

SIR

0.3

Generated by Doxygen 1.8.0

Thu May 24 2012 16:37:41



# Contents

<b>1</b>	<b>Todo List</b>	<b>1</b>
<b>2</b>	<b>Namespace Index</b>	<b>3</b>
2.1	Packages	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class Hierarchy	5
<b>4</b>	<b>Class Index</b>	<b>7</b>
4.1	Class List	7
<b>5</b>	<b>File Index</b>	<b>11</b>
5.1	File List	11
<b>6</b>	<b>Namespace Documentation</b>	<b>13</b>
6.1	Package ControlSystem	13
6.1.1	Detailed Description	14
6.1.2	Enumeration Type Documentation	14
6.1.2.1	enumCloseOpen	14
6.1.2.2	enumIncDec	15
6.1.2.3	enumLeftRight	15
6.1.2.4	enumUpDown	15
6.2	Package RoboGO	15
6.3	Package RoboGO.Properties	15
6.4	Package RoboGO.ViewModels	16
6.5	Package SqlInteraction	16
6.6	Package XamlGeneratedNamespace	16
<b>7</b>	<b>Class Documentation</b>	<b>17</b>
7.1	RoboGO.aboutBox Class Reference	17
7.1.1	Detailed Description	17
7.1.2	Member Function Documentation	17
7.1.2.1	InitializeComponent	17
7.1.2.2	InitializeComponent	18

7.2	ControlSystem.AbsCoordSirVector Class Reference	18
7.2.1	Detailed Description	18
7.2.2	Constructor & Destructor Documentation	19
7.2.2.1	AbsCoordSirVector	19
7.3	ControlSystem.Admin Class Reference	19
7.3.1	Detailed Description	20
7.3.2	Constructor & Destructor Documentation	20
7.3.2.1	Admin	20
7.3.3	Property Documentation	20
7.3.3.1	permissionDictionary	20
7.3.3.2	userName	20
7.4	RoboGO.App Class Reference	20
7.4.1	Detailed Description	21
7.4.2	Member Function Documentation	21
7.4.2.1	InitializeComponent	21
7.4.2.2	InitializeComponent	21
7.4.2.3	Main	21
7.4.2.4	Main	21
7.5	RoboGO.CommandsWindow Class Reference	21
7.5.1	Detailed Description	22
7.5.2	Member Function Documentation	22
7.5.2.1	InitializeComponent	22
7.5.2.2	InitializeComponent	22
7.6	ControlSystem.ConsoleUI Class Reference	22
7.6.1	Detailed Description	23
7.6.2	Member Function Documentation	23
7.6.2.1	write	23
7.6.2.2	writeLine	23
7.7	RoboGO.ViewModels.DelegateCommand Class Reference	24
7.7.1	Detailed Description	24
7.7.2	Constructor & Destructor Documentation	24
7.7.2.1	DelegateCommand	24
7.7.3	Member Function Documentation	24
7.7.3.1	CanExecute	24
7.7.3.2	Execute	25
7.7.4	Event Documentation	25
7.7.4.1	CanExecuteChanged	25
7.8	ControlSystem.DLL Class Reference	25
7.8.1	Detailed Description	27
7.8.2	Member Function Documentation	27

7.8.2.1	CloseGripper	27
7.8.2.2	CloseManual	27
7.8.2.3	CloseWatchDigitalInput	27
7.8.2.4	Control	28
7.8.2.5	DefineVector	28
7.8.2.6	DgateCallBackByteRefArg	28
7.8.2.7	EnterManual	28
7.8.2.8	GetCurrentPosition	29
7.8.2.9	GetJaw	29
7.8.2.10	IsOnLineOk	29
7.8.2.11	MoveLinear	29
7.8.2.12	MoveManual	30
7.8.2.13	OpenGripper	30
7.8.2.14	Speed	30
7.8.2.15	Stop	30
7.8.2.16	Teach	31
7.8.2.17	Time	31
7.9	ControlSystem.DLLImport Class Reference	31
7.9.1	Detailed Description	32
7.9.2	Member Function Documentation	32
7.9.2.1	CloseGripper	32
7.9.2.2	CloseManual	33
7.9.2.3	CloseWatchDigitalInput	33
7.9.2.4	Control	33
7.9.2.5	DefineVector	33
7.9.2.6	EnterManual	33
7.9.2.7	GetCurrentPosition	34
7.9.2.8	GetJaw	34
7.9.2.9	Home	34
7.9.2.10	initialization	34
7.9.2.11	IsOnLineOk	35
7.9.2.12	MoveLinear	35
7.9.2.13	MoveManual	35
7.9.2.14	OpenGripper	36
7.9.2.15	Speed	36
7.9.2.16	Stop	36
7.9.2.17	Teach	36
7.9.2.18	Time	36
7.9.2.19	WatchDigitalInput	37
7.9.2.20	WatchMotion	37

7.10	ControlSystem.ErrorReporter Class Reference	37
7.10.1	Detailed Description	38
7.10.2	Member Function Documentation	38
7.10.2.1	ErrorReported	38
7.11	XamlGeneratedNamespace.GeneratedInternalTypeHelper Class Reference	38
7.11.1	Detailed Description	39
7.11.2	Member Function Documentation	39
7.11.2.1	AddEventHandler	39
7.11.2.2	AddEventHandler	39
7.11.2.3	CreateDelegate	39
7.11.2.4	CreateDelegate	39
7.11.2.5	CreateInstance	39
7.11.2.6	CreateInstance	39
7.11.2.7	GetPropertyValue	40
7.11.2.8	GetPropertyValue	40
7.11.2.9	SetPropertyValue	40
7.11.2.10	SetPropertyValue	40
7.12	RoboGO.GUIManualSteering Class Reference	40
7.12.1	Detailed Description	41
7.12.2	Member Function Documentation	41
7.12.2.1	InitializeComponent	41
7.12.2.2	InitializeComponent	41
7.13	RoboGO.ViewModels.IDEViewModel Class Reference	42
7.13.1	Detailed Description	42
7.13.2	Constructor & Destructor Documentation	43
7.13.2.1	IDEViewModel	43
7.13.3	Member Function Documentation	43
7.13.3.1	CodeClear	43
7.13.3.2	executeCode	43
7.13.4	Property Documentation	43
7.13.4.1	build	43
7.13.4.2	closeTab	43
7.13.4.3	closeTab_CanExecute	43
7.13.4.4	CodeOutput	43
7.13.4.5	ExecuteComd	43
7.13.4.6	IdeTabs	43
7.13.4.7	newTab	44
7.13.4.8	newTab_CanExecute	44
7.13.4.9	open	44
7.13.4.10	open_CanExecute	44

7.13.4.11 saveAs . . . . .	44
7.13.4.12 saveAs_CanExecute . . . . .	44
7.13.4.13 ScriptExecuter . . . . .	44
7.13.5 Event Documentation . . . . .	44
7.13.5.1 PropertyChanged . . . . .	44
7.14 ControlSystem.IDLL Interface Reference . . . . .	45
7.14.1 Detailed Description . . . . .	46
7.14.2 Member Function Documentation . . . . .	46
7.14.2.1 CloseGripper . . . . .	46
7.14.2.2 CloseManual . . . . .	46
7.14.2.3 CloseWatchDigitalInput . . . . .	47
7.14.2.4 Control . . . . .	47
7.14.2.5 DefineVector . . . . .	47
7.14.2.6 EnterManual . . . . .	47
7.14.2.7 GetCurrentPosition . . . . .	48
7.14.2.8 GetJaw . . . . .	48
7.14.2.9 Home . . . . .	48
7.14.2.10 Initialization . . . . .	48
7.14.2.11 IsOnLineOk . . . . .	49
7.14.2.12 MoveLinear . . . . .	49
7.14.2.13 MoveManual . . . . .	49
7.14.2.14 OpenGripper . . . . .	50
7.14.2.15 Speed . . . . .	50
7.14.2.16 Stop . . . . .	50
7.14.2.17 Teach . . . . .	50
7.14.2.18 Time . . . . .	51
7.14.2.19 WatchDigitalInput . . . . .	51
7.14.2.20 WatchMotion . . . . .	51
7.15 ControlSystem.IManualController Interface Reference . . . . .	51
7.15.1 Detailed Description . . . . .	53
7.15.2 Member Function Documentation . . . . .	53
7.15.2.1 moveAxisBase . . . . .	53
7.15.2.2 moveAxisConveyer . . . . .	53
7.15.2.3 moveAxisElbow . . . . .	53
7.15.2.4 moveAxisGripper . . . . .	53
7.15.2.5 moveAxisPitch . . . . .	53
7.15.2.6 moveAxisRoll . . . . .	54
7.15.2.7 moveAxisShoulder . . . . .	54
7.15.2.8 moveCoordPitch . . . . .	54
7.15.2.9 moveCoordRoll . . . . .	54

7.15.2.10 moveCoordX . . . . .	54
7.15.2.11 moveCoordY . . . . .	54
7.15.2.12 moveCoordZ . . . . .	55
7.15.2.13 stopAllMovement . . . . .	55
7.15.3 Property Documentation . . . . .	55
7.15.3.1 RobotConnection . . . . .	55
7.15.3.2 Speed . . . . .	55
7.16 RoboGO.ViewModels.InfoViewModel Class Reference . . . . .	55
7.16.1 Detailed Description . . . . .	56
7.16.2 Constructor & Destructor Documentation . . . . .	56
7.16.2.1 InfoViewModel . . . . .	56
7.16.3 Member Function Documentation . . . . .	56
7.16.3.1 getTableInfo . . . . .	56
7.16.3.2 loadAllTables . . . . .	56
7.16.3.3 tablePrint . . . . .	56
7.16.3.4 tableSave . . . . .	57
7.16.4 Property Documentation . . . . .	57
7.16.4.1 Tables . . . . .	57
7.16.4.2 TableValues . . . . .	57
7.16.5 Event Documentation . . . . .	57
7.16.5.1 PropertyChanged . . . . .	57
7.17 ControlSystem.IScriptRunner Interface Reference . . . . .	57
7.17.1 Detailed Description . . . . .	58
7.17.2 Member Function Documentation . . . . .	58
7.17.2.1 clearOutputStream . . . . .	58
7.17.2.2 ExecuteScript . . . . .	58
7.17.2.3 readFromOutputStream . . . . .	58
7.17.2.4 setRobotInstance . . . . .	58
7.17.2.5 setScriptFromFile . . . . .	58
7.17.2.6 setScriptFromString . . . . .	59
7.18 SqlInteraction.ISqlConnection Interface Reference . . . . .	59
7.18.1 Member Function Documentation . . . . .	60
7.18.1.1 ConnectionClose . . . . .	60
7.18.1.2 ConnectionOpen . . . . .	60
7.18.1.3 CreateCommand . . . . .	60
7.18.2 Property Documentation . . . . .	60
7.18.2.1 Connectionstring . . . . .	60
7.18.2.2 RobotConnectionState . . . . .	60
7.18.2.3 TimeOut . . . . .	60
7.19 SqlInteraction.ISQLHandler Interface Reference . . . . .	60



7.19.1 Detailed Description . . . . .	61
7.19.2 Member Function Documentation . . . . .	61
7.19.2.1 addParameter . . . . .	61
7.19.2.2 changeConnectionparameter . . . . .	62
7.19.2.3 makeCommand . . . . .	62
7.19.2.4 runQuery . . . . .	62
7.19.2.5 setConnection . . . . .	62
7.19.3 Property Documentation . . . . .	63
7.19.3.1 Connection . . . . .	63
7.20 SqlInteraction.ISQLReader Interface Reference . . . . .	63
7.20.1 Detailed Description . . . . .	63
7.20.2 Member Function Documentation . . . . .	63
7.20.2.1 close . . . . .	64
7.20.2.2 readRow . . . . .	64
7.21 ControlSystem.IUI Interface Reference . . . . .	64
7.21.1 Detailed Description . . . . .	64
7.21.2 Member Function Documentation . . . . .	65
7.21.2.1 write . . . . .	65
7.21.2.2 writeLine . . . . .	65
7.22 ControlSystem.IUser Interface Reference . . . . .	65
7.22.1 Detailed Description . . . . .	66
7.22.2 Property Documentation . . . . .	66
7.22.2.1 permissionDictionary . . . . .	66
7.22.2.2 userName . . . . .	66
7.23 ControlSystem.IWrapper Interface Reference . . . . .	66
7.23.1 Detailed Description . . . . .	67
7.23.2 Member Function Documentation . . . . .	68
7.23.2.1 closeGripperWrapped . . . . .	68
7.23.2.2 closeManualWrapped . . . . .	68
7.23.2.3 closeWatchDigitalInputWrapped . . . . .	68
7.23.2.4 controlWrapped . . . . .	68
7.23.2.5 defineVectorWrapped . . . . .	68
7.23.2.6 enterManualWrapped . . . . .	69
7.23.2.7 getCurrentPosition . . . . .	69
7.23.2.8 getJawWrapped . . . . .	69
7.23.2.9 homeWrapped . . . . .	70
7.23.2.10 initializationWrapped . . . . .	70
7.23.2.11 isOnlineOkWrapped . . . . .	70
7.23.2.12 moveLinearWrapped . . . . .	70
7.23.2.13 moveManualWrapped . . . . .	71

7.23.2.14 openGripperWrapped . . . . .	71
7.23.2.15 speedWrapped . . . . .	71
7.23.2.16 stopWrapped . . . . .	71
7.23.2.17 teachWrapped . . . . .	72
7.23.2.18 timeWrapped . . . . .	72
7.23.2.19 watchDigitalInputWrapped . . . . .	72
7.23.2.20 watchMotionWrapped . . . . .	73
7.24 RoboGO.MainWindow Class Reference . . . . .	73
7.24.1 Detailed Description . . . . .	74
7.24.2 Member Function Documentation . . . . .	74
7.24.2.1 InitializeComponent . . . . .	74
7.24.2.2 InitializeComponent . . . . .	74
7.25 RoboGO.MainWindowViewModel Class Reference . . . . .	74
7.25.1 Detailed Description . . . . .	74
7.25.2 Constructor & Destructor Documentation . . . . .	74
7.25.2.1 MainWindowViewModel . . . . .	74
7.25.3 Member Function Documentation . . . . .	75
7.25.3.1 checkIsOnline . . . . .	75
7.25.3.2 setRobotAsRobotInstance . . . . .	75
7.25.3.3 setSimulatorAsRobotInstance . . . . .	75
7.25.3.4 stopRobotInstance . . . . .	75
7.26 ControlSystem.ManualController Class Reference . . . . .	75
7.26.1 Detailed Description . . . . .	77
7.26.2 Member Function Documentation . . . . .	77
7.26.2.1 moveAxisBase . . . . .	77
7.26.2.2 moveAxisConveyer . . . . .	77
7.26.2.3 moveAxisElbow . . . . .	77
7.26.2.4 moveAxisGripper . . . . .	77
7.26.2.5 moveAxisPitch . . . . .	77
7.26.2.6 moveAxisRoll . . . . .	78
7.26.2.7 moveAxisShoulder . . . . .	78
7.26.2.8 moveCoordPitch . . . . .	78
7.26.2.9 moveCoordRoll . . . . .	78
7.26.2.10 moveCoordX . . . . .	78
7.26.2.11 moveCoordY . . . . .	78
7.26.2.12 moveCoordZ . . . . .	79
7.26.2.13 stopAllMovement . . . . .	79
7.26.3 Property Documentation . . . . .	79
7.26.3.1 RobotConnection . . . . .	79
7.26.3.2 Speed . . . . .	79

7.27 RoboGO.PasswordWindow Class Reference . . . . .	79
7.27.1 Detailed Description . . . . .	80
7.27.2 Member Function Documentation . . . . .	80
7.27.2.1 InitializeComponent . . . . .	80
7.27.2.2 InitializeComponent . . . . .	80
7.28 RoboGO.ViewModels.passwordWindowViewModel Class Reference . . . . .	80
7.28.1 Detailed Description . . . . .	81
7.28.2 Constructor & Destructor Documentation . . . . .	81
7.28.2.1 passwordWindowViewModel . . . . .	81
7.28.3 Member Function Documentation . . . . .	81
7.28.3.1 authenticate . . . . .	81
7.29 RoboGO.ViewModels.PositionModel Class Reference . . . . .	81
7.29.1 Detailed Description . . . . .	81
7.29.2 Property Documentation . . . . .	82
7.29.2.1 PositionVec . . . . .	82
7.30 RoboGO.ViewModels.PositionViewModel Class Reference . . . . .	82
7.30.1 Detailed Description . . . . .	82
7.30.2 Constructor & Destructor Documentation . . . . .	82
7.30.2.1 PositionViewModel . . . . .	82
7.30.2.2 PositionViewModel . . . . .	82
7.30.3 Member Function Documentation . . . . .	82
7.30.3.1 getXYZPR . . . . .	82
7.30.3.2 update . . . . .	83
7.30.3.3 update . . . . .	83
7.31 ControlSystem.RelCoordSirVector Class Reference . . . . .	83
7.31.1 Detailed Description . . . . .	84
7.31.2 Constructor & Destructor Documentation . . . . .	84
7.31.2.1 RelCoordSirVector . . . . .	84
7.32 SqlInteraction.RobotSqlConnection Class Reference . . . . .	84
7.32.1 Detailed Description . . . . .	85
7.32.2 Constructor & Destructor Documentation . . . . .	85
7.32.2.1 RobotSqlConnection . . . . .	85
7.32.3 Member Function Documentation . . . . .	85
7.32.3.1 ConnectionClose . . . . .	85
7.32.3.2 ConnectionOpen . . . . .	85
7.32.3.3 CreateCommand . . . . .	85
7.32.4 Property Documentation . . . . .	86
7.32.4.1 Connectionstring . . . . .	86
7.32.4.2 RobotConnectionState . . . . .	86
7.32.4.3 TimeOut . . . . .	86

7.33	ControlSystem.ScriptRunner Class Reference	86
7.33.1	Detailed Description	87
7.33.2	Member Function Documentation	87
7.33.2.1	clearOutputStream	87
7.33.2.2	ExecuteScript	88
7.33.2.3	getInstance	88
7.33.2.4	readFromOutputStream	88
7.33.2.5	setRobotInstance	88
7.33.2.6	setScriptFromFile	88
7.33.2.7	setScriptFromString	88
7.34	ControlSystem.SerialSTK Class Reference	88
7.34.1	Detailed Description	89
7.34.2	Constructor & Destructor Documentation	89
7.34.2.1	SerialSTK	89
7.34.3	Member Function Documentation	89
7.34.3.1	Close	89
7.34.3.2	Open	89
7.34.3.3	ReadADC	89
7.35	ControlSystem.Simulator Class Reference	90
7.35.1	Detailed Description	91
7.35.2	Constructor & Destructor Documentation	91
7.35.2.1	Simulator	91
7.35.3	Member Function Documentation	91
7.35.3.1	closeGripper	91
7.35.3.2	getCurrentPosition	92
7.35.3.3	getJawOpeningWidthMilimeters	92
7.35.3.4	getJawOpeningWidthPercentage	92
7.35.3.5	homeRobot	92
7.35.3.6	isOnline	92
7.35.3.7	moveBase	92
7.35.3.8	moveByAbsoluteCoordinates	93
7.35.3.9	moveByCoordinates	93
7.35.3.10	movebyCoordinates	93
7.35.3.11	moveByPitch	93
7.35.3.12	moveByRelativeCoordinates	93
7.35.3.13	moveByRoll	94
7.35.3.14	moveByXCoordinate	94
7.35.3.15	moveByYCoordinate	94
7.35.3.16	moveByZCoordinate	94
7.35.3.17	moveConveyerBelt	95

7.35.3.18 moveElbow . . . . .	95
7.35.3.19 moveGripper . . . . .	95
7.35.3.20 moveShoulder . . . . .	95
7.35.3.21 moveWristPitch . . . . .	95
7.35.3.22 moveWristRoll . . . . .	96
7.35.3.23 openGripper . . . . .	96
7.35.3.24 Speed . . . . .	96
7.35.3.25 stopAllMovement . . . . .	96
7.35.3.26 Time . . . . .	97
7.35.4 Property Documentation . . . . .	97
7.35.4.1 Currentposition . . . . .	97
7.35.4.2 IUIOutput . . . . .	97
7.36 RoboGO.Simulator Class Reference . . . . .	97
7.36.1 Detailed Description . . . . .	98
7.36.2 Member Function Documentation . . . . .	98
7.36.2.1 InitializeComponent . . . . .	98
7.36.2.2 InitializeComponent . . . . .	98
7.37 RoboGO.ViewModels.SimulatorViewModel Class Reference . . . . .	98
7.37.1 Detailed Description . . . . .	98
7.37.2 Constructor & Destructor Documentation . . . . .	99
7.37.2.1 SimulatorViewModel . . . . .	99
7.37.3 Property Documentation . . . . .	99
7.37.3.1 CurrentPosition . . . . .	99
7.37.3.2 UIText . . . . .	99
7.38 ControlSystem.SIRVector Class Reference . . . . .	99
7.38.1 Detailed Description . . . . .	100
7.38.2 Member Function Documentation . . . . .	100
7.38.2.1 addPoint . . . . .	100
7.38.2.2 getPoint . . . . .	100
7.38.2.3 getSize . . . . .	100
7.38.3 Member Data Documentation . . . . .	100
7.38.3.1 iType . . . . .	100
7.38.4 Property Documentation . . . . .	101
7.38.4.1 Name . . . . .	101
7.38.4.2 Type . . . . .	101
7.39 SqlInteraction.SQLHandler Class Reference . . . . .	101
7.39.1 Detailed Description . . . . .	102
7.39.2 Member Function Documentation . . . . .	102
7.39.2.1 addParameter . . . . .	102
7.39.2.2 changeConnectionparameter . . . . .	102

7.39.2.3	makeCommand	103
7.39.2.4	runQuery	103
7.39.2.5	setConnection	103
7.39.3	Property Documentation	103
7.39.3.1	Connection	103
7.39.3.2	GetInstance	104
7.40	SqlInteraction.SQLiteReader Class Reference	104
7.40.1	Detailed Description	105
7.40.2	Constructor & Destructor Documentation	105
7.40.2.1	SQLiteReader	105
7.40.3	Member Function Documentation	105
7.40.3.1	close	105
7.40.3.2	readRow	105
7.40.4	Property Documentation	105
7.40.4.1	SQLiteCoreReader	105
7.41	ControlSystem.StringUI Class Reference	105
7.41.1	Detailed Description	107
7.41.2	Constructor & Destructor Documentation	107
7.41.2.1	StringUI	107
7.41.3	Member Function Documentation	107
7.41.3.1	clearString	107
7.41.3.2	write	107
7.41.3.3	writeLine	107
7.41.4	Property Documentation	107
7.41.4.1	Buffer	107
7.42	ControlSystem.ThreadHandling Class Reference	108
7.42.1	Detailed Description	108
7.42.2	Constructor & Destructor Documentation	108
7.42.2.1	ThreadHandling	108
7.42.3	Member Function Documentation	108
7.42.3.1	abortAllAndWait	108
7.42.3.2	abortAndWait	109
7.42.3.3	addThread	109
7.42.3.4	addThread	109
7.42.3.5	find	109
7.42.3.6	removeThread	109
7.42.3.7	start	110
7.42.3.8	start	110
7.43	ControlSystem.ThreadHandling.ThreadHolder Class Reference	110
7.43.1	Detailed Description	110

7.43.2	Property Documentation	110
7.43.2.1	stringDescription	110
7.43.2.2	threadPlaceHolder	110
7.44	ControlSystem.User Class Reference	111
7.44.1	Detailed Description	111
7.44.2	Constructor & Destructor Documentation	112
7.44.2.1	User	112
7.44.3	Property Documentation	112
7.44.3.1	permissionDictionary	112
7.44.3.2	userName	112
7.45	ControlSystem.VecPoint Class Reference	112
7.45.1	Detailed Description	112
7.45.2	Constructor & Destructor Documentation	112
7.45.2.1	VecPoint	112
7.45.3	Member Function Documentation	113
7.45.3.1	ToString	113
7.46	RoboGO.ViewModels.ViewModelManualSteering Class Reference	113
7.46.1	Detailed Description	114
7.46.2	Constructor & Destructor Documentation	115
7.46.2.1	ViewModelManualSteering	115
7.46.3	Member Function Documentation	115
7.46.3.1	closeGripper	115
7.46.3.2	moveAxisBaseLeft	115
7.46.3.3	moveAxisBaseRight	115
7.46.3.4	moveAxisConveyerLeft	115
7.46.3.5	moveAxisConveyerRight	115
7.46.3.6	moveAxisElbowLeft	115
7.46.3.7	moveAxisElbowRight	115
7.46.3.8	moveAxisPitchDown	115
7.46.3.9	moveAxisPitchUp	115
7.46.3.10	moveAxisRollLeft	115
7.46.3.11	moveAxisRollRight	116
7.46.3.12	moveAxisShoulderLeft	116
7.46.3.13	moveAxisShoulderRight	116
7.46.3.14	moveCoordPitchDecreasing	116
7.46.3.15	moveCoordPitchIncreasing	116
7.46.3.16	moveCoordRollDecreasing	116
7.46.3.17	moveCoordRollIncreasing	116
7.46.3.18	moveCoordXDecreasing	116
7.46.3.19	moveCoordXIncreasing	116

7.46.3.20	moveCoordYDecreasing	116
7.46.3.21	moveCoordYIncreasing	116
7.46.3.22	moveCoordZDecreasing	116
7.46.3.23	moveCoordZIncreasing	117
7.46.3.24	openGripper	117
7.46.3.25	seekHome	117
7.46.3.26	stopMovement	117
7.46.4	Property Documentation	117
7.46.4.1	CloseGripper	117
7.46.4.2	ManualControl	117
7.46.4.3	OpenGripper	117
7.46.4.4	SeekHome	117
7.46.4.5	Speed	117
7.47	ControlSystem Wrapper Class Reference	117
7.47.1	Detailed Description	120
7.47.2	Member Enumeration Documentation	120
7.47.2.1	enumAxisSettings	120
7.47.2.2	enumBGroup	120
7.47.2.3	enumManualModeWhat	120
7.47.2.4	enumManualType	120
7.47.2.5	enumSystemModes	120
7.47.2.6	enumSystemTypes	120
7.47.3	Member Function Documentation	120
7.47.3.1	closeGripperWrapped	120
7.47.3.2	closeManualWrapped	121
7.47.3.3	closeWatchDigitalInputWrapped	121
7.47.3.4	controlWrapped	121
7.47.3.5	defineVectorWrapped	121
7.47.3.6	enterManualWrapped	122
7.47.3.7	getCurrentPosition	122
7.47.3.8	getInstance	122
7.47.3.9	getJawWrapped	122
7.47.3.10	homeWrapped	123
7.47.3.11	initializationWrapped	123
7.47.3.12	isOnlineOkWrapped	123
7.47.3.13	moveLinearWrapped	123
7.47.3.14	moveManualWrapped	124
7.47.3.15	openGripperWrapped	124
7.47.3.16	speedWrapped	124
7.47.3.17	stopWrapped	124



7.47.3.18 teachWrapped . . . . .	125
7.47.3.19 timeWrapped . . . . .	125
7.47.3.20 watchDigitalInputWrapped . . . . .	125
7.47.3.21 watchMotionWrapped . . . . .	125
7.47.4 Property Documentation . . . . .	126
7.47.4.1 DLL . . . . .	126
7.48 RoboGO.ViewModels.XYCalculate Class Reference . . . . .	126
7.48.1 Detailed Description . . . . .	126
7.48.2 Constructor & Destructor Documentation . . . . .	126
7.48.2.1 XYCalculate . . . . .	126
7.48.3 Member Function Documentation . . . . .	127
7.48.3.1 elbow . . . . .	127
7.48.3.2 gripper . . . . .	127
7.48.4 Member Data Documentation . . . . .	127
7.48.4.1 elbowRotate . . . . .	127
7.48.4.2 gripperX . . . . .	127
7.48.4.3 gripperY . . . . .	127
<b>8 File Documentation</b>	<b>129</b>
8.1 ControlSystem/dll.cs File Reference . . . . .	129
8.1.1 Detailed Description . . . . .	129
8.2 ControlSystem/errorReporter.cs File Reference . . . . .	129
8.2.1 Detailed Description . . . . .	129
8.3 ControlSystem/factory.cs File Reference . . . . .	129
8.3.1 Detailed Description . . . . .	130
8.4 ControlSystem/iDLL.cs File Reference . . . . .	130
8.4.1 Detailed Description . . . . .	130
8.5 ControlSystem/iWrapper.cs File Reference . . . . .	130
8.5.1 Detailed Description . . . . .	130
8.6 ControlSystem/manualController.cs File Reference . . . . .	130
8.6.1 Detailed Description . . . . .	131
8.7 ControlSystem/scriptRunner.cs File Reference . . . . .	131
8.7.1 Detailed Description . . . . .	131
8.8 ControlSystem/serialSTK.cs File Reference . . . . .	131
8.8.1 Detailed Description . . . . .	132
8.9 ControlSystem/simulator.cs File Reference . . . . .	132
8.9.1 Detailed Description . . . . .	132
8.10 ControlSystem/threadHandling.cs File Reference . . . . .	132
8.10.1 Detailed Description . . . . .	132
8.11 ControlSystem/ui.cs File Reference . . . . .	132

8.11.1 Detailed Description . . . . .	133
8.12 ControlSystem/wrapper.cs File Reference . . . . .	133
8.12.1 Detailed Description . . . . .	133
8.13 RoboGO/ViewModels/delegateCommand.cs File Reference . . . . .	133
8.13.1 Detailed Description . . . . .	133
8.14 RoboGO/ViewModels/ideViewModel.cs File Reference . . . . .	133
8.14.1 Detailed Description . . . . .	134
8.15 RoboGO/ViewModels/infoViewModel.cs File Reference . . . . .	134
8.15.1 Detailed Description . . . . .	134
8.16 RoboGO/ViewModels/passwordWindowViewModel.cs File Reference . . . . .	134
8.16.1 Detailed Description . . . . .	134
8.17 RoboGO/ViewModels/positionModel.cs File Reference . . . . .	134
8.17.1 Detailed Description . . . . .	134
8.18 RoboGO/ViewModels/positionViewModel.cs File Reference . . . . .	134
8.18.1 Detailed Description . . . . .	135
8.19 RoboGO/ViewModels/simulatorViewModel.cs File Reference . . . . .	135
8.19.1 Detailed Description . . . . .	135
8.20 RoboGO/ViewModels/viewModelManualSteering.cs File Reference . . . . .	135
8.20.1 Detailed Description . . . . .	135
8.21 SqlInteraction/iSQLHandler.cs File Reference . . . . .	135
8.21.1 Detailed Description . . . . .	136
8.22 SqlInteraction/sqlReader.cs File Reference . . . . .	136
8.22.1 Detailed Description . . . . .	136

# Chapter 1

## Todo List

**Member** [ControlSystem.IWrapper.moveLinearWrapped](#) (string \_sNameOfVector, int \_ilIndex)

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):

<a href="#">ControlSystem</a>	
Class to handle system threads . . . . .	13
<a href="#">RoboGO</a> . . . . .	15
<a href="#">RoboGO.Properties</a> . . . . .	15
<a href="#">RoboGO.ViewModels</a> . . . . .	16
<a href="#">SqlInteraction</a> . . . . .	16
<a href="#">XamlGeneratedNamespace</a> . . . . .	16



## Chapter 3

# Class Index

### 3.1 Class Hierarchy

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

RoboGO.aboutBox . . . . .	17
RoboGO.App . . . . .	20
RoboGO.CommandsWindow . . . . .	21
RoboGO.ViewModels.DelegateCommand . . . . .	24
ControlSystem.DLLImport . . . . .	31
ControlSystem.ErrorReporter . . . . .	37
XamlGeneratedNamespace.GeneratedInternalTypeHelper . . . . .	38
RoboGO.GUIManualSteering . . . . .	40
RoboGO.ViewModels.IDEViewModel . . . . .	42
ControlSystem.IDLL . . . . .	45
ControlSystem.DLL . . . . .	25
ControlSystem.IManualController . . . . .	51
ControlSystem.ManualController . . . . .	75
RoboGO.ViewModels.InfoViewModel . . . . .	55
ControlSystem.IScriptRunner . . . . .	57
ControlSystem.ScriptRunner . . . . .	86
SqlInteraction.ISqlConnection . . . . .	59
SqlInteraction.RobotSqlConnection . . . . .	84
SqlInteraction.ISQLHandler . . . . .	60
SqlInteraction.SQLHandler . . . . .	101
SqlInteraction.ISQLReader . . . . .	63
SqlInteraction.SQLReader . . . . .	104
ControlSystem.IUI . . . . .	64
ControlSystem.ConsoleUI . . . . .	22
ControlSystem.StringUI . . . . .	105
ControlSystem.IUser . . . . .	65
ControlSystem.Admin . . . . .	19
ControlSystem.User . . . . .	111
ControlSystem.IWrapper . . . . .	66
ControlSystem.Wrapper . . . . .	117
RoboGO.MainWindow . . . . .	73
RoboGO.MainWindowViewModel . . . . .	74
RoboGO.PasswordWindow . . . . .	79
RoboGO.ViewModels.passwordWindowViewModel . . . . .	80
RoboGO.ViewModels.PositionModel . . . . .	81

RoboGO.ViewModels.PositionViewModel . . . . .	82
ControlSystem.SerialSTK . . . . .	88
ControlSystem.Simulator . . . . .	90
RoboGO.Simulator . . . . .	97
RoboGO.ViewModels.SimulatorViewModel . . . . .	98
ControlSystem.SIRVector . . . . .	99
ControlSystem.AbsCoordSirVector . . . . .	18
ControlSystem.RelCoordSirVector . . . . .	83
ControlSystem.ThreadHandling . . . . .	108
ControlSystem.ThreadHandling.ThreadHolder . . . . .	110
ControlSystem.VecPoint . . . . .	112
RoboGO.ViewModels.ViewModelManualSteering . . . . .	113
RoboGO.ViewModels.XYCalculate . . . . .	126



## Chapter 4

# Class Index

### 4.1 Class List

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

<a href="#">RoboGO.aboutBox</a>	
AboutBox . . . . .	17
<a href="#">ControlSystem.AbsCoordSirVector</a>	
SIRVector class for absolute positions . . . . .	18
<a href="#">ControlSystem.Admin</a>	
Admin user with ability to see and edit all tables . . . . .	19
<a href="#">RoboGO.App</a>	
App . . . . .	20
<a href="#">RoboGO.CommandsWindow</a>	
CommandsWindow . . . . .	21
<a href="#">ControlSystem.ConsoleUI</a>	
Console UI for writing to the console output . . . . .	22
<a href="#">RoboGO.ViewModels.DelegateCommand</a>	
Command class for executing one function . . . . .	24
<a href="#">ControlSystem.DLL</a>	
Class that with the <a href="#">IDLL</a> interface calls the actual functions from the USBC.dll . . . . .	25
<a href="#">ControlSystem.DLLImport</a>	
Class providing interface for USBC.dll functions . . . . .	31
<a href="#">ControlSystem.ErrorReporter</a>	
Temporary error reporting class . . . . .	37
<a href="#">XamlGeneratedNamespace.GeneratedInternalTypeHelper</a>	
GeneratedInternalTypeHelper . . . . .	38
<a href="#">RoboGO.GUIManualSteering</a>	
GUIManualSteering . . . . .	40
<a href="#">RoboGO.ViewModels.IDEViewModel</a>	
ViewModel between IDEView and ScriptRunner . . . . .	42
<a href="#">ControlSystem.IDLL</a>	
Interface for the functions in the USBC.dll files, in C# format . . . . .	45
<a href="#">ControlSystem.IManualController</a>	
Interface for what manual movement functions there should be available . . . . .	51
<a href="#">RoboGO.ViewModels.InfoViewModel</a>	
ViewModel for the tables.(From the database.) . . . . .	55
<a href="#">ControlSystem.IScriptRunner</a>	
Interface for a script runner . . . . .	57
<a href="#">SqlInteraction.ISqlConnection</a>	59
<a href="#">SqlInteraction.ISQLHandler</a>	
Interface for handling class of sql . . . . .	60

<a href="#">SqlInteraction.ISQLReader</a>	Interface for class that read information from a SQL table . . . . .	63
<a href="#">ControlSystem.IUI</a>	Simple interface for UI interaction . . . . .	64
<a href="#">ControlSystem.IUser</a>	Interface for an user with permissions for the database . . . . .	65
<a href="#">ControlSystem.IWrapper</a>	Interface for a wrapper wrapping the USBC.dll file . . . . .	66
<a href="#">RoboGO.MainWindow</a>		
<a href="#">MainWindow</a>	. . . . .	73
<a href="#">RoboGO.MainWindowViewModel</a>	ViewModel for the mainwindow . . . . .	74
<a href="#">ControlSystem.ManualController</a>	Class that encapsulates controlling manual movement. Instead of using directly Robot or <a href="#">Simulator</a> by interface . . . . .	75
<a href="#">RoboGO.PasswordWindow</a>		
<a href="#">PasswordWindow</a>	. . . . .	79
<a href="#">RoboGO.ViewModels.passwordWindowViewModel</a>	ViewModel for password window . . . . .	80
<a href="#">RoboGO.ViewModels.PositionModel</a>	Class to keep track of simulator position . . . . .	81
<a href="#">RoboGO.ViewModels.PositionViewModel</a>	ViewModel for the position class . . . . .	82
<a href="#">ControlSystem.RelCoordSirVector</a>		
<a href="#">SIRVector</a>	class for relative positions . . . . .	83
<a href="#">SqlInteraction.RobotSqlConnection</a>	SqlConnection to connect to a database . . . . .	84
<a href="#">ControlSystem.ScriptRunner</a>	Used to run IronPython scripts . . . . .	86
<a href="#">ControlSystem.SerialSTK</a>	Class for functions to communicating with the STK kit.(Serial communication.) . . . . .	88
<a href="#">ControlSystem.Simulator</a>	IRobot implementation using <a href="#">IUI</a> output interface for simulating robot behavior . . . . .	90
<a href="#">RoboGO.Simulator</a>		
<a href="#">Simulator</a>	. . . . .	97
<a href="#">RoboGO.ViewModels.SimulatorViewModel</a>	ViewModel for the simulator class . . . . .	98
<a href="#">ControlSystem.SIRVector</a>	Base class for vector used in wrapper . . . . .	99
<a href="#">SqlInteraction.SQLHandler</a>	Class that handles all SQL interaction . . . . .	101
<a href="#">SqlInteraction.SQLReader</a>	Class to read table data . . . . .	104
<a href="#">ControlSystem.StringUI</a>	String UI for writing to a string variable . . . . .	105
<a href="#">ControlSystem.ThreadHandling</a>	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 . . . . .	108
<a href="#">ControlSystem.ThreadHandling.ThreadHolder</a>	Class that holds the description of the thread and the thread itself for usage . . . . .	110
<a href="#">ControlSystem.User</a>	Normal user . . . . .	111
<a href="#">ControlSystem.VecPoint</a>	Class to contain point for use in one of the vector classes . . . . .	112
<a href="#">RoboGO.ViewModels.ViewModelManualSteering</a>	ViewModel for <a href="#">GUIManualSteering</a> . . . . .	113
<a href="#">ControlSystem.Wrapper</a>	Contains a wrapper for the C++ functions in the dll file(USBC.dll) . . . . .	117

[RoboGO.ViewModels.XYCalculate](#)

Class for calculating position of robot . . . . . 126



## Chapter 5

# File Index

### 5.1 File List

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

ControlSystem/ <a href="#">dll.cs</a>	129
ControlSystem/ <a href="#">errorReporter.cs</a>	129
ControlSystem/ <a href="#">factory.cs</a>	129
ControlSystem/ <a href="#">iDLL.cs</a>	130
ControlSystem/ <a href="#">iWrapper.cs</a>	130
ControlSystem/ <a href="#">manualController.cs</a>	130
ControlSystem/ <a href="#">scriptRunner.cs</a>	131
ControlSystem/ <a href="#">serialSTK.cs</a>	131
ControlSystem/ <a href="#">simulator.cs</a>	132
ControlSystem/ <a href="#">threadHandling.cs</a>	132
ControlSystem/ <a href="#">ui.cs</a>	132
ControlSystem/ <a href="#">wrapper.cs</a>	133
RoboGO/ViewModels/ <a href="#">delegateCommand.cs</a>	133
RoboGO/ViewModels/ <a href="#">ideViewModel.cs</a>	133
RoboGO/ViewModels/ <a href="#">infoViewModel.cs</a>	134
RoboGO/ViewModels/ <a href="#">passwordWindowViewModel.cs</a>	134
RoboGO/ViewModels/ <a href="#">positionModel.cs</a>	134
RoboGO/ViewModels/ <a href="#">positionViewModel.cs</a>	134
RoboGO/ViewModels/ <a href="#">simulatorViewModel.cs</a>	135
RoboGO/ViewModels/ <a href="#">viewModelManualSteering.cs</a>	135
SqlInteraction/ <a href="#">SQLHandler.cs</a>	135
SqlInteraction/ <a href="#">sqlReader.cs</a>	136



## 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](#).*

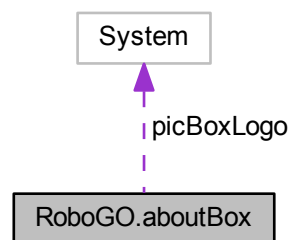
## Chapter 7

# 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.

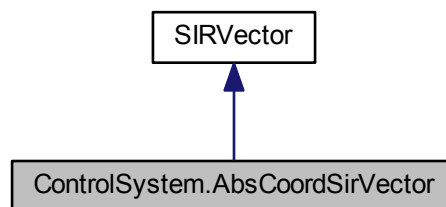
The documentation for this class was generated from the following files:

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

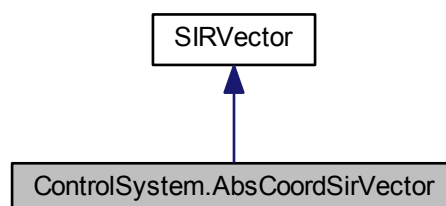
## 7.2 ControlSystem.AbsCoordSirVector Class Reference

[SIRVector](#) class for absolute positions.

Inheritance diagram for ControlSystem.AbsCoordSirVector:



Collaboration diagram for ControlSystem.AbsCoordSirVector:



### Public Member Functions

- [AbsCoordSirVector](#) (string \_sName)  
*Constructors 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 )

Constructors whichs sets up type and name of vector.

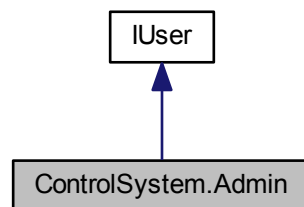
The documentation for this class was generated from the following file:

- ControlSystem/[wrapper.cs](#)

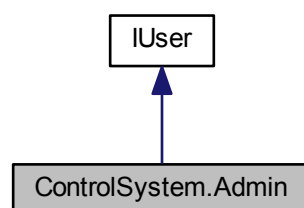
## 7.3 ControlSystem.Admin Class Reference

[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.

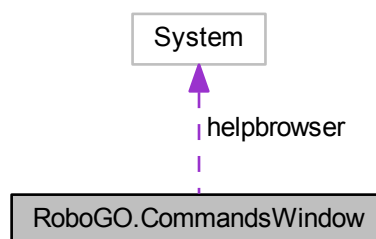
The documentation for this class was generated from the following files:

- 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.*

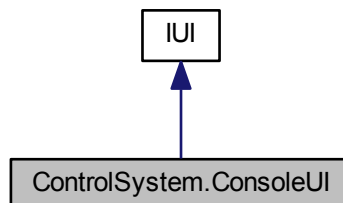
The documentation for this class was generated from the following files:

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

## 7.6 ControlSystem.ConsoleUI Class Reference

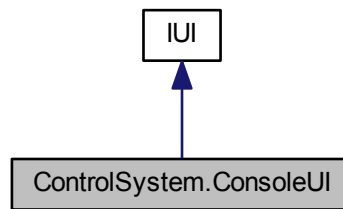
Console UI for writing to the console output.

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

<i>sMsg</i>	The string with the message and argument placement.(Like normal Write())
<i>paramobj-Argument</i>	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

<i>sMsg</i>	The string with the message and argument placement.(Like normal WriteLine())
<i>paramobj-Argument</i>	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

<code>_aMethodToExecute</code>	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

<code>_objParam</code>	Unused from ICommand.
------------------------	-----------------------

**Returns**

Always true.

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

Execute the command from the constructor.

**Parameters**

<i>_objParam</i>	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.

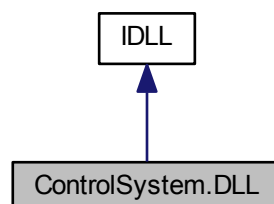
The documentation for this class was generated from the following file:

- RoboGO/ViewModels/[delegateCommand.cs](#)

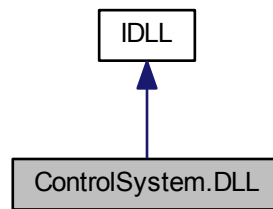
**7.8 ControlSystem.DLL Class Reference**

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 \_blsOn)  
*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

<b>_bAxis</b>	Which Axis to move (0-7)
<b>_lSpeed</b>	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 *\_bIsOn* )

Turns control on and off for certain axis group.

**Parameters**

<i>_bAxis</i>	Axis group to affect.(Use enum)
<i>_bIsOn</i>	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**

<i>_enumGroup</i>	Group can use: Robot(Normally used) Peripherals All
<i>_sVectorName</i>	Name of vector.
<i>_shrtLength</i>	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**

<i>_shrArg</i>	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

<i>_perc</i>	Data in percentage.
<i>_metric</i>	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

<i>_sNameOfVectorThatGotPosition</i>	Navnet på vektoren i robotten.
<i>_shrtPointInVector</i>	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**

<u>_bAxis</u>	Which Axis to move (0-7)
<u>_lSpeed</u>	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**

<u>_bGroup</u>	Which joint to set time (& for all)
<u>_mSpeed</u>	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**

<u>_axis</u>	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**

<code>_sVectorName</code>	The vector with the points.
---------------------------	-----------------------------

**Returns**

Returns true on successfull call.

Implements [ControlSystem.IDLL](#).

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

Set time for movements.

**Parameters**

<code>_bGroup</code>	Which joint to set time (& for all)
<code>_mTime</code>	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 blsOn)  
*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 ISpeed)  
*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 shrtSizeOfVector)  
*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**

<i>bAxis</i>	Axis to apply setting.
<i>bIsOn</i>	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**

<i>bGroup</i>	Group for the vector type.('A' robot, '&' all axes, 'B' peripherals.)
<i>sVectorName</i>	Name of vector.
<i>shrtSizeOfVector</i>	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

<i>shrtArg</i>	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

<i>ibufEnc</i>	Buffer to save values.
<i>ibufJoint</i>	Buffer to save values.
<i>ibufXYZ</i>	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

<i>perc</i>	In percentage.
<i>metric</i>	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

<i>axis</i>	Axis to home.
<i>funcptrCallBack</i>	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

<i>shrtMode</i>	Mode of the robot.
<i>shrtType</i>	Type of connection.
<i>funcptrCallBack</i>	Function to call when initialized.
<i>funcptrCallBack-Error</i>	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

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

Move an axis group.

Note: Call Control and EnterManual first.

## Parameters

<i>bAxis</i>	What axis to move.
<i>ISpeed</i>	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

<i>_bGroup</i>	What axis group.
<i>_mSpeed</i>	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

<i>axis</i>	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

<i>sVectorName</i>	Name of vector.
<i>shrtPoint</i>	What point.(Index.)
<i>iaPointInfo</i>	Point values.
<i>shrtSizeOfArray</i>	Size of point values.(How many values.)
<i>iPointType</i>	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

<code>_bGroup</code>	What axis group.
<code>_mTime</code>	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

<code>funcptrCallback-Event</code>	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

<code>funcptrCallback-End</code>	Function to call when movement has ended.
<code>funcptrCallback-Start</code>	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

<i>_scrpsrcScriptSource</i>	Script.
<i>_sErrorMsg</i>	Error message.
<i>_srcspanSpanning</i>	Span.
<i>_iCode</i>	Code.
<i>sevSeverity</i>	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.

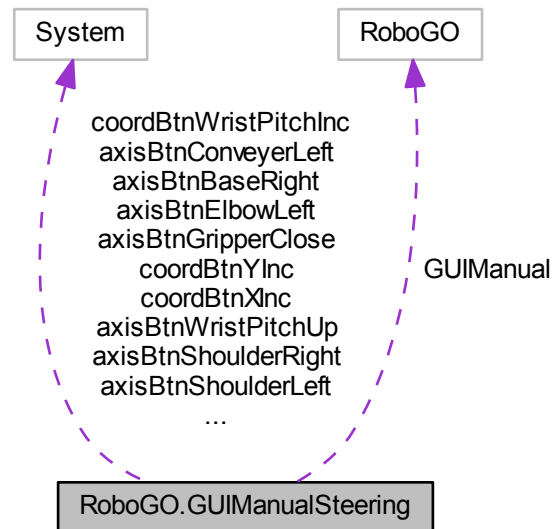
The documentation for this class was generated from the following files:

- 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

<i>_ideTabs</i>	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.)

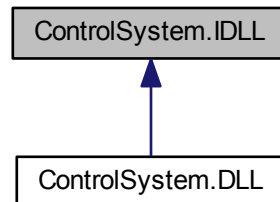
The documentation for this class was generated from the following file:

- 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 \_blsOn)  
*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 \_ISpeed)  
*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

<a href="#">_bAxis</a>	Which Axis to move (0-7)
<a href="#">_ISpeed</a>	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**

<i>bAxis</i>	Axis group to affect.(Use enum)
<i>_bIsOn</i>	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**

<i>_enumGroup</i>	Group can use: Robot(Normally used) Peripherals All
<i>_sVectorName</i>	Name of vector.
<i>_shrtLength</i>	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**

<i>_shrArg</i>	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**

<i>_perc</i>	Data in percentage.
<i>_metric</i>	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**

<i>_axis</i>	The axis group.(Use enum)
<i>_funcptrCallBack</i>	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

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

<code>_sNameOf-VectorThatGot-Position</code>	Navnet på vektoren i robotten.
<code>_shrtPointIn-Vector</code>	Index for punkt.

## Returns

Returns true on successfull call.

Implemented in [ControlSystem.DLL](#).

7.14.2.13 `int ControlSystem.IDLL.MoveManual ( byte _bAxis, int _ISpeed )`

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

<code>_bAxis</code>	Which Axis to move (0-7)
<code>_ISpeed</code>	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**

<code>_bGroup</code>	Which joint to set time (& for all)
<code>_mSpeed</code>	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**

<code>_axis</code>	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**

<code>_sVectorName</code>	The vector with the points.
---------------------------	-----------------------------

**Returns**

Returns true on successful call.

Implemented in [ControlSystem.DLL](#).

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

Set time for movements.

**Parameters**

<i>_bGroup</i>	Which joint to set time (& for all)
<i>_mTime</i>	Time in millisecond

**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**

<i>_funcptr- CallbackEvent</i>	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**

<i>_funcptr- CallbackEnd</i>	Function to be called when motion has ended.
<i>_funcptr- CallbackStart</i>	Function to be called when motion has started.

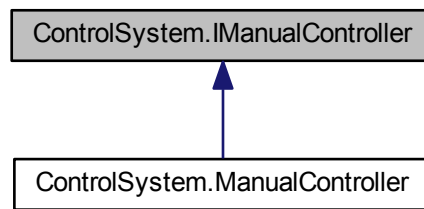
The documentation for this interface was generated from the following file:

- [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

<code>_elrDirection</code>	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

<code>_elrDirection</code>	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

<code>_elrDirection</code>	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

<code>_ecoGripper</code>	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

<code>_eudDirection</code>	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**

<b>_elrDirection</b>	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**

<b>_elrDirection</b>	What direction to move in.
----------------------	----------------------------

Implemented in [ControlSystem.ManualController](#).

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

Change the wrists pitch.

**Parameters**

<b>_eidIncOrDec</b>	Increasing or decreasing.
---------------------	---------------------------

Implemented in [ControlSystem.ManualController](#).

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

Change the roll of the wrist.

**Parameters**

<b>_eidIncOrDec</b>	Increasing or decreasing.
---------------------	---------------------------

Implemented in [ControlSystem.ManualController](#).

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

Change the robots X-coordinate.

**Parameters**

<b>_eidIncOrDec</b>	Increasing or decreasing.
---------------------	---------------------------

Implemented in [ControlSystem.ManualController](#).

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

Change the robots Y-coordinate.



## Parameters

<code>_eidIncOrDec</code>	Increasing or decreasing.
---------------------------	---------------------------

Implemented in [ControlSystem.ManualController](#).

#### 7.15.2.12 void ControlSystem.IManualController.moveCoordZ ( enumIncDec \_eidIncOrDec )

Change the robots Z-coordinate.

## Parameters

<code>_eidIncOrDec</code>	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

<a href="#">_database-Values</a>	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

<a href="#">_objTableName</a>	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.)

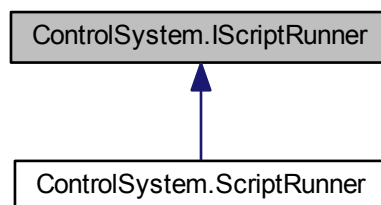
The documentation for this class was generated from the following file:

- 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

<a href="#">_iroboRobot</a>	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.

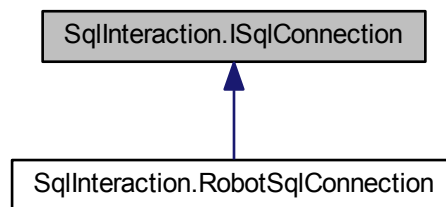
Implemented in [ControlSystem.ScriptRunner](#).

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](#).

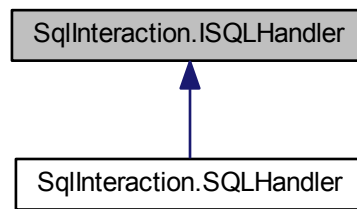
The documentation for this interface was generated from the following file:

- `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

<u>_command</u>	Parameter is the command made in makeCommand
<u>_parameter-Name</u>	Parameter is name of the parameter defined in makeCommand
<u>_parameter-Value</u>	Parameter is value to be used in the command
<u>_parameterType</u>	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

<i>_parameter</i>	Parameter is part of or full parameter name to change
<i>_parameter-Value</i>	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

<i>_commandText</i>	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

<i>_command</i>	Parameter is command to be executed
<i>queryType</i>	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

<i>_server</i>	Parameter for ip/domain to use
<i>_database</i>	Parameter for which database to use
<i>_username</i>	Parameter for username to use
<i>_password</i>	Parameter for password to use
<i>_timeout</i>	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](#).

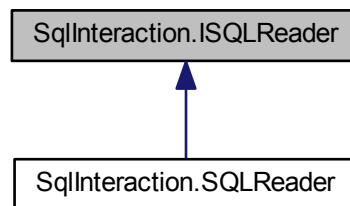
The documentation for this interface was generated from the following file:

- [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](#).

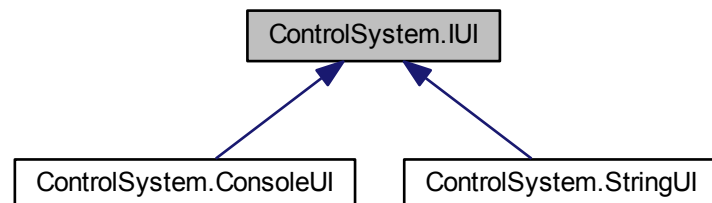
The documentation for this interface was generated from the following file:

- `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

<i>sMsg</i>	The string with the message and argument placement.(Like normal Write())
<i>paramobj-Argument</i>	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

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

Implemented in [ControlSystem.StringUI](#), and [ControlSystem.ConsoleUI](#).

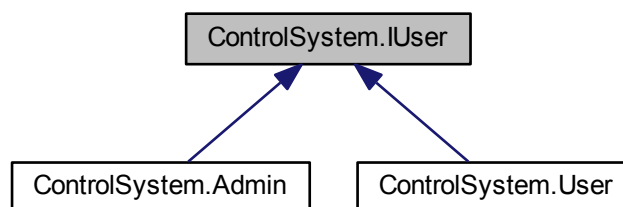
The documentation for this interface was generated from the following file:

- [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](#).

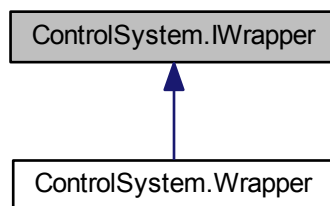
The documentation for this interface was generated from the following file:

- 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 \_ISpeed)  
*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

<code>_axisSettings-Group</code>	Axis group to affect.(Use enum)
<code>_bControlOnOr-Off</code>	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

<code>_enumGroup</code>	Group can use: Robot(Normally used) Peripherals All
<code>_sVectorName</code>	Name of vector.
<code>_shrtLength</code>	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

<code>_enumman-MoveType</code>	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

<code>_shrtPerc</code>	Data in percentage.
<code>_shrtWidth</code>	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

<code>_axisSettings-Group</code>	The axis group.(Use enum)
<code>_funcptrHoming-EventHandler</code>	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

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

<code>_sNameOf-Vector</code>	Navnet på vektoren i robotten.
<code>_iIndex</code>	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 _ISpeed )`

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

<code>_bGroup</code>	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
<code>_mSpeed</code>	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

<code>_bWhatToStop</code>	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**

<i>vecTheSirVector</i>	The vector with the points.
------------------------	-----------------------------

**Returns**

Returns true on successfull 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**

<i>_bGroup</i>	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
<i>_mTime</i>	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**

<i>_funcptr-CallbackEvent</i>	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

<i>_funcptr-CallbackEnd</i>	Function to be called when motion has ended.
<i>_funcptr-CallbackStart</i>	Function to be called when motion has started.

Implemented in [ControlSystem.Wrapper](#).

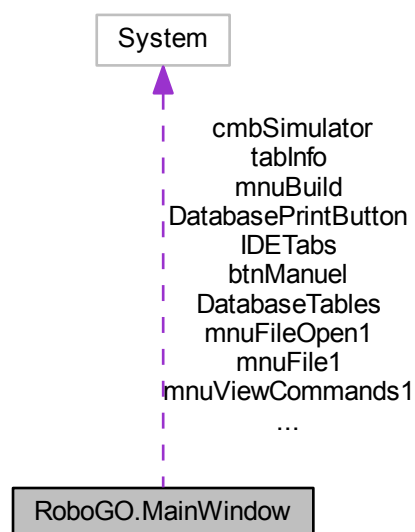
The documentation for this interface was generated from the following file:

- [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

<code>_pb</code>	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.

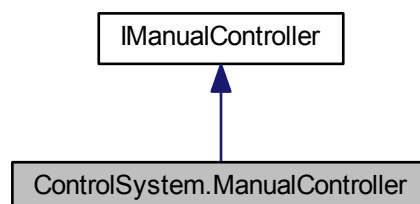
The documentation for this class was generated from the following file:

- RoboGO/ViewModels/mainWindowViewModel.cs

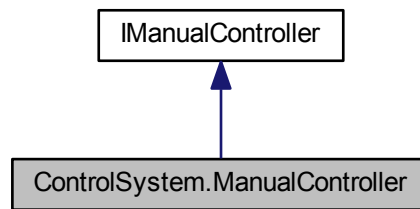
## 7.26 ControlSystem.ManualController Class Reference

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

<code>_elrDirection</code>	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

<code>_elrDirection</code>	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

<code>_elrDirection</code>	What direction to move in.
----------------------------	----------------------------

Implements [ControlSystem.IManualController](#).

#### 7.26.2.4 void ControlSystem.ManualController.moveAxisGripper ( enumCloseOpen \_ecoGripper )

Opens or closes the gripper.

##### Parameters

<code>_ecoGripper</code>	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

<code>_eudDirection</code>	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

<code>_elrDirection</code>	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

<code>_elrDirection</code>	What direction to move in.
----------------------------	----------------------------

Implements [ControlSystem.IManualController](#).

#### 7.26.2.8 void ControlSystem.ManualController.moveCoordPitch ( enumIncDec \_eidIncOrDec )

Change the wrists pitch.

##### Parameters

<code>_eidIncOrDec</code>	Increasing or decreasing.
---------------------------	---------------------------

Implements [ControlSystem.IManualController](#).

#### 7.26.2.9 void ControlSystem.ManualController.moveCoordRoll ( enumIncDec \_eidIncOrDec )

Change the roll of the wrist.

##### Parameters

<code>_eidIncOrDec</code>	Increasing or decreasing.
---------------------------	---------------------------

Implements [ControlSystem.IManualController](#).

#### 7.26.2.10 void ControlSystem.ManualController.moveCoordX ( enumIncDec \_eidIncOrDec )

Change the robots X-coordinate.

##### Parameters

<code>_eidIncOrDec</code>	Increasing or decreasing.
---------------------------	---------------------------

Implements [ControlSystem.IManualController](#).

#### 7.26.2.11 void ControlSystem.ManualController.moveCoordY ( enumIncDec \_eidIncOrDec )

Change the robots Y-coordinate.



## Parameters

<code>_eidIncOrDec</code>	Increasing or decreasing.
---------------------------	---------------------------

Implements [ControlSystem.IManualController](#).

#### 7.26.2.12 void ControlSystem.ManualController.moveCoordZ ( enumIncDec \_eidIncOrDec )

Change the robots Z-coordinate.

## Parameters

<code>_eidIncOrDec</code>	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](#).

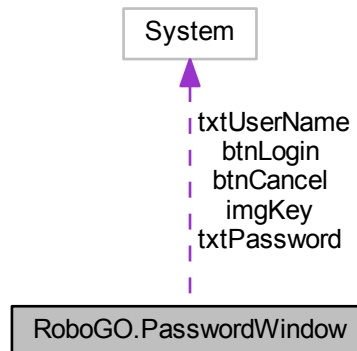
The documentation for this class was generated from the following file:

- 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

<code>_loginName</code>	User name.
<code>_loginPassword</code>	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 explicitconstructor.*
- 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](#) explicitconstructor.

### 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

<code>_vect</code>	The new position.
--------------------	-------------------

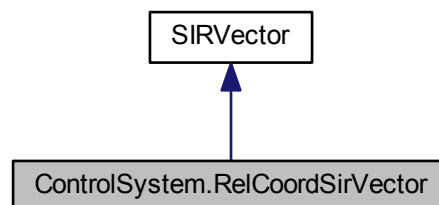
The documentation for this class was generated from the following file:

- RoboGO/ViewModels/[positionViewModel.cs](#)

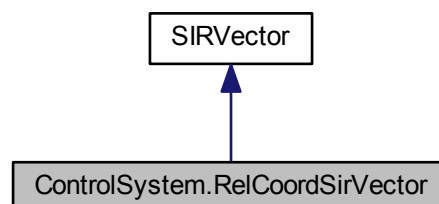
## 7.31 ControlSystem.RelCoordSirVector Class Reference

[SIRVector](#) class for relative positions.

Inheritance diagram for ControlSystem.RelCoordSirVector:



Collaboration diagram for ControlSystem.RelCoordSirVector:



## Public Member Functions

- [RelCoordSirVector](#) (string \_sName)  
*Constructors 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 )`

Constructors whichs sets up type and name of vector.

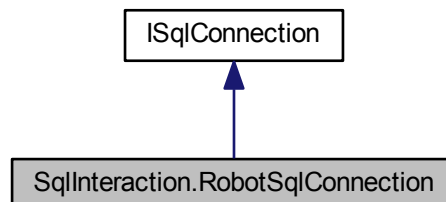
The documentation for this class was generated from the following file:

- `ControlSystem/wrapper.cs`

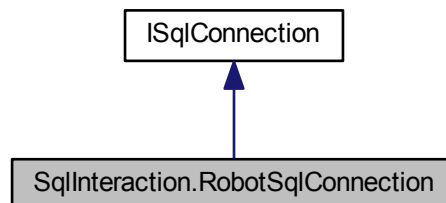
## 7.32 SqlInteraction.RobotSqlConnection Class Reference

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

<a href="#">connectionstring</a>	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](#).

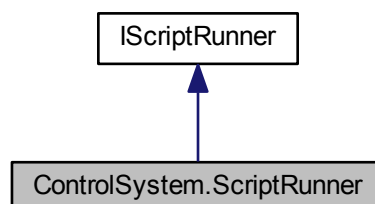
The documentation for this class was generated from the following file:

- `SqlInteraction/robotSqlConnection.cs`

## 7.33 ControlSystem.ScriptRunner Class Reference

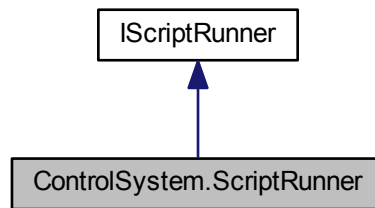
Used to run IronPython scripts.

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

<i>_iroboRobot</i>	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

<i>port</i>	Comport to communicate through
<i>baud</i>	baudrate (has to be the same as the STK-kit)
<i>parity</i>	Parity
<i>dataBits</i>	Databits
<i>stopBits</i>	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 milliseconds.*
  - 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

<i>speed</i>	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**

<i>_x</i>	x-coordinate
<i>_y</i>	y-coordinate
<i>_z</i>	z-coordinate
<i>_pitch</i>	pitch robot arm
<i>_roll</i>	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**

<i>pitch</i>	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**

<i>roll</i>	value for the roll coordinate
-------------	-------------------------------

**Returns**

Always true.

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

Moves just X coordinate.

**Parameters**

<i>x</i>	value for the x coordinate
----------	----------------------------

**Returns**

Always true.

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

Moves just Y coordinate.

**Parameters**

<i>y</i>	value for the y coordinate
----------	----------------------------

**Returns**

Always true.

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

Moves just Z coordinate.

**Parameters**

<i>z</i>	value for the z coordinate
----------	----------------------------

**Returns**

Always true.



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

Moving the Conveyer Belt.

**Parameters**

<i>speed</i>	speed
--------------	-------

**Returns**

Always true.

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

Moving the elbow.

**Parameters**

<i>speed</i>	speed
--------------	-------

**Returns**

Always true.

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

Moving the gripper.

**Parameters**

<i>speed</i>	speed
--------------	-------

**Returns**

Always true.

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

Moving the shoulder.

**Parameters**

<i>speed</i>	speed
--------------	-------

**Returns**

Always true.

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

Moving the wrist Pitch.

## Parameters

<i>speed</i>	speed
--------------	-------

## Returns

Always true.

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

Moving the Wrist Roll.

## Parameters

<i>speed</i>	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

<i>_bGroup</i>	Part of the robot
----------------	-------------------

///

## Parameters

<i>_mSpeed</i>	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 milliseconds.

#### Parameters

<code>_bGroup</code>	Part of the robot
----------------------	-------------------

///

#### Parameters

<code>_mTime</code>	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.

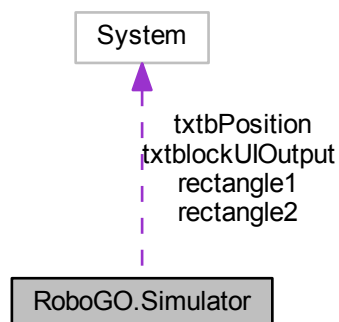
The documentation for this class was generated from the following file:

- 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.

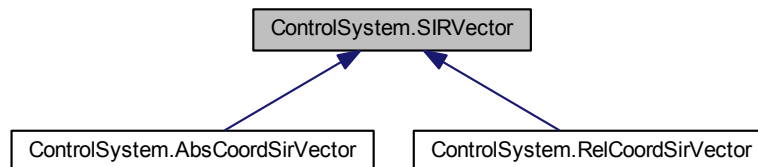
The documentation for this class was generated from the following file:

- 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](#) > **IstPoints**

- 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

<code>pNewVecPoint</code>	
---------------------------	--

#### 7.38.2.2 `VecPoint ControlSystem.SIRVector.getPoint ( int iIndex )`

Gets a point from the vector.

##### Parameters

<code>iIndex</code>	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.)

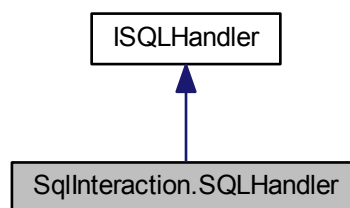
The documentation for this class was generated from the following file:

- `ControlSystem/wrapper.cs`

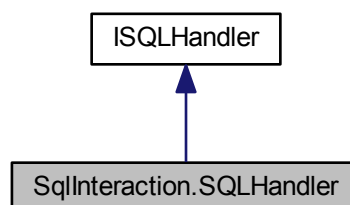
## 7.39 SqlInteraction.SQLHandler Class Reference

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

<a href="#">_command</a>	Parameter is the command made in <a href="#">makeCommand</a>
<a href="#">_parameter-Name</a>	Parameter is name of the parameter defined in <a href="#">makeCommand</a>
<a href="#">_parameter-Value</a>	Parameter is value to be used in the command
<a href="#">_parameterType</a>	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

<a href="#">_parameter</a>	Parameter is part of or full parameter name to change
<a href="#">_parameter-Value</a>	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

<i>_commandText</i>	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

<i>_command</i>	Parameter is command to be executed
<i>queryType</i>	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

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

#### Returns

Returns bool for whether connection was successfully 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.ISQLHandler.GetInstance [static, get]

Variable to get the [SQLHandler](#) singleton instance.

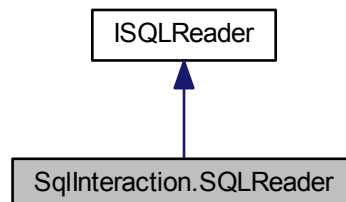
The documentation for this class was generated from the following file:

- SqlInteraction/sqlHandler.cs

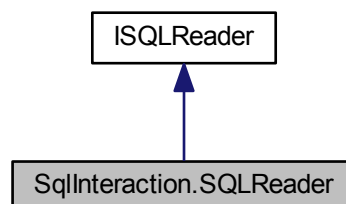
## 7.40 SqlInteraction.SQLReader Class Reference

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

<code>_sqlDataReader</code>	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.

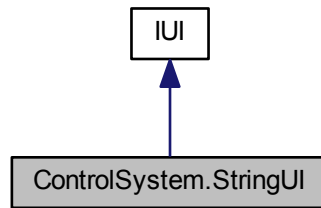
The documentation for this class was generated from the following file:

- [SqlInteraction/sqlReader.cs](#)

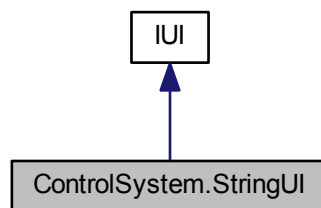
## 7.41 ControlSystem.StringUI Class Reference

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

<i>sMsg</i>	The string with the message and argument placement.(Like normal Write())
<i>paramobj-Argument</i>	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

<i>sMsg</i>	The string with the message and argument placement.(Like normal WriteLine())
<i>paramobj-Argument</i>	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**

<i>_description</i>	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**

<i>_threadStart</i>	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({}))
<i>_description</i>	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**

<i>_parameterizedThreadStart</i>	Parameter is name of function that needs its own thread, as its parameterizedThreadStart, it means it have to be a function with parameter object: (functionname(object example){})
<i>_description</i>	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**

<i>_description</i>	String that describes 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**

<i>_description</i>	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**

<b>_description</b>	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**

<b>_description</b>	Description is a string which the function needs to search and find a thread for in a list
<b>_obj</b>	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:

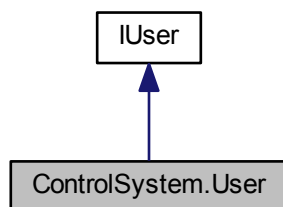
- ControlSystem/[threadHandling.cs](#)



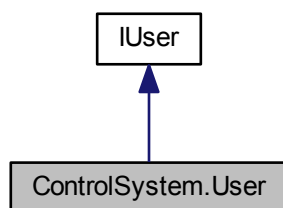
## 7.44 ControlSystem.User Class Reference

Normal user.

Inheritance diagram for ControlSystem.User:



Collaboration diagram for ControlSystem.User:



### Public Member Functions

- [User](#) ()  
*Default constructor 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 constructor 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

<code>_iX</code>	X coordinate.
<code>_iY</code>	Y coordinate.
<code>_iZ</code>	Z coordinate.
<code>_iPitch</code>	Pitch.
<code>_iRoll</code>	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.

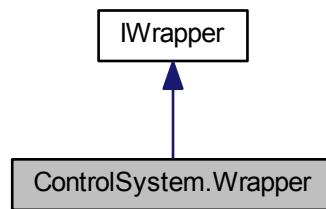
The documentation for this class was generated from the following file:

- RoboGO/ViewModels/[viewModelManualSteering.cs](#)

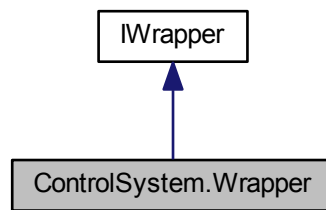
## 7.47 ControlSystem.Wrapper Class Reference

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 \_ISpeed)  
*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 \_ilIndex)  
*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 choosing axis group in certain functions.

#### 7.47.2.2 enum ControlSystem.Wrapper.enumBGroup

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

#### 7.47.2.3 enum ControlSystem.Wrapper.enumManualModeWhat

For choosing 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 choosing 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**

<i>_axisSettings-Group</i>	Axis group to affect.(Use enum)
<i>_bControlOnOr-Off</i>	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**

<i>_enumGroup</i>	Group can use: Robot(Normally used) Peripherals All
<i>_sVectorName</i>	Name of vector.
<i>_shrtLength</i>	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

<code>_enummanMoveType</code>	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

<code>_shrtPerc</code>	Data in percentage.
<code>_shrtWidth</code>	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

<code>_axisSettings-Group</code>	The axis group.(Use enum)
<code>_funcptrHoming-EventHandler</code>	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

<code>_sysmodeMode</code>	Mode.[Normally use online mode]
<code>_systypeType</code>	Type of connection.[Normally use default]
<code>_funcptrSuccess</code>	Function to be called on success.
<code>_funcptrError</code>	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**

<code>_sNameOf- Vector</code>	Name of the vector in the Robot.
<code>_iIndex</code>	Index for point

**Returns**

Returns true on successfull call.

Implements [ControlSystem.IWrapper](#).

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

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**

<code>_bGroup</code>	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
<code>_mSpeed</code>	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**

<i>vecTheSirVector</i>	The vector with the points.
------------------------	-----------------------------

**Returns**

Returns true on successful call.

Implements [ControlSystem.IWrapper](#).

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

Sets the time future movement should take.

**Parameters**

<i>_bGroup</i>	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
<i>_mTime</i>	Time in milliseconds

**Returns**

Returns true if time has been successfully 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**

<i>_funcptrCallbackEvent</i>	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

<code>_funcptr-CallbackEnd</code>	Function to be called when motion has ended.
<code>_funcptr-CallbackStart</code>	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 calculating 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 calculating 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

- abortAllAndWait
  - ControlSystem::ThreadHandling, [108](#)
- abortAndWait
  - ControlSystem::ThreadHandling, [108](#)
- AbsCoordSirVector
  - ControlSystem::AbsCoordSirVector, [19](#)
- AddEventHandler
  - XamlGeneratedNamespace::GeneratedInternalTypeHelper, [39](#)
- addParameter
  - SqlInteraction::ISQLHandler, [61](#)
  - SqlInteraction::SQLHandler, [102](#)
- addPoint
  - ControlSystem::SIRVector, [100](#)
- addThread
  - ControlSystem::ThreadHandling, [109](#)
- Admin
  - ControlSystem::Admin, [20](#)
- authenticate
  - RoboGO::ViewModels::passwordWindowViewModel, [81](#)
- Buffer
  - ControlSystem::StringUI, [107](#)
- build
  - RoboGO::ViewModels::IDEViewModel, [43](#)
- CanExecute
  - RoboGO::ViewModels::DelegateCommand, [24](#)
- CanExecuteChanged
  - RoboGO::ViewModels::DelegateCommand, [25](#)
- changeConnectionparameter
  - SqlInteraction::ISQLHandler, [62](#)
  - SqlInteraction::SQLHandler, [102](#)
- checkIsOnline
  - RoboGO::MainWindowViewModel, [75](#)
- clearOutputStream
  - ControlSystem::IScriptRunner, [58](#)
  - ControlSystem::ScriptRunner, [87](#)
- clearString
  - ControlSystem::StringUI, [107](#)
- Close
  - ControlSystem::SerialSTK, [89](#)
- close
  - SqlInteraction::ISQLReader, [63](#)
  - SqlInteraction::SQLReader, [105](#)
- CloseGripper
  - ControlSystem::DLL, [27](#)
  - ControlSystem::DLLImport, [32](#)
  - ControlSystem::IDLL, [46](#)
- RoboGO::ViewModels::ViewModelManualSteering, [117](#)
- closeGripper
  - ControlSystem::Simulator, [91](#)
  - RoboGO::ViewModels::ViewModelManualSteering, [115](#)
- closeGripperWrapped
  - ControlSystem::IWrapper, [68](#)
  - ControlSystem::Wrapper, [120](#)
- CloseManual
  - ControlSystem::DLL, [27](#)
  - ControlSystem::DLLImport, [33](#)
  - ControlSystem::IDLL, [46](#)
- closeManualWrapped
  - ControlSystem::IWrapper, [68](#)
  - ControlSystem::Wrapper, [120](#)
- closeTab
  - RoboGO::ViewModels::IDEViewModel, [43](#)
- closeTab\_CanExecute
  - RoboGO::ViewModels::IDEViewModel, [43](#)
- CloseWatchDigitalInput
  - ControlSystem::DLL, [27](#)
  - ControlSystem::DLLImport, [33](#)
  - ControlSystem::IDLL, [46](#)
- closeWatchDigitalInputWrapped
  - ControlSystem::IWrapper, [68](#)
  - ControlSystem::Wrapper, [121](#)
- CodeClear
  - RoboGO::ViewModels::IDEViewModel, [43](#)
- CodeOutput
  - RoboGO::ViewModels::IDEViewModel, [43](#)
- Connection
  - SqlInteraction::ISQLHandler, [63](#)
  - SqlInteraction::SQLHandler, [103](#)
- ConnectionClose
  - SqlInteraction::ISqlConnection, [60](#)
  - SqlInteraction::RobotSqlConnection, [85](#)
- ConnectionOpen
  - SqlInteraction::ISqlConnection, [60](#)
  - SqlInteraction::RobotSqlConnection, [85](#)
- Connectionstring
  - SqlInteraction::ISqlConnection, [60](#)
  - SqlInteraction::RobotSqlConnection, [86](#)
- Control
  - ControlSystem::DLL, [28](#)
  - ControlSystem::DLLImport, [33](#)
  - ControlSystem::IDLL, [47](#)
- ControlSystem, [13](#)
  - enumCloseOpen, [14](#)

- enumIncDec, [14](#)
- enumLeftRight, [15](#)
- enumUpDown, [15](#)
- ControlSystem.AbsCoordSirVector, [18](#)
- ControlSystem.Admin, [19](#)
- ControlSystem.ConsoleUI, [22](#)
- ControlSystem.DLL, [25](#)
- ControlSystem.DLLImport, [31](#)
- ControlSystem.ErrorReporter, [37](#)
- ControlSystem.IDLL, [45](#)
- ControlSystem.IManualController, [51](#)
- ControlSystem.IScriptRunner, [57](#)
- ControlSystem.IUI, [64](#)
- ControlSystem.IUser, [65](#)
- ControlSystem.IWrapper, [66](#)
- ControlSystem.ManualController, [75](#)
- ControlSystem.RelCoordSirVector, [83](#)
- ControlSystem.SIRVector, [99](#)
- ControlSystem.ScriptRunner, [86](#)
- ControlSystem.SerialSTK, [88](#)
- ControlSystem.Simulator, [90](#)
- ControlSystem.StringUI, [105](#)
- ControlSystem.ThreadHandling, [108](#)
- ControlSystem.ThreadHandling.ThreadHolder, [110](#)
- ControlSystem.User, [111](#)
- ControlSystem.VecPoint, [112](#)
- ControlSystem Wrapper, [117](#)
- ControlSystem/dll.cs, [129](#)
- ControlSystem/errorReporter.cs, [129](#)
- ControlSystem/factory.cs, [129](#)
- ControlSystem/iDLL.cs, [130](#)
- ControlSystem/iWrapper.cs, [130](#)
- ControlSystem/manualController.cs, [130](#)
- ControlSystem/scriptRunner.cs, [131](#)
- ControlSystem/serialSTK.cs, [131](#)
- ControlSystem/simulator.cs, [132](#)
- ControlSystem/threadHandling.cs, [132](#)
- ControlSystem/ui.cs, [132](#)
- ControlSystem/wrapper.cs, [133](#)
- ControlSystem::AbsCoordSirVector
  - AbsCoordSirVector, [19](#)
- ControlSystem::Admin
  - Admin, [20](#)
  - permissionDictionary, [20](#)
  - userName, [20](#)
- ControlSystem::ConsoleUI
  - write, [23](#)
  - writeLine, [23](#)
- ControlSystem::DLL
  - CloseGripper, [27](#)
  - CloseManual, [27](#)
  - CloseWatchDigitalInput, [27](#)
  - Control, [28](#)
  - DefineVector, [28](#)
  - DgateCallBackByteRefArg, [28](#)
  - EnterManual, [28](#)
  - GetCurrentPosition, [29](#)
  - GetJaw, [29](#)
  - IsOnLineOk, [29](#)
  - MoveLinear, [29](#)
  - MoveManual, [30](#)
  - OpenGripper, [30](#)
  - Speed, [30](#)
  - Stop, [30](#)
  - Teach, [31](#)
  - Time, [31](#)
- ControlSystem::DLLImport
  - CloseGripper, [32](#)
  - CloseManual, [33](#)
  - CloseWatchDigitalInput, [33](#)
  - Control, [33](#)
  - DefineVector, [33](#)
  - EnterManual, [33](#)
  - GetCurrentPosition, [34](#)
  - GetJaw, [34](#)
  - Home, [34](#)
  - initialization, [34](#)
  - IsOnLineOk, [35](#)
  - MoveLinear, [35](#)
  - MoveManual, [35](#)
  - OpenGripper, [35](#)
  - Speed, [36](#)
  - Stop, [36](#)
  - Teach, [36](#)
  - Time, [36](#)
  - WatchDigitalInput, [37](#)
  - WatchMotion, [37](#)
- ControlSystem::ErrorReporter
  - ErrorReported, [38](#)
- ControlSystem::IDLL
  - CloseGripper, [46](#)
  - CloseManual, [46](#)
  - CloseWatchDigitalInput, [46](#)
  - Control, [47](#)
  - DefineVector, [47](#)
  - EnterManual, [47](#)
  - GetCurrentPosition, [48](#)
  - GetJaw, [48](#)
  - Home, [48](#)
  - Initialization, [48](#)
  - IsOnLineOk, [49](#)
  - MoveLinear, [49](#)
  - MoveManual, [49](#)
  - OpenGripper, [50](#)
  - Speed, [50](#)
  - Stop, [50](#)
  - Teach, [50](#)
  - Time, [51](#)
  - WatchDigitalInput, [51](#)
  - WatchMotion, [51](#)
- ControlSystem::IManualController
  - moveAxisBase, [53](#)
  - moveAxisConveyer, [53](#)
  - moveAxisElbow, [53](#)
  - moveAxisGripper, [53](#)
  - moveAxisPitch, [53](#)

- moveAxisRoll, 53
- moveAxisShoulder, 54
- moveCoordPitch, 54
- moveCoordRoll, 54
- moveCoordX, 54
- moveCoordY, 54
- moveCoordZ, 55
- RobotConnection, 55
- Speed, 55
- stopAllMovement, 55
- ControlSystem::IScriptRunner
  - clearOutputStream, 58
  - ExecuteScript, 58
  - readFromOutputStream, 58
  - setRobotInstance, 58
  - setScriptFromFile, 58
  - setScriptFromString, 59
- ControlSystem::IUI
  - write, 65
  - writeLine, 65
- ControlSystem::IUser
  - permissionDictionary, 66
  - userName, 66
- ControlSystem::IWrapper
  - closeGripperWrapped, 68
  - closeManualWrapped, 68
  - closeWatchDigitalInputWrapped, 68
  - controlWrapped, 68
  - defineVectorWrapped, 68
  - enterManualWrapped, 69
  - getCurrentPosition, 69
  - getJawWrapped, 69
  - homeWrapped, 69
  - initializationWrapped, 70
  - isOnlineOkWrapped, 70
  - moveLinearWrapped, 70
  - moveManualWrapped, 71
  - openGripperWrapped, 71
  - speedWrapped, 71
  - stopWrapped, 71
  - teachWrapped, 72
  - timeWrapped, 72
  - watchDigitalInputWrapped, 72
  - watchMotionWrapped, 72
- ControlSystem::ManualController
  - moveAxisBase, 77
  - moveAxisConveyer, 77
  - moveAxisElbow, 77
  - moveAxisGripper, 77
  - moveAxisPitch, 77
  - moveAxisRoll, 78
  - moveAxisShoulder, 78
  - moveCoordPitch, 78
  - moveCoordRoll, 78
  - moveCoordX, 78
  - moveCoordY, 78
  - moveCoordZ, 79
  - RobotConnection, 79
  - Speed, 79
  - stopAllMovement, 79
- ControlSystem::RelCoordSirVector
  - RelCoordSirVector, 84
- ControlSystem::SIRVector
  - addPoint, 100
  - getPoint, 100
  - getSize, 100
  - iType, 100
  - Name, 101
  - Type, 101
- ControlSystem::ScriptRunner
  - clearOutputStream, 87
  - ExecuteScript, 87
  - getInstance, 88
  - readFromOutputStream, 88
  - setRobotInstance, 88
  - setScriptFromFile, 88
  - setScriptFromString, 88
- ControlSystem::SerialSTK
  - Close, 89
  - Open, 89
  - ReadADC, 89
  - SerialSTK, 89
- ControlSystem::Simulator
  - closeGripper, 91
  - Currentposition, 97
  - getCurrentPosition, 91
  - getJawOpeningWidthMilimeters, 92
  - getJawOpeningWidthPercentage, 92
  - homeRobot, 92
  - IUIOutput, 97
  - isOnline, 92
  - moveBase, 92
  - moveByAbsoluteCoordinates, 92
  - moveByCoordinates, 93
  - moveByPitch, 93
  - moveByRelativeCoordinates, 93
  - moveByRoll, 94
  - moveByXCoordinate, 94
  - moveByYCoordinate, 94
  - moveByZCoordinate, 94
  - moveConveyerBelt, 94
  - moveElbow, 95
  - moveGripper, 95
  - moveShoulder, 95
  - moveWristPitch, 95
  - moveWristRoll, 96
  - movebyCoordinates, 93
  - openGripper, 96
  - Simulator, 91
  - Speed, 96
  - stopAllMovement, 96
  - Time, 96
- ControlSystem::StringUI
  - Buffer, 107
  - clearString, 107
  - StringUI, 107

- write, 107
- writeLine, 107
- ControlSystem::ThreadHandling
  - abortAllAndWait, 108
  - abortAndWait, 108
  - addThread, 109
  - find, 109
  - removeThread, 109
  - start, 109, 110
  - ThreadHandling, 108
- ControlSystem::ThreadHandling::ThreadHolder
  - stringDescription, 110
  - threadPlaceholder, 110
- ControlSystem::User
  - permissionDictionary, 112
  - User, 112
  - userName, 112
- ControlSystem::VecPoint
  - ToString, 113
  - VecPoint, 112
- ControlSystem::Wrapper
  - closeGripperWrapped, 120
  - closeManualWrapped, 120
  - closeWatchDigitalInputWrapped, 121
  - controlWrapped, 121
  - DLL, 126
  - defineVectorWrapped, 121
  - enterManualWrapped, 121
  - enumAxisSettings, 120
  - enumBGroup, 120
  - enumManualModeWhat, 120
  - enumManualType, 120
  - enumSystemModes, 120
  - enumSystemTypes, 120
  - getCurrentPosition, 122
  - getInstance, 122
  - getJawWrapped, 122
  - homeWrapped, 122
  - initializationWrapped, 123
  - isOnlineOkWrapped, 123
  - moveLinearWrapped, 123
  - moveManualWrapped, 124
  - openGripperWrapped, 124
  - speedWrapped, 124
  - stopWrapped, 124
  - teachWrapped, 124
  - timeWrapped, 125
  - watchDigitalInputWrapped, 125
  - watchMotionWrapped, 125
- controlWrapped
  - ControlSystem::IWrapper, 68
  - ControlSystem::Wrapper, 121
- CreateCommand
  - SqlInteraction::ISqlConnection, 60
  - SqlInteraction::RobotSqlConnection, 85
- CreateDelegate
  - XamlGeneratedNamespace::GeneratedInternalTypeHelper, 39
- CreateInstance
  - XamlGeneratedNamespace::GeneratedInternalTypeHelper, 39
- CurrentPosition
  - RoboGO::ViewModels::SimulatorViewModel, 99
- Currentposition
  - ControlSystem::Simulator, 97
- DLL
  - ControlSystem::Wrapper, 126
- DefineVector
  - ControlSystem::DLL, 28
  - ControlSystem::DLLImport, 33
  - ControlSystem::IDLL, 47
- defineVectorWrapped
  - ControlSystem::IWrapper, 68
  - ControlSystem::Wrapper, 121
- DelegateCommand
  - RoboGO::ViewModels::DelegateCommand, 24
- DgateCallBackByteRefArg
  - ControlSystem::DLL, 28
- elbow
  - RoboGO::ViewModels::XYCalculate, 127
- elbowRotate
  - RoboGO::ViewModels::XYCalculate, 127
- EnterManual
  - ControlSystem::DLL, 28
  - ControlSystem::DLLImport, 33
  - ControlSystem::IDLL, 47
- enterManualWrapped
  - ControlSystem::IWrapper, 69
  - ControlSystem::Wrapper, 121
- enumAxisSettings
  - ControlSystem::Wrapper, 120
- enumBGroup
  - ControlSystem::Wrapper, 120
- enumCloseOpen
  - ControlSystem, 14
- enumIncDec
  - ControlSystem, 14
- enumLeftRight
  - ControlSystem, 15
- enumManualModeWhat
  - ControlSystem::Wrapper, 120
- enumManualType
  - ControlSystem::Wrapper, 120
- enumSystemModes
  - ControlSystem::Wrapper, 120
- enumSystemTypes
  - ControlSystem::Wrapper, 120
- enumUpDown
  - ControlSystem, 15
- ErrorReported
  - ControlSystem::ErrorReporter, 38
- Execute
  - RoboGO::ViewModels::DelegateCommand, 25
- executeCode
  - RoboGO::ViewModels::IDEViewModel, 43

- ExecuteCmd
  - RoboGO::ViewModels::IDEViewModel, [43](#)
- ExecuteScript
  - ControlSystem::IScriptRunner, [58](#)
  - ControlSystem::ScriptRunner, [87](#)
- find
  - ControlSystem::ThreadHandling, [109](#)
- GetCurrentPosition
  - ControlSystem::DLL, [29](#)
  - ControlSystem::DLLImport, [34](#)
  - ControlSystem::IDLL, [48](#)
- getCurrentPosition
  - ControlSystem::IWrapper, [69](#)
  - ControlSystem::Simulator, [91](#)
  - ControlSystem::Wrapper, [122](#)
- GetInstance
  - SqlInteraction::SQLHandler, [103](#)
- getInstance
  - ControlSystem::ScriptRunner, [88](#)
  - ControlSystem::Wrapper, [122](#)
- GetJaw
  - ControlSystem::DLL, [29](#)
  - ControlSystem::DLLImport, [34](#)
  - ControlSystem::IDLL, [48](#)
- getJawOpeningWidthMilimeters
  - ControlSystem::Simulator, [92](#)
- getJawOpeningWidthPercentage
  - ControlSystem::Simulator, [92](#)
- getJawWrapped
  - ControlSystem::IWrapper, [69](#)
  - ControlSystem::Wrapper, [122](#)
- getPoint
  - ControlSystem::SIRVector, [100](#)
- GetPropertyValue
  - XamlGeneratedNamespace::GeneratedInternalTypeHelper, [39](#), [40](#)
- getSize
  - ControlSystem::SIRVector, [100](#)
- getTableInfo
  - RoboGO::ViewModels::InfoViewModel, [56](#)
- getXYZPR
  - RoboGO::ViewModels::PositionViewModel, [82](#)
- gripper
  - RoboGO::ViewModels::XYCalculate, [127](#)
- gripperX
  - RoboGO::ViewModels::XYCalculate, [127](#)
- gripperY
  - RoboGO::ViewModels::XYCalculate, [127](#)
- Home
  - ControlSystem::DLLImport, [34](#)
  - ControlSystem::IDLL, [48](#)
- homeRobot
  - ControlSystem::Simulator, [92](#)
- homeWrapped
  - ControlSystem::IWrapper, [69](#)
  - ControlSystem::Wrapper, [122](#)
- IDEViewModel
  - RoboGO::ViewModels::IDEViewModel, [43](#)
- iType
  - ControlSystem::SIRVector, [100](#)
- IUIOutput
  - ControlSystem::Simulator, [97](#)
- IdeTabs
  - RoboGO::ViewModels::IDEViewModel, [43](#)
- InfoViewModel
  - RoboGO::ViewModels::InfoViewModel, [56](#)
- Initialization
  - ControlSystem::IDLL, [48](#)
- initialization
  - ControlSystem::DLLImport, [34](#)
- initializationWrapped
  - ControlSystem::IWrapper, [70](#)
  - ControlSystem::Wrapper, [123](#)
- InitializeComponent
  - RoboGO::aboutBox, [17](#)
  - RoboGO::App, [21](#)
  - RoboGO::CommandsWindow, [22](#)
  - RoboGO::GUIManualSteering, [41](#)
  - RoboGO::MainWindow, [74](#)
  - RoboGO::PasswordWindow, [80](#)
  - RoboGO::Simulator, [98](#)
- IsOnLineOk
  - ControlSystem::DLL, [29](#)
  - ControlSystem::DLLImport, [35](#)
  - ControlSystem::IDLL, [49](#)
- isOnline
  - ControlSystem::Simulator, [92](#)
- isOnlineOkWrapped
  - ControlSystem::IWrapper, [70](#)
  - ControlSystem::Wrapper, [123](#)
- loadAllTables
  - RoboGO::ViewModels::InfoViewModel, [56](#)
- Main
  - RoboGO::App, [21](#)
- MainWindowViewModel
  - RoboGO::MainWindowViewModel, [74](#)
- makeCommand
  - SqlInteraction::ISQLHandler, [62](#)
  - SqlInteraction::SQLHandler, [102](#)
- ManualControl
  - RoboGO::ViewModels::ViewModelManualSteering, [117](#)
- moveAxisBase
  - ControlSystem::IManualController, [53](#)
  - ControlSystem::ManualController, [77](#)
- moveAxisBaseLeft
  - RoboGO::ViewModels::ViewModelManualSteering, [115](#)
- moveAxisBaseRight
  - RoboGO::ViewModels::ViewModelManualSteering, [115](#)
- moveAxisConveyer
  - ControlSystem::IManualController, [53](#)

- ControlSystem::ManualController, 77
- moveAxisConveyerLeft
  - RoboGO::ViewModels::ViewModelManualSteering, 115
- moveAxisConveyerRight
  - RoboGO::ViewModels::ViewModelManualSteering, 115
- moveAxisElbow
  - ControlSystem::IManualController, 53
  - ControlSystem::ManualController, 77
- moveAxisElbowLeft
  - RoboGO::ViewModels::ViewModelManualSteering, 115
- moveAxisElbowRight
  - RoboGO::ViewModels::ViewModelManualSteering, 115
- moveAxisGripper
  - ControlSystem::IManualController, 53
  - ControlSystem::ManualController, 77
- moveAxisPitch
  - ControlSystem::IManualController, 53
  - ControlSystem::ManualController, 77
- moveAxisPitchDown
  - RoboGO::ViewModels::ViewModelManualSteering, 115
- moveAxisPitchUp
  - RoboGO::ViewModels::ViewModelManualSteering, 115
- moveAxisRoll
  - ControlSystem::IManualController, 53
  - ControlSystem::ManualController, 78
- moveAxisRollLeft
  - RoboGO::ViewModels::ViewModelManualSteering, 115
- moveAxisRollRight
  - RoboGO::ViewModels::ViewModelManualSteering, 115
- moveAxisShoulder
  - ControlSystem::IManualController, 54
  - ControlSystem::ManualController, 78
- moveAxisShoulderLeft
  - RoboGO::ViewModels::ViewModelManualSteering, 116
- moveAxisShoulderRight
  - RoboGO::ViewModels::ViewModelManualSteering, 116
- moveBase
  - ControlSystem::Simulator, 92
- moveByAbsoluteCoordinates
  - ControlSystem::Simulator, 92
- moveByCoordinates
  - ControlSystem::Simulator, 93
- moveByPitch
  - ControlSystem::Simulator, 93
- moveByRelativeCoordinates
  - ControlSystem::Simulator, 93
- moveByRoll
  - ControlSystem::Simulator, 94
- moveByXCoordinate
  - ControlSystem::Simulator, 94
- moveByYCoordinate
  - ControlSystem::Simulator, 94
- moveByZCoordinate
  - ControlSystem::Simulator, 94
- moveConveyerBelt
  - ControlSystem::Simulator, 94
- moveCoordPitch
  - ControlSystem::IManualController, 54
  - ControlSystem::ManualController, 78
- moveCoordPitchDecreasing
  - RoboGO::ViewModels::ViewModelManualSteering, 116
- moveCoordPitchIncreasing
  - RoboGO::ViewModels::ViewModelManualSteering, 116
- moveCoordRoll
  - ControlSystem::IManualController, 54
  - ControlSystem::ManualController, 78
- moveCoordRollDecreasing
  - RoboGO::ViewModels::ViewModelManualSteering, 116
- moveCoordRollIncreasing
  - RoboGO::ViewModels::ViewModelManualSteering, 116
- moveCoordX
  - ControlSystem::IManualController, 54
  - ControlSystem::ManualController, 78
- moveCoordXDecreasing
  - RoboGO::ViewModels::ViewModelManualSteering, 116
- moveCoordXIncreasing
  - RoboGO::ViewModels::ViewModelManualSteering, 116
- moveCoordY
  - ControlSystem::IManualController, 54
  - ControlSystem::ManualController, 78
- moveCoordYDecreasing
  - RoboGO::ViewModels::ViewModelManualSteering, 116
- moveCoordYIncreasing
  - RoboGO::ViewModels::ViewModelManualSteering, 116
- moveCoordZ
  - ControlSystem::IManualController, 55
  - ControlSystem::ManualController, 79
- moveCoordZDecreasing
  - RoboGO::ViewModels::ViewModelManualSteering, 116
- moveCoordZIncreasing
  - RoboGO::ViewModels::ViewModelManualSteering, 116
- moveElbow
  - ControlSystem::Simulator, 95
- moveGripper
  - ControlSystem::Simulator, 95
- MoveLinear



- ControlSystem::DLL, 29
- ControlSystem::DLLImport, 35
- ControlSystem::IDLL, 49
- moveLinearWrapped
  - ControlSystem::IWrapper, 70
  - ControlSystem::Wrapper, 123
- MoveManual
  - ControlSystem::DLL, 30
  - ControlSystem::DLLImport, 35
  - ControlSystem::IDLL, 49
- moveManualWrapped
  - ControlSystem::IWrapper, 71
  - ControlSystem::Wrapper, 124
- moveShoulder
  - ControlSystem::Simulator, 95
- moveWristPitch
  - ControlSystem::Simulator, 95
- moveWristRoll
  - ControlSystem::Simulator, 96
- movebyCoordinates
  - ControlSystem::Simulator, 93
- Name
  - ControlSystem::SIRVector, 101
- newTab
  - RoboGO::ViewModels::IDEViewModel, 43
- newTab\_CanExecute
  - RoboGO::ViewModels::IDEViewModel, 44
- Open
  - ControlSystem::SerialSTK, 89
- open
  - RoboGO::ViewModels::IDEViewModel, 44
- open\_CanExecute
  - RoboGO::ViewModels::IDEViewModel, 44
- OpenGripper
  - ControlSystem::DLL, 30
  - ControlSystem::DLLImport, 35
  - ControlSystem::IDLL, 50
  - RoboGO::ViewModels::ViewModelManualSteering, 117
- openGripper
  - ControlSystem::Simulator, 96
  - RoboGO::ViewModels::ViewModelManualSteering, 117
- openGripperWrapped
  - ControlSystem::IWrapper, 71
  - ControlSystem::Wrapper, 124
- passwordWindowViewModel
  - RoboGO::ViewModels::passwordWindowViewModel, 81
- permissionDictionary
  - ControlSystem::Admin, 20
  - ControlSystem::User, 66
  - ControlSystem::User, 112
- PositionVec
  - RoboGO::ViewModels::PositionModel, 82
- PositionViewModel
  - RoboGO::ViewModels::PositionViewModel, 82
- PropertyChanged
  - RoboGO::ViewModels::IDEViewModel, 44
  - RoboGO::ViewModels::InfoViewModel, 57
- ReadADC
  - ControlSystem::SerialSTK, 89
- readFromOutputStream
  - ControlSystem::IScriptRunner, 58
  - ControlSystem::ScriptRunner, 88
- readRow
  - SqlInteraction::ISQLReader, 64
  - SqlInteraction::SQLReader, 105
- RelCoordSirVector
  - ControlSystem::RelCoordSirVector, 84
- removeThread
  - ControlSystem::ThreadHandling, 109
- RoboGO, 15
- RoboGO.aboutBox, 17
- RoboGO.App, 20
- RoboGO.CommandsWindow, 21
- RoboGO.GUIManualSteering, 40
- RoboGO.MainWindow, 73
- RoboGO.MainWindowViewModel, 74
- RoboGO.PasswordWindow, 79
- RoboGO.Properties, 15
- RoboGO.Simulator, 97
- RoboGO.ViewModels, 16
- RoboGO.ViewModels.DelegateCommand, 24
- RoboGO.ViewModels.IDEViewModel, 42
- RoboGO.ViewModels.InfoViewModel, 55
- RoboGO.ViewModels.passwordWindowViewModel, 80
- RoboGO.ViewModels.PositionModel, 81
- RoboGO.ViewModels.PositionViewModel, 82
- RoboGO.ViewModels.SimulatorViewModel, 98
- RoboGO.ViewModels.ViewModelManualSteering, 113
- RoboGO.ViewModels.XYCalculate, 126
- RoboGO/ViewModels/delegateCommand.cs, 133
- RoboGO/ViewModels/ideViewModel.cs, 133
- RoboGO/ViewModels/infoViewModel.cs, 134
- RoboGO/ViewModels/passwordWindowViewModel.cs, 134
- RoboGO/ViewModels/positionModel.cs, 134
- RoboGO/ViewModels/positionViewModel.cs, 134
- RoboGO/ViewModels/simulatorViewModel.cs, 135
- RoboGO/ViewModels/viewModelManualSteering.cs, 135
- RoboGO::App
  - InitializeComponent, 21
  - Main, 21
- RoboGO::CommandsWindow
  - InitializeComponent, 22
- RoboGO::GUIManualSteering
  - InitializeComponent, 41
- RoboGO::MainWindow
  - InitializeComponent, 74
- RoboGO::MainWindowViewModel
  - checkIsOnline, 75
  - MainWindowViewModel, 74

- setRobotAsRobotInstance, [75](#)
  - setSimulatorAsRobotInstance, [75](#)
  - stopRobotInstance, [75](#)
- RoboGO::PasswordWindow
  - InitializeComponent, [80](#)
- RoboGO::Simulator
  - InitializeComponent, [98](#)
- RoboGO::ViewModels::DelegateCommand
  - CanExecute, [24](#)
  - CanExecuteChanged, [25](#)
  - DelegateCommand, [24](#)
  - Execute, [25](#)
- RoboGO::ViewModels::IDEViewModel
  - build, [43](#)
  - closeTab, [43](#)
  - closeTab\_CanExecute, [43](#)
  - CodeClear, [43](#)
  - CodeOutput, [43](#)
  - executeCode, [43](#)
  - ExecuteComd, [43](#)
  - IDEViewModel, [43](#)
  - IdeTabs, [43](#)
  - newTab, [43](#)
  - newTab\_CanExecute, [44](#)
  - open, [44](#)
  - open\_CanExecute, [44](#)
  - PropertyChanged, [44](#)
  - saveAs, [44](#)
  - saveAs\_CanExecute, [44](#)
  - ScriptExecuter, [44](#)
- RoboGO::ViewModels::InfoViewModel
  - getTableInfo, [56](#)
  - InfoViewModel, [56](#)
  - loadAllTables, [56](#)
  - PropertyChanged, [57](#)
  - tablePrint, [56](#)
  - tableSave, [57](#)
  - TableValues, [57](#)
  - Tables, [57](#)
- RoboGO::ViewModels::PositionModel
  - PositionVec, [82](#)
- RoboGO::ViewModels::PositionViewModel
  - getXYZPR, [82](#)
  - PositionViewModel, [82](#)
  - update, [82](#), [83](#)
- RoboGO::ViewModels::SimulatorViewModel
  - CurrentPosition, [99](#)
  - SimulatorViewModel, [99](#)
  - UIText, [99](#)
- RoboGO::ViewModels::ViewModelManualSteering
  - CloseGripper, [117](#)
  - closeGripper, [115](#)
  - ManualControl, [117](#)
  - moveAxisBaseLeft, [115](#)
  - moveAxisBaseRight, [115](#)
  - moveAxisConveyerLeft, [115](#)
  - moveAxisConveyerRight, [115](#)
  - moveAxisElbowLeft, [115](#)
  - moveAxisElbowRight, [115](#)
  - moveAxisPitchDown, [115](#)
  - moveAxisPitchUp, [115](#)
  - moveAxisRollLeft, [115](#)
  - moveAxisRollRight, [115](#)
  - moveAxisShoulderLeft, [116](#)
  - moveAxisShoulderRight, [116](#)
  - moveCoordPitchDecreasing, [116](#)
  - moveCoordPitchIncreasing, [116](#)
  - moveCoordRollDecreasing, [116](#)
  - moveCoordRollIncreasing, [116](#)
  - moveCoordXDecreasing, [116](#)
  - moveCoordXIncreasing, [116](#)
  - moveCoordYDecreasing, [116](#)
  - moveCoordYIncreasing, [116](#)
  - moveCoordZDecreasing, [116](#)
  - moveCoordZIncreasing, [116](#)
  - OpenGripper, [117](#)
  - openGripper, [117](#)
  - SeekHome, [117](#)
  - seekHome, [117](#)
  - Speed, [117](#)
  - stopMovement, [117](#)
  - ViewModelManualSteering, [115](#)
- RoboGO::ViewModels::XYCalculate
  - elbow, [127](#)
  - elbowRotate, [127](#)
  - gripper, [127](#)
  - gripperX, [127](#)
  - gripperY, [127](#)
  - XYCalculate, [126](#)
- RoboGO::ViewModels::passwordWindowViewModel
  - authenticate, [81](#)
  - passwordWindowViewModel, [81](#)
- RoboGO::aboutBox
  - InitializeComponent, [17](#)
- RobotConnection
  - ControlSystem::IManualController, [55](#)
  - ControlSystem::ManualController, [79](#)
- RobotConnectionState
  - SqlInteraction::ISqlConnection, [60](#)
  - SqlInteraction::RobotSqlConnection, [86](#)
- RobotSqlConnection
  - SqlInteraction::RobotSqlConnection, [85](#)
- runQuery
  - SqlInteraction::ISQLHandler, [62](#)
  - SqlInteraction::SQLHandler, [103](#)
- SQLCoreReader
  - SqlInteraction::SQLReader, [105](#)
- SQLReader
  - SqlInteraction::SQLReader, [105](#)
- saveAs
  - RoboGO::ViewModels::IDEViewModel, [44](#)
- saveAs\_CanExecute
  - RoboGO::ViewModels::IDEViewModel, [44](#)
- ScriptExecuter
  - RoboGO::ViewModels::IDEViewModel, [44](#)
- SeekHome

- RoboGO::ViewModels::ViewModelManualSteering, 117
- seekHome
  - RoboGO::ViewModels::ViewModelManualSteering, 117
- SerialSTK
  - ControlSystem::SerialSTK, 89
- setConnection
  - SqlInteraction::ISQLHandler, 62
  - SqlInteraction::SQLHandler, 103
- SetPropertyValue
  - XamlGeneratedNamespace::GeneratedInternalTypeHelper, 40
- setRobotAsRobotInstance
  - RoboGO::MainWindowViewModel, 75
- setRobotInstance
  - ControlSystem::IScriptRunner, 58
  - ControlSystem::ScriptRunner, 88
- setScriptFromFile
  - ControlSystem::IScriptRunner, 58
  - ControlSystem::ScriptRunner, 88
- setScriptFromString
  - ControlSystem::IScriptRunner, 59
  - ControlSystem::ScriptRunner, 88
- setSimulatorAsRobotInstance
  - RoboGO::MainWindowViewModel, 75
- Simulator
  - ControlSystem::Simulator, 91
- SimulatorViewModel
  - RoboGO::ViewModels::SimulatorViewModel, 99
- Speed
  - ControlSystem::DLL, 30
  - ControlSystem::DLLImport, 36
  - ControlSystem::IDLL, 50
  - ControlSystem::IManualController, 55
  - ControlSystem::ManualController, 79
  - ControlSystem::Simulator, 96
  - RoboGO::ViewModels::ViewModelManualSteering, 117
- speedWrapped
  - ControlSystem::IWrapper, 71
  - ControlSystem::Wrapper, 124
- SqlInteraction, 16
- SqlInteraction.ISQLHandler, 60
- SqlInteraction.ISQLReader, 63
- SqlInteraction.ISqlConnection, 59
- SqlInteraction.RobotSqlConnection, 84
- SqlInteraction.SQLHandler, 101
- SqlInteraction.SQLReader, 104
- SqlInteraction/iSQLHandler.cs, 135
- SqlInteraction/sqlReader.cs, 136
- SqlInteraction::ISQLHandler
  - addParameter, 61
  - changeConnectionparameter, 62
  - Connection, 63
  - makeCommand, 62
  - runQuery, 62
  - setConnection, 62
- SqlInteraction::ISQLReader
  - close, 63
  - readRow, 64
- SqlInteraction::ISqlConnection
  - ConnectionClose, 60
  - ConnectionOpen, 60
  - ConnectionString, 60
  - CreateCommand, 60
  - RobotConnectionState, 60
  - Timeout, 60
- SqlInteraction::RobotSqlConnection
  - ConnectionClose, 85
  - ConnectionOpen, 85
  - ConnectionString, 86
  - CreateCommand, 85
  - RobotConnectionState, 86
  - RobotSqlConnection, 85
  - Timeout, 86
- SqlInteraction::SQLHandler
  - addParameter, 102
  - changeConnectionparameter, 102
  - Connection, 103
  - GetInstance, 103
  - makeCommand, 102
  - runQuery, 103
  - setConnection, 103
- SqlInteraction::SQLReader
  - close, 105
  - readRow, 105
  - SQLCoreReader, 105
  - SQLReader, 105
- start
  - ControlSystem::ThreadHandling, 109, 110
- Stop
  - ControlSystem::DLL, 30
  - ControlSystem::DLLImport, 36
  - ControlSystem::IDLL, 50
- stopAllMovement
  - ControlSystem::IManualController, 55
  - ControlSystem::ManualController, 79
  - ControlSystem::Simulator, 96
- stopMovement
  - RoboGO::ViewModels::ViewModelManualSteering, 117
- stopRobotInstance
  - RoboGO::MainWindowViewModel, 75
- stopWrapped
  - ControlSystem::IWrapper, 71
  - ControlSystem::Wrapper, 124
- stringDescription
  - ControlSystem::ThreadHandling::ThreadHolder, 110
- StringUI
  - ControlSystem::StringUI, 107
- tablePrint
  - RoboGO::ViewModels::InfoViewModel, 56
- tableSave
  - RoboGO::ViewModels::InfoViewModel, 57

TableValues  
     RoboGO::ViewModels::InfoViewModel, 57  
 Tables  
     RoboGO::ViewModels::InfoViewModel, 57  
 Teach  
     ControlSystem::DLL, 31  
     ControlSystem::DLLImport, 36  
     ControlSystem::IDLL, 50  
 teachWrapped  
     ControlSystem::IWrapper, 72  
     ControlSystem::Wrapper, 124  
 ThreadHandling  
     ControlSystem::ThreadHandling, 108  
 threadPlaceholder  
     ControlSystem::ThreadHandling::ThreadHolder, 110  
 Time  
     ControlSystem::DLL, 31  
     ControlSystem::DLLImport, 36  
     ControlSystem::IDLL, 51  
     ControlSystem::Simulator, 96  
 TimeOut  
     SqlInteraction::ISqlConnection, 60  
     SqlInteraction::RobotSqlConnection, 86  
 timeWrapped  
     ControlSystem::IWrapper, 72  
     ControlSystem::Wrapper, 125  
 ToString  
     ControlSystem::VecPoint, 113  
 Type  
     ControlSystem::SIRVector, 101  
  
 UIText  
     RoboGO::ViewModels::SimulatorViewModel, 99  
 update  
     RoboGO::ViewModels::PositionViewModel, 82, 83  
 User  
     ControlSystem::User, 112  
 userName  
     ControlSystem::Admin, 20  
     ControlSystem::IUser, 66  
     ControlSystem::User, 112  
  
 VecPoint  
     ControlSystem::VecPoint, 112  
 ViewModelManualSteering  
     RoboGO::ViewModels::ViewModelManualSteering, 115  
  
 WatchDigitalInput  
     ControlSystem::DLLImport, 37  
     ControlSystem::IDLL, 51  
 watchDigitalInputWrapped  
     ControlSystem::IWrapper, 72  
     ControlSystem::Wrapper, 125  
 WatchMotion  
     ControlSystem::DLLImport, 37  
     ControlSystem::IDLL, 51  
 watchMotionWrapped  
     ControlSystem::IWrapper, 72  
     ControlSystem::Wrapper, 125  
  
 write  
     ControlSystem::ConsoleUI, 23  
     ControlSystem::IUI, 65  
     ControlSystem::StringUI, 107  
 writeLine  
     ControlSystem::ConsoleUI, 23  
     ControlSystem::IUI, 65  
     ControlSystem::StringUI, 107  
  
 XYCalculate  
     RoboGO::ViewModels::XYCalculate, 126  
 XamlGeneratedNamespace, 16  
 XamlGeneratedNamespace.GeneratedInternalTypeHelper, 38  
 XamlGeneratedNamespace::GeneratedInternalTypeHelper  
     AddEventHandler, 39  
     CreateDelegate, 39  
     CreateInstance, 39  
     GetPropertyValue, 39, 40  
     SetPropertyValue, 40