( ) `[virtual]`

Returns a State with proper type and all the data from the widget.

Implements MyTypeWidget.

3.28.3.2 void stopGenWidget::reportError (QString *msgString*) `[signal]`

Signal to parent, that an error has ocured.

3.28.3.3 void stopGenWidget::setState (BaseState ∗ *state*) `[virtual]`

Function opening state for edition in the widget.

Implements MyTypeWidget.

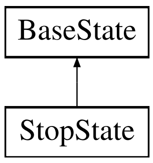
The documentation for this class was generated from the following file:

- StateTypeWidgets.h

3.29 StopState Class Reference

`#include <States.h>`

Inheritance diagram for StopState:



Public Member Functions

- StopState ()
- StopState (StopState &old)
- ∼StopState ()

- bool equals (BaseState ∗other)
- void Print (QXmlStreamWriter ∗writer)
- std::string Print ()
- QStringList LoadFromXML (QXmlStreamReader ∗reader)
- int itemCount ()
- TreeItem ∗ getChild (int i, TreeItem ∗parent)

3.29.1 Detailed Description

State representing end of a task/subtask.

3.29.2 Constructor & Destructor Documentation

3.29.2.1 StopState::StopState ( ) `[inline]`

Empty constructor setting stateType.

3.29.2.2 StopState::StopState (StopState & *old*) `[inline]`

Copy constructor copying all date from state old.

3.29.2.3 StopState::∼StopState ( ) `[inline]`

Empty destructor.

3.29.3 Member Function Documentation

3.29.3.1 bool StopState::equals (BaseState ∗ *other*) `[inline, virtual]`

Function checking if the other object is equal to this.

Returns

true if objects data is the same.

Reimplemented from BaseState.

3.29.3.2 TreeItem∗ StopState::getChild (int *i*, TreeItem ∗ *parent*) `[inline, virtual]`

Returns the i'th children of this state.

Returns

NULL, because no children are present.

Reimplemented from BaseState.

3.29.3.3 int StopState::itemCount ( ) `[inline, virtual]`

Function used to count how many children should there be for state's TreeView item.

Reimplemented from BaseState.

3.29.3.4 QStringList StopState::LoadFromXML (QXmlStreamReader ∗ *reader*) `[virtual]`

Function loading a State from XML reader Stream.

Reimplemented from BaseState.

3.29.3.5 std::string StopState::Print ( ) [virtual]

Function printing the attributes of the state into a String.

Reimplemented from BaseState.

3.29.3.6 void StopState::Print (QXmlStreamWriter * *writer*) [virtual]

Function printing the state to XML writer stream.

Reimplemented from BaseState.

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

- States.h

3.30 SubtaskWidget Class Reference

```
#include <subtaskWidget.h>
```

Signals

- void UncheckTasksAction ()
- void added (QString)
- void removed (QString)
- void changed (QString oldName, QString newName)
- void reportError (QString)

Public Member Functions

- **SubtaskWidget** (QWidget *parent, Model *mod)
- void refreshData ()

3.30.1 Detailed Description

Widget responsible for editing, creating and deleting tasks.

3.30.2 Member Function Documentation

3.30.2.1 void SubtaskWidget::added (QString) [signal]

Signals that a new task should be added.

3.30.2.2 void SubtaskWidget::changed (QString *oldName*, QString *newName*) [signal]

Signals, that a name should be changed

3.30.2.3 void SubtaskWidget::refreshData ( )

Downloads list of tasks from the model.

3.30.2.4 void SubtaskWidget::removed (QString) [signal]

Signals, that the selected task should be removed.

3.30.2.5 void SubtaskWidget::reportError (QString) `[signal]`

Signals to parent Widget, that an error has occured.

3.30.2.6 void SubtaskWidget::UncheckTasksAction ( ) `[signal]`

Signals that an action has been done and the icon representing this widget should be unchecked.

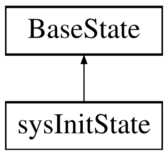
The documentation for this class was generated from the following file:

- subtaskWidget.h

3.31 sysInitState Class Reference

`#include <States.h>`

Inheritance diagram for sysInitState:



Public Member Functions

- sysInitState ()
- sysInitState (sysInitState &old)
- ∼sysInitState ()
- bool equals (BaseState ∗other)
- std::vector< robotInit > getInits ()
- void setInits (std::vector< robotInit > newInits)
- Transmitter getTransmitter ()
- void setTransmitter (Transmitter newTrans)
- std::vector< Sensor > getSensors ()
- void setSensors (std::vector< Sensor > newSensors)
- void Print (QXmlStreamWriter ∗writer)
- std::string Print ()
- QStringList LoadFromXML (QXmlStreamReader ∗reader)
- int itemCount ()
- TreeItem ∗ getChild (int i, TreeItem ∗parent)

3.31.1 Detailed Description

State representing system initialization.

3.31.2 Constructor & Destructor Documentation

3.31.2.1 sysInitState::sysInitState ( ) `[inline]`

Empty constructor setting stateType.

3.31.2.2 sysInitState::sysInitState (sysInitState & *old*) `[inline]`

Copy constructor copying all date from state old.

3.31.2.3 sysInitState::∼sysInitState ( ) [inline]

Empty destructor.

3.31.3 Member Function Documentation

3.31.3.1 bool sysInitState::equals (BaseState ∗ *other*) [virtual]

Function checking if the other object is equal to this.

Returns

true if objects data is the same.

Reimplemented from BaseState.

3.31.3.2 TreeItem∗ sysInitState::getChild (int *i*, TreeItem ∗ *parent*) [virtual]

Returns the i'th children of this state.

Reimplemented from BaseState.

3.31.3.3 std::vector<robotInit> sysInitState::getInits ( ) [inline]

Getter function for inits.

3.31.3.4 std::vector<Sensor> sysInitState::getSensors ( ) [inline]

Getter function for sensors.

3.31.3.5 Transmitter sysInitState::getTransmitter ( ) [inline]

Getter function for transmitter.

3.31.3.6 int sysInitState::itemCount ( ) [inline, virtual]

Function used to count how many children should there be for state's TreeView item.

Reimplemented from BaseState.

3.31.3.7 QStringList sysInitState::LoadFromXML (QXmlStreamReader ∗ *reader*) [virtual]

Function loading a State from XML reader Stream.

Reimplemented from BaseState.

3.31.3.8 void sysInitState::Print (QXmlStreamWriter ∗ *writer*) [virtual]

Function printing the state to XML writer stream.

Reimplemented from BaseState.

3.31.3.9 std::string sysInitState::Print ( ) [virtual]

Function printing the attributes of the state into a String.

Reimplemented from BaseState.

3.31.3.10 void sysInitState::setInits (std::vector< robotInit > *newInits*) `[inline]`

Setter function for inits.

3.31.3.11 void sysInitState::setSensors (std::vector< Sensor > *newSensors*) `[inline]`

Setter function for sensors.

3.31.3.12 void sysInitState::setTransmitter (Transmitter *newTrans*) `[inline]`

Setter function for transmitter.

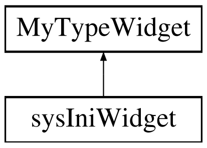
The documentation for this class was generated from the following file:

- States.h

3.32 sysIniWidget Class Reference

`#include <StateTypeWidgets.h>`

Inheritance diagram for sysIniWidget:



Signals

- void reportError (QString msgString)

Public Member Functions

- sysIniWidget (QWidget ∗parent, Model ∗newmod)
- BaseState ∗ getStateObject ()
- void setState (BaseState ∗state)

3.32.1 Detailed Description

Widget allowing to edit sysInitState.

3.32.2 Constructor & Destructor Documentation

3.32.2.1 sysIniWidget::sysIniWidget (QWidget ∗ *parent*, Model ∗ *newmod*)

Constructor creating this widget and all it's sub-widgets.

3.32.3 Member Function Documentation

3.32.3.1 BaseState∗ sysIniWidget::getStateObject ( ) `[virtual]`

Returns a State with proper type and all the data from the widget.

Implements MyTypeWidget.

3.32.3.2 void sysIniWidget::reportError (QString *msgString*) `[signal]`

Signal to parent, that an error has ocured.

3.32.3.3 void sysIniWidget::setState (BaseState ∗ *state*) `[virtual]`

Function opening state for edition in the widget.

Implements MyTypeWidget.

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

- StateTypeWidgets.h

3.33 TransDialog Class Reference

`#include <TransDialog.h>`

Signals

- void TransitionSelected (Transition ∗tr)
- void reportError (QString)

Public Member Functions

- TransDialog (QWidget ∗parent, Model ∗mod)
- ∼TransDialog ()
- void openForAState (BaseState ∗tmp)

3.33.1 Detailed Description

DialogBox allowing to change order of transitions for a state.

3.33.2 Constructor & Destructor Documentation

3.33.2.1 TransDialog::TransDialog (QWidget ∗ *parent,* Model ∗ *mod*)

Constructor creating the widget and all it's elements.

3.33.2.2 TransDialog::∼TransDialog ( ) `[inline]`

Destructor for this widget.

3.33.3 Member Function Documentation

3.33.3.1 void TransDialog::openForAState (BaseState ∗ *tmp*)

Opens the dialogbox for a state loading it's transitions from the model.

3.33.3.2 void TransDialog::reportError (QString) `[signal]`

reports to parent widget, that an error has occured.

3.33.3.3 void TransDialog::TransitionSelected (**Transition** ∗ *tr*) `[signal]`

Signals that a transition should be selected.

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

- TransDialog.h

3.34 Transition Class Reference

```
#include <Transition.h>
```

Public Types

- enum { **Type** = UserType + 4 }

Public Slots

- void updatePosition ()

Public Member Functions

- QGraphicsScene ∗ getScene ()
- void setScene (QGraphicsScene ∗sc)
- Transition (BaseState ∗startItem, BaseState ∗endItem, QGraphicsItem ∗parent=0, QGraphicsScene ∗scene=0)
- ∼Transition ()
- ConditionType getCondType ()
- void setCondType (ConditionType newCondType)
- int type () const
- QRectF boundingRect () const
- QPainterPath shape () const
- void setStartItem (BaseState ∗newStartItem)
- void setEndItem (BaseState ∗newEndItem)
- void setZValue (qreal z)
- void removeSubtask ()
- BaseState ∗ startItem () const
- BaseState ∗ endItem () const
- QString getCondition ()
- void setCondition (QString newCondition)
- BaseState ∗ getSubtask ()
- void setSubtask (BaseState ∗newSubtask)
- std::string Print ()
- void Print (QXmlStreamWriter ∗writer)

Protected Member Functions

- void paint (QPainter ∗painter, const QStyleOptionGraphicsItem ∗option, QWidget ∗widget=0)

Protected Attributes

- ConditionType CondType
- QGraphicsScene ∗ scene
- std::vector< QGraphicsLineItem ∗ > lines
- BaseState ∗ myStartItem
- BaseState ∗ myEndItem
- QPolygonF TransitionHead
- QString condition
- BaseState ∗ subtask
- QGraphicsTextItem ∗ subtaskItem

3.34.1 Detailed Description

Class representing a transition between the tasks, and it's graphic representation.

3.34.2 Constructor & Destructor Documentation

3.34.2.1 Transition::Transition (BaseState ∗ *startItem*, BaseState ∗ *endItem*, QGraphicsItem ∗ *parent* = `0`, QGraphicsScene ∗ *scene* = `0`)

Constructor creating Transition between startItem and EndItem.

3.34.2.2 Transition::∼Transition ( )

Destructor removing this item from subtask list of the state pointed as subtask and removes this item from scene.

3.34.3 Member Function Documentation

3.34.3.1 QRectF Transition::boundingRect ( ) const

Returns QRectf bounding the transition.

3.34.3.2 BaseState∗ Transition::endItem ( ) const `[inline]`

Getter function for myEndItem.

3.34.3.3 QString Transition::getCondition ( ) `[inline]`

Getter function for condition.

3.34.3.4 ConditionType Transition::getCondType ( ) `[inline]`

Getter function for CondType.

3.34.3.5 QGraphicsScene∗ Transition::getScene ( ) `[inline]`

Getter function for scene.

3.34.3.6 BaseState∗ Transition::getSubtask ( ) `[inline]`

Getter function for subtask.

3.34.3.7 void Transition::paint (QPainter ∗ *painter,* const QStyleOptionGraphicsItem ∗ *option,* QWidget ∗ *widget* = 0) [protected]

Paints line, and head of the transition with colour dependent on isSelected().

3.34.3.8 std::string Transition::Print ( ) [inline]

Creates a string describing the coordinates attributes.

Returns

String with the description of the Coordinates

3.34.3.9 void Transition::Print (QXmlStreamWriter ∗ *writer*) [inline]

Writes the data of the state to the XML stream.

Parameters

writer	Stream to which the data is written

3.34.3.10 void Transition::removeSubtask ( ) [inline]

Sets subtask to NULL pointer.

3.34.3.11 void Transition::setCondition (QString *newCondition*) [inline]

Setter function for subtask.

3.34.3.12 void Transition::setCondType (ConditionType *newCondType*) [inline]

Setter function for CondType.

3.34.3.13 void Transition::setEndItem (BaseState ∗ *newEndItem*) [inline]

Sets state, which is an end state for this transition

3.34.3.14 void Transition::setScene (QGraphicsScene ∗ *sc*)

Setter function for scene adding this item, all elements of lines vector and subtaskItem.

3.34.3.15 void Transition::setStartItem (BaseState ∗ *newStartItem*) [inline]

Sets state, which is a start state for this transition.

3.34.3.16 void Transition::setSubtask (BaseState ∗ *newSubtask*) [inline]

Sets new subtask value (state pointer) and removes this transition from the list of pointers of the old state, and adds it to the new state.

3.34.3.17 void Transition::setZValue (qreal *z*)

Changes the z value (position of overlapping towards other elements).

3.34.3.18 QPainterPath Transition::shape ( ) const

Returns the path of the shape of transition.

3.34.3.19 BaseState∗ Transition::startItem ( ) const [inline]

Getter function for myStartItem.

3.34.3.20 int Transition::type ( ) const [inline]

Getter function for type.

3.34.3.21 void Transition::updatePosition ( ) [slot]

Creates new, actual line of the transition.

3.34.4 Member Data Documentation

3.34.4.1 QString Transition::condition [protected]

String representing the condition of this transition.

3.34.4.2 ConditionType Transition::CondType [protected]

Type of the condition.

3.34.4.3 std::vector<QGraphicsLineItem ∗> Transition::lines [protected]

Additional lines usedfor transition whose start and end element are the same.

3.34.4.4 BaseState∗ Transition::myEndItem [protected]

End item of the transition.

3.34.4.5 BaseState∗ Transition::myStartItem [protected]

Start item of the transition.

3.34.4.6 QGraphicsScene∗ Transition::scene [protected]

Scene on which this item is.

3.34.4.7 BaseState∗ Transition::subtask [protected]

Pointer to the subtask starting point of the transition.

3.34.4.8 QGraphicsTextItem∗ Transition::subtaskItem [protected]

GraphicsItem representing name of the State, which is a subtask to this transition.

3.34.4.9 QPolygonF Transition::TransitionHead `[protected]`

Polygon representing the arrowhead of the transition.

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

- Transition.h

3.35 transition_t Struct Reference

Public Types

- typedef edge_property_tag **kind**

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

- Graph.h

3.36 TransWidget Class Reference

```
#include <transWidget.h>
```

Signals

- void reportError (QString)

Public Member Functions

- TransWidget (QWidget ∗parent, Model ∗newmod)
- void refreshData ()
- void TransSelected (Transition ∗)
- void setOKButtonDisabled ()

3.36.1 Detailed Description

a Widget, which allows to edit and create Transitions.

3.36.2 Constructor & Destructor Documentation

3.36.2.1 TransWidget::TransWidget (QWidget ∗ *parent*, Model ∗ *newmod*)

Constructor creating this widget and all its elements.

3.36.3 Member Function Documentation

3.36.3.1 void TransWidget::refreshData ( )

Refreshes data, especially the subtasks list.

3.36.3.2 void TransWidget::reportError (QString) `[signal]`

Signals,that a transition will be inserted, gibving it's attributes. Reports to the parent widget, that an error has occured.

3.36.3.3 void TransWidget::setOKButtonDisabled ( ) `[inline]`

Disables OK button (used when condition.size()==0).

3.36.3.4 void TransWidget::TransSelected (**Transition** $*$)

Loads data of a Transition to edit.

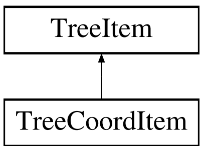
The documentation for this class was generated from the following file:

- transWidget.h

3.37 TreeCoordItem Class Reference

`#include <TreeItem.h>`

Inheritance diagram for TreeCoordItem:



Public Member Functions

- TreeCoordItem (int row, TreeItem $*$parent=0)
- ∼TreeCoordItem ()
- void setCoords (Coordinates $*$_coords)
- QGraphicsItem $*$ getGraphicsItem ()
- int childNodesCount ()
- TreeItem $*$ child (int i)
- QString Name ()
- QString Attr ()

3.37.1 Detailed Description

Item of the treeview, which represents Coordinates.

3.37.2 Constructor & Destructor Documentation

3.37.2.1 TreeCoordItem::TreeCoordItem (int *row*, **TreeItem** $*$ *parent* = 0) `[inline]`

Creates an item with given parent, and saves the row value.

3.37.2.2 TreeCoordItem::∼TreeCoordItem ( ) `[inline]`

Destructor of the item.

3.37.3 Member Function Documentation

3.37.3.1 QString TreeCoordItem::Attr ( ) `[inline, virtual]`

Returns Value (2nd column value) for this element.

Reimplemented from TreeItem.

3.37.3.2 TreeItem∗ TreeCoordItem::child (int *i*) `[virtual]`

Returns the i-th child of this element.

Reimplemented from TreeItem.

3.37.3.3 int TreeCoordItem::childNodesCount ( ) `[inline, virtual]`

Returns number of children, which this item has.

Reimplemented from TreeItem.

3.37.3.4 QGraphicsItem∗ TreeCoordItem::getGraphicsItem ( ) `[inline, virtual]`

Returns the first met graphicsItem while going up the item tree.

Reimplemented from TreeItem.

3.37.3.5 QString TreeCoordItem::Name ( ) `[inline, virtual]`

Returns name (1st column value) for this element.

Reimplemented from TreeItem.

3.37.3.6 void TreeCoordItem::setCoords (Coordinates ∗ *_coords*) `[inline]`

Sets coordinates, which are represented by this item.

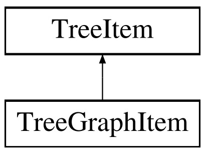
The documentation for this class was generated from the following file:

- TreeItem.h

3.38 TreeGraphItem Class Reference

`#include <TreeItem.h>`

Inheritance diagram for TreeGraphItem:



Public Member Functions

- TreeGraphItem (int row, TreeItem ∗parent=0)
- ∼TreeGraphItem ()
- void setGraph (MyGraphType ∗_gr, TreeModel ∗mod)
- int childNodesCount ()
- TreeItem ∗ child (int i)
- QGraphicsItem ∗ getGraphicsItem ()
- QString Name ()
- QString Attr ()

3.38.1 Detailed Description

Item of the treeview, which represents a Graph; it's a parent to all other items of the TreeView.

3.38.2 Constructor & Destructor Documentation

3.38.2.1 TreeGraphItem::TreeGraphItem (int *row*, TreeItem ∗ *parent* = 0) `[inline]`

Creates an item with given parent, and saves the row value.

3.38.2.2 TreeGraphItem::∼TreeGraphItem ( ) `[inline]`

Destructor of the item.

3.38.3 Member Function Documentation

3.38.3.1 QString TreeGraphItem::Attr ( ) `[inline, virtual]`

Returns Value (2nd column value) for this element.

Reimplemented from TreeItem.

3.38.3.2 TreeItem∗ TreeGraphItem::child (int *i*) `[virtual]`

Returns the i-th child of this element.

Reimplemented from TreeItem.

3.38.3.3 int TreeGraphItem::childNodesCount ( ) `[inline, virtual]`

Returns number of children, which this item has.

Reimplemented from TreeItem.

3.38.3.4 QGraphicsItem∗ TreeGraphItem::getGraphicsItem ( ) `[inline, virtual]`

Returns the first met graphicsItem while going up the item tree.

Reimplemented from TreeItem.

3.38.3.5 QString TreeGraphItem::Name ( ) `[inline, virtual]`

Returns name (1st column value) for this element.

Reimplemented from TreeItem.

3.38.3.6 void TreeGraphItem::setGraph (MyGraphType ∗ *_gr*, TreeModel ∗ *mod*) `[inline]`

Sets Graph, which is represented by this item and sets TreeModel, which is used to get list of states of this Graph.

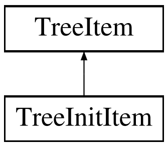
The documentation for this class was generated from the following file:

- TreeItem.h

3.39 TreeInitItem Class Reference

```
#include <TreeItem.h>
```

Inheritance diagram for TreeInitItem:



Public Member Functions

- TreeInitItem (int row, TreeItem ∗parent=0)
- ∼TreeInitItem ()
- void setInit (robotInit _init)
- int childNodesCount ()
- TreeItem ∗ child (int i)
- QGraphicsItem ∗ getGraphicsItem ()
- QString Name ()
- QString Attr ()

3.39.1 Detailed Description

Item of the treeview, which represents a robotInit.

3.39.2 Constructor & Destructor Documentation

3.39.2.1 TreeInitItem::TreeInitItem (int *row*, TreeItem ∗ *parent* = 0) `[inline]`

Creates an item with given parent, and saves the row value.

3.39.2.2 TreeInitItem::∼TreeInitItem ( ) `[inline]`

Destructor of the item.

3.39.3 Member Function Documentation

3.39.3.1 QString TreeInitItem::Attr ( ) `[inline, virtual]`

Returns Value (2nd column value) for this element.

Reimplemented from TreeItem.

3.39.3.2 TreeItem∗ TreeInitItem::child (int *i*) `[virtual]`

Returns the i-th child of this element.

Reimplemented from TreeItem.

3.39.3.3 int TreeInitItem::childNodesCount ( ) `[inline, virtual]`

Returns number of children, which this item has.

Reimplemented from TreeItem.

3.39.3.4 QGraphicsItem∗ TreeInitItem::getGraphicsItem ( ) `[inline, virtual]`

Returns the first met graphicsItem while going up the item tree.

Reimplemented from TreeItem.

3.39.3.5 QString TreeInitItem::Name ( ) `[inline, virtual]`

Returns name (1st column value) for this element.

Reimplemented from TreeItem.

3.39.3.6 void TreeInitItem::setInit (robotInit *_init*) `[inline]`

Sets robotInit which is represented by this item.

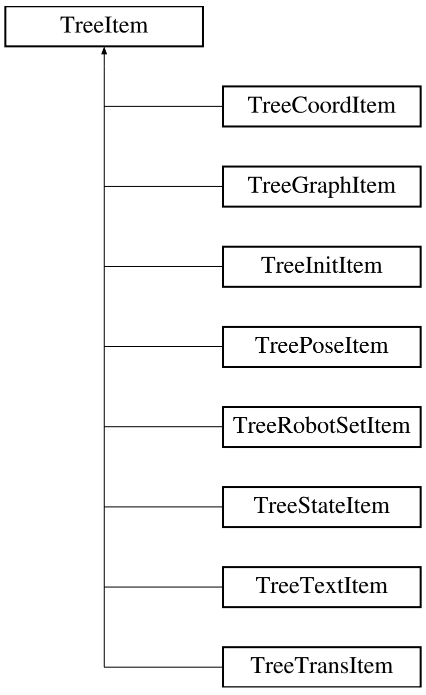
The documentation for this class was generated from the following file:

- TreeItem.h

3.40 TreeItem Class Reference

`#include <TreeItem.h>`

Inheritance diagram for TreeItem:



Public Member Functions

- TreeItem (int row, TreeItem ∗parent=0)
- ∼TreeItem ()
- TreeItem ∗ parent ()
- int row ()
- int getType ()
- virtual int childNodesCount ()
- virtual QString Name ()

- virtual QString Attr ()
- virtual TreeItem ∗ child (int i)
- virtual QGraphicsItem ∗ getGraphicsItem ()

Protected Attributes

- int Type
- QHash< int, TreeItem ∗ > childItems
- TreeItem ∗ parentItem
- int rowNumber

3.40.1 Detailed Description

Base class of item of the TreeView.

3.40.2 Constructor & Destructor Documentation

3.40.2.1 TreeItem::TreeItem (int *row,* TreeItem ∗ *parent* = 0)

Creates an item with given parent, and saves the row value.

3.40.2.2 TreeItem::∼TreeItem ( )

Destructor for base item.

3.40.3 Member Function Documentation

3.40.3.1 virtual QString TreeItem::Attr ( ) `[inline, virtual]`

Returns Value (2nd column value) for this element.

Reimplemented in TreeStateItem, TreeTransItem, TreeCoordItem, TreeRobotSetItem, TreeInitItem, TreePoseItem, TreeTextItem, and TreeGraphItem.

3.40.3.2 virtual TreeItem∗ TreeItem::child (int *i*) `[inline, virtual]`

Returns the i-th child of this element.

Reimplemented in TreeStateItem, TreeTransItem, TreeCoordItem, TreeRobotSetItem, TreeInitItem, TreePoseItem, TreeTextItem, and TreeGraphItem.

3.40.3.3 virtual int TreeItem::childNodesCount ( ) `[inline, virtual]`

Returns number of children, which this item has.

Reimplemented in TreeStateItem, TreeTransItem, TreeCoordItem, TreeRobotSetItem, TreeInitItem, TreePoseItem, TreeTextItem, and TreeGraphItem.

3.40.3.4 virtual QGraphicsItem∗ TreeItem::getGraphicsItem ( ) `[inline, virtual]`

Returns the first met graphicsItem while going up the item tree.

Reimplemented in TreeStateItem, TreeTransItem, TreeCoordItem, TreeRobotSetItem, TreeInitItem, TreePoseItem, TreeTextItem, and TreeGraphItem.

3.40.3.5 int TreeItem::getType ( ) [inline]

Getter function for Type.

3.40.3.6 virtual QString TreeItem::Name ( ) [inline, virtual]

Returns name (1st column value) for this element.

Reimplemented in TreeStateItem, TreeTransItem, TreeCoordItem, TreeRobotSetItem, TreeInitItem, TreePoseItem, TreeTextItem, and TreeGraphItem.

3.40.3.7 TreeItem∗ TreeItem::parent ( )

Returns parent of this element.

3.40.3.8 int TreeItem::row ( )

Returns number of the row in that level (considering only children of parent of this element) on which this element is.

3.40.4 Member Data Documentation

3.40.4.1 QHash<int,TreeItem∗> TreeItem::childItems [protected]

Table of items, used to store items, not to create them with every use of child(i).

3.40.4.2 TreeItem∗ TreeItem::parentItem [protected]

Parentitem of this element.

3.40.4.3 int TreeItem::rowNumber [protected]

Number of the row, in which this item is in the list of the children of parentelement.

3.40.4.4 int TreeItem::Type [protected]

Type of the item.

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

- TreeItem.h

3.41 TreeModel Class Reference

```
#include <TreeModel.h>
```

Public Member Functions

- TreeModel (QObject ∗parent, Model ∗mod, QString Name)
- ∼TreeModel ()
- QVariant data (const QModelIndex &index, int role) const
- Qt::ItemFlags flags (const QModelIndex &index) const
- QVariant headerData (int section, Qt::Orientation orientation, int role=Qt::DisplayRole) const
- QModelIndex index (int row, int column, const QModelIndex &parent=QModelIndex()) const

- QModelIndex parent (const QModelIndex &child) const
- int rowCount (const QModelIndex &parent=QModelIndex()) const
- int columnCount (const QModelIndex &parent=QModelIndex()) const
- QGraphicsItem ∗ getItemOrParent (QModelIndex index)

Public Attributes

- Model ∗ mod

3.41.1 Detailed Description

Class representing model of the TreeView.

3.41.2 Constructor & Destructor Documentation

3.41.2.1 TreeModel::TreeModel (QObject ∗ *parent*, Model ∗ *mod*, QString *Name*)

Creates the model for the treeview.

3.41.2.2 TreeModel::∼TreeModel ( )

Destructor for the model of the treeview.

3.41.3 Member Function Documentation

3.41.3.1 int TreeModel::columnCount (const QModelIndex & *parent* = `QModelIndex()`) const

Returns number of colums in the view.

3.41.3.2 QVariant TreeModel::data (const QModelIndex & *index*, int *role*) const

Returns data (text) from the index.

3.41.3.3 Qt::ItemFlags TreeModel::flags (const QModelIndex & *index*) const

Returns flags for the index.

3.41.3.4 QGraphicsItem∗ TreeModel::getItemOrParent (QModelIndex *index*)

Returns graphics item of the item at the given index, or of it's parent (recursively).

3.41.3.5 QVariant TreeModel::headerData (int *section*, Qt::Orientation *orientation*, int *role* = `Qt::DisplayRole`) const

Returns data for headers of the view.

3.41.3.6 QModelIndex TreeModel::index (int *row*, int *column*, const QModelIndex & *parent* = `QModelIndex()`) const

Returns index fromt he current row and column.

3.41.3.7 QModelIndex TreeModel::parent (const QModelIndex & *child*) const

Returns parent of the given item.

Parameters

child	Item, of which the parent is needed.

3.41.3.8 int TreeModel::rowCount (const QModelIndex & *parent* = `QModelIndex()`) const

Returns number of childitems of the current item.

3.41.4 Member Data Documentation

3.41.4.1 Model∗ TreeModel::mod

Model of the project.

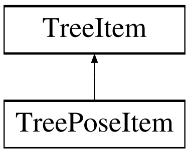
The documentation for this class was generated from the following file:

- TreeModel.h

3.42 TreePoseItem Class Reference

```
#include <TreeItem.h>
```

Inheritance diagram for TreePoseItem:



Public Member Functions

- TreePoseItem (int row, TreeItem ∗parent=0)
- ∼TreePoseItem ()
- void setPos (Pose ∗_pos)
- int childNodesCount ()
- TreeItem ∗ child (int i)
- QGraphicsItem ∗ getGraphicsItem ()
- QString Name ()
- QString Attr ()

3.42.1 Detailed Description

Item of the treeview, which represents a Pose.

3.42.2 Constructor & Destructor Documentation

3.42.2.1 TreePoseItem::TreePoseItem (int *row*, TreeItem ∗ *parent* = 0) [inline]

Creates an item with given parent, and saves the row value.

3.42.2.2 TreePoseItem::∼TreePoseItem ( ) `[inline]`

Destructor of the item.

3.42.3 Member Function Documentation

3.42.3.1 QString TreePoseItem::Attr ( ) `[inline, virtual]`

Returns Value (2nd column value) for this element.

Reimplemented from TreeItem.

3.42.3.2 TreeItem∗ TreePoseItem::child (int *i*) `[virtual]`

Returns the i-th child of this element.

Reimplemented from TreeItem.

3.42.3.3 int TreePoseItem::childNodesCount ( ) `[inline, virtual]`

Returns number of children, which this item has.

Reimplemented from TreeItem.

3.42.3.4 QGraphicsItem∗ TreePoseItem::getGraphicsItem ( ) `[inline, virtual]`

Returns the first met graphicsItem while going up the item tree.

Reimplemented from TreeItem.

3.42.3.5 QString TreePoseItem::Name ( ) `[inline, virtual]`

Returns name (1st column value) for this element.

Reimplemented from TreeItem.

3.42.3.6 void TreePoseItem::setPos (Pose ∗ *_pos*) `[inline]`

Sets pose, which is represented by this item.

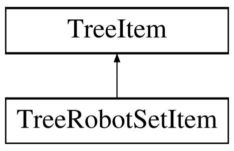
The documentation for this class was generated from the following file:

- TreeItem.h

3.43 TreeRobotSetItem Class Reference

```
#include <TreeItem.h>
```

Inheritance diagram for TreeRobotSetItem:



Public Member Functions

- TreeRobotSetItem (int row, TreeItem ∗parent=0)
- ∼TreeRobotSetItem ()
- void setSet (std::vector< Robot > _set)
- int childNodesCount ()
- TreeItem ∗ child (int i)
- QGraphicsItem ∗ getGraphicsItem ()
- QString Name ()
- QString Attr ()

3.43.1 Detailed Description

Item of the treeview, which represents a RobotSet.

3.43.2 Constructor & Destructor Documentation

3.43.2.1 TreeRobotSetItem::TreeRobotSetItem (int *row*, TreeItem ∗ *parent* = 0) `[inline]`

Creates an item with given parent, and saves the row value.

3.43.2.2 TreeRobotSetItem::∼TreeRobotSetItem ( ) `[inline]`

Destructor of the item.

3.43.3 Member Function Documentation

3.43.3.1 QString TreeRobotSetItem::Attr ( ) `[inline, virtual]`

Returns Value (2nd column value) for this element.

Reimplemented from TreeItem.

3.43.3.2 TreeItem∗ TreeRobotSetItem::child (int *i*) `[virtual]`

Returns the i-th child of this element.

Reimplemented from TreeItem.

3.43.3.3 int TreeRobotSetItem::childNodesCount ( ) `[inline, virtual]`

Returns number of children, which this item has.

Reimplemented from TreeItem.

3.43.3.4 QGraphicsItem∗ TreeRobotSetItem::getGraphicsItem ( ) `[inline, virtual]`

Returns the first met graphicsItem while going up the item tree.

Reimplemented from TreeItem.

3.43.3.5 QString TreeRobotSetItem::Name ( ) `[inline, virtual]`

Returns name (1st column value) for this element.

Reimplemented from TreeItem.

3.43.3.6 void TreeRobotSetItem::setSet (std::vector< Robot > *_set*) `[inline]`

Sets set, which is represented by this item, also defines if it's first or second set.

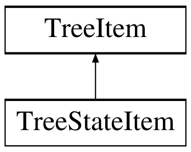
The documentation for this class was generated from the following file:

- TreeItem.h

3.44 TreeStateItem Class Reference

`#include <TreeItem.h>`

Inheritance diagram for TreeStateItem:



Public Member Functions

- TreeStateItem (int row, TreeItem ∗parent=0)
- ∼TreeStateItem ()
- void setState (BaseState ∗st)
- QGraphicsItem ∗ getGraphicsItem ()
- int childNodesCount ()
- TreeItem ∗ child (int i)
- QString Name ()
- QString Attr ()

3.44.1 Detailed Description

Item of the treeview, which represents a State (uses it's name and StateType).

3.44.2 Constructor & Destructor Documentation

3.44.2.1 TreeStateItem::TreeStateItem (int *row*, TreeItem ∗ *parent* = 0) `[inline]`

Creates an item with given parent, and saves the row value.

3.44.2.2 TreeStateItem::∼TreeStateItem ( ) `[inline]`

Destructor of the item.

3.44.3 Member Function Documentation

3.44.3.1 QString TreeStateItem::Attr ( ) `[inline, virtual]`

Returns Value (2nd column value) for this element.

Reimplemented from TreeItem.

3.44.3.2 TreeItem∗ TreeStateItem::child (int *i*) `[virtual]`

Returns the i-th child of this element.

Reimplemented from TreeItem.

3.44.3.3 int TreeStateItem::childNodesCount ( ) `[inline, virtual]`

Returns number of children, which this item has.

Reimplemented from TreeItem.

3.44.3.4 QGraphicsItem∗ TreeStateItem::getGraphicsItem ( ) `[inline, virtual]`

Returns the first met graphicsItem while going up the item tree.

Reimplemented from TreeItem.

3.44.3.5 QString TreeStateItem::Name ( ) `[inline, virtual]`

Returns name (1st column value) for this element.

Reimplemented from TreeItem.

3.44.3.6 void TreeStateItem::setState (BaseState ∗ *st*) `[inline]`

Sets State which is represented by this item.

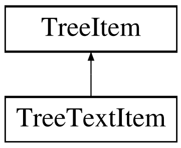
The documentation for this class was generated from the following file:

- TreeItem.h

3.45 TreeTextItem Class Reference

`#include <TreeItem.h>`

Inheritance diagram for TreeTextItem:



Public Member Functions

- TreeTextItem (int row, TreeItem ∗parent=0)
- ∼TreeTextItem ()
- void setNameAttr (QString _name, QString _attr)
- int childNodesCount ()
- TreeItem ∗ child (int i)
- QGraphicsItem ∗ getGraphicsItem ()
- QString Name ()
- QString Attr ()

3.45.1 Detailed Description

Item of the treeview, which represents various items without children.

3.45.2 Constructor & Destructor Documentation

3.45.2.1 TreeTextItem::TreeTextItem (int *row*, TreeItem ∗ *parent* = 0) `[inline]`

Creates an item with given parent, and saves the row value.

3.45.2.2 TreeTextItem::∼TreeTextItem ( ) `[inline]`

Destructor of the item.

3.45.3 Member Function Documentation

3.45.3.1 QString TreeTextItem::Attr ( ) `[inline, virtual]`

Returns Value (2nd column value) for this element.

Reimplemented from TreeItem.

3.45.3.2 TreeItem∗ TreeTextItem::child (int *i*) `[inline, virtual]`

Returns the i-th child of this element.

Reimplemented from TreeItem.

3.45.3.3 int TreeTextItem::childNodesCount ( ) `[inline, virtual]`

Returns number of children, which this item has.

Reimplemented from TreeItem.

3.45.3.4 QGraphicsItem∗ TreeTextItem::getGraphicsItem ( ) `[inline, virtual]`

Returns the first met graphicsItem while going up the item tree.

Reimplemented from TreeItem.

3.45.3.5 QString TreeTextItem::Name ( ) `[inline, virtual]`

Returns name (1st column value) for this element.

Reimplemented from TreeItem.

3.45.3.6 void TreeTextItem::setNameAttr (QString *_name*, QString *_attr*) `[inline]`

Sets name and value, which are represented by this value.

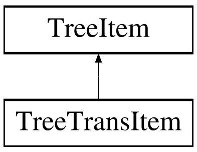
The documentation for this class was generated from the following file:

- TreeItem.h

3.46 TreeTransItem Class Reference

`#include <TreeItem.h>`

Inheritance diagram for TreeTransItem:



Public Member Functions

- TreeTransItem (int row, TreeItem ∗parent=0)
- ∼TreeTransItem ()
- void setTrans (Transition ∗_tr)
- QGraphicsItem ∗ getGraphicsItem ()
- int childNodesCount ()
- TreeItem ∗ child (int i)
- QString Name ()
- QString Attr ()

3.46.1 Detailed Description

Item of the treeview, which represents a Transition.

3.46.2 Constructor & Destructor Documentation

3.46.2.1 TreeTransItem::TreeTransItem (int *row*, TreeItem ∗ *parent* = 0) `[inline]`

Creates an item with given parent, and saves the row value.

3.46.2.2 TreeTransItem::∼TreeTransItem ( ) `[inline]`

Destructor of the item.

3.46.3 Member Function Documentation

3.46.3.1 QString TreeTransItem::Attr ( ) `[inline, virtual]`

Returns Value (2nd column value) for this element.

Reimplemented from TreeItem.

3.46.3.2 TreeItem∗ TreeTransItem::child (int *i*) `[virtual]`

Returns the i-th child of this element.

Reimplemented from TreeItem.

3.46.3.3 int TreeTransItem::childNodesCount ( ) `[inline, virtual]`

Returns number of children, which this item has.

Reimplemented from TreeItem.

3.46.3.4 QGraphicsItem∗ TreeTransItem::getGraphicsItem ( ) `[inline, virtual]`

Returns the first met graphicsItem while going up the item tree.

Reimplemented from TreeItem.

3.46.3.5 QString TreeTransItem::Name ( ) `[inline, virtual]`

Returns name (1st column value) for this element.

Reimplemented from TreeItem.

3.46.3.6 void TreeTransItem::setTrans (Transition ∗ *_tr*) `[inline]`

Sets Transition, which is represented by this item.

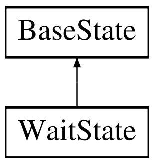
The documentation for this class was generated from the following file:

- TreeItem.h

3.47 WaitState Class Reference

`#include <States.h>`

Inheritance diagram for WaitState:



Public Member Functions

- WaitState ()
- WaitState (WaitState &old)
- ∼WaitState ()
- bool equals (BaseState ∗other)
- long long int getTimespan ()
- void setTimespan (long long int newTimeSpan)
- void Print (QXmlStreamWriter ∗writer)
- std::string Print ()
- QStringList LoadFromXML (QXmlStreamReader ∗reader)
- int itemCount ()
- TreeItem ∗ getChild (int i, TreeItem ∗parent)

3.47.1 Detailed Description

State representing a time delay in the system.

3.47.2 Constructor & Destructor Documentation

3.47.2.1 WaitState::WaitState ( ) `[inline]`

Empty constructor setting stateType.

3.47.2.2 WaitState::WaitState (WaitState & *old*) [inline]

Copy constructor copying all date from state old.

3.47.2.3 WaitState::∼WaitState ( ) [inline]

Empty destructor.

3.47.3 Member Function Documentation

3.47.3.1 bool WaitState::equals (BaseState ∗ *other*) [virtual]

Function checking if the other object is equal to this.

Returns

true if objects data is the same.

Reimplemented from BaseState.

3.47.3.2 TreeItem∗ WaitState::getChild (int *i*, TreeItem ∗ *parent*) [virtual]

Returns the i'th children of this state.

Reimplemented from BaseState.

3.47.3.3 long long int WaitState::getTimespan ( ) [inline]

Getter function for Timespan.

3.47.3.4 int WaitState::itemCount ( ) [inline, virtual]

Function used to count how many children should there be for state's TreeView item.

Reimplemented from BaseState.

3.47.3.5 QStringList WaitState::LoadFromXML (QXmlStreamReader ∗ *reader*) [virtual]

Function loading a State from XML reader Stream.

Reimplemented from BaseState.

3.47.3.6 void WaitState::Print (QXmlStreamWriter ∗ *writer*) [virtual]

Function printing the state to XML writer stream.

Reimplemented from BaseState.

3.47.3.7 std::string WaitState::Print ( ) [virtual]

Function printing the attributes of the state into a String.

Reimplemented from BaseState.

3.47.3.8 void WaitState::setTimespan (long long int *newTimeSpan*) `[inline]`

Setter function for Timespan.

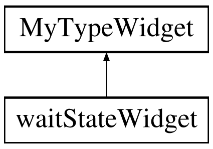
The documentation for this class was generated from the following file:

- States.h

3.48 waitStateWidget Class Reference

`#include <StateTypeWidgets.h>`

Inheritance diagram for waitStateWidget:



Public Member Functions

- waitStateWidget (QWidget ∗parent, Model ∗newmod)
- BaseState ∗ getStateObject ()
- void setState (BaseState ∗state)

3.48.1 Detailed Description

Widget allowing to edit waitState.

3.48.2 Constructor & Destructor Documentation

3.48.2.1 waitStateWidget::waitStateWidget (QWidget ∗ *parent*, Model ∗ *newmod*)

Constructor creating this widget and all it's sub-widgets.

3.48.3 Member Function Documentation

3.48.3.1 BaseState∗ waitStateWidget::getStateObject ( ) `[virtual]`

Returns a State with proper type and all the data from the widget.

Implements MyTypeWidget.

3.48.3.2 void waitStateWidget::setState (BaseState ∗ *state*) `[virtual]`

Function opening state for edition in the widget.

Implements MyTypeWidget.

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

- StateTypeWidgets.h