SIR

0.3

Generated by Doxygen 1.8.0

Thu May 24 2012 16:37:41

Contents

Chapter 1

Todo List

Member ControlSystem.IWrapper.moveLinearWrapped (string _sNameOfVector, int _iIndex)

Refactor.

Class ControlSystem.Wrapper

Add behind factory class.

Member ControlSystem.Wrapper.initializationWrapped (enumSystemModes _sysmodeMode, enumSystemTypes _systypeType, DLL.DgateCallBack _funcptrSuccess, DLL.DgateCallBack _funcptrError)

Refactor delegate to contain ConfigData and ErrorInfo if found necessary.

Member ControlSystem.Wrapper.stopWrapped (enumAxisSettings _bWhatToStop)

Refactor.

Class RoboGO.ViewModels.ViewModelManualSteering

Way to inform View about errors, like not being connected to robot.(Example messaging.)

Chapter 2

Namespace Index

2.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 3

Class Index

3.1 Class Hierarchy

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

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

Chapter 6

Namespace Documentation

6.1 Package ControlSystem

Class to handle system threads.

Classes

- class DLL

  Class that with the IDLL interface calls the actual functions from the USBC.dll.
- class DLLImport

  Class providing interface for USBC.dll functions.
- class ErrorReporter

  Temporary error reporting class.
- class **Factory**

  Manages all global class's as a singleton and factory.
- interface IDLL

  Interface for the functions in the USBC.dll files, in C# format.
- interface IWrapper

  Interface for a wrapper wrapping the USBC.dll file.
- interface IManualController

  Interface for what manual movement functions there should be available.
- class ManualController

  Class that encapsulates controlling manual movement. Instead of using directly Robot or Simulator by interface.
- interface IScriptRunner

  Interface for a script runner.
- class ScriptRunner

  Used to run IronPython scripts.
- class SerialSTK

  Class for functions to communicating with the STK kit.(Serial communication.)
- class Simulator

  IRobot implementation using IUI output interface for simulating robot behavior.
- class ThreadHandling

  Class to handle all threads in system, everything handled by a unique description tag that stays with a thread from moment it gets added till it gets removed.
- interface IUI

  Simple interface for UI interaction.
- class ConsoleUI

Console UI for writing to the console output.

- class StringUI

  String UI for writing to a string variable.
- interface IUser

  Interface for an user with permissions for the database.
- class User

  Normal user.
- class Admin

  Admin user with ability to see and edit all tables.
- class VecPoint

  Class to contain point for use in one of the vector classes.
- class SIRVector

  Base class for vector used in wrapper.
- class AbsCoordSirVector

  SIRVector class for absolute positions.
- class RelCoordSirVector

  SIRVector class for relative positions.
- class Wrapper

  Contains a wrapper for the C++ functions in the dll file(USBC.dll).

Enumerations

- enum enumLeftRight

  What direction to move in when moving by axes.
- enum enumUpDown

  What direction to move in when moving by axes.(Wrist)
- enum enumIncDec

  Move increasing or decreasing when moving by coordinates.
- enum enumCloseOpen

  To close or open gripper.

6.1.1 Detailed Description

Class to handle system threads.

Author

Robotic Global Organization(RoboGO)
Robotic Global Organization(RoboGO)

Date

18-03-2012

Note

Thread functions are to be defined as following - With start parameter: void functionname(object o) {} - Without start parameter: void functionname()

6.1.2 Enumeration Type Documentation

6.1.2.1 enum ControlSystem.enumCloseOpen

To close or open gripper.

6.1.2.2 enum ControlSystem.enumIncDec

Move increasing or decreasing when moving by coordinates.

6.1.2.3 enum ControlSystem.enumLeftRight

What direction to move in when moving by axes.

6.1.2.4 enum ControlSystem.enumUpDown

What direction to move in when moving by axes.(Wrist)

6.2 Package RoboGO

Packages

- package Properties
- package ViewModels

Classes

- class aboutBox

  aboutBox
- class App

  App.
- class CommandsWindow

  CommandsWindow.
- class GUIManualSteering

  GUIManualSteering.
- class MainWindow

  MainWindow.
- class PasswordWindow

  PasswordWindow.
- class Simulator

  Simulator.
- class **UIService**

  Service for UI related services.
- class MainWindowViewModel

  ViewModel for the mainwindow.

6.3 Package RoboGO.Properties

Classes

- class **Resources**

  A strongly-typed resource class, for looking up localized strings, etc.
- class **Settings**

6.4 Package RoboGO.ViewModels

Classes

- class DelegateCommand

  Command class for executing one function.
- class IDEViewModel

  ViewModel between IDEView and ScriptRunner.
- class InfoViewModel

  ViewModel for the tables.(From the database.)
- class passwordWindowViewModel

  ViewModel for password window.
- class PositionModel

  Class to keep track of simulator position.
- class PositionViewModel

  ViewModel for the position class.
- class SimulatorViewModel

  ViewModel for the simulator class.
- class XYCalculate

  Class for calculating position of robot.
- class ViewModelManualSteering

  ViewModel for GUIManualSteering.

6.5 Package SqlInteraction

Classes

- interface ISqlConnection
- interface ISQLHandler

  Interface for handling class of sql.
- interface ISQLReader

  Interface for class that read information from a SQL table.
- class RobotSqlConnection

  SQLConnection to connect to a database.
- class SQLHandler

  Class that handles all SQL interaction.
- class SQLReader

  Class to read table data.

6.6 Package XamlGeneratedNamespace

Classes

- class GeneratedInternalTypeHelper

  GeneratedInternalTypeHelper.

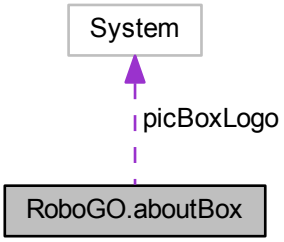
Chapter 7

Class Documentation

7.1 RoboGO.aboutBox Class Reference

aboutBox

Collaboration diagram for RoboGO.aboutBox:



Public Member Functions

- void InitializeComponent ()

  InitializeComponent.
- void InitializeComponent ()

  InitializeComponent.

7.1.1 Detailed Description

aboutBox

7.1.2 Member Function Documentation

7.1.2.1 void RoboGO.aboutBox.InitializeComponent ( )

InitializeComponent.

7.1.2.2 void RoboGO.aboutBox.InitializeComponent ( )

InitializeComponent.

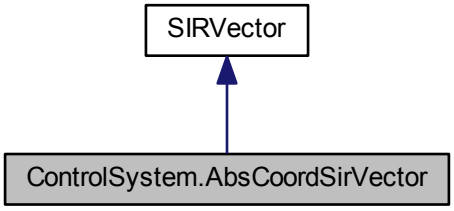
The documentation for this class was generated from the following files:

- RoboGO/obj/x86/Debug/aboutBox.g.cs
- RoboGO/obj/x86/Debug/aboutBox.g.i.cs

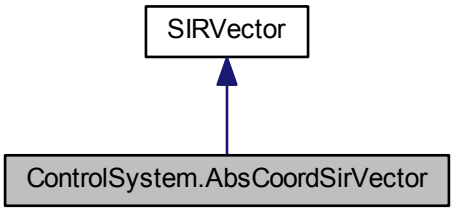
7.2 ControlSystem.AbsCoordSirVector Class Reference

SIRVector class for absolute positions.

Inheritance diagram for ControlSystem.AbsCoordSirVector:



Collaboration diagram for ControlSystem.AbsCoordSirVector:



Public Member Functions

- AbsCoordSirVector (string _sName)

  Contructors whichs sets up type and name of vector.

7.2.1 Detailed Description

SIRVector class for absolute positions.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 ControlSystem.AbsCoordSirVector.AbsCoordSirVector (string *_sName*)

Contructors whichs sets up type and name of vector.

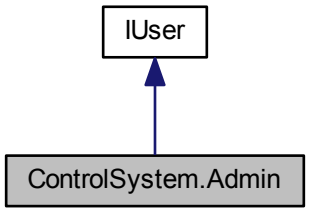
The documentation for this class was generated from the following file:

- ControlSystem/wrapper.cs

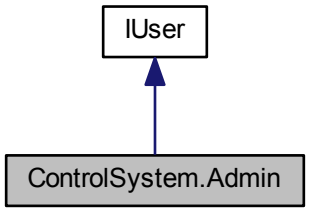
7.3 ControlSystem.Admin Class Reference

Admin user with ability to see and edit all tables.

Inheritance diagram for ControlSystem.Admin:



Collaboration diagram for ControlSystem.Admin:



Public Member Functions

- Admin ()

  Default constructor setting up permissions.

Properties

- Dictionary< string, bool > permissionDictionary `[get]`

Set of permissions.

- string userName [get, set]

  Name of the user.

7.3.1 Detailed Description

Admin user with ability to see and edit all tables.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 ControlSystem.Admin.Admin ( )

Default constructor setting up permissions.

7.3.3 Property Documentation

7.3.3.1 Dictionary<string, bool> ControlSystem.Admin.permissionDictionary [get]

Set of permissions.

Implements ControlSystem.IUser.

7.3.3.2 string ControlSystem.Admin.userName [get, set]

Name of the user.

Implements ControlSystem.IUser.

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

- ControlSystem/User.cs

7.4 RoboGO.App Class Reference

App.

Public Member Functions

- void InitializeComponent ()

  InitializeComponent.

- void InitializeComponent ()

  InitializeComponent.

Static Public Member Functions

- static void Main ()

  Application Entry Point.

- static void Main ()

  Application Entry Point.

7.4.1 Detailed Description

App.

7.4.2 Member Function Documentation

7.4.2.1 void RoboGO.App.InitializeComponent ( )

InitializeComponent.

7.4.2.2 void RoboGO.App.InitializeComponent ( )

InitializeComponent.

7.4.2.3 static void RoboGO.App.Main ( ) [static]

Application Entry Point.

7.4.2.4 static void RoboGO.App.Main ( ) [static]

Application Entry Point.

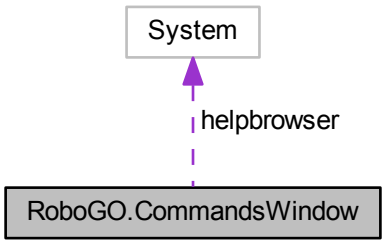
The documentation for this class was generated from the following files:

- RoboGO/obj/x86/Debug/App.g.cs
- RoboGO/obj/x86/Debug/App.g.i.cs

7.5 RoboGO.CommandsWindow Class Reference

CommandsWindow.

Collaboration diagram for RoboGO.CommandsWindow:



Public Member Functions

- void InitializeComponent ()

  InitializeComponent.

- void InitializeComponent ()

  InitializeComponent.

7.5.1 Detailed Description

CommandsWindow.

7.5.2 Member Function Documentation

7.5.2.1 void RoboGO.CommandsWindow.InitializeComponent ( )

InitializeComponent.

7.5.2.2 void RoboGO.CommandsWindow.InitializeComponent ( )

InitializeComponent.

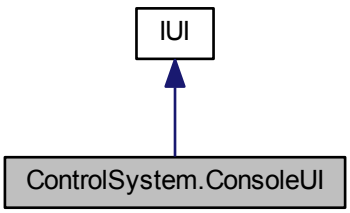
The documentation for this class was generated from the following files:

- RoboGO/obj/x86/Debug/CommandsWindow.g.cs
- RoboGO/obj/x86/Debug/CommandsWindow.g.i.cs

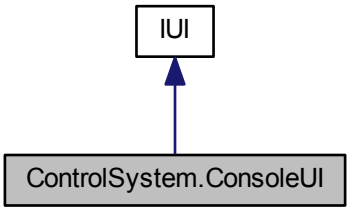
7.6 ControlSystem.ConsoleUI Class Reference

Console UI for writing to the console output.

Inheritance diagram for ControlSystem.ConsoleUI:



Collaboration diagram for ControlSystem.ConsoleUI:



Public Member Functions

- void write (string _sMsg, params object[] _paramobjArgument)

  Writes the string with arguments to the UI.
- void writeLine (string _sMsg, params object[] _paramobjArgument)

  Writes the string with arguments to the UI.

7.6.1 Detailed Description

Console UI for writing to the console output.

7.6.2 Member Function Documentation

7.6.2.1 void ControlSystem.ConsoleUI.write (string *_sMsg*, params object[] *_paramobjArgument*)

Writes the string with arguments to the UI.

No newline character written.

Parameters

sMsg	The string with the message and argument placement.(Like normal Write())
paramobj-Argument	Arguments to be placed in the string.

Implements ControlSystem.IUI.

7.6.2.2 void ControlSystem.ConsoleUI.writeLine (string *sMsg*, params object[] *paramobjArgument*)

Writes the string with arguments to the UI.

Newline character appended to end of string.

Parameters

sMsg	The string with the message and argument placement.(Like normal WriteLine())
paramobj-Argument	Arguments to be placed in the string.

Implements ControlSystem.IUI.

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

- ControlSystem/ui.cs

7.7 RoboGO.ViewModels.DelegateCommand Class Reference

Command class for executing one function.

Public Member Functions

- DelegateCommand (Action _aMethodToExecute)
  Constructor with function to call.
- bool CanExecute (object _objParam)
  Able to execute.
- void Execute (object _objParam)
  Execute the command from the constructor.

Events

- EventHandler CanExecuteChanged
  EventHandler for if able execute.

7.7.1 Detailed Description

Command class for executing one function.

Function type: void functionName(void)

7.7.2 Constructor & Destructor Documentation

7.7.2.1 RoboGO.ViewModels.DelegateCommand.DelegateCommand (Action *_aMethodToExecute*)

Constructor with function to call.

Parameters

_aMethodToExecute	Function to call when command is used.

7.7.3 Member Function Documentation

7.7.3.1 bool RoboGO.ViewModels.DelegateCommand.CanExecute (object *_objParam*)

Able to execute.

Parameters

_objParam	Unused from ICommand.

Returns

Always true.

7.7.3.2 void RoboGO.ViewModels.DelegateCommand.Execute (object *_objParam*)

Execute the command from the constructor.

Parameters

_objParam	Unused from ICommand.

7.7.4 Event Documentation

7.7.4.1 EventHandler RoboGO.ViewModels.DelegateCommand.CanExecuteChanged

EventHandler for if able execute.

Note: Not used.

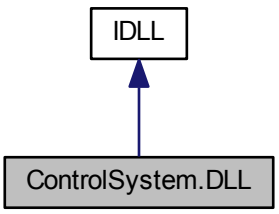
The documentation for this class was generated from the following file:

- RoboGO/ViewModels/delegateCommand.cs

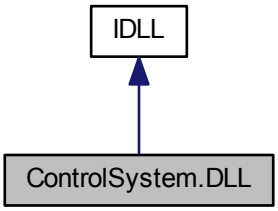
7.8 ControlSystem.DLL Class Reference

Class that with the IDLL interface calls the actual functions from the USBC.dll.

Inheritance diagram for ControlSystem.DLL:



Collaboration diagram for ControlSystem.DLL:



Public Member Functions

- delegate void **DgateCallBack** (IntPtr voidptrConfigData)
- delegate void **DgateCallBackCharArg** (Byte bArg)
- delegate void **DgateCallBackLongArg** (long lArg)
- delegate void DgateCallBackByteRefArg (ref byte bArg)
- int **Initialization** (short _shrtMode, short _shrtType, DgateCallBack _funcprtCallBack, DgateCallBack _funcptrCallBackError)
- int Control (byte _bAxis, bool _bIsOn)

  Turns control on and off for certain axis group.
- int **Home** (byte _axis, DgateCallBackByteRefArg _funcptrCallBack)
- int OpenGripper ()

  Opens the gripper.
- int CloseGripper ()

  Closes the gripper.
- int GetJaw (ref short _perc, ref short _metric)

  Gives information about how much open the gripper is.(Between the 'fingers')
- int EnterManual (short _shrtArg)

  Must be called to use manual movement. Seems to stop previous movement of any object(Axis) that was moving before.
- int CloseManual ()

  Stops manual mode.
- int MoveManual (byte _bAxis, int _lSpeed)

  Moves the robot. homeWrapped must have been called if moving by coordinates. enterManual seems have to be called before each call to this function. Use stopWrapped to stop motion afterwards.(Moving some other part of the system also stops the previous movement, since the system can only handle one object(Axis) moving at a time.)

  Parameters

__bAxis	*Which Axis to move (0-7)*
_lSpeed	*The move speed of Axis 0-100%*

  Returns

  Returns true on successful call.
- int Stop (byte _axis)

  Stops movement of axis.
- DgateCallBackCharArg **WatchMotion** (DgateCallBackCharArg _funcptrCallbackEnd, DgateCallBackCharArg _funcptrCallbackStart)
- int **WatchDigitalInput** (DgateCallBackLongArg _funcptrCallbackEvent)

- int CloseWatchDigitalInput ()

  Stops watching of digital inputs.
- int IsOnLineOk ()

  Tells about the robot being online.
- int MoveLinear ([MarshalAs(UnmanagedType.LPStr)] string _sNameOfVectorThatGotPosition, short _shrtPointInVector,[MarshalAs(UnmanagedType.LPStr)] string _sSecondaryPos, short _shrtPointToMoveTo)

  Move the Robot to a specific point.
- int DefineVector (byte _bGroup,[MarshalAs(UnmanagedType.LPStr)] string _sVectorName, short _shrtSizeOfVector)

  Defines a new vector in robot memory.
- int Teach ([MarshalAs(UnmanagedType.LPStr)] string _sVectorName, short _shrtPoint, int[] _iaPointInfo, short _shrtSizeOfArray, int _iPointType)

  Add the vector points to the vector with the same name.
- int GetCurrentPosition (ref int[] _ibufEnc, ref int[] _ibufJoint, ref int[] _ibufXYZ)

  Get the current position by reference.
- int Time (byte _bGroup, long _mTime)

  Set time for movements.
- int Speed (byte _bGroup, long mSpeed)

  Set speed for movements.

7.8.1 Detailed Description

Class that with the IDLL interface calls the actual functions from the USBC.dll.

Note: Uses static imports from DLLImport.

7.8.2 Member Function Documentation

7.8.2.1 int ControlSystem.DLL.CloseGripper ( )

Closes the gripper.

Returns

Returns 1 on successful call.

Implements ControlSystem.IDLL.

7.8.2.2 int ControlSystem.DLL.CloseManual ( )

Stops manual mode.

Returns

Returns 1 on successful call.

Implements ControlSystem.IDLL.

7.8.2.3 int ControlSystem.DLL.CloseWatchDigitalInput ( )

Stops watching of digital inputs.

Note: Probably means no more events.

Returns

Returns 1 if successful call.

Implements ControlSystem.IDLL.

7.8.2.4 int ControlSystem.DLL.Control (byte *_bAxis,* bool *_blsOn*)

Turns control on and off for certain axis group.

Parameters

bAxis	Axis group to affect.(Use enum)
_blsOn	To have it turned off or on.

Returns

Returns 1 on successful call.

Implements ControlSystem.IDLL.

7.8.2.5 int ControlSystem.DLL.DefineVector (byte *bGroup,* [MarshalAs(UnmanagedType.LPStr)] string *_sVectorName,* short *_shrtSizeOfVector*)

Defines a new vector in robot memory.

Note: Good idea to have in program one of the SIRVector classes to contains vector information.

Parameters

_enumGroup	Group can use: Robot(Normally used) Peripherals All
_sVectorName	Name of vector.
_shrtLength	Length of vector.(Number of points.)

Returns

Returns true on successfull call.

Implements ControlSystem.IDLL.

7.8.2.6 delegate void ControlSystem.DLL.DgateCallBackByteRefArg (ref byte *bArg*)

Warning

Using long.

7.8.2.7 int ControlSystem.DLL.EnterManual (short *_shrtArg*)

Must be called to use manual movement. Seems to stop previous movement of any object(Axis) that was moving before.

Parameters

_shrArg	What to move by.(Axis(0), Coordinates(1))

Returns

Returns 1 on successful call.

Implements ControlSystem.IDLL.

7.8.2.8 int ControlSystem.DLL.GetCurrentPosition (ref int[] *_ibufEnc,* ref int[] *_ibufJoint,* ref int[] *_ibufXYZ*)

Get the current position by reference.

Returns

Returns 1 if called

Implements ControlSystem.IDLL.

7.8.2.9 int ControlSystem.DLL.GetJaw (ref short *_perc,* ref short *_metric*)

Gives information about how much open the gripper is.(Between the 'fingers')

Note: Probably most useful to use the _shrtWidth arg.

Parameters

_perc	Data in percentage.
_metric	Data in width.(mm)

Returns

Returns 1 on successful call.

Implements ControlSystem.IDLL.

7.8.2.10 int ControlSystem.DLL.IsOnLineOk ( )

Tells about the robot being online.

Returns

Returns 1 if it is, 0 otherwise.

Implements ControlSystem.IDLL.

7.8.2.11 int ControlSystem.DLL.MoveLinear ([MarshalAs(UnmanagedType.LPStr)] string *_sNameOfVectorThatGotPosition,* short *_shrtPointInVector,* [MarshalAs(UnmanagedType.LPStr)] string *_sSecondaryPos,* short *_shrtPointToMoveTo*)

Move the Robot to a specific point.

Parameters

_sNameOfVectorThatGotPosition	Navnet på vektoren i robotten.
_shrtPointInVector	Index for punkt.

Returns

Returns true on successfull call.

Implements ControlSystem.IDLL.

7.8.2.12 int ControlSystem.DLL.MoveManual (byte *_bAxis*, int *_lSpeed*)

Moves the robot. homeWrapped must have been called if moving by coordinates. enterManual seems have to be called before each call to this function. Use stopWrapped to stop motion afterwards.(Moving some other part of the system also stops the previous movement, since the system can only handle one object(Axis) moving at a time.)

Parameters

__bAxis	Which Axis to move (0-7)
_lSpeed	The move speed of Axis 0-100%

Returns

Returns true on successful call.

Implements ControlSystem.IDLL.

7.8.2.13 int ControlSystem.DLL.OpenGripper ( )

Opens the gripper.

Returns

Returns 1 on successful call.

Implements ControlSystem.IDLL.

7.8.2.14 int ControlSystem.DLL.Speed (byte *_bGroup*, long *_mSpeed*)

Set speed for movements.

Parameters

_bGroup	Which joint to set time (& for all)
_mSpeed	speed from 0-100%

Returns

Returns 1 if called

Implements ControlSystem.IDLL.

7.8.2.15 int ControlSystem.DLL.Stop (byte *_axis*)

Stops movement of axis.

Parameters

_axis	Axis to stop.

Returns

Returns 1 on successful call.

Implements ControlSystem.IDLL.

7.8.2.16 int ControlSystem.DLL.Teach ([MarshalAs(UnmanagedType.LPStr)] string *_sVectorName,* short *_shrtPoint,* int[ ] *_iaPointInfo,* short *_shrtSizeOfArray,* int *_iPointType*)

Add the vector points to the vector with the same name.

Note: Should call 'defineVectorWrapped' first.

Parameters

_sVectorName	The vector with the points.

Returns

Returns true on succeessfull call.

Implements ControlSystem.IDLL.

7.8.2.17 int ControlSystem.DLL.Time (byte *_bGroup,* long *_mTime*)

Set time for movements.

Parameters

_bGroup	Which joint to set time (& for all)
_mTime	Time in milisecond

Returns

Returns 1 if called

Implements ControlSystem.IDLL.

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

- ControlSystem/dll.cs

7.9 ControlSystem.DLLImport Class Reference

Class providing interface for USBC.dll functions.

Public Member Functions

- static int initialization (short shrtMode, short shrtType, DLL.DgateCallBack funcptrCallBack, DLL.DgateCallBack funcptrCallBackError)

  Initialize the robot.
- static int Control (byte bAxis, bool bIsOn)

  Turn control on/off for axis group.
- static int Home (byte axis, DLL.DgateCallBackByteRefArg funcptrCallBack)

  Home the axis group or the whole robot.

- static int OpenGripper ()

  Opens the gripper.
- static int CloseGripper ()

  Closes the gripper.
- static int GetJaw (ref short perc, ref short metric)

  Get values for how open the gripper are.
- static int EnterManual (short shrtArg)

  Enter manual steering mode.
- static int CloseManual ()

  Stops manual steering mode.
- static int MoveManual (byte bAxis, int lSpeed)

  Move an axis group.
- static int Stop (byte axis)

  Stop movement of an axis group.
- static DLL.DgateCallBackCharArg WatchMotion (DLL.DgateCallBackCharArg funcptrCallbackEnd, DLL.-DgateCallBackCharArg funcptrCallbackStart)

  Adds callback for robot movement functions.
- static int WatchDigitalInput (DLL.DgateCallBackLongArg funcptrCallbackEvent)

  Adds callback for digital input signal.
- static int CloseWatchDigitalInput ()

  Stops watching for digital input.
- static int IsOnLineOk ()

  Cehcks for being online.
- static int MoveLinear ([MarshalAs(UnmanagedType.LPStr)] string sNameOfVectorThatGotFirstPosition, short shrtPointInVector,[MarshalAs(UnmanagedType.LPStr)] string sNameOfVectorThatGotSecondPosition, short shrtPointToMoveTo)

  Moves the robot towards a position and then another.
- static int DefineVector (byte bGroup,[MarshalAs(UnmanagedType.LPStr)] string sVectorName, short shrtSize-OfVector)

  Define a new vector in the robot.
- static int Teach ([MarshalAs(UnmanagedType.LPStr)] string sVectorName, short shrtPoint, int[] iaPointInfo, short shrtSizeOfArray, int iPointType)

  Saves a point in the vector.
- static int GetCurrentPosition (ref int[] ibufEnc, ref int[] ibufJoint, ref int[] ibufXYZ)

  Get the current position.
- static int Time (byte _bGroup, long _mTime)

  Sets the value for how long movement should take place.(Manual steering.)
- static int Speed (byte _bGroup, long _mSpeed)

  Sets the speed of movement.

7.9.1 Detailed Description

Class providing interface for USBC.dll functions.

Note: Please use the Wrapper instead.

7.9.2 Member Function Documentation

7.9.2.1 static int ControlSystem.DLLImport.CloseGripper ( )

Closes the gripper.

Returns

True on successfull call.

7.9.2.2 static int ControlSystem.DLLImport.CloseManual ( )

Stops manual steering mode.

Returns

True on successfull call.

7.9.2.3 static int ControlSystem.DLLImport.CloseWatchDigitalInput ( )

Stops watching for digital input.

Returns

True on successfull call.

7.9.2.4 static int ControlSystem.DLLImport.Control (byte *bAxis,* bool *bIsOn*)

Turn control on/off for axis group.

Parameters

bAxis	Axis to apply setting.
bIsOn	On or off.

Returns

True on successfull call.

7.9.2.5 static int ControlSystem.DLLImport.DefineVector (byte *bGroup,* [MarshalAs(UnmanagedType.LPStr)] string *sVectorName,* short *shrtSizeOfVector*)

Define a new vector in the robot.

Parameters

bGroup	Group for the vector type.('A' robot, '&' all axes, 'B' peripherals.)
sVectorName	Name of vector.
shrtSizeOfVector	Size of vector.(Number of points.)

Returns

True on successfull call.

7.9.2.6 static int ControlSystem.DLLImport.EnterManual (short *shrtArg*)

Enter manual steering mode.

Parameters

shrtArg	Type: Axis(0) or by coordinates(1).

Returns

True on successfull call.

7.9.2.7 static int ControlSystem.DLLImport.GetCurrentPosition (ref int[] *ibufEnc,* ref int[] *ibufJoint,* ref int[] *ibufXYZ*)

Get the current position.

Parameters

ibufEnc	Buffer to save values.
ibufJoint	Buffer to save values.
ibufXYZ	Buffer to save values.

Returns

True on successfull call.

7.9.2.8 static int ControlSystem.DLLImport.GetJaw (ref short *perc,* ref short *metric*)

Get values for how open the gripper are.

Parameters

perc	In percentage.
metric	In metric value.

Returns

True on successfull call.

7.9.2.9 static int ControlSystem.DLLImport.Home (byte *axis,* DLL.DgateCallBackByteRefArg *funcptrCallBack*)

Home the axis group or the whole robot.

Parameters

axis	Axis to home.
funcptrCallBack	Function to call when homing axis.

Returns

True on successfull call.

7.9.2.10 static int ControlSystem.DLLImport.initialization (short *shrtMode,* short *shrtType,* DLL.DgateCallBack *funcptrCallBack,* DLL.DgateCallBack *funcptrCallBackError*)

Initialize the robot.

Parameters

shrtMode	Mode of the robot.
shrtType	Type of connection.
funcptrCallBack	Function to call when initialized.
funcptrCallBack-Error	Function to call when errors happen.

Returns

True on successfull call.

7.9.2.11 static int ControlSystem.DLLImport.IsOnLineOk ( )

Cehcks for being online.

Returns

Online(1)/offline(0):

7.9.2.12 static int ControlSystem.DLLImport.MoveLinear ([MarshalAs(UnmanagedType.LPStr)] string *sNameOfVectorThatGotFirstPosition,* short *shrtPointInVector,* [MarshalAs(UnmanagedType.LPStr)] string *sNameOfVectorThatGotSecondPosition,* short *shrtPointToMoveTo*)

Moves the robot towards a position and then another.

Parameters

sNameOfVector-ThatGotPosition	Vector with first position.
shrtPointInVector	What point in the vector to move to.(Index.)
sSecondaryPos	Vector with second position
shrtPointTo-MoveTo	What point in the vector to move to.(Index.)

Returns

True on successfull call.

7.9.2.13 static int ControlSystem.DLLImport.MoveManual (byte *bAxis,* int *lSpeed*)

Move an axis group.

Note: Call Control and EnterManual first.

Parameters

bAxis	What axis to move.
lSpeed	Speed to move in. Negative value for opposite direction.(This value is a percentage of max speed.)

Returns

True on successfull call.

7.9.2.14 static int ControlSystem.DLLImport.OpenGripper ( )

Opens the gripper.

Returns

True on successfull call.

7.9.2.15 static int ControlSystem.DLLImport.Speed (byte *_bGroup,* long *_mSpeed*)

Sets the speed of movement.

Parameters

_bGroup	What axis group.
_mSpeed	Speed of movement.(Value in percentage.)

Returns

True on successfull call.

7.9.2.16 static int ControlSystem.DLLImport.Stop (byte *axis*)

Stop movement of an axis group.

Parameters

axis	What axis to stop.

Returns

True on successfull call.

7.9.2.17 static int ControlSystem.DLLImport.Teach ([MarshalAs(UnmanagedType.LPStr)] string *sVectorName,* short *shrtPoint,* int[] *iaPointInfo,* short *shrtSizeOfArray,* int *iPointType*)

Saves a point in the vector.

Parameters

sVectorName	Name of vector.
shrtPoint	What point.(Index.)
iaPointInfo	Point values.
shrtSizeOfArray	Size of point values.(How many values.)
iPointType	What kind of point.(Relative(-32767) or Absolute(-32766).)

Returns

True on successfull call.

7.9.2.18 static int ControlSystem.DLLImport.Time (byte *_bGroup,* long *_mTime*)

Sets the value for how long movement should take place.(Manual steering.)

Parameters

_bGroup	What axis group.
_mTime	Time in milliseconds.

Returns

True on successfull call.

7.9.2.19 static int ControlSystem.DLLImport.WatchDigitalInput (DLL.DgateCallBackLongArg *funcptrCallbackEvent*)

Adds callback for digital input signal.

Parameters

funcptrCallback-Event	Function to call when signal.

Returns

True on successfull call.

7.9.2.20 static DLL.DgateCallBackCharArg ControlSystem.DLLImport.WatchMotion (DLL.DgateCallBackCharArg *funcptrCallbackEnd,* DLL.DgateCallBackCharArg *funcptrCallbackStart*)

Adds callback for robot movement functions.

Parameters

funcptrCallback-End	Function to call when movement has ended.
funcptrCallback-Start	Function to call when movement starts.

Returns

True on successfull call.

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

- ControlSystem/dll.cs

7.10 ControlSystem.ErrorReporter Class Reference

Temporary error reporting class.

Public Member Functions

- override void ErrorReported (ScriptSource _scrpsrcScriptSource, string _sErrorMsg, SourceSpan _srcspan-Spanning, int _iCode, Severity sevSeverity)

  Adds error to the error list.

Static Public Attributes

- static List< String > **Errorlist** = new List<string>()

7.10.1 Detailed Description

Temporary error reporting class.

Warning

Temp. class.

7.10.2 Member Function Documentation

7.10.2.1 override void ControlSystem.ErrorReporter.ErrorReported (ScriptSource *_scrpsrcScriptSource,* string *_sErrorMsg,* SourceSpan *_srcspanSpanning,* int *_iCode,* Severity *sevSeverity*)

Adds error to the error list.

Parameters

_scrpsrcScript-Source	Script.
_sErrorMsg	Error message.
_srcspan-Spanning	Span.
_iCode	Code.
sevSeverity	Severity.

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

- ControlSystem/errorReporter.cs

7.11 XamlGeneratedNamespace.GeneratedInternalTypeHelper Class Reference

GeneratedInternalTypeHelper.

Protected Member Functions

- override object CreateInstance (System.Type type, System.Globalization.CultureInfo culture)

  CreateInstance.
- override object GetPropertyValue (System.Reflection.PropertyInfo propertyInfo, object target, System.-Globalization.CultureInfo culture)

  GetPropertyValue.
- override void SetPropertyValue (System.Reflection.PropertyInfo propertyInfo, object target, object value, System.Globalization.CultureInfo culture)

  SetPropertyValue.
- override System.Delegate CreateDelegate (System.Type delegateType, object target, string handler)

  CreateDelegate.
- override void AddEventHandler (System.Reflection.EventInfo eventInfo, object target, System.Delegate handler)

  AddEventHandler.
- override object CreateInstance (System.Type type, System.Globalization.CultureInfo culture)

CreateInstance.

- override object GetPropertyValue (System.Reflection.PropertyInfo propertyInfo, object target, System.Globalization.CultureInfo culture)

  GetPropertyValue.

- override void SetPropertyValue (System.Reflection.PropertyInfo propertyInfo, object target, object value, System.Globalization.CultureInfo culture)

  SetPropertyValue.

- override System.Delegate CreateDelegate (System.Type delegateType, object target, string handler)

  CreateDelegate.

- override void AddEventHandler (System.Reflection.EventInfo eventInfo, object target, System.Delegate handler)

  AddEventHandler.

7.11.1 Detailed Description

GeneratedInternalTypeHelper.

7.11.2 Member Function Documentation

7.11.2.1 override void XamlGeneratedNamespace.GeneratedInternalTypeHelper.AddEventHandler (System.Reflection.EventInfo *eventInfo,* object *target,* System.Delegate *handler*) [protected]

AddEventHandler.

7.11.2.2 override void XamlGeneratedNamespace.GeneratedInternalTypeHelper.AddEventHandler (System.Reflection.EventInfo *eventInfo,* object *target,* System.Delegate *handler*) [protected]

AddEventHandler.

7.11.2.3 override System.Delegate XamlGeneratedNamespace.GeneratedInternalTypeHelper.CreateDelegate (System.Type *delegateType,* object *target,* string *handler*) [protected]

CreateDelegate.

7.11.2.4 override System.Delegate XamlGeneratedNamespace.GeneratedInternalTypeHelper.CreateDelegate (System.Type *delegateType,* object *target,* string *handler*) [protected]

CreateDelegate.

7.11.2.5 override object XamlGeneratedNamespace.GeneratedInternalTypeHelper.CreateInstance (System.Type *type,* System.Globalization.CultureInfo *culture*) [protected]

CreateInstance.

7.11.2.6 override object XamlGeneratedNamespace.GeneratedInternalTypeHelper.CreateInstance (System.Type *type,* System.Globalization.CultureInfo *culture*) [protected]

CreateInstance.

7.11.2.7 override object XamlGeneratedNamespace.GeneratedInternalTypeHelper.GetPropertyValue (System.Reflection.PropertyInfo *propertyInfo,* object *target,* System.Globalization.CultureInfo *culture*) [protected]

GetPropertyValue.

7.11.2.8 override object XamlGeneratedNamespace.GeneratedInternalTypeHelper.GetPropertyValue (System.Reflection.PropertyInfo *propertyInfo,* object *target,* System.Globalization.CultureInfo *culture*) [protected]

GetPropertyValue.

7.11.2.9 override void XamlGeneratedNamespace.GeneratedInternalTypeHelper.SetPropertyValue (System.Reflection.PropertyInfo *propertyInfo,* object *target,* object *value,* System.Globalization.CultureInfo *culture*) [protected]

SetPropertyValue.

7.11.2.10 override void XamlGeneratedNamespace.GeneratedInternalTypeHelper.SetPropertyValue (System.Reflection.PropertyInfo *propertyInfo,* object *target,* object *value,* System.Globalization.CultureInfo *culture*) [protected]

SetPropertyValue.

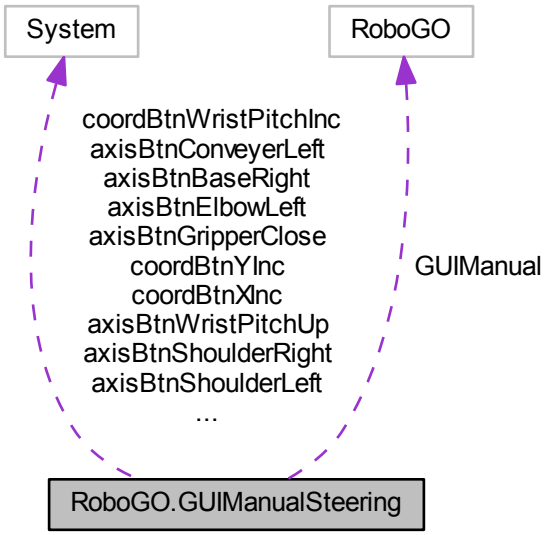
The documentation for this class was generated from the following files:

- RoboGO/obj/x86/Debug/GeneratedInternalTypeHelper.g.cs
- RoboGO/obj/x86/Debug/GeneratedInternalTypeHelper.g.i.cs

7.12 RoboGO.GUIManualSteering Class Reference

GUIManualSteering.

Collaboration diagram for RoboGO.GUIManualSteering:



Public Member Functions

- void InitializeComponent ()

  InitializeComponent.
- void InitializeComponent ()

  InitializeComponent.

7.12.1 Detailed Description

GUIManualSteering.

7.12.2 Member Function Documentation

7.12.2.1 void RoboGO.GUIManualSteering.InitializeComponent ( )

InitializeComponent.

7.12.2.2 void RoboGO.GUIManualSteering.InitializeComponent ( )

InitializeComponent.

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

- RoboGO/obj/x86/Debug/guiManualSteering.g.cs
- RoboGO/obj/x86/Debug/guiManualSteering.g.i.cs

7.13 RoboGO.ViewModels.IDEViewModel Class Reference

ViewModel between IDEView and ScriptRunner.

Public Member Functions

- IDEViewModel (TabControl _ideTabs)

  Constructor which uses TabControl.
- void executeCode ()

  Execute the code.
- void CodeClear ()

  Clear the code output shown.

Properties

- TabControl IdeTabs `[get, set]`

  TabControl for all the textboxes.
- string CodeOutput `[get, set]`

  Where print statements gets printed.
- IScriptRunner ScriptExecuter `[get, set]`

  ScriptRunner using to execute code.
- DelegateCommand ExecuteComd `[get]`

  Executes the code.
- RelayCommand saveAs `[get, set]`

  Save file from current tab.
- RelayCommand open `[get, set]`

  Open file.
- RelayCommand closeTab `[get, set]`

  Close current tab.
- RelayCommand newTab `[get, set]`

  Make a new tab.
- RelayCommand build `[get, set]`

  Build the current code.
- bool saveAs_CanExecute `[get]`

  Tells if able to save code.
- bool open_CanExecute `[get]`

  Tells if able to open file.
- bool closeTab_CanExecute `[get]`

  Tells if able to close tab.
- bool newTab_CanExecute `[get]`

  Tells if able to open a new tab.

Events

- PropertyChangedEventHandler PropertyChanged

  Called when dependency properties changed.(Used in view.)

7.13.1 Detailed Description

ViewModel between IDEView and ScriptRunner.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 RoboGO.ViewModels.IDEViewModel.IDEViewModel (TabControl *_ideTabs*)

Constructor which uses TabControl.

TabControl so can add and remove tab content.

Parameters

_ideTabs	TabControl used in main program in the IDE.

7.13.3 Member Function Documentation

7.13.3.1 void RoboGO.ViewModels.IDEViewModel.CodeClear ( )

Clear the code output shown.

7.13.3.2 void RoboGO.ViewModels.IDEViewModel.executeCode ( )

Execute the code.

7.13.4 Property Documentation

7.13.4.1 RelayCommand RoboGO.ViewModels.IDEViewModel.build [get, set]

Build the current code.

7.13.4.2 RelayCommand RoboGO.ViewModels.IDEViewModel.closeTab [get, set]

Close current tab.

7.13.4.3 bool RoboGO.ViewModels.IDEViewModel.closeTab_CanExecute [get, protected]

Tells if able to close tab.

Returns

True if 1 or more tabs.

7.13.4.4 string RoboGO.ViewModels.IDEViewModel.CodeOutput [get, set]

Where print statements gets printed.

7.13.4.5 DelegateCommand RoboGO.ViewModels.IDEViewModel.ExecuteComd [get]

Executes the code.

7.13.4.6 TabControl RoboGO.ViewModels.IDEViewModel.IdeTabs [get, set]

TabControl for all the textboxes.

7.13.4.7 RelayCommand RoboGO.ViewModels.IDEViewModel.newTab [get, set]

Make a new tab.

7.13.4.8 bool RoboGO.ViewModels.IDEViewModel.newTab_CanExecute [get, protected]

Tells if able to open a new tab.

Returns

Always true.

7.13.4.9 RelayCommand RoboGO.ViewModels.IDEViewModel.open [get, set]

Open file.

7.13.4.10 bool RoboGO.ViewModels.IDEViewModel.open_CanExecute [get, protected]

Tells if able to open file.

Returns

Always return true.

7.13.4.11 RelayCommand RoboGO.ViewModels.IDEViewModel.saveAs [get, set]

Save file from current tab.

7.13.4.12 bool RoboGO.ViewModels.IDEViewModel.saveAs_CanExecute [get, protected]

Tells if able to save code.

Returns

True if tab selected./returns>

7.13.4.13 IScriptRunner RoboGO.ViewModels.IDEViewModel.ScriptExecuter [get, set]

ScriptRunner using to execute code.

7.13.5 Event Documentation

7.13.5.1 PropertyChangedEventHandler RoboGO.ViewModels.IDEViewModel.PropertyChanged

Called when dependency properties changed.(Used in view.)

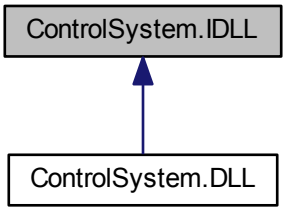
The documentation for this class was generated from the following file:

- RoboGO/ViewModels/ideViewModel.cs

7.14 ControlSystem.IDLL Interface Reference

Interface for the functions in the USBC.dll files, in C# format.

Inheritance diagram for ControlSystem.IDLL:



Public Member Functions

- int Initialization (short _shrtMode, short _shrtType, DLL.DgateCallBack _funcprtCallBack, DLL.DgateCallBack _funcptrCallBackError)
- int Control (byte _bAxis, bool _bIsOn)

  Turns control on and off for certain axis group.
- int Home (byte _axis, DLL.DgateCallBackByteRefArg _funcptrCallBack)

  Homes a axis group. Should be called before calling most movement functions.
- int OpenGripper ()

  Opens the gripper.
- int CloseGripper ()

  Closes the gripper.
- int GetJaw (ref short _perc, ref short _metric)

  Gives information about how much open the gripper is.(Between the 'fingers')
- int EnterManual (short _shrtArg)

  Must be called to use manual movement. Seems to stop previous movement of any object(Axis) that was moving before.
- int CloseManual ()

  Stops manual mode.
- int MoveManual (byte _bAxis, int _lSpeed)

  Moves the robot. homeWrapped must have been called if moving by coordinates. enterManual seems have to be called before each call to this function. Use stopWrapped to stop motion afterwards.(Moving some other part of the system also stops the previous movement, since the system can only handle one object(Axis) moving at a time.)

  Parameters

__bAxis	*Which Axis to move (0-7)*
_lSpeed	*The move speed of Axis 0-100%*

  Returns

  Returns true on successful call.
- int Stop (byte _axis)

  Stops movement of axis.
- DLL.DgateCallBackCharArg WatchMotion (DLL.DgateCallBackCharArg _funcptrCallbackEnd, DLL.DgateCallBackCharArg _funcptrCallbackStart)

  Adds functions to be called when motion starts and motion ends.

- int WatchDigitalInput (DLL.DgateCallBackLongArg _funcptrCallbackEvent)

  Adds a function to be called when digital input changes.
- int CloseWatchDigitalInput ()

  Stops watching of digital inputs.
- int IsOnLineOk ()

  Tells about the robot being online.
- int MoveLinear ([MarshalAs(UnmanagedType.LPStr)] string _sNameOfVectorThatGotPosition, short _shrtPointInVector,[MarshalAs(UnmanagedType.LPStr)] string _sSecondaryPos, short _shrtPointToMoveTo)

  Move the Robot to a specific point.
- int DefineVector (byte bGroup,[MarshalAs(UnmanagedType.LPStr)] string _sVectorName, short _shrtSizeOfVector)

  Defines a new vector in robot memory.
- int Teach ([MarshalAs(UnmanagedType.LPStr)] string _sVectorName, short _shrtPoint, int[] _iaPointInfo, short _shrtSizeOfArray, int _iPointType)

  Add the vector points to the vector with the same name.
- int GetCurrentPosition (ref int[] _ibufEnc, ref int[] _ibufJoint, ref int[] _ibufXYZ)

  Get the current position by reference.
- int Time (byte _bGroup, long _mTime)

  Set time for movements.
- int Speed (byte _bGroup, long _mSpeed)

  Set speed for movements.

7.14.1 Detailed Description

Interface for the functions in the USBC.dll files, in C# format.

7.14.2 Member Function Documentation

7.14.2.1 int ControlSystem.IDLL.CloseGripper ( )

Closes the gripper.

Returns

Returns 1 on successful call.

Implemented in ControlSystem.DLL.

7.14.2.2 int ControlSystem.IDLL.CloseManual ( )

Stops manual mode.

Returns

Returns 1 on successful call.

Implemented in ControlSystem.DLL.

7.14.2.3 int ControlSystem.IDLL.CloseWatchDigitalInput ( )

Stops watching of digital inputs.

Note: Probably means no more events.

Returns

Returns 1 if successful call.

Implemented in ControlSystem.DLL.

7.14.2.4 int ControlSystem.IDLL.Control (byte *_bAxis,* bool *_bIsOn*)

Turns control on and off for certain axis group.

Parameters

bAxis	Axis group to affect.(Use enum)
_bIsOn	To have it turned off or on.

Returns

Returns 1 on successful call.

Implemented in ControlSystem.DLL.

7.14.2.5 int ControlSystem.IDLL.DefineVector (byte *bGroup,* [MarshalAs(UnmanagedType.LPStr)] string *_sVectorName,* short *_shrtSizeOfVector*)

Defines a new vector in robot memory.

Note: Good idea to have in program one of the SIRVector classes to contains vector information.

Parameters

_enumGroup	Group can use: Robot(Normally used) Peripherals All
_sVectorName	Name of vector.
_shrtLength	Length of vector.(Number of points.)

Returns

Returns true on successfull call.

Implemented in ControlSystem.DLL.

7.14.2.6 int ControlSystem.IDLL.EnterManual (short *_shrtArg*)

Must be called to use manual movement. Seems to stop previous movement of any object(Axis) that was moving before.

Parameters

_shrArg	What to move by.(Axis(0), Coordinates(1))

Returns

Returns 1 on successful call.

Implemented in ControlSystem.DLL.

7.14.2.7 int ControlSystem.IDLL.GetCurrentPosition (ref int[] *_ibufEnc,* ref int[] *_ibufJoint,* ref int[] *_ibufXYZ*)

Get the current position by reference.

Returns

Returns 1 if called

Implemented in ControlSystem.DLL.

7.14.2.8 int ControlSystem.IDLL.GetJaw (ref short *_perc,* ref short *_metric*)

Gives information about how much open the gripper is.(Between the 'fingers')

Note: Probably most useful to use the _shrtWidth arg.

Parameters

_perc	Data in percentage.
_metric	Data in width.(mm)

Returns

Returns 1 on successful call.

Implemented in ControlSystem.DLL.

7.14.2.9 int ControlSystem.IDLL.Home (byte *_axis,* DLL.DgateCallBackByteRefArg *_funcptrCallBack*)

Homes a axis group. Should be called before calling most movement functions.

Parameters

_axis	The axis group.(Use enum)
_funcptrCallBack	Function to be called for homing events.

Values being passed in event: 0xff: Homing started 1 - 8: Axis n being homed. 0x40: Homing ended.

Returns

Returns 1 on successful call.

7.14.2.10 int ControlSystem.IDLL.Initialization (short *_shrtMode,* short *_shrtType,* DLL.DgateCallBack *_funcprtCallBack,* DLL.DgateCallBack *_funcptrCallBackError*)

Initializes the robot.

Note: Should wait for it to be done before calling other functions.

Parameters

_shrtMode	Mode.(Use one of constants[Normally use online mode])
_shrtType	Type of connection.(Use one of constants[Normally use default])
_funcprtCallBack	Function to be called on success.
_funcptrCall-BackError	Function to be called on error.

Returns

Returns 1 on successful call.(But errors can still happen)

7.14.2.11 int ControlSystem.IDLL.IsOnLineOk ( )

Tells about the robot being online.

Returns

Returns 1 if it is, 0 otherwise.

Implemented in ControlSystem.DLL.

7.14.2.12 int ControlSystem.IDLL.MoveLinear ([MarshalAs(UnmanagedType.LPStr)] string *_sNameOfVectorThatGotPosition,* short *_shrtPointInVector,* [MarshalAs(UnmanagedType.LPStr)] string *_sSecondaryPos,* short *_shrtPointToMoveTo*)

Move the Robot to a specific point.

Parameters

_sNameOf-VectorThatGot-Position	Navnet på vektoren i robotten.
_shrtPointIn-Vector	Index for punkt.

Returns

Returns true on successfull call.

Implemented in ControlSystem.DLL.

7.14.2.13 int ControlSystem.IDLL.MoveManual (byte *_bAxis,* int *_lSpeed*)

Moves the robot. homeWrapped must have been called if moving by coordinates. enterManual seems have to be called before each call to this function. Use stopWrapped to stop motion afterwards.(Moving some other part of the system also stops the previous movement, since the system can only handle one object(Axis) moving at a time.)

Parameters

__bAxis	Which Axis to move (0-7)
_lSpeed	The move speed of Axis 0-100%

Returns

Returns true on successful call.

Implemented in ControlSystem.DLL.

7.14.2.14 int ControlSystem.IDLL.OpenGripper ( )

Opens the gripper.

Returns

Returns 1 on successful call.

Implemented in ControlSystem.DLL.

7.14.2.15 int ControlSystem.IDLL.Speed (byte *_bGroup*, long *_mSpeed*)

Set speed for movements.

Parameters

_bGroup	Which joint to set time (& for all)
_mSpeed	speed from 0-100%

Returns

Returns 1 if called

Implemented in ControlSystem.DLL.

7.14.2.16 int ControlSystem.IDLL.Stop (byte *_axis*)

Stops movement of axis.

Parameters

_axis	Axis to stop.

Returns

Returns 1 on successful call.

Implemented in ControlSystem.DLL.

7.14.2.17 int ControlSystem.IDLL.Teach ([MarshalAs(UnmanagedType.LPStr)] string *_sVectorName*, short *_shrtPoint*, int[] *_iaPointInfo*, short *_shrtSizeOfArray*, int *_iPointType*)

Add the vector points to the vector with the same name.

Note: Should call 'defineVectorWrapped' first.

Parameters

_sVectorName	The vector with the points.

Returns

Returns true on succeessfull call.

Implemented in ControlSystem.DLL.

7.14.2.18 int ControlSystem.IDLL.Time (byte *_bGroup*, long *_mTime*)

Set time for movements.

Parameters

_bGroup	Which joint to set time (& for all)
_mTime	Time in milisecond

Returns

Returns 1 if called

Implemented in ControlSystem.DLL.

7.14.2.19 int ControlSystem.IDLL.WatchDigitalInput (DLL.DgateCallBackLongArg *_funcptrCallbackEvent*)

Adds a function to be called when digital input changes.

Parameters

_funcptr-CallbackEvent	The function to be called.

Returns

Returns 1 if successful call.

7.14.2.20 DLL.DgateCallBackCharArg ControlSystem.IDLL.WatchMotion (DLL.DgateCallBackCharArg *_funcptrCallbackEnd*, DLL.DgateCallBackCharArg *_funcptrCallbackStart*)

Adds functions to be called when motion starts and motion ends.

Note: Ignoring return value.

Parameters

_funcptr-CallbackEnd	Function to be called when motion has ended.
_funcptr-CallbackStart	Function to be called when motion has started.

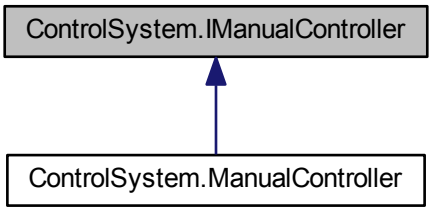
The documentation for this interface was generated from the following file:

- ControlSystem/iDLL.cs

7.15 ControlSystem.IManualController Interface Reference

Interface for what manual movement functions there should be available.

Inheritance diagram for ControlSystem.IManualController:



Public Member Functions

- void moveAxisBase (enumLeftRight _elrDirection)

  Move the base in the desired direction in 'speed' percentage of maximum speed.
- void moveAxisShoulder (enumLeftRight _elrDirection)

  Moves the shoulder in the desired direction.
- void moveAxisElbow (enumLeftRight _elrDirection)

  Moves the elbow in the desired direction.
- void moveAxisGripper (enumCloseOpen _ecoGripper)

  Opens or closes the gripper.
- void moveAxisPitch (enumUpDown _eudDirection)

  Moves the wrists pitch in the desired direction.
- void moveAxisRoll (enumLeftRight _elrDirection)

  Rolls the wrist in the desired direction.
- void moveAxisConveyer (enumLeftRight _elrDirection)

  Move the conveyer belt in the desired direction.
- void moveCoordX (enumIncDec _eidIncOrDec)

  Change the robots X-coordinate.
- void moveCoordY (enumIncDec _eidIncOrDec)

  Change the robots Y-coordinate.
- void moveCoordZ (enumIncDec _eidIncOrDec)

  Change the robots Z-coordinate.
- void moveCoordPitch (enumIncDec _eidIncOrDec)

  Change the wrists pitch.
- void moveCoordRoll (enumIncDec _eidIncOrDec)

  Change the roll of the wrist.
- void stopAllMovement ()

  Stops movement of all axes.

Properties

- int Speed `[get, set]`

  Speed in percentage.
- IRobot RobotConnection `[get, set]`

  What to steer.

7.15.1 Detailed Description

Interface for what manual movement functions there should be available.

7.15.2 Member Function Documentation

7.15.2.1 void ControlSystem.IManualController.moveAxisBase (enumLeftRight *_elrDirection*)

Move the base in the desired direction in 'speed' percentage of maximum speed.

Parameters

_elrDirection	Where ya wanna go?

Implemented in ControlSystem.ManualController.

7.15.2.2 void ControlSystem.IManualController.moveAxisConveyer (enumLeftRight *_elrDirection*)

Move the conveyer belt in the desired direction.

Parameters

_elrDirection	What direction to move in.

Implemented in ControlSystem.ManualController.

7.15.2.3 void ControlSystem.IManualController.moveAxisElbow (enumLeftRight *_elrDirection*)

Moves the elbow in the desired direction.

Parameters

_elrDirection	What direction to move in.

Implemented in ControlSystem.ManualController.

7.15.2.4 void ControlSystem.IManualController.moveAxisGripper (enumCloseOpen *_ecoGripper*)

Opens or closes the gripper.

Parameters

_ecoGripper	To open or close.

Implemented in ControlSystem.ManualController.

7.15.2.5 void ControlSystem.IManualController.moveAxisPitch (enumUpDown *_eudDirection*)

Moves the wrists pitch in the desired direction.

Parameters

_eudDirection	What direction to move in.

Implemented in ControlSystem.ManualController.

7.15.2.6 void ControlSystem.IManualController.moveAxisRoll (enumLeftRight *_elrDirection*)

Rolls the wrist in the desired direction.

Parameters

_elrDirection	What direction to move in.

Implemented in ControlSystem.ManualController.

7.15.2.7 void ControlSystem.IManualController.moveAxisShoulder (enumLeftRight *_elrDirection*)

Moves the shoulder in the desired direction.

Parameters

_elrDirection	What direction to move in.

Implemented in ControlSystem.ManualController.

7.15.2.8 void ControlSystem.IManualController.moveCoordPitch (enumIncDec *_eidIncOrDec*)

Change the wrists pitch.

Parameters

_eidIncOrDec	Increasing or decreasing.

Implemented in ControlSystem.ManualController.

7.15.2.9 void ControlSystem.IManualController.moveCoordRoll (enumIncDec *_eidIncOrDec*)

Change the roll of the wrist.

Parameters

_eidIncOrDec	Increasing or decreasing.

Implemented in ControlSystem.ManualController.

7.15.2.10 void ControlSystem.IManualController.moveCoordX (enumIncDec *_eidIncOrDec*)

Change the robots X-coordinate.

Parameters

_eidIncOrDec	Increasing or decreasing.

Implemented in ControlSystem.ManualController.

7.15.2.11 void ControlSystem.IManualController.moveCoordY (enumIncDec *_eidIncOrDec*)

Change the robots Y-coordinate.

Parameters

_eidIncOrDec	Increasing or decreasing.

Implemented in ControlSystem.ManualController.

7.15.2.12 void ControlSystem.IManualController.moveCoordZ (enumIncDec *_eidIncOrDec*)

Change the robots Z-coordinate.

Parameters

_eidIncOrDec	Increasing or decreasing.

Implemented in ControlSystem.ManualController.

7.15.2.13 void ControlSystem.IManualController.stopAllMovement ( )

Stops movement of all axes.

Implemented in ControlSystem.ManualController.

7.15.3 Property Documentation

7.15.3.1 IRobot ControlSystem.IManualController.RobotConnection [get, set]

What to steer.

Could be for example Robot(ER4) or Simulator.

Implemented in ControlSystem.ManualController.

7.15.3.2 int ControlSystem.IManualController.Speed [get, set]

Speed in percentage.

So should be between 0 and 100.

Implemented in ControlSystem.ManualController.

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

- ControlSystem/manualController.cs

7.16 RoboGO.ViewModels.InfoViewModel Class Reference

ViewModel for the tables.(From the database.)

Public Member Functions

- InfoViewModel (DataGrid _databaseValues)

  Constructor taking a DataGrid for showing values.(Also setting permissions[Read/Edit].)
- void loadAllTables ()

  Loads information about all tables in database.(Loads in other thread.)
- void getTableInfo (string _objTableName)

Gets information from table.(Saved in TableValues.)

- void tableSave ()

  Saves table edits.

- void tablePrint ()

  'Prints' the currently selected table to a CSV comma seperated file.

Properties

- DataTable Tables `[get, set]`

  Table list.

- DataTable TableValues `[get, set]`

  Table information.

Events

- PropertyChangedEventHandler PropertyChanged

  Called when dependency properties changed.(Used in view.)

7.16.1 Detailed Description

ViewModel for the tables.(From the database.)

7.16.2 Constructor & Destructor Documentation

7.16.2.1 RoboGO.ViewModels.InfoViewModel.InfoViewModel (DataGrid *_databaseValues*)

Constructor taking a DataGrid for showing values.(Also setting permissions[Read/Edit].)

Parameters

_database-Values	DataGrid from GUI.

7.16.3 Member Function Documentation

7.16.3.1 void RoboGO.ViewModels.InfoViewModel.getTableInfo (string *_objTableName*)

Gets information from table.(Saved in TableValues.)

Parameters

_objTableName	Name of table.

7.16.3.2 void RoboGO.ViewModels.InfoViewModel.loadAllTables ( )

Loads information about all tables in database.(Loads in other thread.)

7.16.3.3 void RoboGO.ViewModels.InfoViewModel.tablePrint ( )

'Prints' the currently selected table to a CSV comma seperated file.

Warning

Using UI dialog.

7.16.3.4 void RoboGO.ViewModels.InfoViewModel.tableSave ( )

Saves table edits.

7.16.4 Property Documentation

7.16.4.1 DataTable RoboGO.ViewModels.InfoViewModel.Tables [get, set]

Table list.

7.16.4.2 DataTable RoboGO.ViewModels.InfoViewModel.TableValues [get, set]

Table information.

7.16.5 Event Documentation

7.16.5.1 PropertyChangedEventHandler RoboGO.ViewModels.InfoViewModel.PropertyChanged

Called when dependency properties changed.(Used in view.)

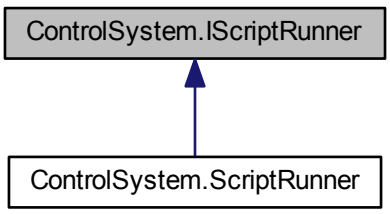
The documentation for this class was generated from the following file:

- RoboGO/ViewModels/infoViewModel.cs

7.17 ControlSystem.IScriptRunner Interface Reference

Interface for a script runner.

Inheritance diagram for ControlSystem.IScriptRunner:



Public Member Functions

- void setRobotInstance (IRobot _iroboRobot)

  Sets the underlying IRobot that the scripts are used on.

- void setScriptFromFile (string _sPath)

  Loads script from a file.
- void setScriptFromString (string _sScript)

  Loads script from a string.
- string readFromOutputStream ()

  Returns all input from the program.
- void clearOutputStream ()

  Clear the output.
- void ExecuteScript ()

  Executes loaded script.

7.17.1 Detailed Description

Interface for a script runner.

Used for Unit testing primarily.

7.17.2 Member Function Documentation

7.17.2.1 void ControlSystem.IScriptRunner.clearOutputStream ( )

Clear the output.

Implemented in ControlSystem.ScriptRunner.

7.17.2.2 void ControlSystem.IScriptRunner.ExecuteScript ( )

Executes loaded script.

Implemented in ControlSystem.ScriptRunner.

7.17.2.3 string ControlSystem.IScriptRunner.readFromOutputStream ( )

Returns all input from the program.

Returns

Output from program.

Implemented in ControlSystem.ScriptRunner.

7.17.2.4 void ControlSystem.IScriptRunner.setRobotInstance (IRobot *_iroboRobot*)

Sets the underlying IRobot that the scripts are used on.

Parameters

_iroboRobot	Robot to run script on.

Implemented in ControlSystem.ScriptRunner.

7.17.2.5 void ControlSystem.IScriptRunner.setScriptFromFile (string *_sPath*)

Loads script from a file.

Implemented in ControlSystem.ScriptRunner.

7.17.2.6 void ControlSystem.IScriptRunner.setScriptFromString (string *_sScript*)

Loads script from a string.

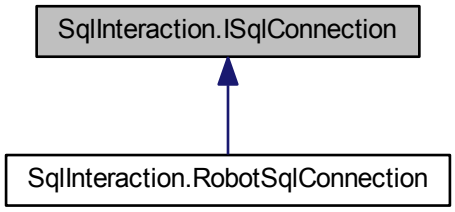
Implemented in ControlSystem.ScriptRunner.

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

- ControlSystem/scriptRunner.cs

7.18 SqlInteraction.ISqlConnection Interface Reference

Inheritance diagram for SqlInteraction.ISqlConnection:



Public Member Functions

- void ConnectionOpen ()

  Open the connection.
- void ConnectionClose ()

  Close the connection.
- SqlCommand CreateCommand ()

  Create a SQLCommand to be able to get/set information in the database.

Properties

- ConnectionState RobotConnectionState `[get]`

  State of the connection.
- string Connectionstring `[get, set]`

  The connection string used for the connection.
- int TimeOut `[get]`

  How long it will try to connect before timing out.

7.18.1 Member Function Documentation

7.18.1.1 void SqlInteraction.ISqlConnection.ConnectionClose ( )

Close the connection.

Implemented in SqlInteraction.RobotSqlConnection.

7.18.1.2 void SqlInteraction.ISqlConnection.ConnectionOpen ( )

Open the connection.

Implemented in SqlInteraction.RobotSqlConnection.

7.18.1.3 SqlCommand SqlInteraction.ISqlConnection.CreateCommand ( )

Create a SQLCommand to be able to get/set information in the database.

Returns

Implemented in SqlInteraction.RobotSqlConnection.

7.18.2 Property Documentation

7.18.2.1 string SqlInteraction.ISqlConnection.Connectionstring [get, set]

The connection string used for the connection.

Implemented in SqlInteraction.RobotSqlConnection.

7.18.2.2 ConnectionState SqlInteraction.ISqlConnection.RobotConnectionState [get]

State of the connection.

Implemented in SqlInteraction.RobotSqlConnection.

7.18.2.3 int SqlInteraction.ISqlConnection.TimeOut [get]

How long it will try to connect before timing out.

Implemented in SqlInteraction.RobotSqlConnection.

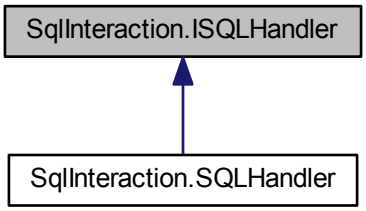
The documentation for this interface was generated from the following file:

- SqlInteraction/iSQLConnection.cs

7.19 SqlInteraction.ISQLHandler Interface Reference

Interface for handling class of sql.

Inheritance diagram for SqlInteraction.ISQLHandler:



Public Member Functions

- bool setConnection (string _server, string _database, string _username, string _password, string _timeout)

  Sets a new connection to use.
- SqlCommand makeCommand (string _commandText)

  Creates a command that can be used for sql interaction.
- void addParameter (SqlCommand _command, string _parameterName, object _parameterValue, SqlDbType _parameterType)

  Function adds a parameter to the command.
- ISQLReader runQuery (SqlCommand _command, string queryType)

  Executes the supplied command on SQL server.
- void changeConnectionparameter (string _parameter, string _parameterValue)

  Changes 1 specific parameter in connection info.

Properties

- ISqlConnection Connection `[get, set]`

  Variable to get the connection definition.

7.19.1 Detailed Description

Interface for handling class of sql.

7.19.2 Member Function Documentation

7.19.2.1 void SqlInteraction.ISQLHandler.addParameter (SqlCommand *_command*, string *_parameterName*, object *_parameterValue*, SqlDbType *_parameterType*)

Function adds a parameter to the command.

Parameters

_command	Parameter is the command made in makeCommand
_parameter-Name	Parameter is name of the parameter defined in makeCommand
_parameter-Value	Parameter is value to be used in the command
_parameterType	Parameter is of which type the parameter is

Implemented in SqlInteraction.SQLHandler.

7.19.2.2 void SqlInteraction.ISQLHandler.changeConnectionparameter (string *_parameter,* string *_parameterValue*)

Changes 1 specific parameter in connection info.

Parameters

_parameter	Parameter is part of or full parameter name to change
_parameter-Value	Parameter is value to change the connection info to

Implemented in SqlInteraction.SQLHandler.

7.19.2.3 SqlCommand SqlInteraction.ISQLHandler.makeCommand (string *_commandText*)

Creates a command that can be used for sql interaction.

Parameters

_commandText	Parameter for the command text

Returns

Returns the SqlCommand made from the given parameters

Implemented in SqlInteraction.SQLHandler.

7.19.2.4 ISQLReader SqlInteraction.ISQLHandler.runQuery (SqlCommand *_command,* string *queryType*)

Executes the supplied command on SQL server.

Parameters

_command	Parameter is command to be executed
queryType	Parameter is what kind of query to execute (write/read)

Returns

Returns Null if write, if read returns a Datareader

Implemented in SqlInteraction.SQLHandler.

7.19.2.5 bool SqlInteraction.ISQLHandler.setConnection (string *_server,* string *_database,* string *_username,* string *_password,* string *_timeout*)

Sets a new connection to use.

Parameters

_server	Paramater for ip/domain to use
_database	Parameter for which database to use
_username	Parameter for username to use
_password	Parameter for password to use
_timeout	Parameter for what timeout should be

Returns

Returns bool for whether connection was succesfully set

Implemented in SqlInteraction.SQLHandler.

7.19.3 Property Documentation

7.19.3.1 ISqlConnection SqlInteraction.ISQLHandler.Connection [get, set]

Variable to get the connection definition.

Implemented in SqlInteraction.SQLHandler.

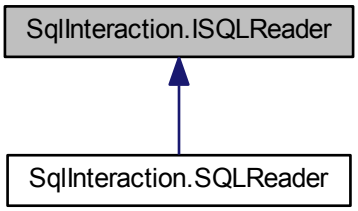
The documentation for this interface was generated from the following file:

- SqlInteraction/iSQLHandler.cs

7.20 SqlInteraction.ISQLReader Interface Reference

Interface for class that read information from a SQL table.

Inheritance diagram for SqlInteraction.ISQLReader:



Public Member Functions

- List< object > readRow ()

  Read one table row.
- void close ()

  Closes the reader.

7.20.1 Detailed Description

Interface for class that read information from a SQL table.

Used primarily as holder for already gathered information from a database.

7.20.2 Member Function Documentation

7.20.2.1 void SqlInteraction.ISQLReader.close ( )

Closes the reader.

Implemented in SqlInteraction.SQLReader.

7.20.2.2 List<object> SqlInteraction.ISQLReader.readRow ( )

Read one table row.

For each call the next row is read.

Returns

List of objects from the table. Empty list if end of table or no data in table.

Implemented in SqlInteraction.SQLReader.

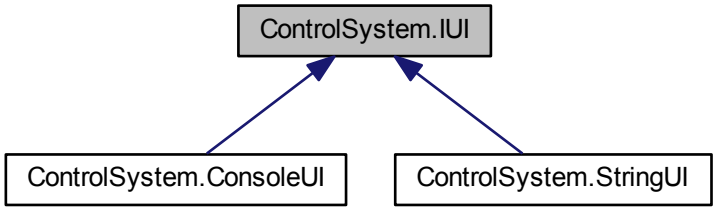
The documentation for this interface was generated from the following file:

- SqlInteraction/iSQLReader.cs

7.21 ControlSystem.IUI Interface Reference

Simple interface for UI interaction.

Inheritance diagram for ControlSystem.IUI:



Public Member Functions

- void write (string _sMsg, params object[] _paramobjArgument)

  Writes the string with arguments to the UI.
- void writeLine (string sMsg, params object[] paramobjArgument)

  Writes the string with arguments to the UI.

7.21.1 Detailed Description

Simple interface for UI interaction.

7.21.2 Member Function Documentation

7.21.2.1 void ControlSystem.IUI.write (string *_sMsg,* params object[] *_paramobjArgument*)

Writes the string with arguments to the UI.

No newline character written.

Parameters

sMsg	The string with the message and argument placement.(Like normal Write())
paramobj-Argument	Arguments to be placed in the string.

Implemented in ControlSystem.StringUI, and ControlSystem.ConsoleUI.

7.21.2.2 void ControlSystem.IUI.writeLine (string *sMsg,* params object[] *paramobjArgument*)

Writes the string with arguments to the UI.

Newline character appended to end of string.

Parameters

sMsg	The string with the message and argument placement.(Like normal WriteLine())
paramobj-Argument	Arguments to be placed in the string.

Implemented in ControlSystem.StringUI, and ControlSystem.ConsoleUI.

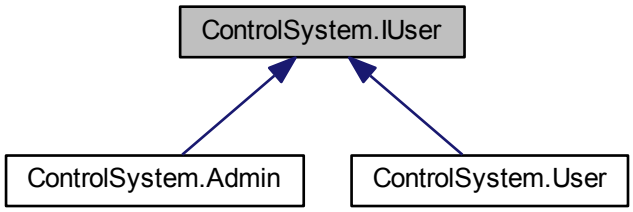
The documentation for this interface was generated from the following file:

- ControlSystem/ui.cs

7.22 ControlSystem.IUser Interface Reference

Interface for an user with permissions for the database.

Inheritance diagram for ControlSystem.IUser:



Properties

- string userName `[get, set]`

Name of the user.

- Dictionary< string, bool > permissionDictionary [get]

  Set of permissions.

7.22.1 Detailed Description

Interface for an user with permissions for the database.

7.22.2 Property Documentation

7.22.2.1 Dictionary<string, bool> ControlSystem.IUser.permissionDictionary [get]

Set of permissions.

Implemented in ControlSystem.Admin, and ControlSystem.User.

7.22.2.2 string ControlSystem.IUser.userName [get, set]

Name of the user.

Implemented in ControlSystem.Admin, and ControlSystem.User.

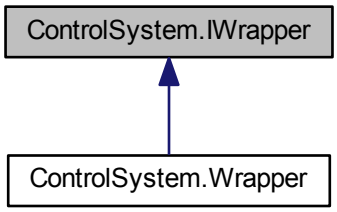
The documentation for this interface was generated from the following file:

- ControlSystem/User.cs

7.23 ControlSystem.IWrapper Interface Reference

Interface for a wrapper wrapping the USBC.dll file.

Inheritance diagram for ControlSystem.IWrapper:



Public Member Functions

- bool initializationWrapped (Wrapper.enumSystemModes _sysmodeMode, Wrapper.enumSystemTypes _systypeType, DLL.DgateCallBack _funcptrSuccess, DLL.DgateCallBack _funcptrError)

  Initializes the robot.

- bool controlWrapped (Wrapper.enumAxisSettings _axisSettingsGroup, bool _bControlOnOrOff)

  Turns control on and off for certain axis group.

- bool isOnlineOkWrapped ()

  Tells about the robot being online.

- bool homeWrapped (Wrapper.enumAxisSettings _axisSettingsGroup, DLL.DgateCallBackByteRefArg _funcptrHomingEventHandler)

  Homes a axis group. Should be called before calling most movement functions.

- bool enterManualWrapped (Wrapper.enumManualType _enummanMoveType)

  Must be called to use manual movement. Seems to stop previous movement of any object(Axis) that was moving before.

- bool closeManualWrapped ()

  Stops manual mode.

- bool moveManualWrapped (Wrapper.enumManualModeWhat _enumWhatToMove, int _lSpeed)

  Moves the robot. homeWrapped must have been called if moving by coordinates. enterManual seems have to be called before each call to this function. Use stopWrapped to stop motion afterwards.(Moving some other part of the system also stops the previous movement, since the system can only handle one object(Axis) moving at a time.)

- bool stopWrapped (Wrapper.enumAxisSettings _bWhatToStop)

  Stops movement of axis.

- bool moveLinearWrapped (string _sNameOfVector, int _iIndex)

  Flytter robotten til et punkt.

- bool openGripperWrapped ()

  Opens the gripper.

- bool closeGripperWrapped ()

  Closes the gripper.

- bool getJawWrapped (ref short _shrtPerc, ref short _shrtWidth)

  Gives information about how much open the gripper is.(Between the 'fingers')

- void watchMotionWrapped (DLL.DgateCallBackCharArg _funcptrCallbackEnd, DLL.DgateCallBackCharArg _funcptrCallbackStart)

  Adds functions to be called when motion starts and motion ends.

- bool watchDigitalInputWrapped (DLL.DgateCallBackLongArg _funcptrCallbackEvent)

  Adds a function to be called when digital input changes.

- bool closeWatchDigitalInputWrapped ()

  Stops watching of digital inputs.

- bool defineVectorWrapped (Wrapper.enumAxisSettings _enumGroup, string _sVectorName, short _shrtLength)

  Defines a new vector in robot memory.

- bool teachWrapped (SIRVector vecTheSirVector)

  Add the vector points to the vector with the same name.

- VecPoint getCurrentPosition ()

  Returns the position of the robot.

- bool timeWrapped (Wrapper.enumBGroup _bGroup, long _mTime)

  Sets the time future movement should take.

- bool speedWrapped (Wrapper.enumBGroup _bGroup, long _mSpeed)

  Sets the speed future movement should take.

7.23.1 Detailed Description

Interface for a wrapper wrapping the USBC.dll file.

7.23.2 Member Function Documentation

7.23.2.1 bool ControlSystem.IWrapper.closeGripperWrapped ( )

Closes the gripper.

Returns

Returns true on successful call.

Implemented in ControlSystem.Wrapper.

7.23.2.2 bool ControlSystem.IWrapper.closeManualWrapped ( )

Stops manual mode.

Returns

Returns true on successful call.

Implemented in ControlSystem.Wrapper.

7.23.2.3 bool ControlSystem.IWrapper.closeWatchDigitalInputWrapped ( )

Stops watching of digital inputs.

Note: Probably means no more events.

Returns

Returns true if successful call.

Implemented in ControlSystem.Wrapper.

7.23.2.4 bool ControlSystem.IWrapper.controlWrapped (Wrapper.enumAxisSettings *_axisSettingsGroup,* bool *_bControlOnOrOff*)

Turns control on and off for certain axis group.

Parameters

_axisSettingsGroup	Axis group to affect.(Use enum)
_bControlOnOrOff	To have it turned off or on.

Returns

Returns true on successful call.

Implemented in ControlSystem.Wrapper.

7.23.2.5 bool ControlSystem.IWrapper.defineVectorWrapped (Wrapper.enumAxisSettings *_enumGroup,* string *_sVectorName,* short *_shrtLength*)

Defines a new vector in robot memory.

Note: Good idea to have in program one of the SIRVector classes to contains vector information.

Parameters

_enumGroup	Group can use: Robot(Normally used) Peripherals All
_sVectorName	Name of vector.
_shrtLength	Length of vector.(Number of points.)

Returns

Returns true on successfull call.

Implemented in ControlSystem.Wrapper.

7.23.2.6 bool ControlSystem.IWrapper.enterManualWrapped (Wrapper.enumManualType *_enummanMoveType*)

Must be called to use manual movement. Seems to stop previous movement of any object(Axis) that was moving before.

Parameters

_enummanMoveType	What to move by.(Axis(0), Coordinates(1))

Returns

Returns true on successful call.

Implemented in ControlSystem.Wrapper.

7.23.2.7 VecPoint ControlSystem.IWrapper.getCurrentPosition ( )

Returns the position of the robot.

Returns

Returns current position.

Implemented in ControlSystem.Wrapper.

7.23.2.8 bool ControlSystem.IWrapper.getJawWrapped (ref short *_shrtPerc,* ref short *_shrtWidth*)

Gives information about how much open the gripper is.(Between the 'fingers')

Note: Probably most useful to use the _shrtWidth arg.

Parameters

_shrtPerc	Data in percentage.
_shrtWidth	Data in width.(mm)

Returns

Returns true on successful call.

Implemented in ControlSystem.Wrapper.

7.23.2.9 bool ControlSystem.IWrapper.homeWrapped (Wrapper.enumAxisSettings *_axisSettingsGroup,* DLL.DgateCallBackByteRefArg *_funcptrHomingEventHandler*)

Homes a axis group. Should be called before calling most movement functions.

Parameters

_axisSettingsGroup	The axis group.(Use enum)
_funcptrHomingEventHandler	Function to be called for homing events.

Values being passed in event: 0xff: Homing started 1 - 8: Axis n being homed. 0x40: Homing ended.

Returns

Returns true on successful call.

Implemented in ControlSystem.Wrapper.

7.23.2.10 bool ControlSystem.IWrapper.initializationWrapped (Wrapper.enumSystemModes *_sysmodeMode,* Wrapper.enumSystemTypes *_systypeType,* DLL.DgateCallBack *_funcptrSuccess,* DLL.DgateCallBack *_funcptrError*)

Initializes the robot.

Note: Should wait for it to be done before calling other functions.

Parameters

_shrtMode	Mode.(Use one of constants[Normally use online mode])
_shrtType	Type of connection.(Use one of constants[Normally use default])
_funcptrSuccess	Function to be called on success.
_funcptrError	Function to be called on error.

Returns

Returns true on successful call.(But errors can still happen)

Implemented in ControlSystem.Wrapper.

7.23.2.11 bool ControlSystem.IWrapper.isOnlineOkWrapped ( )

Tells about the robot being online.

Returns

Returns true if it is, false otherwise.

Implemented in ControlSystem.Wrapper.

7.23.2.12 bool ControlSystem.IWrapper.moveLinearWrapped (string *_sNameOfVector,* int *_iIndex*)

Flytter robotten til et punkt.

Todo Refactor.

Parameters

_sNameOf-Vector	Navnet på vektoren i robotten.
_iIndex	Index for punkt.

Returns

Returns true on successfull call.

Implemented in ControlSystem.Wrapper.

7.23.2.13 bool ControlSystem.IWrapper.moveManualWrapped (Wrapper.enumManualModeWhat *_enumWhatToMove,* int *_lSpeed*)

Moves the robot. homeWrapped must have been called if moving by coordinates. enterManual seems have to be called before each call to this function. Use stopWrapped to stop motion afterwards.(Moving some other part of the system also stops the previous movement, since the system can only handle one object(Axis) moving at a time.)

Implemented in ControlSystem.Wrapper.

7.23.2.14 bool ControlSystem.IWrapper.openGripperWrapped ( )

Opens the gripper.

Returns

Returns true on successful call.

Implemented in ControlSystem.Wrapper.

7.23.2.15 bool ControlSystem.IWrapper.speedWrapped (Wrapper.enumBGroup *_bGroup,* long *_mSpeed*)

Sets the speed future movement should take.

Parameters

_bGroup	bool ucGroup Axis group to which the time should be applied '&' for all axes '0'-'7' for axis movements 'A' for robot movements 'B' for peripheral movements 'G' for gripper movements
_mSpeed	Speed in percent of max speed

Returns

Returns true if the speed has been succesfully set, false otherwise..

Implemented in ControlSystem.Wrapper.

7.23.2.16 bool ControlSystem.IWrapper.stopWrapped (Wrapper.enumAxisSettings *_bWhatToStop*)

Stops movement of axis.

Parameters

_bWhatToStop	Axis to stop.

Returns

Returns true on successful call.

Implemented in ControlSystem.Wrapper.

7.23.2.17 bool ControlSystem.IWrapper.teachWrapped (SIRVector *vecTheSirVector*)

Add the vector points to the vector with the same name.

Note: Should call 'defineVectorWrapped' first.

Parameters

vecTheSirVector	The vector with the points.

Returns

Returns true on succeessfull call.

Implemented in ControlSystem.Wrapper.

7.23.2.18 bool ControlSystem.IWrapper.timeWrapped (Wrapper.enumBGroup *_bGroup,* long *_mTime*)

Sets the time future movement should take.

Parameters

_bGroup	bool ucGroup Axis group to which the time should be applied '&' for all axes '0'-'7' for axis movements 'A' for robot movements 'B' for peripheral movements 'G' for gripper movements
_mTime	Time in milliseconds

Returns

Returns true if time has been succesfully set, false otherwise.

Implemented in ControlSystem.Wrapper.

7.23.2.19 bool ControlSystem.IWrapper.watchDigitalInputWrapped (DLL.DgateCallBackLongArg *_funcptrCallbackEvent*)

Adds a function to be called when digital input changes.

Parameters

_funcptr-CallbackEvent	The function to be called.

Returns

Returns true if successful call.

Implemented in ControlSystem.Wrapper.

7.23.2.20 void ControlSystem.IWrapper.watchMotionWrapped (DLL.DgateCallBackCharArg *_funcptrCallbackEnd*, DLL.DgateCallBackCharArg *_funcptrCallbackStart*)

Adds functions to be called when motion starts and motion ends.

Parameters

_funcptr-CallbackEnd	Function to be called when motion has ended.
_funcptr-CallbackStart	Function to be called when motion has started.

Implemented in ControlSystem.Wrapper.

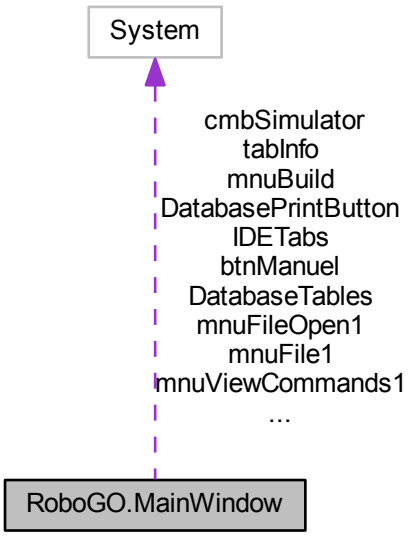
The documentation for this interface was generated from the following file:

- ControlSystem/iWrapper.cs

7.24 RoboGO.MainWindow Class Reference

MainWindow.

Collaboration diagram for RoboGO.MainWindow:



Public Member Functions

- void InitializeComponent ()

  InitializeComponent.
- void InitializeComponent ()

  InitializeComponent.

7.24.1 Detailed Description

MainWindow.

7.24.2 Member Function Documentation

7.24.2.1 void RoboGO.MainWindow.InitializeComponent ( )

InitializeComponent.

7.24.2.2 void RoboGO.MainWindow.InitializeComponent ( )

InitializeComponent.

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

- RoboGO/obj/x86/Debug/MainWindow.g.cs
- RoboGO/obj/x86/Debug/MainWindow.g.i.cs

7.25 RoboGO.MainWindowViewModel Class Reference

ViewModel for the mainwindow.

Public Member Functions

- MainWindowViewModel (ProgressBar _pb)

  Constructor with progressbar for showing connection status.
- void setSimulatorAsRobotInstance ()

  Sets the current robot as being a simulator.
- bool setRobotAsRobotInstance ()

  Sets the curren robot as being the SCORBOT.
- void stopRobotInstance ()

  Stops the robot from continuing any action.
- void checkIsOnline ()

  Check for being online.(The robot.)

7.25.1 Detailed Description

ViewModel for the mainwindow.

7.25.2 Constructor & Destructor Documentation

7.25.2.1 RoboGO.MainWindowViewModel.MainWindowViewModel (ProgressBar *_pb*)

Constructor with progressbar for showing connection status.

Parameters

_pb	Progressbar for showing connection status

7.25.3 Member Function Documentation

7.25.3.1 void RoboGO.MainWindowViewModel.checkIsOnline ( )

Check for being online.(The robot.)

7.25.3.2 bool RoboGO.MainWindowViewModel.setRobotAsRobotInstance ( )

Sets the curren robot as being the SCORBOT.

Returns

False if DLL missing.

7.25.3.3 void RoboGO.MainWindowViewModel.setSimulatorAsRobotInstance ( )

Sets the current robot as being a simulator.

7.25.3.4 void RoboGO.MainWindowViewModel.stopRobotInstance ( )

Stops the robot from continuing any action.

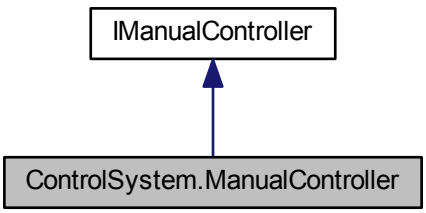
The documentation for this class was generated from the following file:

- RoboGO/ViewModels/mainWindowViewModel.cs

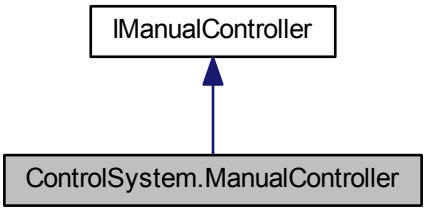
7.26 ControlSystem.ManualController Class Reference

Class that encapsulates controlling manual movement. Instead of using directly Robot or Simulator by interface.

Inheritance diagram for ControlSystem.ManualController:



Collaboration diagram for ControlSystem.ManualController:



Public Member Functions

- void moveAxisBase (enumLeftRight _elrDirection)

  Move the base in the desired direction in 'speed' percentage of maximum speed.
- void moveAxisShoulder (enumLeftRight _elrDirection)

  Moves the shoulder in the desired direction.
- void moveAxisElbow (enumLeftRight _elrDirection)

  Moves the elbow in the desired direction.
- void moveAxisGripper (enumCloseOpen _ecoGripper)

  Opens or closes the gripper.
- void moveAxisPitch (enumUpDown _eudDirection)

  Moves the wrists pitch in the desired direction.
- void moveAxisRoll (enumLeftRight _elrDirection)

  Rolls the wrist in the desired direction.
- void moveAxisConveyer (enumLeftRight _elrDirection)

  Move the conveyer belt in the desired direction.
- void moveCoordX (enumIncDec _eidIncOrDec)

  Change the robots X-coordinate.
- void moveCoordY (enumIncDec _eidIncOrDec)

  Change the robots Y-coordinate.
- void moveCoordZ (enumIncDec _eidIncOrDec)

  Change the robots Z-coordinate.
- void moveCoordPitch (enumIncDec _eidIncOrDec)

  Change the wrists pitch.
- void moveCoordRoll (enumIncDec _eidIncOrDec)

  Change the roll of the wrist.
- void stopAllMovement ()

  Stops movement of all axes.

Properties

- int Speed `[get, set]`

  Speed in percentage.
- IRobot RobotConnection `[get, set]`

  What to steer.

7.26.1 Detailed Description

Class that encapsulates controlling manual movement. Instead of using directly Robot or Simulator by interface.

Note: Uses IRobot, so it is able to use either a Robot or a Simulator.

7.26.2 Member Function Documentation

7.26.2.1 void ControlSystem.ManualController.moveAxisBase (enumLeftRight *_elrDirection*)

Move the base in the desired direction in 'speed' percentage of maximum speed.

Parameters

_elrDirection	Where ya wanna go?

Implements ControlSystem.IManualController.

7.26.2.2 void ControlSystem.ManualController.moveAxisConveyer (enumLeftRight *_elrDirection*)

Move the conveyer belt in the desired direction.

Parameters

_elrDirection	What direction to move in.

Implements ControlSystem.IManualController.

7.26.2.3 void ControlSystem.ManualController.moveAxisElbow (enumLeftRight *_elrDirection*)

Moves the elbow in the desired direction.

Parameters

_elrDirection	What direction to move in.

Implements ControlSystem.IManualController.

7.26.2.4 void ControlSystem.ManualController.moveAxisGripper (enumCloseOpen *_ecoGripper*)

Opens or closes the gripper.

Parameters

_ecoGripper	To open or close.

Implements ControlSystem.IManualController.

7.26.2.5 void ControlSystem.ManualController.moveAxisPitch (enumUpDown *_eudDirection*)

Moves the wrists pitch in the desired direction.

Parameters

_eudDirection	What direction to move in.

Implements ControlSystem.IManualController.

7.26.2.6 void ControlSystem.ManualController.moveAxisRoll (enumLeftRight *_elrDirection*)

Rolls the wrist in the desired direction.

Parameters

_elrDirection	What direction to move in.

Implements ControlSystem.IManualController.

7.26.2.7 void ControlSystem.ManualController.moveAxisShoulder (enumLeftRight *_elrDirection*)

Moves the shoulder in the desired direction.

Parameters

_elrDirection	What direction to move in.

Implements ControlSystem.IManualController.

7.26.2.8 void ControlSystem.ManualController.moveCoordPitch (enumIncDec *_eidIncOrDec*)

Change the wrists pitch.

Parameters

_eidIncOrDec	Increasing or decreasing.

Implements ControlSystem.IManualController.

7.26.2.9 void ControlSystem.ManualController.moveCoordRoll (enumIncDec *_eidIncOrDec*)

Change the roll of the wrist.

Parameters

_eidIncOrDec	Increasing or decreasing.

Implements ControlSystem.IManualController.

7.26.2.10 void ControlSystem.ManualController.moveCoordX (enumIncDec *_eidIncOrDec*)

Change the robots X-coordinate.

Parameters

_eidIncOrDec	Increasing or decreasing.

Implements ControlSystem.IManualController.

7.26.2.11 void ControlSystem.ManualController.moveCoordY (enumIncDec *_eidIncOrDec*)

Change the robots Y-coordinate.

Parameters

_eidIncOrDec	Increasing or decreasing.

Implements ControlSystem.IManualController.

7.26.2.12 void ControlSystem.ManualController.moveCoordZ (enumIncDec *_eidIncOrDec*)

Change the robots Z-coordinate.

Parameters

_eidIncOrDec	Increasing or decreasing.

Implements ControlSystem.IManualController.

7.26.2.13 void ControlSystem.ManualController.stopAllMovement ( )

Stops movement of all axes.

Implements ControlSystem.IManualController.

7.26.3 Property Documentation

7.26.3.1 IRobot ControlSystem.ManualController.RobotConnection [get, set]

What to steer.

Could be for example Robot(ER4) or Simulator.

Implements ControlSystem.IManualController.

7.26.3.2 int ControlSystem.ManualController.Speed [get, set]

Speed in percentage.

So should be between 0 and 100.

Implements ControlSystem.IManualController.

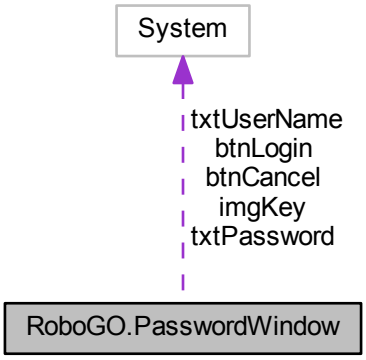
The documentation for this class was generated from the following file:

- ControlSystem/manualController.cs

7.27 RoboGO.PasswordWindow Class Reference

PasswordWindow.

Collaboration diagram for RoboGO.PasswordWindow:



Public Member Functions

- void InitializeComponent ()

  InitializeComponent.
- void InitializeComponent ()

  InitializeComponent.

7.27.1 Detailed Description

PasswordWindow.

7.27.2 Member Function Documentation

7.27.2.1 void RoboGO.PasswordWindow.InitializeComponent ( )

InitializeComponent.

7.27.2.2 void RoboGO.PasswordWindow.InitializeComponent ( )

InitializeComponent.

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

- RoboGO/obj/x86/Debug/PasswordWindow.g.cs
- RoboGO/obj/x86/Debug/PasswordWindow.g.i.cs

7.28 RoboGO.ViewModels.passwordWindowViewModel Class Reference

ViewModel for password window.

Public Member Functions

- passwordWindowViewModel ()

  Default constructor.
- bool authenticate (string _loginName, string _loginPassword)

  Checks the user.

7.28.1 Detailed Description

ViewModel for password window.

7.28.2 Constructor & Destructor Documentation

7.28.2.1 RoboGO.ViewModels.passwordWindowViewModel.passwordWindowViewModel ( )

Default constructor.

7.28.3 Member Function Documentation

7.28.3.1 bool RoboGO.ViewModels.passwordWindowViewModel.authenticate (string *_loginName,* string *_loginPassword*)

Checks the user.

Parameters

_loginName	User name.
_loginPassword	Password for user.

Returns

Returns true if matching user and password.

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

- RoboGO/ViewModels/passwordWindowViewModel.cs

7.29 RoboGO.ViewModels.PositionModel Class Reference

Class to keep track of simulator position.

Properties

- VecPoint PositionVec `[get, set]`

  Get and set Position as a VectPoint.

7.29.1 Detailed Description

Class to keep track of simulator position.

7.29.2 Property Documentation

7.29.2.1 VecPoint RoboGO.ViewModels.PositionModel.PositionVec [get, set]

Get and set Position as a VectPoint.

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

- RoboGO/ViewModels/positionModel.cs

7.30 RoboGO.ViewModels.PositionViewModel Class Reference

ViewModel for the position class.

Public Member Functions

- PositionViewModel ()

  PositionViewModel classconstructor.
- PositionViewModel (PositionModel _pm)

  PositionViewModel explixitconstructor.
- void update ()

  Update function, getting the current position from the current IRobotinstance. Set the Position as vectpoint in PositionModel.
- void update (VecPoint _vect)

  Sets the current position.
- string getXYZPR ()

  Get the current position.

7.30.1 Detailed Description

ViewModel for the position class.

7.30.2 Constructor & Destructor Documentation

7.30.2.1 RoboGO.ViewModels.PositionViewModel.PositionViewModel ( )

PositionViewModel classconstructor.

7.30.2.2 RoboGO.ViewModels.PositionViewModel.PositionViewModel (PositionModel *_pm*)

PositionViewModel explixitconstructor.

7.30.3 Member Function Documentation

7.30.3.1 string RoboGO.ViewModels.PositionViewModel.getXYZPR ( )

Get the current position.

Returns

String of position Vectpoint

7.30.3.2 void RoboGO.ViewModels.PositionViewModel.update ( )

Update function, getting the current position from the current IRobotinstance. Set the Position as vectpoint in PositionModel.

7.30.3.3 void RoboGO.ViewModels.PositionViewModel.update (VecPoint *_vect*)

Sets the current position.

Parameters

_vect	The new position.

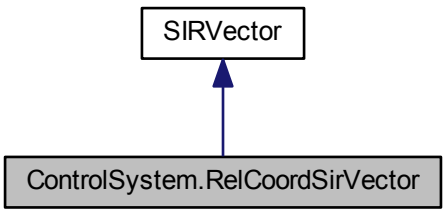
The documentation for this class was generated from the following file:

- RoboGO/ViewModels/positionViewModel.cs

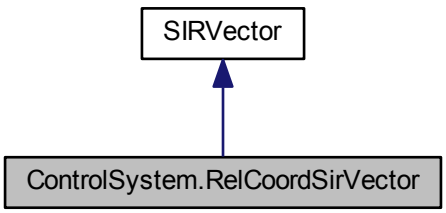
7.31 ControlSystem.RelCoordSirVector Class Reference

SIRVector class for relative positions.

Inheritance diagram for ControlSystem.RelCoordSirVector:



Collaboration diagram for ControlSystem.RelCoordSirVector:



Public Member Functions

- RelCoordSirVector (string _sName)

  Contructors whichs sets up type and name of vector.

7.31.1 Detailed Description

SIRVector class for relative positions.

7.31.2 Constructor & Destructor Documentation

7.31.2.1 ControlSystem.RelCoordSirVector.RelCoordSirVector (string *_sName*)

Contructors whichs sets up type and name of vector.

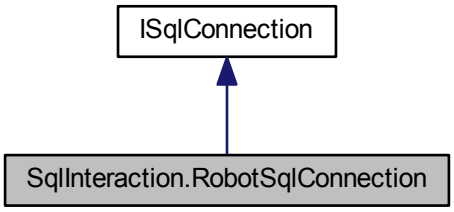
The documentation for this class was generated from the following file:

- ControlSystem/wrapper.cs

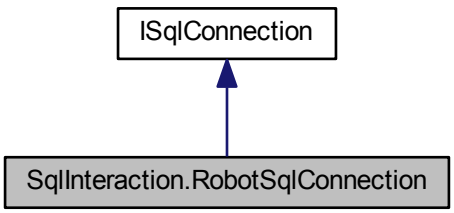
7.32 SqlInteraction.RobotSqlConnection Class Reference

SQLConnection to connect to a database.

Inheritance diagram for SqlInteraction.RobotSqlConnection:



Collaboration diagram for SqlInteraction.RobotSqlConnection:



Public Member Functions

- RobotSqlConnection (string connectionstring)

  Constructor that connects to a database with the information specified.
- void ConnectionOpen ()

  Open the connection.
- void ConnectionClose ()

  Close the connection.
- SqlCommand CreateCommand ()

  Create a SQLCommand to be able to get/set information in the database.

Properties

- ConnectionState RobotConnectionState `[get]`

  State of the connection.
- string Connectionstring `[get, set]`

  The connection string used for the connection.
- int TimeOut `[get]`

  How long it will try to connect before timing out.

7.32.1 Detailed Description

SQLConnection to connect to a database.

7.32.2 Constructor & Destructor Documentation

7.32.2.1 SqlInteraction.RobotSqlConnection.RobotSqlConnection (string *connectionstring*)

Constructor that connects to a database with the information specified.

Parameters

connectionstring	Information for connection.

7.32.3 Member Function Documentation

7.32.3.1 void SqlInteraction.RobotSqlConnection.ConnectionClose ( )

Close the connection.

Implements SqlInteraction.ISqlConnection.

7.32.3.2 void SqlInteraction.RobotSqlConnection.ConnectionOpen ( )

Open the connection.

Implements SqlInteraction.ISqlConnection.

7.32.3.3 SqlCommand SqlInteraction.RobotSqlConnection.CreateCommand ( )

Create a SQLCommand to be able to get/set information in the database.

Returns

Implements SqlInteraction.ISqlConnection.

7.32.4 Property Documentation

7.32.4.1 string SqlInteraction.RobotSqlConnection.Connectionstring [get, set]

The connection string used for the connection.

Implements SqlInteraction.ISqlConnection.

7.32.4.2 ConnectionState SqlInteraction.RobotSqlConnection.RobotConnectionState [get]

State of the connection.

Implements SqlInteraction.ISqlConnection.

7.32.4.3 int SqlInteraction.RobotSqlConnection.TimeOut [get]

How long it will try to connect before timing out.

Implements SqlInteraction.ISqlConnection.

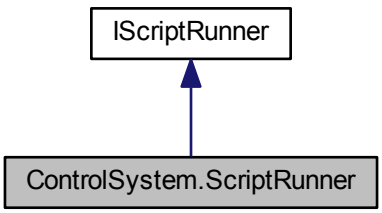
The documentation for this class was generated from the following file:

- SqlInteraction/robotSqlConnection.cs

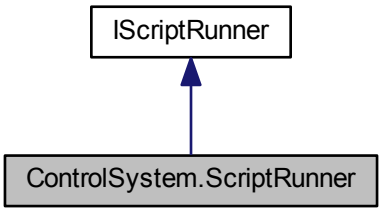
7.33 ControlSystem.ScriptRunner Class Reference

Used to run IronPython scripts.

Inheritance diagram for ControlSystem.ScriptRunner:



Collaboration diagram for ControlSystem.ScriptRunner:



Public Member Functions

- void setRobotInstance (IRobot _iroboRobot)

  Sets the underlying IRobot that the scripts are used on.
- void setScriptFromFile (string _sPath)

  Loads script from a file.
- void setScriptFromString (string _sScript)

  Loads script from a string.
- string readFromOutputStream ()

  Returns all input from the program.
- void clearOutputStream ()

  Clear the output.
- void ExecuteScript ()

  Executes loaded script.

Static Public Member Functions

- static ScriptRunner getInstance ()

  Gets the instance of the ScriptRunner.

7.33.1 Detailed Description

Used to run IronPython scripts.

Note: Needs IronPython, IronPython.Modules and Microsoft.Scripting assemblies as reference

7.33.2 Member Function Documentation

7.33.2.1 void ControlSystem.ScriptRunner.clearOutputStream ( )

Clear the output.

Implements ControlSystem.IScriptRunner.

7.33.2.2 void ControlSystem.ScriptRunner.ExecuteScript ( )

Executes loaded script.

Implements ControlSystem.IScriptRunner.

7.33.2.3 static ScriptRunner ControlSystem.ScriptRunner.getInstance ( ) [static]

Gets the instance of the ScriptRunner.

Returns

7.33.2.4 string ControlSystem.ScriptRunner.readFromOutputStream ( )

Returns all input from the program.

Returns

Output from program.

Implements ControlSystem.IScriptRunner.

7.33.2.5 void ControlSystem.ScriptRunner.setRobotInstance (IRobot *_iroboRobot*)

Sets the underlying IRobot that the scripts are used on.

Parameters

_iroboRobot	Robot to run script on.

Implements ControlSystem.IScriptRunner.

7.33.2.6 void ControlSystem.ScriptRunner.setScriptFromFile (string *_sPath*)

Loads script from a file.

Implements ControlSystem.IScriptRunner.

7.33.2.7 void ControlSystem.ScriptRunner.setScriptFromString (string *_sScript*)

Loads script from a string.

Implements ControlSystem.IScriptRunner.

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

- ControlSystem/scriptRunner.cs

7.34 ControlSystem.SerialSTK Class Reference

Class for functions to communicating with the STK kit.(Serial communication.)

Public Member Functions

- SerialSTK (int baud=9600, Parity parity=Parity.None, int dataBits=8, StopBits stopBits=StopBits.One)

  Classconstructor.
- bool Open ()

  Opens for communication.
- bool Close ()

  Closes for communication.
- double ReadADC ()

  Reads an ADC value.

7.34.1 Detailed Description

Class for functions to communicating with the STK kit.(Serial communication.)

7.34.2 Constructor & Destructor Documentation

7.34.2.1 ControlSystem.SerialSTK.SerialSTK (int *baud* = `9600`, Parity *parity* = `Parity.None`, int *dataBits* = `8`, StopBits *stopBits* = `StopBits.One`)

Classconstructor.

Parameters

port	Comport to communicate through
baud	baudrate (has to be the same as the STK-kit)
parity	Parity
dataBits	Databits
stopBits	Stopbits

7.34.3 Member Function Documentation

7.34.3.1 bool ControlSystem.SerialSTK.Close ( )

Closes for communication.

Returns

True if closed, false otherwise

7.34.3.2 bool ControlSystem.SerialSTK.Open ( )

Opens for communication.

Returns

True if opened, false otherwise

7.34.3.3 double ControlSystem.SerialSTK.ReadADC ( )

Reads an ADC value.

Returns

Value from the ADC

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

- ControlSystem/serialSTK.cs

7.35 ControlSystem.Simulator Class Reference

IRobot implementation using IUI output interface for simulating robot behavior.

Public Member Functions

- Simulator ()
  Default constructor.
- bool stopAllMovement ()
  Writes out that All movement is stopped.
- bool closeGripper ()
  Close the gripper.
- bool openGripper ()
  Opens the gripper.
- bool moveByCoordinates (int _x, int _y, int _z, int _pitch, int _roll)
  Function for moving by coordinates.
- short getJawOpeningWidthMilimeters ()
  Gets width of jaw opening in milimeters.
- short getJawOpeningWidthPercentage ()
  Gets width of jaw opening in percent.
- bool homeRobot ()
  Setting robot to start position.
- bool isOnline ()
  Connect to the robot.
- bool moveBase (int speed)
  Moving the base.
- bool moveShoulder (int speed)
  Moving the shoulder.
- bool moveWristPitch (int speed)
  Moving the wrist Pitch.
- bool moveWristRoll (int speed)
  Moving the Wrist Roll.
- bool moveElbow (int speed)
  Moving the elbow.
- bool moveGripper (int speed)
  Moving the gripper.
- bool moveConveyerBelt (int speed)
  Moving the Conveyer Belt.
- bool moveByAbsoluteCoordinates (int x, int y, int z, int pitch, int roll)
  Moving from the absolute position (home position)
- bool moveByRelativeCoordinates (int _iX, int _iY, int _iZ, int _iPitch, int _iRoll)
  Moving from the relative position (current position)
- bool moveByXCoordinate (int x)

Moves just X coordinate.

- bool moveByYCoordinate (int y)

  Moves just Y coordinate.
- bool moveByZCoordinate (int z)

  Moves just Z coordinate.
- bool moveByPitch (int pitch)

  Moves just pitch coordinate.
- bool moveByRoll (int roll)

  Moves just roll coordinate.
- bool Time (Wrapper.enumBGroup _bGroup, long _mTime)

  Time for future movements in miliseconds.
- bool Speed (Wrapper.enumBGroup _bGroup, long _mSpeed)

  Speed for future movements in percent.
- bool movebyCoordinates (int _iX, int _iY, int _iZ)

  Moves all coordinates.
- VecPoint getCurrentPosition ()

  Gets current position.
- bool **moveToCubePosition** (int _iCubeID)
- string **getCurrentPositionAsString** ()
- double **getWeight** ()

Properties

- IUI IUIOutput `[get, set]`

  Output for writing robot operations.
- VecPoint Currentposition `[get, set]`

  Its current position.

7.35.1 Detailed Description

IRobot implementation using IUI output interface for simulating robot behavior.

7.35.2 Constructor & Destructor Documentation

7.35.2.1 ControlSystem.Simulator.Simulator ( )

Default constructor.

Note: Uses console output as standard.

7.35.3 Member Function Documentation

7.35.3.1 bool ControlSystem.Simulator.closeGripper ( )

Close the gripper.

Returns

Returns true if the gripper closes

7.35.3.2 VecPoint ControlSystem.Simulator.getCurrentPosition ( )

Gets current position.

Returns

Currentposition

7.35.3.3 short ControlSystem.Simulator.getJawOpeningWidthMilimeters ( )

Gets width of jaw opening in milimeters.

Returns

7.35.3.4 short ControlSystem.Simulator.getJawOpeningWidthPercentage ( )

Gets width of jaw opening in percent.

Returns

7.35.3.5 bool ControlSystem.Simulator.homeRobot ( )

Setting robot to start position.

Returns

true if robot goes home.

7.35.3.6 bool ControlSystem.Simulator.isOnline ( )

Connect to the robot.

Returns

true if there are connection

7.35.3.7 bool ControlSystem.Simulator.moveBase (int *speed*)

Moving the base.

Parameters

speed	speed

Returns

Always true.

7.35.3.8 bool ControlSystem.Simulator.moveByAbsoluteCoordinates (int *x*, int *y*, int *z*, int *pitch*, int *roll*)

Moving from the absolute position (home position)

<params name="ALL">represents each coordinate values</params>

Returns

Always true.

7.35.3.9 bool ControlSystem.Simulator.moveByCoordinates (int *_x*, int *_y*, int *_z*, int *_pitch*, int *_roll*)

Function for moving by coordinates.

Parameters

_x	x-coordinate
_y	y-coordinate
_z	z-coordinate
_pitch	pitch robot arm
_roll	roll of robot arm

Returns

7.35.3.10 bool ControlSystem.Simulator.movebyCoordinates (int *_iX*, int *_iY*, int *_iZ*)

Moves all coordinates.

<params>value for the three coordinates</params>

Returns

Always true.

7.35.3.11 bool ControlSystem.Simulator.moveByPitch (int *pitch*)

Moves just pitch coordinate.

Parameters

pitch	value for the pitch coordinate

Returns

Always true.

7.35.3.12 bool ControlSystem.Simulator.moveByRelativeCoordinates (int *_iX*, int *_iY*, int *_iZ*, int *_iPitch*, int *_iRoll*)

Moving from the relative position (current position)

<params name="ALL">represents each coordinate values</params>

Returns

Always true.

7.35.3.13 bool ControlSystem.Simulator.moveByRoll (int *roll*)

Moves just roll coordinate.

Parameters

roll	value for the roll coordinate

Returns

Always true.

7.35.3.14 bool ControlSystem.Simulator.moveByXCoordinate (int *x*)

Moves just X coordinate.

Parameters

x	value for the x coordinate

Returns

Always true.

7.35.3.15 bool ControlSystem.Simulator.moveByYCoordinate (int *y*)

Moves just Y coordinate.

Parameters

y	value for the y coordinate

Returns

Always true.

7.35.3.16 bool ControlSystem.Simulator.moveByZCoordinate (int *z*)

Moves just Z coordinate.

Parameters

z	value for the z coordinate

Returns

Always true.

7.35.3.17 bool ControlSystem.Simulator.moveConveyerBelt (int *speed*)

Moving the Conveyer Belt.

Parameters

speed	speed

Returns

Always true.

7.35.3.18 bool ControlSystem.Simulator.moveElbow (int *speed*)

Moving the elbow.

Parameters

speed	speed

Returns

Always true.

7.35.3.19 bool ControlSystem.Simulator.moveGripper (int *speed*)

Moving the gripper.

Parameters

speed	speed

Returns

Always true.

7.35.3.20 bool ControlSystem.Simulator.moveShoulder (int *speed*)

Moving the shoulder.

Parameters

speed	speed

Returns

Always true.

7.35.3.21 bool ControlSystem.Simulator.moveWristPitch (int *speed*)

Moving the wrist Pitch.

Parameters

speed	speed

Returns

Always true.

7.35.3.22 bool ControlSystem.Simulator.moveWristRoll (int *speed*)

Moving the Wrist Roll.

Parameters

speed	speed

Returns

Always true.

7.35.3.23 bool ControlSystem.Simulator.openGripper ( )

Opens the gripper.

Returns

returns true if the gripper opens

7.35.3.24 bool ControlSystem.Simulator.Speed (Wrapper.enumBGroup *_bGroup,* long *_mSpeed*)

Speed for future movements in percent.

Parameters

_bGroup	Part of the robot

///

Parameters

_mSpeed	Value for speed

Returns

Always true.

7.35.3.25 bool ControlSystem.Simulator.stopAllMovement ( )

Writes out that All movement is stopped.

Returns

returns true if it is stopped

7.35.3.26 bool ControlSystem.Simulator.Time (Wrapper.enumBGroup *_bGroup*, long *_mTime*)

Time for future movements in miliseconds.

Parameters

_bGroup	Part of the robot

///

Parameters

_mTime	Value for time

Returns

Always true.

7.35.4 Property Documentation

7.35.4.1 VecPoint ControlSystem.Simulator.Currentposition [get, set]

Its current position.

7.35.4.2 IUI ControlSystem.Simulator.IUIOutput [get, set]

Output for writing robot operations.

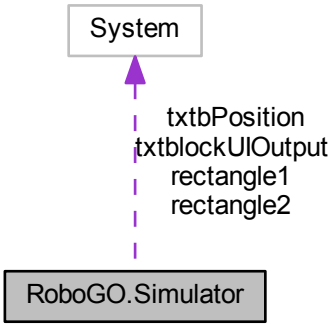
The documentation for this class was generated from the following file:

- ControlSystem/simulator.cs

7.36 RoboGO.Simulator Class Reference

Simulator.

Collaboration diagram for RoboGO.Simulator:



Public Member Functions

- void InitializeComponent ()

  InitializeComponent.
- void InitializeComponent ()

  InitializeComponent.

7.36.1 Detailed Description

Simulator.

7.36.2 Member Function Documentation

7.36.2.1 void RoboGO.Simulator.InitializeComponent ( )

InitializeComponent.

7.36.2.2 void RoboGO.Simulator.InitializeComponent ( )

InitializeComponent.

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

- RoboGO/obj/x86/Debug/Simulator.g.cs
- RoboGO/obj/x86/Debug/Simulator.g.i.cs

7.37 RoboGO.ViewModels.SimulatorViewModel Class Reference

ViewModel for the simulator class.

Public Member Functions

- SimulatorViewModel ()

  Default constructor setting up simulator.

Properties

- StringUI UIText `[get]`

  Simulator output.
- string CurrentPosition `[get]`

  Current position of the simulator.

7.37.1 Detailed Description

ViewModel for the simulator class.

Note: Uses StringUI class for output.

7.37.2 Constructor & Destructor Documentation

7.37.2.1 RoboGO.ViewModels.SimulatorViewModel.SimulatorViewModel ( )

Default constructor setting up simulator.

Note: Edits Factory->Simulator.

7.37.3 Property Documentation

7.37.3.1 string RoboGO.ViewModels.SimulatorViewModel.CurrentPosition [get]

Current position of the simulator.

7.37.3.2 StringUI RoboGO.ViewModels.SimulatorViewModel.UIText [get]

Simulator output.

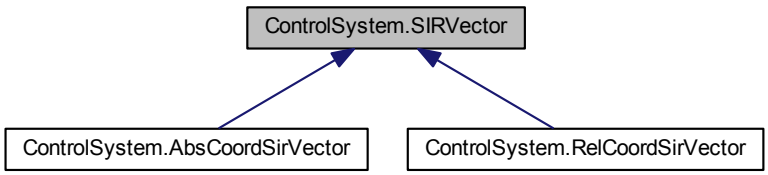
The documentation for this class was generated from the following file:

- RoboGO/ViewModels/simulatorViewModel.cs

7.38 ControlSystem.SIRVector Class Reference

Base class for vector used in wrapper.

Inheritance diagram for ControlSystem.SIRVector:



Public Member Functions

- int getSize ()

  Number of points in vector.
- void addPoint (VecPoint pNewVecPoint)

  Add a point to the vector.
- VecPoint getPoint (int iIndex)

  Gets a point from the vector.

Protected Attributes

- string **sName**
- List< VecPoint > **lstPoints**

- int iType

  Type of vector.(Should be set in classes inheriting from this.)

Properties

- string Name [get]

  Name of the vector.
- int Type [get]

  Type of the vector.(Relative or Absolute.)

7.38.1 Detailed Description

Base class for vector used in wrapper.

Should use the derived classes.

7.38.2 Member Function Documentation

7.38.2.1 void ControlSystem.SIRVector.addPoint (VecPoint *pNewVecPoint*)

Add a point to the vector.

Parameters

pNewVecPoint	

7.38.2.2 VecPoint ControlSystem.SIRVector.getPoint (int *iIndex*)

Gets a point from the vector.

Parameters

iIndex	Index of the point.

Returns

The point.

7.38.2.3 int ControlSystem.SIRVector.getSize ( )

Number of points in vector.

Returns

Number of points.

7.38.3 Member Data Documentation

7.38.3.1 int ControlSystem.SIRVector.iType [protected]

Type of vector.(Should be set in classes inheriting from this.)

7.38.4 Property Documentation

7.38.4.1 string ControlSystem.SIRVector.Name [get]

Name of the vector.

7.38.4.2 int ControlSystem.SIRVector.Type [get]

Type of the vector.(Relative or Absolute.)

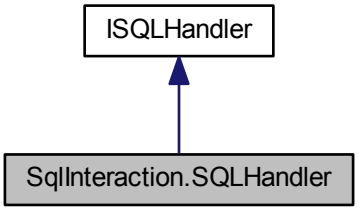
The documentation for this class was generated from the following file:

- ControlSystem/wrapper.cs

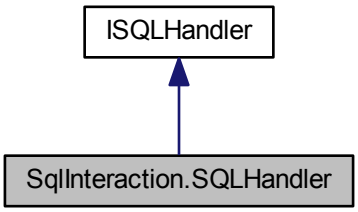
7.39 SqlInteraction.SQLHandler Class Reference

Class that handles all SQL interaction.

Inheritance diagram for SqlInteraction.SQLHandler:



Collaboration diagram for SqlInteraction.SQLHandler:



Public Member Functions

- bool setConnection (string _server, string _database, string _username, string _password, string _timeout)

Sets a new connection to use.

- SqlCommand makeCommand (string _commandText)

  Creates a command that can be used for sql interaction.
- void addParameter (SqlCommand _command, string _parameterName, object _parameterValue, SqlDbType _parameterType)

  Function adds a parameter to the command.
- ISQLReader runQuery (SqlCommand _command, string queryType)

  Executes the supplied command on SQL server.
- void changeConnectionparameter (string _parameter, string _parameterValue)

  Changes 1 specific parameter in connection info.

Properties

- static ISQLHandler GetInstance `[get]`

  Variable to get the SQLHandler singleton instance.
- ISqlConnection Connection `[get, set]`

  Variable to get the connection definition.

7.39.1 Detailed Description

Class that handles all SQL interaction.

7.39.2 Member Function Documentation

7.39.2.1 void SqlInteraction.SQLHandler.addParameter (SqlCommand *_command,* string *_parameterName,* object *_parameterValue,* SqlDbType *_parameterType*)

Function adds a parameter to the command.

Parameters

_command	Parameter is the command made in makeCommand
_parameter-Name	Parameter is name of the parameter defined in makeCommand
_parameter-Value	Parameter is value to be used in the command
_parameterType	Parameter is of which type the parameter is

Implements SqlInteraction.ISQLHandler.

7.39.2.2 void SqlInteraction.SQLHandler.changeConnectionparameter (string *_parameter,* string *_parameterValue*)

Changes 1 specific parameter in connection info.

Parameters

_parameter	Parameter is part of or full parameter name to change
_parameter-Value	Parameter is value to change the connection info to

Implements SqlInteraction.ISQLHandler.

7.39.2.3 SqlCommand SqlInteraction.SQLHandler.makeCommand (string *_commandText*)

Creates a command that can be used for sql interaction.

Parameters

_commandText	Parameter for the command text

Returns

Returns the SqlCommand made from the given parameters

Implements SqlInteraction.ISQLHandler.

7.39.2.4 ISQLReader SqlInteraction.SQLHandler.runQuery (SqlCommand *_command,* string *queryType*)

Executes the supplied command on SQL server.

Parameters

_command	Parameter is command to be executed
queryType	Parameter is what kind of query to execute (write/read)

Returns

Returns Null if write, if read returns a Datareader

Implements SqlInteraction.ISQLHandler.

7.39.2.5 bool SqlInteraction.SQLHandler.setConnection (string *_server,* string *_database,* string *_username,* string *_password,* string *_timeout*)

Sets a new connection to use.

Parameters

_server	Paramater for ip/domain to use
_database	Parameter for which database to use
_username	Parameter for username to use
_password	Parameter for password to use
_timeout	Parameter for what timeout should be

Returns

Returns bool for whether connection was succesfully set

Implements SqlInteraction.ISQLHandler.

7.39.3 Property Documentation

7.39.3.1 ISqlConnection SqlInteraction.SQLHandler.Connection [get, set]

Variable to get the connection definition.

Implements SqlInteraction.ISQLHandler.

7.39.3.2 ISQLHandler SqlInteraction.SQLHandler.GetInstance [static, get]

Variable to get the SQLHandler singleton instance.

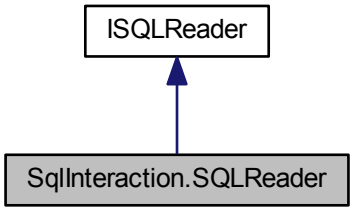
The documentation for this class was generated from the following file:

- SqlInteraction/sqlHandler.cs

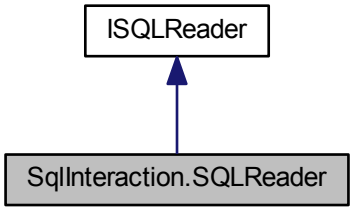
7.40 SqlInteraction.SQLReader Class Reference

Class to read table data.

Inheritance diagram for SqlInteraction.SQLReader:



Collaboration diagram for SqlInteraction.SQLReader:



Public Member Functions

- SQLReader (SqlDataReader _sqlDataReader)

  Constructor which takes a SqlDataReader used for the functions.
- List< object > readRow ()

  Read one table row.
- void close ()

  Closes the reader and its connection with the database connection.

Properties

- SqlDataReader SQLCoreReader [get, set]

  Core reader used for functions.

7.40.1 Detailed Description

Class to read table data.

7.40.2 Constructor & Destructor Documentation

7.40.2.1 SqlInteraction.SQLReader.SQLReader (SqlDataReader *_sqlDataReader*)

Constructor which takes a SqlDataReader used for the functions.

Parameters

_sqlDataReader	Data reader for the functions.

7.40.3 Member Function Documentation

7.40.3.1 void SqlInteraction.SQLReader.close ( )

Closes the reader and its connection with the database connection.

Use this or you can lock other functions from using the database connection.

Implements SqlInteraction.ISQLReader.

7.40.3.2 List<object> SqlInteraction.SQLReader.readRow ( )

Read one table row.

For each call the next row is read.

Returns

List of objects from the table. Empty list if end of table or no data in table.

Implements SqlInteraction.ISQLReader.

7.40.4 Property Documentation

7.40.4.1 SqlDataReader SqlInteraction.SQLReader.SQLCoreReader [get, set]

Core reader used for functions.

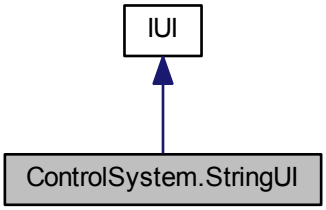
The documentation for this class was generated from the following file:

- SqlInteraction/sqlReader.cs

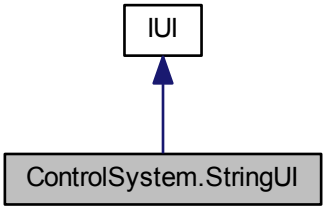
7.41 ControlSystem.StringUI Class Reference

String UI for writing to a string variable.

Inheritance diagram for ControlSystem.StringUI:



Collaboration diagram for ControlSystem.StringUI:



Public Member Functions

- StringUI ()

  Default constructor that sets up string buffer.
- void write (string _sMsg, params object[] _paramobjArgument)

  Writes the string with arguments to the UI.
- void writeLine (string _sMsg, params object[] _paramobjArgument)

  Writes the string with arguments to the UI.
- void clearString ()

  Clears the buffer.

Protected Member Functions

- void **NotifyPropertyChanged** (string name)

Properties

- string Buffer `[get]`

  Buffer used for output.

Events

- PropertyChangedEventHandler **PropertyChanged**

7.41.1 Detailed Description

String UI for writing to a string variable.

7.41.2 Constructor & Destructor Documentation

7.41.2.1 ControlSystem.StringUI.StringUI ( )

Default constructor that sets up string buffer.

7.41.3 Member Function Documentation

7.41.3.1 void ControlSystem.StringUI.clearString ( )

Clears the buffer.

7.41.3.2 void ControlSystem.StringUI.write (string *_sMsg*, params object[] *_paramobjArgument*)

Writes the string with arguments to the UI.

No newline character written.

Parameters

sMsg	The string with the message and argument placement.(Like normal Write())
paramobj-Argument	Arguments to be placed in the string.

Implements ControlSystem.IUI.

7.41.3.3 void ControlSystem.StringUI.writeLine (string *sMsg*, params object[] *paramobjArgument*)

Writes the string with arguments to the UI.

Newline character appended to end of string.

Parameters

sMsg	The string with the message and argument placement.(Like normal WriteLine())
paramobj-Argument	Arguments to be placed in the string.

Implements ControlSystem.IUI.

7.41.4 Property Documentation

7.41.4.1 string ControlSystem.StringUI.Buffer [get]

Buffer used for output.

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

- ControlSystem/ui.cs

7.42 ControlSystem.ThreadHandling Class Reference

Class to handle all threads in system, everything handled by a unique description tag that stays with a thread from moment it gets added till it gets removed.

Classes

- class ThreadHolder

  Class that holds the description of the thread and the thread itself for usage.

Public Member Functions

- ThreadHandling ()

  Constructor for class : Makes a new list for usage.
- void addThread (ThreadStart _threadStart, string _description)

  addThread function, adds a function as a thread to a list, which needs its own thread for running. This specific function handles ThreadStart parameter, which means it can only take functions with no parameters as argument
- void addThread (ParameterizedThreadStart _parameterizedThreadStart, string _description)

  Overloaded addThread function, so that it can add threads with a parameterizedThreadStart to a thread list by a description.
- void removeThread (string _description)

  Removes the specified thread from the thread list (Stops and waits for it to finish if it was running).
- void abortAndWait (string _description)

  Terminates thread with supplied description, then waits for it to finish terminating.
- void abortAllAndWait ()

  Functions aborts all threads currently running and waits for them to finish.
- void start (string _description)

  Starts the thread from supplied description if found.
- void start (string _description, object _obj)

  Starts the thread from supplied description if found but with a given parameter object.
- ThreadHolder find (string _description)

  Function that looks up saved threads by description.

7.42.1 Detailed Description

Class to handle all threads in system, everything handled by a unique description tag that stays with a thread from moment it gets added till it gets removed.

7.42.2 Constructor & Destructor Documentation

7.42.2.1 ControlSystem.ThreadHandling.ThreadHandling ( )

Constructor for class : Makes a new list for usage.

7.42.3 Member Function Documentation

7.42.3.1 void ControlSystem.ThreadHandling.abortAllAndWait ( )

Functions aborts all threads currently running and waits for them to finish.

7.42.3.2 void ControlSystem.ThreadHandling.abortAndWait (string _description)

Terminates thread with supplied description, then waits for it to finish terminating.

Parameters

_description	Description which the function needs to terminate thread for and wait for

7.42.3.3 void ControlSystem.ThreadHandling.addThread (ThreadStart _threadStart, string _description)

addThread function, adds a function as a thread to a list, which needs its own thread for running. This specific function handles ThreadStart parameter, which means it can only take functions with no parameters as argument

Parameters

_threadStart	Parameter is name of function that needs its own thread, as its Threadstart, it means it have to be a function with no parameters: (functionname(){})
_description	Description which the thread needs to be saved as (unique)

7.42.3.4 void ControlSystem.ThreadHandling.addThread (ParameterizedThreadStart _parameterizedThreadStart, string _description)

Overloaded addThread function, so that it can add threads with a parameterizedThreadStart to a thread list by a description.

Parameters

_parameterized-ThreadStart	Parameter is name of function that needs its own thread, as its paramterizedThreadStart, it means it have to be a function with parameter object: (functionname(object example){})
_description	Description which the thread needs to be saved as (unique)

7.42.3.5 ThreadHolder ControlSystem.ThreadHandling.find (string _description)

Function that looks up saved threads by description.

Parameters

_description	String that descripes the thread looking for

Returns

Returns Threadholder object if one found with description else null

7.42.3.6 void ControlSystem.ThreadHandling.removeThread (string _description)

Removes the specified thread from the thread list (Stops and waits for it to finish if it was running).

Parameters

_description	Description which the function need to find the thread that should be deleted

7.42.3.7 void ControlSystem.ThreadHandling.start (string *_description*)

Starts the thread from supplied description if found.

Parameters

_description	Description is a string which the function needs to search and find a thread for in a list

7.42.3.8 void ControlSystem.ThreadHandling.start (string *_description*, object *_obj*)

Starts the thread from supplied description if found but with a given parameter object.

Parameters

_description	Description is a string which the function needs to search and find a thread for in a list
_obj	An object that needs to be passed on to the thread as start parameter

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

- ControlSystem/threadHandling.cs

7.43 ControlSystem.ThreadHandling.ThreadHolder Class Reference

Class that holds the description of the thread and the thread itself for usage.

Properties

- string stringDescription `[get, set]`

  Variable that holds the description for a thread.
- Thread threadPlaceHolder `[get, set]`

  Variable that holds the thread instance.

7.43.1 Detailed Description

Class that holds the description of the thread and the thread itself for usage.

7.43.2 Property Documentation

7.43.2.1 string ControlSystem.ThreadHandling.ThreadHolder.stringDescription `[get, set]`

Variable that holds the description for a thread.

7.43.2.2 Thread ControlSystem.ThreadHandling.ThreadHolder.threadPlaceHolder `[get, set]`

Variable that holds the thread instance.

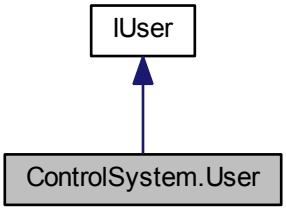
The documentation for this class was generated from the following file:

- ControlSystem/threadHandling.cs

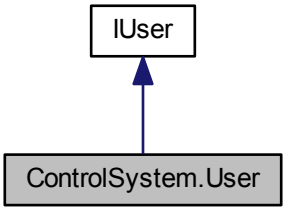
7.44 ControlSystem.User Class Reference

Normal user.

Inheritance diagram for ControlSystem.User:



Collaboration diagram for ControlSystem.User:



Public Member Functions

- User ()

  Default contructor setting up permissions.

Properties

- Dictionary< string, bool > permissionDictionary `[get]`

  Set of permissions.
- string userName `[get, set]`

  Name of the user.

7.44.1 Detailed Description

Normal user.

7.44.2 Constructor & Destructor Documentation

7.44.2.1 ControlSystem.User.User ( )

Default contructor setting up permissions.

7.44.3 Property Documentation

7.44.3.1 Dictionary<string, bool> ControlSystem.User.permissionDictionary [get]

Set of permissions.

Implements ControlSystem.IUser.

7.44.3.2 string ControlSystem.User.userName [get, set]

Name of the user.

Implements ControlSystem.IUser.

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

- ControlSystem/User.cs

7.45 ControlSystem.VecPoint Class Reference

Class to contain point for use in one of the vector classes.

Public Member Functions

- VecPoint (int _iX, int _iY, int _iZ, int _iPitch, int _iRoll)
  Constructor setting up the point location.
- override string ToString ()
  The position in string format.

Public Attributes

- int **iX**
- int **iY**
- int **iZ**
- int **iPitch**
- int **iRoll**

7.45.1 Detailed Description

Class to contain point for use in one of the vector classes.

7.45.2 Constructor & Destructor Documentation

7.45.2.1 ControlSystem.VecPoint.VecPoint (int *_iX,* int *_iY,* int *_iZ,* int *_iPitch,* int *_iRoll*)

Constructor setting up the point location.

Parameters

_iX	X coordinate.
_iY	Y coordinate.
_iZ	Z coordinate.
_iPitch	Pitch.
_iRoll	Roll.

7.45.3 Member Function Documentation

7.45.3.1 override string ControlSystem.VecPoint.ToString ( )

The position in string format.

Returns

The position.("(x,y,z,pitch,roll)")

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

- ControlSystem/wrapper.cs

7.46 RoboGO.ViewModels.ViewModelManualSteering Class Reference

ViewModel for GUIManualSteering.

Public Member Functions

- ViewModelManualSteering ()
  Default constructor.
- void moveAxisBaseRight ()
  Moves the base of the robot to the right.
- void moveAxisBaseLeft ()
  Moves the base of the robot to the left.
- void moveAxisShoulderRight ()
  Moves the shoulder of the robot to the right.
- void moveAxisShoulderLeft ()
  Moves the shoulder of the robot to the left.
- void moveAxisElbowRight ()
  Moves the elbow of the robot to the right.
- void moveAxisElbowLeft ()
  Moves the elbow of the robot to the left.
- void openGripper ()
  Opens the gripper.
- void closeGripper ()
  Closes the gripper.
- void moveAxisPitchUp ()
  Moves the pitch of the robot hand up.
- void moveAxisPitchDown ()
  Moves the pitch of the robot hand down.
- void moveAxisRollRight ()
  Rolls the robot hand to the right.

- void moveAxisRollLeft ()

  Rolls the robot hand to the left.
- void moveAxisConveyerRight ()

  Moves the conveyer to the right.
- void moveAxisConveyerLeft ()

  Moves the conveyer to the left.
- void moveCoordXIncreasing ()

  Moves the robot hand increasing in the X-axis.
- void moveCoordXDecreasing ()

  Moves the robot hand decreasing in the X-axis.
- void moveCoordYIncreasing ()

  Moves the robot hand increasing in the Y-axis.
- void moveCoordYDecreasing ()

  Moves the robot hand decreasing in the Y-axis.
- void moveCoordZIncreasing ()

  Moves the robot hand increasing the Z-axis.
- void moveCoordZDecreasing ()

  Moves the robot hand decreasing in the Z-axis.
- void moveCoordPitchIncreasing ()

  Increase the pitch of the robot hand while keeping jaw fixed in position.
- void moveCoordPitchDecreasing ()

  Decrease the pitch of the robot hand while keeping jaw fixed in position.
- void moveCoordRollIncreasing ()

  Roll the robot hand.
- void moveCoordRollDecreasing ()

  Roll the robot hand.
- void seekHome ()

  The robot begins seeking home for all axes.
- void stopMovement ()

  Stops all movement of the robot.

Properties

- IManualController ManualControl `[get, set]`

  Manual controller used for the functions.
- int Speed `[get, set]`

  Speed of the movements.
- ICommand OpenGripper `[get]`

  Open the gripper.
- ICommand CloseGripper `[get]`

  Close the gripper.
- ICommand SeekHome `[get]`

  Home the robot.

7.46.1 Detailed Description

ViewModel for GUIManualSteering.

Todo Way to inform View about errors, like not being connected to robot.(Example messaging.)

7.46.2 Constructor & Destructor Documentation

7.46.2.1 RoboGO.ViewModels.ViewModelManualSteering.ViewModelManualSteering ( )

Default constructor.

Warning

Must be checked up later.

7.46.3 Member Function Documentation

7.46.3.1 void RoboGO.ViewModels.ViewModelManualSteering.closeGripper ( )

Closes the gripper.

7.46.3.2 void RoboGO.ViewModels.ViewModelManualSteering.moveAxisBaseLeft ( )

Moves the base of the robot to the left.

7.46.3.3 void RoboGO.ViewModels.ViewModelManualSteering.moveAxisBaseRight ( )

Moves the base of the robot to the right.

7.46.3.4 void RoboGO.ViewModels.ViewModelManualSteering.moveAxisConveyerLeft ( )

Moves the conveyer to the left.

7.46.3.5 void RoboGO.ViewModels.ViewModelManualSteering.moveAxisConveyerRight ( )

Moves the conveyer to the right.

7.46.3.6 void RoboGO.ViewModels.ViewModelManualSteering.moveAxisElbowLeft ( )

Moves the elbow of the robot to the left.

7.46.3.7 void RoboGO.ViewModels.ViewModelManualSteering.moveAxisElbowRight ( )

Moves the elbow of the robot to the right.

7.46.3.8 void RoboGO.ViewModels.ViewModelManualSteering.moveAxisPitchDown ( )

Moves the pitch of the robot hand down.

7.46.3.9 void RoboGO.ViewModels.ViewModelManualSteering.moveAxisPitchUp ( )

Moves the pitch of the robot hand up.

7.46.3.10 void RoboGO.ViewModels.ViewModelManualSteering.moveAxisRollLeft ( )

Rolls the robot hand to the left.

7.46.3.11 void RoboGO.ViewModels.ViewModelManualSteering.moveAxisRollRight ( )

Rolls the robot hand to the right.

7.46.3.12 void RoboGO.ViewModels.ViewModelManualSteering.moveAxisShoulderLeft ( )

Moves the shoulder of the robot to the left.

7.46.3.13 void RoboGO.ViewModels.ViewModelManualSteering.moveAxisShoulderRight ( )

Moves the shoulder of the robot to the right.

7.46.3.14 void RoboGO.ViewModels.ViewModelManualSteering.moveCoordPitchDecreasing ( )

Decrease the pitch of the robot hand while keeping jaw fixed in position.

7.46.3.15 void RoboGO.ViewModels.ViewModelManualSteering.moveCoordPitchIncreasing ( )

Increase the pitch of the robot hand while keeping jaw fixed in position.

7.46.3.16 void RoboGO.ViewModels.ViewModelManualSteering.moveCoordRollDecreasing ( )

Roll the robot hand.

7.46.3.17 void RoboGO.ViewModels.ViewModelManualSteering.moveCoordRollIncreasing ( )

Roll the robot hand.

7.46.3.18 void RoboGO.ViewModels.ViewModelManualSteering.moveCoordXDecreasing ( )

Moves the robot hand decreasing in the X-axis.

7.46.3.19 void RoboGO.ViewModels.ViewModelManualSteering.moveCoordXIncreasing ( )

Moves the robot hand increasing in the X-axis.

7.46.3.20 void RoboGO.ViewModels.ViewModelManualSteering.moveCoordYDecreasing ( )

Moves the robot hand decreasing in the Y-axis.

7.46.3.21 void RoboGO.ViewModels.ViewModelManualSteering.moveCoordYIncreasing ( )

Moves the robot hand increasing in the Y-axis.

7.46.3.22 void RoboGO.ViewModels.ViewModelManualSteering.moveCoordZDecreasing ( )

Moves the robot hand decreasing in the Z-axis.

7.46.3.23 void RoboGO.ViewModels.ViewModelManualSteering.moveCoordZIncreasing ( )

Moves the robot hand increasing the Z-axis.

7.46.3.24 void RoboGO.ViewModels.ViewModelManualSteering.openGripper ( )

Opens the gripper.

7.46.3.25 void RoboGO.ViewModels.ViewModelManualSteering.seekHome ( )

The robot begins seeking home for all axes.

7.46.3.26 void RoboGO.ViewModels.ViewModelManualSteering.stopMovement ( )

Stops all movement of the robot.

7.46.4 Property Documentation

7.46.4.1 ICommand RoboGO.ViewModels.ViewModelManualSteering.CloseGripper [get]

Close the gripper.

7.46.4.2 IManualController RoboGO.ViewModels.ViewModelManualSteering.ManualControl [get, set]

Manual controller used for the functions.

7.46.4.3 ICommand RoboGO.ViewModels.ViewModelManualSteering.OpenGripper [get]

Open the gripper.

7.46.4.4 ICommand RoboGO.ViewModels.ViewModelManualSteering.SeekHome [get]

Home the robot.

7.46.4.5 int RoboGO.ViewModels.ViewModelManualSteering.Speed [get, set]

Speed of the movements.

Value 0->100 is equal to percentage of max speed.

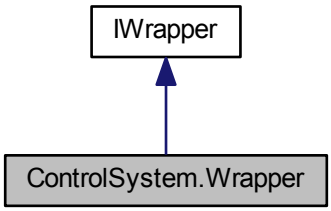
The documentation for this class was generated from the following file:

- RoboGO/ViewModels/viewModelManualSteering.cs

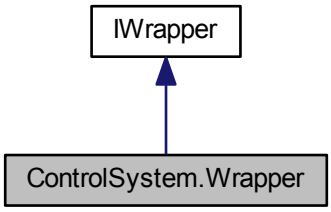
7.47 ControlSystem.Wrapper Class Reference

Contains a wrapper for the C++ functions in the dll file(USBC.dll).

Inheritance diagram for ControlSystem.Wrapper:



Collaboration diagram for ControlSystem.Wrapper:



Public Types

- enum enumSystemModes

  For mode in initialization.
- enum enumSystemTypes

  For type in initialization.
- enum enumAxisSettings

  For chosing axis group in certain functions.
- enum enumManualType

  For chosing type of movement when enabling manual movement.
- enum enumManualModeWhat

  For chosing what part to move when moving manually.
- enum enumBGroup

  For chosing what part to move when setting Time or Speed.

Public Member Functions

- bool initializationWrapped (enumSystemModes _sysmodeMode, enumSystemTypes _systypeType, DLL.DgateCallBack _funcptrSuccess, DLL.DgateCallBack _funcptrError)

  Initializes the robot.

- bool controlWrapped (enumAxisSettings _axisSettingsGroup, bool _bControlOnOrOff)

  Turns control on and off for certain axis group.
- bool isOnlineOkWrapped ()

  Tells about the robot being online.
- bool timeWrapped (enumBGroup _bGroup, long _mTime)

  Sets the time future movement should take.
- bool speedWrapped (enumBGroup _bGroup, long _mSpeed)

  Sets the speed future movement should take.
- bool homeWrapped (enumAxisSettings _axisSettingsGroup, DLL.DgateCallBackByteRefArg _funcptrHomingEventHandler)

  Homes a axis group. Should be called before calling most movement functions.
- bool enterManualWrapped (enumManualType _enummanMoveType)

  Must be called to use manual movement. Seems to stop previous movement of any object(Axis) that was moving before.
- bool closeManualWrapped ()

  Stops manual mode.
- bool moveManualWrapped (enumManualModeWhat _enumWhatToMove, int _lSpeed)

  Moves the robot. homeWrapped must have been called if moving by coordinates. enterManual seems have to be called before each call to this function. Use stopWrapped to stop motion afterwards.(Moving some other part of the system also stops the previous movement, since the system can only handle one object(Axis) moving at a time.)
- bool stopWrapped (enumAxisSettings _bWhatToStop)
- bool moveLinearWrapped (string _sNameOfVector, int _iIndex)

  Moves the robot in a linear motion.
- bool openGripperWrapped ()

  Opens the gripper.
- bool closeGripperWrapped ()

  Closes the gripper.
- bool getJawWrapped (ref short _shrtPerc, ref short _shrtWidth)

  Gives information about how much open the gripper is.(Between the 'fingers')
- void watchMotionWrapped (DLL.DgateCallBackCharArg _funcptrCallbackEnd, DLL.DgateCallBackCharArg _funcptrCallbackStart)

  Adds functions to be called when motion starts and motion ends.
- bool watchDigitalInputWrapped (DLL.DgateCallBackLongArg _funcptrCallbackEvent)

  Adds a function to be called when digital input changes.
- bool closeWatchDigitalInputWrapped ()

  Stops watching of digital inputs.
- bool defineVectorWrapped (enumAxisSettings _enumGroup, string _sVectorName, short _shrtLength)

  Defines a new vector in robot memory.
- bool teachWrapped (SIRVector vecTheSirVector)

  Add the vector points to the vector with the same name.
- VecPoint getCurrentPosition ()

  Returns the position of the robot.

Static Public Member Functions

- static Wrapper getInstance ()

  Gets the wrapper.

Properties

- IDLL DLL `[get, set]`

  Dll used for functions.

7.47.1 Detailed Description

Contains a wrapper for the C++ functions in the dll file(USBC.dll).

Good idea to check USBC-documentation.pdf.

Notes: Uses IntPtr arg for different types of C++ pointers. Same function names as C++ but has "Wrapped" at the end. Try to have handlers in delegates which in entire use of wrapper, so the memory for the handler doesn´t get removed by GC.

Todo Add behind factory class.

7.47.2 Member Enumeration Documentation

7.47.2.1 enum ControlSystem.Wrapper.enumAxisSettings

For chosing axis group in certain functions.

7.47.2.2 enum ControlSystem.Wrapper.enumBGroup

For chosing what part to move when setting Time or Speed.

7.47.2.3 enum ControlSystem.Wrapper.enumManualModeWhat

For chosing what part to move when moving manually.

Note: Some used for moving by axes and some used for moving by coordinates.

7.47.2.4 enum ControlSystem.Wrapper.enumManualType

For chosing type of movement when enabling manual movement.

7.47.2.5 enum ControlSystem.Wrapper.enumSystemModes

For mode in initialization.

(MODE_ONLINE is normally used)

7.47.2.6 enum ControlSystem.Wrapper.enumSystemTypes

For type in initialization.

(SYSTEM_TYPE_DEFAULT normally used)

7.47.3 Member Function Documentation

7.47.3.1 bool ControlSystem.Wrapper.closeGripperWrapped ( )

Closes the gripper.

Returns

Returns true on successful call.

Implements ControlSystem.IWrapper.

7.47.3.2 bool ControlSystem.Wrapper.closeManualWrapped ( )

Stops manual mode.

Returns

Returns true on successful call.

Implements ControlSystem.IWrapper.

7.47.3.3 bool ControlSystem.Wrapper.closeWatchDigitalInputWrapped ( )

Stops watching of digital inputs.

Note: Probably means no more events.

Returns

Returns true if successful call.

Implements ControlSystem.IWrapper.

7.47.3.4 bool ControlSystem.Wrapper.controlWrapped (enumAxisSettings *_axisSettingsGroup,* bool *_bControlOnOrOff*)

Turns control on and off for certain axis group.

Parameters

_axisSettings-Group	Axis group to affect.(Use enum)
_bControlOnOr-Off	To have it turned off or on.

Returns

Returns true on successful call.

Implements ControlSystem.IWrapper.

7.47.3.5 bool ControlSystem.Wrapper.defineVectorWrapped (enumAxisSettings *_enumGroup,* string *_sVectorName,* short *_shrtLength*)

Defines a new vector in robot memory.

Note: Good idea to have in program one of the SIRVector classes to contains vector information.

Parameters

_enumGroup	Group can use: Robot(Normally used) Peripherals All
_sVectorName	Name of vector.
_shrtLength	Length of vector.(Number of points.)

Returns

Returns true on successfull call.

Implements ControlSystem.IWrapper.

7.47.3.6 bool ControlSystem.Wrapper.enterManualWrapped (enumManualType *_enummanMoveType*)

Must be called to use manual movement. Seems to stop previous movement of any object(Axis) that was moving before.

Parameters

_enummanMoveType	What to move by.(Axis(0), Coordinates(1))

Returns

Returns true on successful call.

Implements ControlSystem.IWrapper.

7.47.3.7 VecPoint ControlSystem.Wrapper.getCurrentPosition ( )

Returns the position of the robot.

Warning

Ignoring wrapper int return value.
Not sure about buffer type to use in impl.

Returns

Returns current position.

Implements ControlSystem.IWrapper.

7.47.3.8 static Wrapper ControlSystem.Wrapper.getInstance ( ) [static]

Gets the wrapper.

Returns

The wrapper.

7.47.3.9 bool ControlSystem.Wrapper.getJawWrapped (ref short *_shrtPerc,* ref short *_shrtWidth*)

Gives information about how much open the gripper is.(Between the 'fingers')

Note: Probably most useful to use the _shrtWidth arg.

Parameters

_shrtPerc	Data in percentage.
_shrtWidth	Data in width.(mm)

Returns

Returns true on successful call.

Implements ControlSystem.IWrapper.

7.47.3.10 bool ControlSystem.Wrapper.homeWrapped (enumAxisSettings *_axisSettingsGroup,* DLL.DgateCallBackByteRefArg *_funcptrHomingEventHandler*)

Homes a axis group. Should be called before calling most movement functions.

Parameters

_axisSettingsGroup	The axis group.(Use enum)
_funcptrHomingEventHandler	Function to be called for homing events.

Values being passed in event: 0xff: Homing started 1 - 8: Axis n being homed. 0x40: Homing ended.

Returns

Returns true on successful call.

Implements ControlSystem.IWrapper.

7.47.3.11 bool ControlSystem.Wrapper.initializationWrapped (enumSystemModes *_sysmodeMode,* enumSystemTypes *_systypeType,* DLL.DgateCallBack *_funcptrSuccess,* DLL.DgateCallBack *_funcptrError*)

Initializes the robot.

Note: Should wait for it to be done before calling other functions.

Todo Refactor delegate to contain ConfigData and ErrorInfo if found necessary.

Parameters

_sysmodeMode	Mode.[Normally use online mode]
_systypeType	Type of connection.[Normally use default]
_funcptrSuccess	Function to be called on success.
_funcptrError	Function to be called on error.

Returns

Returns true on successful call.(But errors can still happen)

Implements ControlSystem.IWrapper.

7.47.3.12 bool ControlSystem.Wrapper.isOnlineOkWrapped ( )

Tells about the robot being online.

Returns

Returns true if it is, false otherwise.

Implements ControlSystem.IWrapper.

7.47.3.13 bool ControlSystem.Wrapper.moveLinearWrapped (string *_sNameOfVector,* int *_iIndex*)

Moves the robot in a linear motion.

Warning

Not using pos 2 in wrapped dll function.
Seems to be unfunctional.

Parameters

_sNameOf-Vector	Name of the vector in the Robot.
_iIndex	Index for point

Returns

Returns true on successfull call.

Implements ControlSystem.IWrapper.

7.47.3.14 bool ControlSystem.Wrapper.moveManualWrapped (enumManualModeWhat *_enumWhatToMove,* int *_lSpeed*)

Moves the robot. homeWrapped must have been called if moving by coordinates. enterManual seems have to be called before each call to this function. Use stopWrapped to stop motion afterwards.(Moving some other part of the system also stops the previous movement, since the system can only handle one object(Axis) moving at a time.)

Implements ControlSystem.IWrapper.

7.47.3.15 bool ControlSystem.Wrapper.openGripperWrapped ( )

Opens the gripper.

Returns

Returns true on successful call.

Implements ControlSystem.IWrapper.

7.47.3.16 bool ControlSystem.Wrapper.speedWrapped (enumBGroup *_bGroup,* long *_mSpeed*)

Sets the speed future movement should take.

Parameters

_bGroup	bool ucGroup Axis group to which the time should be applied '&' for all axes '0'-'7' for axis movements 'A' for robot movements 'B' for peripheral movements 'G' for gripper movements
_mSpeed	Speed in percent of max speed

Returns

Returns true if the speed has been succesfully set, false otherwise..

Implements ControlSystem.IWrapper.

7.47.3.17 bool ControlSystem.Wrapper.stopWrapped (enumAxisSettings *_bWhatToStop*)

Todo Refactor.

Implements ControlSystem.IWrapper.

7.47.3.18 bool ControlSystem.Wrapper.teachWrapped (SIRVector *vecTheSirVector*)

Add the vector points to the vector with the same name.

Note: Should call 'defineVectorWrapped' first.

Parameters

vecTheSirVector	The vector with the points.

Returns

Returns true on succeessfull call.

Implements ControlSystem.IWrapper.

7.47.3.19 bool ControlSystem.Wrapper.timeWrapped (enumBGroup *_bGroup,* long *_mTime*)

Sets the time future movement should take.

Parameters

_bGroup	bool ucGroup Axis group to which the time should be applied '&' for all axes '0'-'7' for axis movements 'A' for robot movements 'B' for peripheral movements 'G' for gripper movements
_mTime	Time in milliseconds

Returns

Returns true if time has been succesfully set, false otherwise.

Implements ControlSystem.IWrapper.

7.47.3.20 bool ControlSystem.Wrapper.watchDigitalInputWrapped (DLL.DgateCallBackLongArg *_funcptrCallbackEvent*)

Adds a function to be called when digital input changes.

Parameters

_funcptr-CallbackEvent	The function to be called.

Returns

Returns true if successful call.

Implements ControlSystem.IWrapper.

7.47.3.21 void ControlSystem.Wrapper.watchMotionWrapped (DLL.DgateCallBackCharArg *_funcptrCallbackEnd,* DLL.DgateCallBackCharArg *_funcptrCallbackStart*)

Adds functions to be called when motion starts and motion ends.

Note: Ignoring return value.

Parameters

_funcptr-CallbackEnd	Function to be called when motion has ended.
_funcptr-CallbackStart	Function to be called when motion has started.

Implements ControlSystem.IWrapper.

7.47.4 Property Documentation

7.47.4.1 IDLL ControlSystem.Wrapper.DLL [get, set]

Dll used for functions.

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

- ControlSystem/wrapper.cs

7.48 RoboGO.ViewModels.XYCalculate Class Reference

Class for calculating position of robot.

Public Member Functions

- XYCalculate (VecPoint p)
  Constructor taking a VecPoint that it uses for calculting position.
- void elbow ()
  Calculate elbow position.
- void gripper ()
  Calculate gripper position.

Public Attributes

- double elbowRotate
  Rotation of the elbow.
- double gripperX
  Gripper x position.
- double gripperY
  Gripper y position.

7.48.1 Detailed Description

Class for calculating position of robot.

Note: Used by simulator.

7.48.2 Constructor & Destructor Documentation

7.48.2.1 RoboGO.ViewModels.XYCalculate.XYCalculate (VecPoint *p*)

Constructor taking a VecPoint that it uses for calculting position.

Parameters

p	

7.48.3 Member Function Documentation

7.48.3.1 void RoboGO.ViewModels.XYCalculate.elbow ( )

Calculate elbow position.

7.48.3.2 void RoboGO.ViewModels.XYCalculate.gripper ( )

Calculate gripper position.

7.48.4 Member Data Documentation

7.48.4.1 double RoboGO.ViewModels.XYCalculate.elbowRotate

Rotation of the elbow.

7.48.4.2 double RoboGO.ViewModels.XYCalculate.gripperX

Gripper x position.

7.48.4.3 double RoboGO.ViewModels.XYCalculate.gripperY

Gripper y position.

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

- RoboGO/ViewModels/simulatorViewModel.cs

Chapter 8

File Documentation

8.1 ControlSystem/dll.cs File Reference

Classes

- class ControlSystem.DLL

  Class that with the IDLL interface calls the actual functions from the USBC.dll.
- class ControlSystem.DLLImport

  Class providing interface for USBC.dll functions.

Packages

- package ControlSystem

  Class to handle system threads.

8.1.1 Detailed Description

8.2 ControlSystem/errorReporter.cs File Reference

Classes

- class ControlSystem.ErrorReporter

  Temporary error reporting class.

Packages

- package ControlSystem

  Class to handle system threads.

8.2.1 Detailed Description

8.3 ControlSystem/factory.cs File Reference

Classes

- class **ControlSystem.Factory**

Manages all global class's as a singleton and factory.

Packages

- package ControlSystem

  Class to handle system threads.

8.3.1 Detailed Description

8.4 ControlSystem/iDLL.cs File Reference

Classes

- interface ControlSystem.IDLL

  Interface for the functions in the USBC.dll files, in C# format.

Packages

- package ControlSystem

  Class to handle system threads.

8.4.1 Detailed Description

8.5 ControlSystem/iWrapper.cs File Reference

Classes

- interface ControlSystem.IWrapper

  Interface for a wrapper wrapping the USBC.dll file.

Packages

- package ControlSystem

  Class to handle system threads.

8.5.1 Detailed Description

8.6 ControlSystem/manualController.cs File Reference

Classes

- interface ControlSystem.IManualController

  Interface for what manual movement functions there should be available.
- class ControlSystem.ManualController

  Class that encapsulates controlling manual movement. Instead of using directly Robot or Simulator by interface.

Packages

- package ControlSystem

  Class to handle system threads.

Enumerations

- enum ControlSystem.enumLeftRight

  What direction to move in when moving by axes.
- enum ControlSystem.enumUpDown

  What direction to move in when moving by axes.(Wrist)
- enum ControlSystem.enumIncDec

  Move increasing or decreasing when moving by coordinates.
- enum ControlSystem.enumCloseOpen

  To close or open gripper.

8.6.1 Detailed Description

8.7 ControlSystem/scriptRunner.cs File Reference

Classes

- interface ControlSystem.IScriptRunner

  Interface for a script runner.
- class ControlSystem.ScriptRunner

  Used to run IronPython scripts.

Packages

- package ControlSystem

  Class to handle system threads.

8.7.1 Detailed Description

8.8 ControlSystem/serialSTK.cs File Reference

Classes

- class ControlSystem.SerialSTK

  Class for functions to communicating with the STK kit.(Serial communication.)

Packages

- package ControlSystem

  Class to handle system threads.

8.8.1 Detailed Description

8.9 ControlSystem/simulator.cs File Reference

Classes

- class ControlSystem.Simulator

  IRobot implementation using IUI output interface for simulating robot behavior.

Packages

- package ControlSystem

  Class to handle system threads.

8.9.1 Detailed Description

8.10 ControlSystem/threadHandling.cs File Reference

Classes

- class ControlSystem.ThreadHandling

  Class to handle all threads in system, everything handled by a unique description tag that stays with a thread from moment it gets added till it gets removed.
- class ControlSystem.ThreadHandling.ThreadHolder

  Class that holds the description of the thread and the thread itself for usage.

Packages

- package ControlSystem

  Class to handle system threads.

8.10.1 Detailed Description

8.11 ControlSystem/ui.cs File Reference

Classes

- interface ControlSystem.IUI

  Simple interface for UI interaction.
- class ControlSystem.ConsoleUI

  Console UI for writing to the console output.
- class ControlSystem.StringUI

  String UI for writing to a string variable.

Packages

- package ControlSystem

  Class to handle system threads.

8.11.1 Detailed Description

8.12 ControlSystem/wrapper.cs File Reference

Classes

- class ControlSystem.VecPoint

  Class to contain point for use in one of the vector classes.
- class ControlSystem.SIRVector

  Base class for vector used in wrapper.
- class ControlSystem.AbsCoordSirVector

  SIRVector class for absolute positions.
- class ControlSystem.RelCoordSirVector

  SIRVector class for relative positions.
- class ControlSystem.Wrapper

  Contains a wrapper for the C++ functions in the dll file(USBC.dll).

Packages

- package ControlSystem

  Class to handle system threads.

8.12.1 Detailed Description

8.13 RoboGO/ViewModels/delegateCommand.cs File Reference

Classes

- class RoboGO.ViewModels.DelegateCommand

  Command class for executing one function.

Packages

- package RoboGO.ViewModels

8.13.1 Detailed Description

8.14 RoboGO/ViewModels/ideViewModel.cs File Reference

Classes

- class RoboGO.ViewModels.IDEViewModel

  ViewModel between IDEView and ScriptRunner.

Packages

- package RoboGO.ViewModels

8.14.1 Detailed Description

8.15 RoboGO/ViewModels/infoViewModel.cs File Reference

Classes

- class RoboGO.ViewModels.InfoViewModel

  ViewModel for the tables.(From the database.)

Packages

- package RoboGO.ViewModels

8.15.1 Detailed Description

8.16 RoboGO/ViewModels/passwordWindowViewModel.cs File Reference

Classes

- class RoboGO.ViewModels.passwordWindowViewModel

  ViewModel for password window.

Packages

- package RoboGO.ViewModels

8.16.1 Detailed Description

8.17 RoboGO/ViewModels/positionModel.cs File Reference

Classes

- class RoboGO.ViewModels.PositionModel

  Class to keep track of simulator position.

Packages

- package RoboGO.ViewModels

8.17.1 Detailed Description

8.18 RoboGO/ViewModels/positionViewModel.cs File Reference

Classes

- class RoboGO.ViewModels.PositionViewModel

  ViewModel for the position class.

Packages

- package RoboGO.ViewModels

8.18.1 Detailed Description

8.19 RoboGO/ViewModels/simulatorViewModel.cs File Reference

Classes

- class RoboGO.ViewModels.SimulatorViewModel

  ViewModel for the simulator class.
- class RoboGO.ViewModels.XYCalculate

  Class for calculating position of robot.

Packages

- package RoboGO.ViewModels

8.19.1 Detailed Description

8.20 RoboGO/ViewModels/viewModelManualSteering.cs File Reference

Classes

- class RoboGO.ViewModels.ViewModelManualSteering

  ViewModel for GUIManualSteering.

Packages

- package RoboGO.ViewModels

8.20.1 Detailed Description

8.21 SqlInteraction/iSQLHandler.cs File Reference

Classes

- interface SqlInteraction.ISQLHandler

  Interface for handling class of sql.

Packages

- package SqlInteraction

8.21.1 Detailed Description

8.22 SqlInteraction/sqlReader.cs File Reference

Classes

- class SqlInteraction.SQLReader

  Class to read table data.

Packages

- package SqlInteraction

8.22.1 Detailed Description

Index