

Ev3Dev

0.1.1

Generated by Doxygen 1.8.11



# Contents

<b>1</b>	<b>Hierarchical Index</b>	<b>1</b>
1.1	Class Hierarchy . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>File Index</b>	<b>7</b>
3.1	File List . . . . .	7
<b>4</b>	<b>Class Documentation</b>	<b>9</b>
4.1	ev3::Action Class Reference . . . . .	9
4.1.1	Detailed Description . . . . .	10
4.1.2	Member Enumeration Documentation . . . . .	11
4.1.2.1	ActionType . . . . .	11
4.1.3	Constructor & Destructor Documentation . . . . .	11
4.1.3.1	Action(CommandsVector commands, ActionType type) . . . . .	11
4.1.3.2	Action(CommandsVector commands) . . . . .	11
4.1.3.3	Action(ActionType type) . . . . .	11
4.1.4	Member Function Documentation . . . . .	12
4.1.4.1	getActionPrototype() . . . . .	12
4.1.4.2	getString() . . . . .	12
4.1.4.3	getType() . . . . .	12
4.1.4.4	isExecuted() . . . . .	12
4.1.4.5	isFinished() . . . . .	12
4.1.4.6	setCommands(CommandsVector commands) . . . . .	12

4.1.4.7	<a href="#">setEndCondition(EndCondition condition)</a>	13
4.1.5	<a href="#">Member Data Documentation</a>	13
4.1.5.1	<a href="#">_endCondition</a>	13
4.2	<a href="#">ev3::ActionDriveDistance Class Reference</a>	13
4.2.1	<a href="#">Detailed Description</a>	14
4.2.2	<a href="#">Constructor &amp; Destructor Documentation</a>	14
4.2.2.1	<a href="#">ActionDriveDistance(int distance)</a>	14
4.2.2.2	<a href="#">ActionDriveDistance(CommandsVector commands, int distance)</a>	14
4.2.3	<a href="#">Member Function Documentation</a>	15
4.2.3.1	<a href="#">getActionPrototype()</a>	15
4.2.3.2	<a href="#">getDistance()</a>	15
4.2.3.3	<a href="#">getString() override</a>	15
4.3	<a href="#">ev3::ActionDriveForever Class Reference</a>	15
4.3.1	<a href="#">Detailed Description</a>	16
4.3.2	<a href="#">Constructor &amp; Destructor Documentation</a>	16
4.3.2.1	<a href="#">ActionDriveForever(bool forward=true)</a>	16
4.3.2.2	<a href="#">ActionDriveForever(CommandsVector commands, bool forward=true)</a>	16
4.3.3	<a href="#">Member Function Documentation</a>	17
4.3.3.1	<a href="#">getActionPrototype()</a>	17
4.3.3.2	<a href="#">getString() override</a>	17
4.3.3.3	<a href="#">isForward()</a>	17
4.4	<a href="#">ev3::ActionRepeat Class Reference</a>	17
4.4.1	<a href="#">Detailed Description</a>	18
4.4.2	<a href="#">Constructor &amp; Destructor Documentation</a>	18
4.4.2.1	<a href="#">ActionRepeat(StoredActions actions, unsigned int n)</a>	18
4.4.3	<a href="#">Member Function Documentation</a>	18
4.4.3.1	<a href="#">getString()</a>	18
4.5	<a href="#">ev3::ActionRotate Class Reference</a>	19
4.5.1	<a href="#">Detailed Description</a>	20
4.5.2	<a href="#">Constructor &amp; Destructor Documentation</a>	20

4.5.2.1	ActionRotate(int rotation)	20
4.5.2.2	ActionRotate(CommandsVector commands, int rotation)	20
4.5.3	Member Function Documentation	20
4.5.3.1	getActionPrototype()	20
4.5.3.2	getRotation()	20
4.5.3.3	getString() override	21
4.6	ev3::ActionRotateRandDirection Class Reference	21
4.6.1	Detailed Description	21
4.6.2	Constructor & Destructor Documentation	21
4.6.2.1	ActionRotateRandDirection(int rotation)	21
4.6.2.2	ActionRotateRandDirection(CommandsVector commands, int rotation)	22
4.6.3	Member Function Documentation	22
4.6.3.1	execute() override	22
4.6.3.2	getActionPrototype()	22
4.6.3.3	getString() override	22
4.7	ev3::ActionStop Class Reference	23
4.7.1	Detailed Description	23
4.7.2	Constructor & Destructor Documentation	23
4.7.2.1	ActionStop(CommandsVector commands)	23
4.7.3	Member Function Documentation	23
4.7.3.1	getActionPrototype()	23
4.7.3.2	getString() override	24
4.8	ev3::Agent Class Reference	24
4.8.1	Detailed Description	25
4.8.2	Member Function Documentation	25
4.8.2.1	getCommId()	25
4.8.2.2	getId()	25
4.8.2.3	processMessage(Message *message, Message *retMessage)	25
4.8.2.4	setBehaviour(SharedPtrBehaviour behaviour)	26
4.8.2.5	setCommId(const unsigned int commId)	26

4.8.2.6	<a href="#">setId(const unsigned int id)</a>	26
4.8.2.7	<a href="#">setMeasurement(Measurements measurements)</a>	26
4.8.2.8	<a href="#">updateLastMessage(Message *message)</a>	26
4.8.3	<a href="#">Member Data Documentation</a>	26
4.8.3.1	<a href="#">_measurements</a>	27
4.9	<a href="#">ev3::Behaviour Class Reference</a>	27
4.9.1	<a href="#">Detailed Description</a>	28
4.9.2	<a href="#">Member Enumeration Documentation</a>	28
4.9.2.1	<a href="#">BehaviourType</a>	28
4.9.3	<a href="#">Constructor &amp; Destructor Documentation</a>	29
4.9.3.1	<a href="#">Behaviour(BehaviourType type, BehaviourStates states)</a>	29
4.9.3.2	<a href="#">Behaviour(BehaviourType type)</a>	29
4.9.4	<a href="#">Member Function Documentation</a>	29
4.9.4.1	<a href="#">getPrototype()</a>	29
4.9.4.2	<a href="#">getString()</a>	29
4.9.4.3	<a href="#">react(Event::EventType type)</a>	29
4.9.4.4	<a href="#">setMeasurements(Measurements measurements)</a>	30
4.9.4.5	<a href="#">setReactionStates(BehaviourStates reactionStates)</a>	30
4.9.4.6	<a href="#">setStates(BehaviourStates states)</a>	30
4.9.4.7	<a href="#">setStopState(BehaviourState state)</a>	30
4.10	<a href="#">ev3::BehaviourDriveOnSquare Class Reference</a>	31
4.10.1	<a href="#">Detailed Description</a>	31
4.10.2	<a href="#">Constructor &amp; Destructor Documentation</a>	31
4.10.2.1	<a href="#">BehaviourDriveOnSquare(unsigned int side, bool turningRight)</a>	31
4.10.2.2	<a href="#">BehaviourDriveOnSquare(BehaviourStates states, unsigned int side, bool turningRight)</a>	32
4.10.3	<a href="#">Member Function Documentation</a>	32
4.10.3.1	<a href="#">getPrototype()</a>	32
4.10.3.2	<a href="#">getString()</a>	32
4.11	<a href="#">ev3::BehaviourExploreRandom Class Reference</a>	33
4.11.1	<a href="#">Detailed Description</a>	33

4.11.2	Constructor & Destructor Documentation	33
4.11.2.1	BehaviourExploreRandom(BehaviourStates states)	33
4.11.3	Member Function Documentation	33
4.11.3.1	getPrototype()	33
4.11.3.2	getString()	34
4.12	ev3::BehaviourState Class Reference	34
4.12.1	Detailed Description	35
4.12.2	Constructor & Destructor Documentation	35
4.12.2.1	BehaviourState(const BehaviourState &other)=default	35
4.12.2.2	BehaviourState(SharedPtrAction action, unsigned int nextState, bool isStop← State=false)	35
4.12.2.3	BehaviourState(SharedPtrAction action, unsigned int nextState, Reactions← Transitions reactions)	35
4.12.3	Member Function Documentation	36
4.12.3.1	getAction()	36
4.12.3.2	getReaction(Event::EventType type)	36
4.12.3.3	isStopState()	36
4.12.3.4	process()	36
4.12.3.5	setNextState(const unsigned int next)	36
4.12.3.6	setReactions(ReactionsTransitions reactions)	37
4.13	ev3::CommUtils::Buffer Struct Reference	37
4.13.1	Detailed Description	37
4.14	ev3dev::button Class Reference	37
4.15	ev3::CircularBuffer< T > Class Template Reference	38
4.15.1	Detailed Description	39
4.15.2	Constructor & Destructor Documentation	39
4.15.2.1	CircularBuffer(unsigned int limit)	39
4.15.3	Member Function Documentation	39
4.15.3.1	contain(T object)	39
4.15.3.2	push(T object)	40
4.16	ev3dev::color_sensor Class Reference	40

4.17	<a href="#">ev3::ColorUtils Class Reference</a>	41
4.17.1	<a href="#">Detailed Description</a>	42
4.18	<a href="#">ev3::Command Class Reference</a>	43
4.18.1	<a href="#">Detailed Description</a>	43
4.18.2	<a href="#">Member Function Documentation</a>	43
4.18.2.1	<a href="#">getString()</a>	43
4.19	<a href="#">ev3::CommandMotor Class Reference</a>	44
4.19.1	<a href="#">Detailed Description</a>	44
4.19.2	<a href="#">Constructor &amp; Destructor Documentation</a>	44
4.19.2.1	<a href="#">CommandMotor(Motor &amp;motor)</a>	44
4.19.3	<a href="#">Member Function Documentation</a>	45
4.19.3.1	<a href="#">getMotor()</a>	45
4.20	<a href="#">ev3::CommandMotorReset Class Reference</a>	45
4.20.1	<a href="#">Detailed Description</a>	45
4.20.2	<a href="#">Constructor &amp; Destructor Documentation</a>	45
4.20.2.1	<a href="#">CommandMotorReset(Motor &amp;motor)</a>	45
4.21	<a href="#">ev3::CommandMotorRunForever Class Reference</a>	46
4.21.1	<a href="#">Detailed Description</a>	46
4.21.2	<a href="#">Constructor &amp; Destructor Documentation</a>	46
4.21.2.1	<a href="#">CommandMotorRunForever(Motor &amp;motor)</a>	46
4.22	<a href="#">ev3::CommandMotorSetSpeed Class Reference</a>	47
4.22.1	<a href="#">Detailed Description</a>	47
4.22.2	<a href="#">Constructor &amp; Destructor Documentation</a>	47
4.22.2.1	<a href="#">CommandMotorSetSpeed(Motor &amp;motor, int value)</a>	47
4.23	<a href="#">ev3::CommandMotorSetSpeedRegEnabled Class Reference</a>	48
4.23.1	<a href="#">Detailed Description</a>	49
4.23.2	<a href="#">Constructor &amp; Destructor Documentation</a>	49
4.23.2.1	<a href="#">CommandMotorSetSpeedRegEnabled(Motor &amp;motor, bool value)</a>	49
4.24	<a href="#">ev3::CommandMotorSetStopMode Class Reference</a>	49
4.24.1	<a href="#">Detailed Description</a>	50



4.24.2	Member Enumeration Documentation	50
4.24.2.1	StopMode	50
4.24.3	Constructor & Destructor Documentation	50
4.24.3.1	CommandMotorSetStopMode(Motor &motor, StopMode mode)	50
4.25	ev3::CommandMotorStop Class Reference	51
4.25.1	Detailed Description	51
4.25.2	Constructor & Destructor Documentation	51
4.25.2.1	CommandMotorStop(Motor &motor)	51
4.26	ev3::CommandSensor Class Reference	52
4.26.1	Detailed Description	52
4.26.2	Constructor & Destructor Documentation	52
4.26.2.1	CommandSensor(Sensor &sensor)	52
4.26.3	Member Function Documentation	52
4.26.3.1	getSensor()	53
4.27	ev3::Communication Class Reference	53
4.27.1	Detailed Description	54
4.27.2	Member Function Documentation	54
4.27.2.1	createThread(Queue< Message > *sendQueue, Queue< Message > *receiveQueue, bool isMaster=false)	54
4.27.2.2	run(Queue< Message > *sendQueue, Queue< Message > *receiveQueue, bool isMaster=false)	54
4.28	ev3::CommUtils Class Reference	54
4.28.1	Detailed Description	55
4.28.2	Member Function Documentation	56
4.28.2.1	getBufferFromString(const std::string message)	56
4.28.2.2	getStringFromBuffer(const Buffer buffer)	56
4.28.2.3	makeSockAddr(std::string ipAddress, int portNumber, struct sockaddr↵ in *sockaddr)	56
4.28.2.4	preparePassiveSocket(unsigned int portNumber)	56
4.28.2.5	receiveMessage(unsigned int socket, Message &message, NetworkNode &sender)	57
4.28.2.6	receiveMessageDelay(unsigned int socket, Message &message, NetworkNode &sender, unsigned int msDelay=DEFAULT_RECEIVE_DELAY)	57

4.28.2.7	<a href="#">sendMessage(unsigned int socket, unsigned int port, std::string message)</a>	57
4.28.2.8	<a href="#">sendMessage(unsigned int socket, unsigned int port, Message &amp;message, std::string &amp;proto, bool isMaster, unsigned int repeat=SENT_MESSAGE_COPIES)</a>	58
4.28.2.9	<a href="#">sendMessageTo(unsigned int socket, std::string ipAddress, unsigned int destinationPort, std::string message)</a>	58
4.28.3	<a href="#">Member Data Documentation</a>	58
4.28.3.1	<a href="#">_packetBuffer</a>	58
4.29	<a href="#">ev3dev::dc_motor Class Reference</a>	59
4.30	<a href="#">ev3dev::device Class Reference</a>	60
4.31	<a href="#">ev3::Devices Class Reference</a>	61
4.31.1	<a href="#">Detailed Description</a>	62
4.31.2	<a href="#">Constructor &amp; Destructor Documentation</a>	62
4.31.2.1	<a href="#">Devices(const Devices &amp;other)</a>	62
4.31.3	<a href="#">Member Function Documentation</a>	63
4.31.3.1	<a href="#">addListener(Sensor::SensorType type)</a>	63
4.31.3.2	<a href="#">checkDevices(RequiredDevices &amp;devices)</a>	63
4.31.3.3	<a href="#">getInstance()</a>	63
4.31.3.4	<a href="#">getMotor(ev3dev::port_type port)</a>	63
4.31.3.5	<a href="#">getSensor(ev3dev::port_type port)</a>	63
4.31.3.6	<a href="#">operator=(const Devices &amp;other)</a>	64
4.31.3.7	<a href="#">removeListener(Sensor::SensorType type)</a>	64
4.31.3.8	<a href="#">setProximitySensor(ev3dev::port_type port)</a>	64
4.31.3.9	<a href="#">setSafetyTouchSensor(ev3dev::port_type port)</a>	64
4.32	<a href="#">ev3::Event Class Reference</a>	65
4.32.1	<a href="#">Detailed Description</a>	65
4.32.2	<a href="#">Member Enumeration Documentation</a>	66
4.32.2.1	<a href="#">EventType</a>	66
4.32.3	<a href="#">Constructor &amp; Destructor Documentation</a>	66
4.32.3.1	<a href="#">Event(EventType type)</a>	66
4.32.4	<a href="#">Member Function Documentation</a>	66
4.32.4.1	<a href="#">getStringType()</a>	66

4.32.4.2	<a href="#">getType()</a>	66
4.33	<a href="#">ev3::EventAction Class Reference</a>	67
4.33.1	<a href="#">Detailed Description</a>	67
4.33.2	<a href="#">Constructor &amp; Destructor Documentation</a>	67
4.33.2.1	<a href="#">EventAction(EventType eventType, Action::ActionType actionType)</a>	67
4.33.3	<a href="#">Member Function Documentation</a>	67
4.33.3.1	<a href="#">getActionType()</a>	68
4.34	<a href="#">ev3::EventQueue Class Reference</a>	68
4.34.1	<a href="#">Detailed Description</a>	69
4.34.2	<a href="#">Constructor &amp; Destructor Documentation</a>	69
4.34.2.1	<a href="#">EventQueue(const EventQueue &amp;other)</a>	69
4.34.3	<a href="#">Member Function Documentation</a>	69
4.34.3.1	<a href="#">empty()</a>	69
4.34.3.2	<a href="#">getInstance()</a>	69
4.34.3.3	<a href="#">operator=(const EventQueue &amp;other)</a>	70
4.34.3.4	<a href="#">pop()</a>	71
4.34.3.5	<a href="#">push(SharedPtrEvent event)</a>	71
4.34.3.6	<a href="#">size()</a>	71
4.35	<a href="#">ev3::EventSensorWatch Class Reference</a>	71
4.35.1	<a href="#">Detailed Description</a>	72
4.35.2	<a href="#">Constructor &amp; Destructor Documentation</a>	72
4.35.2.1	<a href="#">EventSensorWatch(Sensor::SensorType type, SensorValue value)</a>	72
4.35.3	<a href="#">Member Function Documentation</a>	72
4.35.3.1	<a href="#">getType()</a>	72
4.35.3.2	<a href="#">getValue()</a>	73
4.36	<a href="#">ev3dev::button::file_descriptor Struct Reference</a>	73
4.37	<a href="#">ev3dev::gyro_sensor Class Reference</a>	73
4.38	<a href="#">ev3dev::i2c_sensor Class Reference</a>	74
4.39	<a href="#">ev3dev::infrared_sensor Class Reference</a>	75
4.40	<a href="#">ev3dev::large_motor Class Reference</a>	75

4.41	ev3dev::lcd Class Reference	76
4.42	ev3dev::led Class Reference	76
4.43	ev3::LedControl Class Reference	78
4.43.1	Detailed Description	79
4.43.2	Member Enumeration Documentation	79
4.43.2.1	LedColors	79
4.43.2.2	LedType	79
4.43.3	Member Function Documentation	79
4.43.3.1	flash(unsigned int leds, unsigned int msInterval, unsigned int repeat=1, unsigned int brightnessRed=MAX_BRIGHTNESS, unsigned int brightnessGreen=MAX_BRIGHTNESS)	79
4.43.3.2	flashColor(LedColors color, unsigned int msInterval, unsigned int repeat=1)	80
4.43.3.3	off(unsigned int leds=LedType::ALL)	80
4.43.3.4	on(unsigned int leds=LedType::ALL, unsigned int brightness=MAX_BRIGHTNESS)	80
4.43.3.5	onExclusive(unsigned int leds=LedType::ALL, unsigned int brightness=MAX_BRIGHTNESS)	80
4.43.3.6	setColor(LedColors color)	80
4.44	ev3dev::lego_port Class Reference	81
4.45	ev3dev::light_sensor Class Reference	81
4.46	ev3::Logger Class Reference	82
4.46.1	Detailed Description	83
4.46.2	Member Enumeration Documentation	84
4.46.2.1	LogLevel	84
4.46.2.2	LogOutput	84
4.46.3	Constructor & Destructor Documentation	84
4.46.3.1	Logger(const Logger &other)	84
4.46.4	Member Function Documentation	84
4.46.4.1	getColor(LogLevel level, LogOutput output)	84
4.46.4.2	getInstance()	85
4.46.4.3	getLabel(LogLevel level, LogOutput output)	85
4.46.4.4	log(std::string message, LogLevel level, LogOutput output=STD_OUT)	85

4.46.4.5	<code>operator=(const Logger &amp;other)</code>	85
4.46.4.6	<code>setLogLevel(LogLevel level)</code>	86
4.46.4.7	<code>setLogLevel(std::string level)</code>	86
4.46.4.8	<code>setLogOutput(LogOutput output)</code>	86
4.46.5	Member Data Documentation	86
4.46.5.1	<code>_loggerForced</code>	86
4.47	<code>ev3::Master</code> Class Reference	86
4.47.1	Detailed Description	87
4.47.2	Member Function Documentation	87
4.47.2.1	<code>createThread(Queue&lt; Message &gt; *sendQueue, Queue&lt; Message &gt; *receiveQueue)</code>	87
4.47.2.2	<code>run(Queue&lt; Message &gt; *sendQueue, Queue&lt; Message &gt; *receiveQueue)</code>	88
4.47.2.3	<code>send(Message message, bool recordMessage=true)</code>	88
4.48	<code>ev3dev::medium_motor</code> Class Reference	88
4.49	<code>ev3::Message</code> Class Reference	89
4.49.1	Detailed Description	90
4.49.2	Member Enumeration Documentation	91
4.49.2.1	<code>MessageType</code>	91
4.49.3	Constructor & Destructor Documentation	91
4.49.3.1	<code>Message(unsigned int senderId, unsigned int receiverId, unsigned int messageId, MessageType type, StringVector parameters={})</code>	91
4.49.4	Member Function Documentation	91
4.49.4.1	<code>decodeMessage(const std::string message)</code>	92
4.49.4.2	<code>empty()</code>	92
4.49.4.3	<code>encodeMessage(Message &amp;message)</code>	92
4.49.4.4	<code>getMessageId()</code>	92
4.49.4.5	<code>getParameters()</code>	92
4.49.4.6	<code>getReceiverId()</code>	93
4.49.4.7	<code>getSenderId()</code>	93
4.49.4.8	<code>getString()</code>	93
4.49.4.9	<code>getStringType()</code>	93

4.49.4.10	<a href="#">getType()</a>	93
4.49.4.11	<a href="#">setMessageId(unsigned int id)</a>	93
4.49.4.12	<a href="#">setParameters(StringVector parameters)</a>	94
4.49.4.13	<a href="#">setReceiverId(unsigned int id)</a>	94
4.49.4.14	<a href="#">setSenderId(unsigned int id)</a>	94
4.49.4.15	<a href="#">setType(MessageType type)</a>	94
4.50	<a href="#">ev3::Motor Class Reference</a>	95
4.50.1	<a href="#">Detailed Description</a>	95
4.50.2	<a href="#">Constructor &amp; Destructor Documentation</a>	95
4.50.2.1	<a href="#">Motor(ev3dev::motor motor)</a>	95
4.50.3	<a href="#">Member Function Documentation</a>	95
4.50.3.1	<a href="#">getMotor()</a>	95
4.51	<a href="#">ev3dev::motor Class Reference</a>	96
4.52	<a href="#">ev3::CommUtils::NetworkNode Struct Reference</a>	98
4.52.1	<a href="#">Detailed Description</a>	98
4.53	<a href="#">ev3dev::power_supply Class Reference</a>	98
4.54	<a href="#">ev3::Queue&lt; T &gt; Class Template Reference</a>	99
4.54.1	<a href="#">Detailed Description</a>	99
4.54.2	<a href="#">Member Function Documentation</a>	100
4.54.2.1	<a href="#">empty()</a>	100
4.54.2.2	<a href="#">pop()</a>	100
4.54.2.3	<a href="#">push(T element)</a>	100
4.55	<a href="#">ev3dev::remote_control Class Reference</a>	100
4.56	<a href="#">ev3::Robot Class Reference</a>	101
4.56.1	<a href="#">Detailed Description</a>	103
4.56.2	<a href="#">Constructor &amp; Destructor Documentation</a>	103
4.56.2.1	<a href="#">Robot(Devices::RequiredDevices devices, AvailableActions actions)</a>	103
4.56.3	<a href="#">Member Function Documentation</a>	103
4.56.3.1	<a href="#">createThread(Queue&lt; Message &gt; *sendQueue, Queue&lt; Message &gt; *receiveQueue)</a>	103
4.56.3.2	<a href="#">generateBehaviour(Behaviour::BehaviourType type, StringVector parameters)</a>	104

4.56.3.3	<a href="#">getString()</a>	104
4.56.3.4	<a href="#">run(Queue&lt; Message &gt; *sendQueue, Queue&lt; Message &gt; *receiveQueue)</a>	104
4.56.3.5	<a href="#">send(Message message)</a>	105
4.56.4	Member Data Documentation	105
4.56.4.1	<a href="#">_behaviourSet</a>	105
4.56.4.2	<a href="#">_pulsePerUnitRatio</a>	105
4.57	<a href="#">ev3::RobotModelA Class Reference</a>	105
4.57.1	Detailed Description	106
4.57.2	Member Function Documentation	106
4.57.2.1	<a href="#">generateAction(SharedPtrAction action, Action::ActionType type)</a>	106
4.57.2.2	<a href="#">getString() override</a>	106
4.58	<a href="#">ev3::RobotState Class Reference</a>	107
4.58.1	Detailed Description	108
4.58.2	Member Enumeration Documentation	108
4.58.2.1	<a href="#">States</a>	108
4.58.3	Constructor & Destructor Documentation	108
4.58.3.1	<a href="#">RobotState(ChangeMap changes, LedControl *led)</a>	108
4.58.4	Member Function Documentation	109
4.58.4.1	<a href="#">changeState(States state)</a>	109
4.58.4.2	<a href="#">getBehaviour()</a>	109
4.58.4.3	<a href="#">getPendingMessage()</a>	109
4.58.4.4	<a href="#">isPendingEnabled()</a>	109
4.58.4.5	<a href="#">process(Message msg)</a>	109
4.58.4.6	<a href="#">setBehaviour(SharedPtrBehaviour behaviour)</a>	110
4.58.4.7	<a href="#">switchState(Message::MessageType type)</a>	110
4.59	<a href="#">ev3::RobotStateActive Class Reference</a>	110
4.59.1	Detailed Description	111
4.59.2	Constructor & Destructor Documentation	111
4.59.2.1	<a href="#">RobotStateActive(LedControl *led)</a>	111
4.60	<a href="#">ev3::RobotStateIdle Class Reference</a>	111

4.60.1	Detailed Description	112
4.60.2	Constructor & Destructor Documentation	112
4.60.2.1	RobotStateIdle(LedControl *led)	112
4.61	ev3::RobotStatePanic Class Reference	112
4.61.1	Detailed Description	112
4.61.2	Constructor & Destructor Documentation	113
4.61.2.1	RobotStatePanic(LedControl *led)	113
4.62	ev3::RobotStatePaused Class Reference	113
4.62.1	Detailed Description	113
4.62.2	Constructor & Destructor Documentation	113
4.62.2.1	RobotStatePaused(LedControl *led)	113
4.63	ev3::RobotStateWorking Class Reference	114
4.63.1	Detailed Description	114
4.63.2	Constructor & Destructor Documentation	114
4.63.2.1	RobotStateWorking(LedControl *led)	114
4.64	ev3dev::sensor Class Reference	115
4.65	ev3::Sensor Class Reference	116
4.65.1	Detailed Description	117
4.65.2	Member Enumeration Documentation	117
4.65.2.1	SensorType	117
4.65.3	Constructor & Destructor Documentation	118
4.65.3.1	Sensor(ev3dev::sensor sensor, SensorType type)	118
4.65.4	Member Function Documentation	119
4.65.4.1	getDecimals()	119
4.65.4.2	getNumValues()	119
4.65.4.3	getSensor()	119
4.65.4.4	getType()	119
4.65.4.5	getValue(unsigned int n)	119
4.65.4.6	getValueF(unsigned int n)	120
4.65.4.7	prepareMessage(SensorValue value, SensorType type)	120
4.66	ev3dev::servo_motor Class Reference	120
4.67	ev3::SignalHandler Class Reference	121
4.67.1	Detailed Description	122
4.67.2	Member Function Documentation	122
4.67.2.1	HandleSignal(int signum)	122
4.68	ev3dev::sound Class Reference	122
4.69	ev3dev::sound_sensor Class Reference	123
4.70	ev3dev::touch_sensor Class Reference	123
4.71	ev3dev::ultrasonic_sensor Class Reference	124



<b>5</b>	<b>File Documentation</b>	<b>125</b>
5.1	<a href="#">/home/panda/Dokumenty/Repos/Ev3Dev/include/action/Action.h File Reference</a>	125
5.1.1	Detailed Description	126
5.1.2	<a href="#">Typedef Documentation</a>	126
5.1.2.1	StoredActions	126
5.2	<a href="#">/home/panda/Dokumenty/Repos/Ev3Dev/include/action/Behaviour.h File Reference</a>	126
5.2.1	Detailed Description	127
5.3	<a href="#">/home/panda/Dokumenty/Repos/Ev3Dev/include/action/BehaviourState.h File Reference</a>	127
5.3.1	Detailed Description	127
5.4	<a href="#">/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/Communication.h File Reference</a>	127
5.4.1	Detailed Description	128
5.5	<a href="#">/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/CommUtils.h File Reference</a>	128
5.5.1	Detailed Description	129
5.6	<a href="#">/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/Message.h File Reference</a>	129
5.6.1	Detailed Description	129
5.7	<a href="#">/home/panda/Dokumenty/Repos/Ev3Dev/include/robot/Devices.h File Reference</a>	129
5.7.1	Detailed Description	130
5.8	<a href="#">/home/panda/Dokumenty/Repos/Ev3Dev/include/utils/EventQueue.h File Reference</a>	130
5.8.1	Detailed Description	130
	<b>Index</b>	<b>131</b>



# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

ev3::Action . . . . .	9
ev3::ActionDriveDistance . . . . .	13
ev3::ActionDriveForever . . . . .	15
ev3::ActionRepeat . . . . .	17
ev3::ActionRotate . . . . .	19
ev3::ActionRotateRandDirection . . . . .	21
ev3::ActionStop . . . . .	23
ev3::Agent . . . . .	24
ev3::Behaviour . . . . .	27
ev3::BehaviourDriveOnSquare . . . . .	31
ev3::BehaviourExploreRandom . . . . .	33
ev3::BehaviourState . . . . .	34
ev3::CommUtils::Buffer . . . . .	37
ev3dev::button . . . . .	37
ev3::CircularBuffer< T > . . . . .	38
ev3::CircularBuffer< std::string > . . . . .	38
ev3::ColorUtils . . . . .	41
ev3::Command . . . . .	43
ev3::CommandMotor . . . . .	44
ev3::CommandMotorReset . . . . .	45
ev3::CommandMotorRunForever . . . . .	46
ev3::CommandMotorSetSpeed . . . . .	47
ev3::CommandMotorSetSpeedRegEnabled . . . . .	48
ev3::CommandMotorSetStopMode . . . . .	49
ev3::CommandMotorStop . . . . .	51
ev3::CommandSensor . . . . .	52
ev3::Communication . . . . .	53
ev3::CommUtils . . . . .	54
ev3dev::device . . . . .	60
ev3dev::dc_motor . . . . .	59
ev3dev::led . . . . .	76
ev3dev::lego_port . . . . .	81
ev3dev::motor . . . . .	96
ev3dev::large_motor . . . . .	75

ev3dev::medium_motor . . . . .	88
ev3dev::power_supply . . . . .	98
ev3dev::sensor . . . . .	115
ev3dev::color_sensor . . . . .	40
ev3dev::gyro_sensor . . . . .	73
ev3dev::i2c_sensor . . . . .	74
ev3dev::infrared_sensor . . . . .	75
ev3dev::light_sensor . . . . .	81
ev3dev::sound_sensor . . . . .	123
ev3dev::touch_sensor . . . . .	123
ev3dev::ultrasonic_sensor . . . . .	124
ev3dev::servo_motor . . . . .	120
ev3::Devices . . . . .	61
ev3::Event . . . . .	65
ev3::EventAction . . . . .	67
ev3::EventSensorWatch . . . . .	71
ev3::EventQueue . . . . .	68
ev3dev::button::file_descriptor . . . . .	73
ev3dev::lcd . . . . .	76
ev3::LedControl . . . . .	78
ev3::Logger . . . . .	82
ev3::Master . . . . .	86
ev3::Message . . . . .	89
ev3::Motor . . . . .	95
ev3::CommUtils::NetworkNode . . . . .	98
ev3::Queue< T > . . . . .	99
ev3::Queue< ev3::Message > . . . . .	99
ev3dev::remote_control . . . . .	100
ev3::Robot . . . . .	101
ev3::RobotModelA . . . . .	105
ev3::RobotState . . . . .	107
ev3::RobotStateActive . . . . .	110
ev3::RobotStateIdle . . . . .	111
ev3::RobotStatePanic . . . . .	112
ev3::RobotStatePaused . . . . .	113
ev3::RobotStateWorking . . . . .	114
ev3::Sensor . . . . .	116
ev3::SignalHandler . . . . .	121
ev3dev::sound . . . . .	122

## Chapter 2

# Class Index

### 2.1 Class List

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

<a href="#">ev3::Action</a>	Base class for all <a href="#">Action</a> controlling classes . . . . .	9
<a href="#">ev3::ActionDriveDistance</a>	Implements <a href="#">Robot</a> simple task to drive straight for a given distance . . . . .	13
<a href="#">ev3::ActionDriveForever</a>	Implements <a href="#">Robot</a> simple task to drive straight forever . . . . .	15
<a href="#">ev3::ActionRepeat</a>	Stores many Actions in a vector and executes them in loop . . . . .	17
<a href="#">ev3::ActionRotate</a>	Implements <a href="#">Robot</a> simple task to rotate a given angle, while not driving . . . . .	19
<a href="#">ev3::ActionRotateRandDirection</a>	Implements <a href="#">Robot</a> simple task to rotate a random angle . . . . .	21
<a href="#">ev3::ActionStop</a>	Implements <a href="#">Robot</a> simple task to stop all active motors . . . . .	23
<a href="#">ev3::Agent</a>	Master-side representative of a robot unit . . . . .	24
<a href="#">ev3::Behaviour</a>	Base class for all defined behaviours . . . . .	27
<a href="#">ev3::BehaviourDriveOnSquare</a>	Implements complex behaviour of driving on a square-shaped route . . . . .	31
<a href="#">ev3::BehaviourExploreRandom</a>	Implements complex behaviour of exploring the surrounding with random rotation . . . . .	33
<a href="#">ev3::BehaviourState</a>	Encapsulates action and other information in a form of a state . . . . .	34
<a href="#">ev3::CommUtils::Buffer</a>	Contains buffer and its size . . . . .	37
<a href="#">ev3dev::button</a>	. . . . .	37
<a href="#">ev3::CircularBuffer&lt; T &gt;</a>	Template class for storing N objects of a particular class . . . . .	38
<a href="#">ev3dev::color_sensor</a>	. . . . .	40
<a href="#">ev3::ColorUtils</a>	Stores all available color escape codes . . . . .	41
<a href="#">ev3::Command</a>	Base class for all command controlling classes . . . . .	43
<a href="#">ev3::CommandMotor</a>	Base class for all motor controlling commands . . . . .	44

<a href="#">ev3::CommandMotorReset</a>	
Calls <code>reset()</code> method of containing <a href="#">Motor</a>	45
<a href="#">ev3::CommandMotorRunForever</a>	
Calls <code>run_forever()</code> method of containing <a href="#">Motor</a>	46
<a href="#">ev3::CommandMotorSetSpeed</a>	
Call <code>set_speed_sp()</code> method of containing <a href="#">Motor</a>	47
<a href="#">ev3::CommandMotorSetSpeedRegEnabled</a>	
Calls <code>set_speed_regulation_enabled()</code> method of containing <a href="#">Motor</a>	48
<a href="#">ev3::CommandMotorSetStopMode</a>	
Calls <code>set_stop_command()</code> method of containing <a href="#">Motor</a>	49
<a href="#">ev3::CommandMotorStop</a>	
Calls <code>stop()</code> method of containing <a href="#">Motor</a>	51
<a href="#">ev3::CommandSensor</a>	
Base class for all sensor controlling commands	52
<a href="#">ev3::Communication</a>	
Encapsulates low-level communication and adds logic concerning sending and receiving <a href="#">Message</a> queueing	53
<a href="#">ev3::CommUtils</a>	
Responsible for low-level communication	54
<a href="#">ev3dev::dc_motor</a>	59
<a href="#">ev3dev::device</a>	60
<a href="#">ev3::Devices</a>	
Singleton class responsible for managing devices connected to the robot	61
<a href="#">ev3::Event</a>	
Base class for all <a href="#">Event</a> classes	65
<a href="#">ev3::EventAction</a>	
<a href="#">Event</a> class triggered when something happened with <a href="#">Action</a>	67
<a href="#">ev3::EventQueue</a>	
Singleton class responsible for managing <a href="#">Event</a> objects	68
<a href="#">ev3::EventSensorWatch</a>	
Triggered when measurement of certain <a href="#">Sensor</a> occurred	71
<a href="#">ev3dev::button::file_descriptor</a>	73
<a href="#">ev3dev::gyro_sensor</a>	73
<a href="#">ev3dev::i2c_sensor</a>	74
<a href="#">ev3dev::infrared_sensor</a>	75
<a href="#">ev3dev::large_motor</a>	75
<a href="#">ev3dev::lcd</a>	76
<a href="#">ev3dev::led</a>	76
<a href="#">ev3::LedControl</a>	
Class specifically designed to eliminate <code>ev3dev</code> library limitations of controlling LED panel	78
<a href="#">ev3dev::lego_port</a>	81
<a href="#">ev3dev::light_sensor</a>	81
<a href="#">ev3::Logger</a>	
Singleton class responsible for displaying information about events, messages, exceptions and executed methods	82
<a href="#">ev3::Master</a>	
Controls the whole system and knows about every <a href="#">Agent</a>	86
<a href="#">ev3dev::medium_motor</a>	88
<a href="#">ev3::Message</a>	
Stores information passed between physical system units (another robots or master)	89
<a href="#">ev3::Motor</a>	
Encapsulates <a href="#">ev3dev::motor</a>	95
<a href="#">ev3dev::motor</a>	96
<a href="#">ev3::CommUtils::NetworkNode</a>	
Stores information about a particular node in the network	98
<a href="#">ev3dev::power_supply</a>	98
<a href="#">ev3::Queue&lt; T &gt;</a>	
Template class implementing synchronized queue	99

<a href="#">ev3dev::remote_control</a> . . . . .	100
<a href="#">ev3::Robot</a>	
Main class representing actual robot . . . . .	101
<a href="#">ev3::RobotModelA</a>	
Describes particular <a href="#">Robot</a> construction and its way of implementing actions and running behaviours . . . . .	105
<a href="#">ev3::RobotState</a>	
Base class for all <a href="#">Robot</a> states . . . . .	107
<a href="#">ev3::RobotStateActive</a>	
State in which <a href="#">Robot</a> is connected but has no assigned <a href="#">Behaviour</a> . . . . .	110
<a href="#">ev3::RobotStateIdle</a>	
State in which <a href="#">Robot</a> is powered but not connected to <a href="#">Master</a> . . . . .	111
<a href="#">ev3::RobotStatePanic</a>	
State in which <a href="#">Robot</a> lost connection with <a href="#">Master</a> or had no connection at all . . . . .	112
<a href="#">ev3::RobotStatePaused</a>	
State in which <a href="#">Robot</a> 's <a href="#">Behaviour</a> processing is paused . . . . .	113
<a href="#">ev3::RobotStateWorking</a>	
State in which <a href="#">Robot</a> is processing assigned <a href="#">Behaviour</a> . . . . .	114
<a href="#">ev3dev::sensor</a> . . . . .	115
<a href="#">ev3::Sensor</a>	
Encapsulates <a href="#">ev3dev::sensor</a> . . . . .	116
<a href="#">ev3dev::servo_motor</a> . . . . .	120
<a href="#">ev3::SignalHandler</a>	
Simple class catching system signals . . . . .	121
<a href="#">ev3dev::sound</a> . . . . .	122
<a href="#">ev3dev::sound_sensor</a> . . . . .	123
<a href="#">ev3dev::touch_sensor</a> . . . . .	123
<a href="#">ev3dev::ultrasonic_sensor</a> . . . . .	124





## File Index

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

/home/panda/Dokumenty/Repos/Ev3Dev/include/action/ <a href="#">Action.h</a>	125
Contains all Action classes	
/home/panda/Dokumenty/Repos/Ev3Dev/include/action/ <a href="#">Behaviour.h</a>	126
Contains all Behaviour classes	
/home/panda/Dokumenty/Repos/Ev3Dev/include/action/ <a href="#">BehaviourState.h</a>	127
Contains BehaviourState class	
/home/panda/Dokumenty/Repos/Ev3Dev/include/action/ <b>Command.h</b>	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/action/ <b>CommandMotor.h</b>	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/action/ <b>CommandSensor.h</b>	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/ <a href="#">Communication.h</a>	127
Contains Communication class	
/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/ <a href="#">CommUtils.h</a>	128
Contains CommUtils class	
/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/ <b>Event.h</b>	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/ <a href="#">Message.h</a>	129
Contains Message class	
/home/panda/Dokumenty/Repos/Ev3Dev/include/control/ <b>LedControl.h</b>	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ <b>ev3dev.h</b>	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/master/ <b>Agent.h</b>	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/master/ <b>Master.h</b>	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/robot/ <a href="#">Devices.h</a>	129
Contains Devices classes	
/home/panda/Dokumenty/Repos/Ev3Dev/include/robot/ <b>Motor.h</b>	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/robot/ <b>Robot.h</b>	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/robot/ <b>RobotModelA.h</b>	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/robot/ <b>RobotState.h</b>	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/robot/ <b>Sensor.h</b>	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/utils/ <b>CircularBuffer.h</b>	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/utils/ <b>ColorUtils.h</b>	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/utils/ <a href="#">EventQueue.h</a>	130
Contains EventQueue class	
/home/panda/Dokumenty/Repos/Ev3Dev/include/utils/ <b>Logger.h</b>	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/utils/ <b>Queue.h</b>	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/utils/ <b>SignalHandler.h</b>	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/utils/ <b>Utils.h</b>	??



## Chapter 4

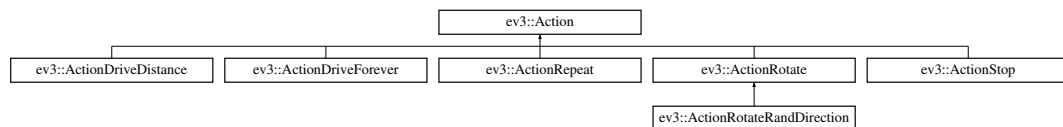
# Class Documentation

### 4.1 ev3::Action Class Reference

Base class for all [Action](#) controlling classes.

```
#include <Action.h>
```

Inheritance diagram for ev3::Action:



#### Public Types

- enum [ActionType](#) {  
  [NOP](#), [REPEAT](#), [DRIVE\\_DISTANCE](#), [ROTATE](#),  
  [ROTATE\\_RANDOM\\_DIR](#), [STOP](#), [DRIVE\\_FOREVER](#) }  
  Type of [Action](#).
- typedef std::function< bool(void) > [EndCondition](#)  
  Type for lambda functions to store end of [Action](#) condition.

#### Public Member Functions

- [Action](#) ([CommandsVector](#) commands, [ActionType](#) type)  
  Constructor with [CommandsVector](#) and [ActionType](#) parameters.
- [Action](#) ([CommandsVector](#) commands)  
  Constructor with [CommandsVector](#) parameter.
- [Action](#) ([ActionType](#) type)  
  Constructor with [ActionType](#) parameter.
- virtual [~Action](#) ()  
  Default destructor.
- virtual void [execute](#) ()  
  Executes stored [Commands](#) in a sequence.

- virtual bool `isFinished ()`  
*Check if [Action](#) condition is fulfilled.*
- virtual bool `isExecuted ()`  
*Check if action was executed.*
- virtual std::string `getActionPrototype ()`  
*Generate std::string prototype for [Action](#).*
- virtual std::string `getString ()`  
*Get human-readable [Action](#) name.*
- void `setCommands (CommandsVector commands)`  
*Set [Commands](#) to be executed.*
- void `setEndCondition (EndCondition condition)`  
*Set end condition for [Action](#).*
- [ActionType](#) `getType ()`  
*Get current [Action](#) type.*

### Static Public Attributes

- static const std::string `EMPTY_PROTO`  
*String for empty [Action](#) prototype.*

### Protected Attributes

- [ActionType](#) `_type`  
*[Action](#) type.*
- [CommandsVector](#) `_commands`  
*Vector of [Commands](#).*
- [EndCondition](#) `_endCondition`  
*Lambda function defining [Action](#) end condition.*
- bool `_isExecuted` = false  
*True if action is already executed, false otherwise.*

#### 4.1.1 Detailed Description

Base class for all [Action](#) controlling classes.

Each [Action](#) contains of a sequence of many [Commands](#) and all of them are executed immediately, one after another. [Action](#) is valid, until specific [Event](#) occurs or its `endCondition` function returns true.

[Action](#) objects are instantiated accordingly to [Robot](#) model that uses them. Actions are predefined and cannot be dynamically created.

## 4.1.2 Member Enumeration Documentation

### 4.1.2.1 enum ev3::Action::ActionType

Type of [Action](#).

It directly points to derived class being used.

See also

[Robot::AvailableActions](#)

Enumerator

**NOP** No operation.

**REPEAT** Repeats execution of other [Actions](#).

**DRIVE\_DISTANCE** Power [Motor](#) to reach certain distance.

**ROTATE** Rotate [Robot](#) for given angle.

**ROTATE\_RANDOM\_DIR** Rotate for given angle, clockwise or counterclockwise at random.

**STOP** Stop all active motors.

**DRIVE\_FOREVER** Drive forward or backward infinitely.

## 4.1.3 Constructor & Destructor Documentation

### 4.1.3.1 Action::Action ( CommandsVector *commands*, ActionType *type* )

Constructor with CommandsVector and ActionType parameters.

Parameters

<i>commands</i>	<a href="#">Commands</a> stored within this <a href="#">Action</a> .
<i>type</i>	Type of <a href="#">Action</a> used.

### 4.1.3.2 Action::Action ( CommandsVector *commands* )

Constructor with CommandsVector parameter.

[Action type](#) is set to [Action::NOP](#) .

Parameters

<i>commands</i>	<a href="#">Commands</a> stored within this <a href="#">Action</a> .
-----------------	--

### 4.1.3.3 Action::Action ( ActionType *type* )

Constructor with ActionType parameter.

## Parameters

<i>type</i>	Type of <a href="#">Action</a> used.
-------------	--------------------------------------

#### 4.1.4 Member Function Documentation

##### 4.1.4.1 `std::string Action::getActionPrototype ( )` [virtual]

Generate `std::string` prototype for [Action](#).

## Returns

Encoded [Action](#) data into `std::string`.

Reimplemented in [ev3::ActionDriveForever](#), [ev3::ActionStop](#), [ev3::ActionRotateRandDirection](#), [ev3::ActionRotate](#), and [ev3::ActionDriveDistance](#).

##### 4.1.4.2 `std::string Action::getString ( )` [virtual]

Get human-readable [Action](#) name.

## Returns

String containing [Action](#) name.

Reimplemented in [ev3::ActionDriveForever](#), [ev3::ActionStop](#), [ev3::ActionRotateRandDirection](#), [ev3::ActionRotate](#), [ev3::ActionDriveDistance](#), and [ev3::ActionRepeat](#).

##### 4.1.4.3 `Action::ActionType Action::getType ( )`

Get current [Action](#) type.

## Returns

ActionType value.

##### 4.1.4.4 `bool Action::isExecuted ( )` [virtual]

Check if action was executed.

## Returns

True if action was already executed, false otherwise.

##### 4.1.4.5 `bool Action::isFinished ( )` [virtual]

Check if [Action](#) condition is fulfilled.

## Returns

Value returned from [Action::\\_endCondition](#).

##### 4.1.4.6 `void Action::setCommands ( CommandsVector commands )`

Set [Commands](#) to be executed.

## Parameters

<i>commands</i>	CommandsVector with pointers to commands.
-----------------	---

4.1.4.7 void Action::setEndCondition ( EndCondition *condition* )

Set end condition for [Action](#).

## Parameters

<i>condition</i>	Lambda function returning bool value.
------------------	---------------------------------------

## 4.1.5 Member Data Documentation

## 4.1.5.1 EndCondition ev3::Action::\_endCondition [protected]

## Initial value:

```
= [] ()
{
    return true;
}
```

Lambda function defining [Action](#) end condition.

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

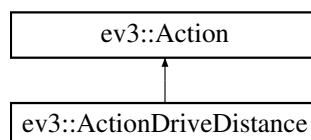
- /home/panda/Dokumenty/Repos/Ev3Dev/include/action/[Action.h](#)
- /home/panda/Dokumenty/Repos/Ev3Dev/src/action/Action.cpp

## 4.2 ev3::ActionDriveDistance Class Reference

Implements [Robot](#) simple task to drive straight for a given distance.

```
#include <Action.h>
```

Inheritance diagram for ev3::ActionDriveDistance:



## Public Member Functions

- [ActionDriveDistance](#) (int distance)  
*Constructor with distance parameter.*
- [ActionDriveDistance](#) ([CommandsVector](#) commands, int distance)  
*Constructor with CommandsVector and distance parameters.*
- int [getDistance](#) ()  
*Get distance the [Robot](#) has to drive.*
- virtual std::string [getActionPrototype](#) ()  
*Get [ActionDriveDistance](#) encoded name and its parameters.*
- virtual std::string [getString](#) () override  
*Get [ActionDriveDistance](#) human-readable name.*

## Private Attributes

- int [\\_distance](#)  
*Distance for the robot to drive in units.*

## Additional Inherited Members

### 4.2.1 Detailed Description

Implements [Robot](#) simple task to drive straight for a given distance.

### 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 [ActionDriveDistance::ActionDriveDistance](#) ( int *distance* )

Constructor with distance parameter.

##### Parameters

<i>distance</i>	Integer value in <a href="#">Robot</a> units to be driven.
-----------------	--

#### 4.2.2.2 [ActionDriveDistance::ActionDriveDistance](#) ( [CommandsVector](#) *commands*, int *distance* )

Constructor with CommandsVector and distance parameters.

##### Parameters

<i>commands</i>	Sequence of commands to be executed.
<i>distance</i>	Integer value in <a href="#">Robot</a> units to be driven.



### 4.2.3 Member Function Documentation

#### 4.2.3.1 `std::string ActionDriveDistance::getActionPrototype ( )` `[virtual]`

Get [ActionDriveDistance](#) encoded name and its parameters.

##### Returns

String with encoded name and parameters.

Reimplemented from [ev3::Action](#).

#### 4.2.3.2 `int ActionDriveDistance::getDistance ( )`

Get distance the [Robot](#) has to drive.

##### Returns

Integer value in [Robot](#) units.

#### 4.2.3.3 `std::string ActionDriveDistance::getString ( )` `[override],[virtual]`

Get [ActionDriveDistance](#) human-readable name.

##### Returns

String with name and parameters

Reimplemented from [ev3::Action](#).

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

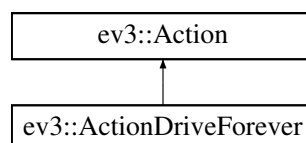
- `/home/panda/Dokumenty/Repos/Ev3Dev/include/action/Action.h`
- `/home/panda/Dokumenty/Repos/Ev3Dev/src/action/Action.cpp`

## 4.3 ev3::ActionDriveForever Class Reference

Implements [Robot](#) simple task to drive straight forever.

```
#include <Action.h>
```

Inheritance diagram for `ev3::ActionDriveForever`:



## Public Member Functions

- [ActionDriveForever](#) (bool forward=true)  
*Constructor with direction parameter.*
- [ActionDriveForever](#) ([CommandsVector](#) commands, bool forward=true)  
*Constructor with CommandsVector and direction parameter.*
- virtual std::string [getActionPrototype](#) ()  
*Get [ActionDriveForever](#) encoded name and its parameters.*
- virtual std::string [getString](#) () override  
*Get [ActionDriveForever](#) human-readable name.*
- bool [isForward](#) ()  
*Return specified direction.*

## Private Attributes

- bool [\\_isForward](#)  
*Direction of driving. Either forward (true) or backward (false).*

## Additional Inherited Members

### 4.3.1 Detailed Description

Implements [Robot](#) simple task to drive straight forever.

### 4.3.2 Constructor & Destructor Documentation

#### 4.3.2.1 [ActionDriveForever::ActionDriveForever](#) ( bool *forward* = true )

Constructor with direction parameter.

##### Parameters

<i>forward</i>	True to drive forward, false otherwise.
----------------	---

#### 4.3.2.2 [ActionDriveForever::ActionDriveForever](#) ( [CommandsVector](#) *commands*, bool *forward* = true )

Constructor with CommandsVector and direction parameter.

##### Parameters

<i>commands</i>	Sequence of commands to be executed.
<i>forward</i>	True to drive forward, false otherwise.

### 4.3.3 Member Function Documentation

#### 4.3.3.1 `std::string ActionDriveForever::getActionPrototype ( )` `[virtual]`

Get [ActionDriveForever](#) encoded name and its parameters.

##### Returns

String with encoded name and parameters.

Reimplemented from [ev3::Action](#).

#### 4.3.3.2 `std::string ActionDriveForever::getString ( )` `[override], [virtual]`

Get [ActionDriveForever](#) human-readable name.

##### Returns

String with name and parameters

Reimplemented from [ev3::Action](#).

#### 4.3.3.3 `bool ActionDriveForever::isForward ( )`

Return specified direction.

##### Returns

True for forward, false for backward.

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

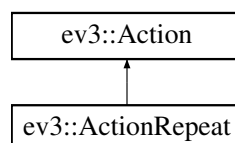
- `/home/panda/Dokumenty/Repos/Ev3Dev/include/action/Action.h`
- `/home/panda/Dokumenty/Repos/Ev3Dev/src/action/Action.cpp`

## 4.4 ev3::ActionRepeat Class Reference

Stores many Actions in a vector and executes them in loop.

```
#include <Action.h>
```

Inheritance diagram for `ev3::ActionRepeat`:



## Public Member Functions

- [ActionRepeat](#) ([StoredActions](#) actions, unsigned int n)  
*Constructor with StoredActions and iterations parameters.*
- virtual void [execute](#) ()  
*Continue with executing stored Actions.*
- virtual std::string [getString](#) ()  
*Return human-readable [ActionRepeat](#) name.*

## Private Attributes

- [StoredActions](#) [\\_actions](#)  
*Vector of stored Actions to be executed.*
- unsigned int [\\_n](#)  
*Number of iterations.*
- unsigned int [\\_currentIteration](#) = 0  
*Keeps track of iterations already passed.*
- unsigned int [\\_currentAction](#) = 0  
*Keeps track of which [Action](#) is currently in progress.*

## Additional Inherited Members

### 4.4.1 Detailed Description

Stores many Actions in a vector and executes them in loop.

Number of iterations is given and may be infinite.

### 4.4.2 Constructor & Destructor Documentation

#### 4.4.2.1 [ActionRepeat::ActionRepeat](#) ( [StoredActions](#) actions, unsigned int *n* )

Constructor with StoredActions and iterations parameters.

##### Parameters

<i>actions</i>	Vector of Actions to be executed in a loop.
<i>n</i>	Number of iterations. If 0 is given, loop will be infinite.

### 4.4.3 Member Function Documentation

#### 4.4.3.1 std::string [ActionRepeat::getString](#) ( ) [virtual]

Return human-readable [ActionRepeat](#) name.

**Returns**

String containing [Action](#) name.

Reimplemented from [ev3::Action](#).

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

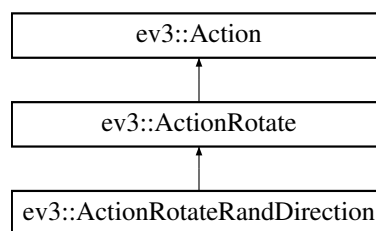
- [/home/panda/Dokumenty/Repos/Ev3Dev/include/action/Action.h](#)
- [/home/panda/Dokumenty/Repos/Ev3Dev/src/action/Action.cpp](#)

## 4.5 ev3::ActionRotate Class Reference

Implements [Robot](#) simple task to rotate a given angle, while not driving.

```
#include <Action.h>
```

Inheritance diagram for ev3::ActionRotate:

**Public Member Functions**

- [ActionRotate](#) (int rotation)  
*Constructor with rotation parameter in degrees.*
- [ActionRotate](#) ([CommandsVector](#) commands, int rotation)  
*Constructor with CommandsVector and rotation parameters.*
- int [getRotation](#) ()  
*Get [Robot](#) rotation.*
- virtual std::string [getActionPrototype](#) ()  
*Get [ActionRotate](#) encoded name and its parameters.*
- virtual std::string [getString](#) () override  
*Get [ActionRotate](#) human-readable name.*

**Protected Attributes**

- int [\\_rotation](#)  
*Angle of rotation in degrees for the [Robot](#).*

## Additional Inherited Members

### 4.5.1 Detailed Description

Implements [Robot](#) simple task to rotate a given angle, while not driving.

Rotation is made in place.

### 4.5.2 Constructor & Destructor Documentation

#### 4.5.2.1 `ActionRotate::ActionRotate ( int rotation )`

Constructor with rotation parameter in degrees.

##### Parameters

<i>rotation</i>	Number of degrees to rotate. Positive value rotates clockwise.
-----------------	--

#### 4.5.2.2 `ActionRotate::ActionRotate ( CommandsVector commands, int rotation )`

Constructor with CommandsVector and rotation parameters.

##### Parameters

<i>commands</i>	Sequence of commands to be executed.
<i>rotation</i>	Integer value of <a href="#">Robot</a> rotation in degrees.

### 4.5.3 Member Function Documentation

#### 4.5.3.1 `std::string ActionRotate::getActionPrototype ( ) [virtual]`

Get [ActionRotate](#) encoded name and its parameters.

##### Returns

String with encoded name and parameters.

Reimplemented from [ev3::Action](#).

Reimplemented in [ev3::ActionRotateRandDirection](#).

#### 4.5.3.2 `int ActionRotate::getRotation ( )`

Get [Robot](#) rotation.

##### Returns

Integer value of rotation in degrees.

4.5.3.3 `std::string ActionRotate::getString ( ) [override],[virtual]`

Get [ActionRotate](#) human-readable name.

Returns

String with name and parameters

Reimplemented from [ev3::Action](#).

Reimplemented in [ev3::ActionRotateRandDirection](#).

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

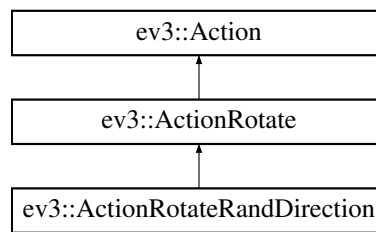
- `/home/panda/Dokumenty/Repos/Ev3Dev/include/action/Action.h`
- `/home/panda/Dokumenty/Repos/Ev3Dev/src/action/Action.cpp`

## 4.6 ev3::ActionRotateRandDirection Class Reference

Implements [Robot](#) simple task to rotate a random angle.

```
#include <Action.h>
```

Inheritance diagram for `ev3::ActionRotateRandDirection`:



### Public Member Functions

- [ActionRotateRandDirection](#) (int rotation)  
*Constructor with rotation parameter in degrees.*
- [ActionRotateRandDirection](#) ([CommandsVector](#) commands, int rotation)  
*Constructor with CommandsVector and rotation parameters.*
- virtual `std::string getActionPrototype ( )`  
*Get [ActionRotateRandDirection](#) encoded name and its parameters.*
- virtual `std::string getString ( )` override  
*Get [ActionRotateRandDirection](#) human-readable name.*
- virtual `void execute ( )` override

### Additional Inherited Members

#### 4.6.1 Detailed Description

Implements [Robot](#) simple task to rotate a random angle.

Rotation is performed in place. Maximum angle in degrees is passed via constructor argument.

#### 4.6.2 Constructor & Destructor Documentation

##### 4.6.2.1 `ActionRotateRandDirection::ActionRotateRandDirection ( int rotation )`

Constructor with rotation parameter in degrees.

## Parameters

<i>rotation</i>	Upper limit of degrees to rotate randomly. Positive value rotates clockwise.
-----------------	--

4.6.2.2 ActionRotateRandDirection::ActionRotateRandDirection ( **CommandsVector** *commands*, int *rotation* )

Constructor with CommandsVector and rotation parameters.

## Parameters

<i>commands</i>	Sequence of commands to be executed.
<i>rotation</i>	Upper limit of degrees to rotate randomly. Positive value rotates clockwise.

## 4.6.3 Member Function Documentation

## 4.6.3.1 void ActionRotateRandDirection::execute ( ) [override],[virtual]

See also

[Action::execute](#)

Reimplemented from [ev3::Action](#).

## 4.6.3.2 std::string ActionRotateRandDirection::getActionPrototype ( ) [virtual]

Get [ActionRotateRandDirection](#) encoded name and its parameters.

## Returns

String with encoded name and parameters.

Reimplemented from [ev3::ActionRotate](#).

## 4.6.3.3 std::string ActionRotateRandDirection::getString ( ) [override],[virtual]

Get [ActionRotateRandDirection](#) human-readable name.

## Returns

String with name and parameters

Reimplemented from [ev3::ActionRotate](#).

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/action/[Action.h](#)
- /home/panda/Dokumenty/Repos/Ev3Dev/src/action/[Action.cpp](#)

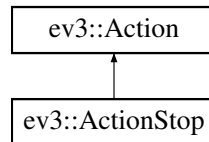


## 4.7 ev3::ActionStop Class Reference

Implements [Robot](#) simple task to stop all active motors.

```
#include <Action.h>
```

Inheritance diagram for ev3::ActionStop:



### Public Member Functions

- [ActionStop](#) ()  
*Default constructor.*
- [ActionStop](#) ([CommandsVector](#) commands)  
*Constructor with CommandsVector parameter.*
- virtual std::string [getActionPrototype](#) ()  
*Get [ActionStop](#) encoded name and its parameters.*
- virtual std::string [getString](#) () override  
*Get [ActionStop](#) human-readable name.*

### Additional Inherited Members

#### 4.7.1 Detailed Description

Implements [Robot](#) simple task to stop all active motors.

#### 4.7.2 Constructor & Destructor Documentation

##### 4.7.2.1 ActionStop::ActionStop ( [CommandsVector](#) commands )

Constructor with [CommandsVector](#) parameter.

Parameters

<i>commands</i>	Sequence of commands to be executed.
-----------------	--------------------------------------

#### 4.7.3 Member Function Documentation

##### 4.7.3.1 std::string ActionStop::getActionPrototype ( ) [virtual]

Get [ActionStop](#) encoded name and its parameters.

**Returns**

String with encoded name and parameters.

Reimplemented from [ev3::Action](#).

#### 4.7.3.2 `std::string ActionStop::getString ( )` `[override],[virtual]`

Get [ActionStop](#) human-readable name.

**Returns**

String with name and parameters

Reimplemented from [ev3::Action](#).

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

- `/home/panda/Dokumenty/Repos/Ev3Dev/include/action/Action.h`
- `/home/panda/Dokumenty/Repos/Ev3Dev/src/action/Action.cpp`

## 4.8 [ev3::Agent](#) Class Reference

Master-side representative of a robot unit.

```
#include <Agent.h>
```

**Public Member Functions**

- unsigned int [getId](#) ()  
*Agent id getter.*
- void [setId](#) (const unsigned int id)  
*Agent id setter.*
- unsigned int [getCommId](#) ()  
*Current communication id getter.*
- void [setCommId](#) (const unsigned int commId)  
*Communication id setter.*
- void [processMessage](#) ([Message](#) \*message, [Message](#) \*retMessage)  
*Process received Message to produce response.*
- void [updateLastMessage](#) ([Message](#) \*message)  
*Update data concerning last message sent.*
- void [setBehaviour](#) ([SharedPtrBehaviour](#) behaviour)  
*Set currently executing Behaviour.*
- void [setMeasurement](#) ([Measurements](#) measurements)  
*Set measurements that must be done on corresponding Robot.*

## Private Attributes

- [SharedPtrBehaviour \\_currentBehaviour](#)  
*Currently active [Behaviour](#).*
- [Measurements \\_measurements](#)  
*Vector with [Sensor](#) types.*
- [RobotState::States \\_state](#) = [RobotState::IDLE](#)  
*Current state of the corresponding [Robot](#).*
- unsigned int [\\_id](#)  
*Assigned [Agent](#) id.*
- unsigned int [\\_commId](#) = 0  
*[Message](#) id.*
- [Message::MessageType \\_lastMessageType](#)  
*Type of the last [Message](#) sent.*

### 4.8.1 Detailed Description

Master-side representative of a robot unit.

Lacks all device references and action execution.

### 4.8.2 Member Function Documentation

#### 4.8.2.1 unsigned int Agent::getCommId ( )

Current communication id getter.

##### Returns

Id of [Message](#) id synchronised between [Agent](#) and [Robot](#).

#### 4.8.2.2 unsigned int Agent::getId ( )

[Agent](#) id getter.

##### Returns

Id given by [Master](#).

#### 4.8.2.3 void Agent::processMessage ( [Message](#) \* *message*, [Message](#) \* *retMessage* )

Process received [Message](#) to produce response.

##### Parameters

<i>message</i>	<a href="#">Message</a> to be analyzed.
<i>retMessage</i>	Modified <a href="#">Message</a> to be sent to <a href="#">Robot</a> .

#### 4.8.2.4 void Agent::setBehaviour ( SharedPtrBehaviour *behaviour* )

Set currently executing [Behaviour](#).

##### Parameters

<i>behaviour</i>	<a href="#">Behaviour</a> shared_ptr object.
------------------	--

#### 4.8.2.5 void Agent::setCommId ( const unsigned int *commId* )

[Communication](#) id setter.

##### Parameters

<i>commId</i>	New communication id.
---------------	-----------------------

#### 4.8.2.6 void Agent::setId ( const unsigned int *id* )

[Agent](#) id setter.

##### Parameters

<i>id</i>	New id for this <a href="#">Agent</a> .
-----------	---

#### 4.8.2.7 void Agent::setMeasurement ( Measurements *measurements* )

Set measurements that must be done on corresponding [Robot](#).

##### Parameters

<i>measurements</i>	Vector of <a href="#">Sensor</a> types.
---------------------	---

#### 4.8.2.8 void Agent::updateLastMessage ( Message \* *message* )

Update data concerning last message sent.

##### Parameters

<i>message</i>	Last <a href="#">Message</a> sent to corresponding <a href="#">Robot</a> .
----------------	--

### 4.8.3 Member Data Documentation

#### 4.8.3.1 Measurements ev3::Agent::\_measurements [private]

Vector with [Sensor](#) types.

These Sensors measure values that are sent to the master.

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

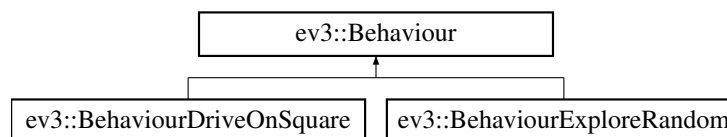
- /home/panda/Dokumenty/Repos/Ev3Dev/include/master/Agent.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/master/Agent.cpp

## 4.9 ev3::Behaviour Class Reference

Base class for all defined behaviours.

```
#include <Behaviour.h>
```

Inheritance diagram for ev3::Behaviour:



### Public Types

- enum [BehaviourType](#) { [CUSTOM](#), [DRIVE\\_ON\\_SQUARE](#), [EXPLORE\\_RANDOM](#) }  
Type of [Behaviour](#).

### Public Member Functions

- [Behaviour](#) ()=default  
Default constructor.
- [Behaviour](#) ([BehaviourType](#) type, [BehaviourStates](#) states)  
Constructor with type and states vector parameters.
- [Behaviour](#) ([BehaviourType](#) type)  
Constructor with behaviour type.
- void [setStates](#) ([BehaviourStates](#) states)  
Available states setter.
- void [setReactionStates](#) ([BehaviourStates](#) reactionStates)  
Special reaction states which occur when event is fired.
- void [setStopState](#) ([BehaviourState](#) state)  
Special stop state, used mainly to get precise sensor measurements.
- void [setMeasurements](#) ([Measurements](#) measurements)  
[Sensor](#) which measurements will be required.
- virtual StringVector [getPrototype](#) ()  
Get [Behaviour](#) encoded name and its parameters.

- virtual std::string `getString` ()  
*Get [Behaviour](#) human-readable name.*
- virtual void `process` ()  
*Updates behaviour in every iteration.*
- void `stop` ()  
*Stops [Behaviour](#) execution definitely.*
- void `pause` ()  
*Pauses [Behaviour](#) execution until it's resumed.*
- void `resume` ()  
*Resumes paused [Behaviour](#).*
- void `start` ()  
*Starts [Behaviour](#) execution.*
- void `react` ([Event::EventType](#) type)  
*Performs special actions based on [Event](#) passed.*

## Protected Attributes

- [BehaviourType](#) `_type`  
*Type of [Behaviour](#).*
- [BehaviourState](#) `_currentState`  
*Currently processed [Behaviour](#).*
- [BehaviourState](#) `_stopState`  
*Special stop state for measurements and accurate action execution.*
- [BehaviourStates](#) `_states`  
*Vector with all [Behaviour](#) available states.*
- [BehaviourStates](#) `_reactionStates`  
*Vector with all reaction states, occuring after specific events.*
- [Measurements](#) `_measurements`  
*Vector of all [Sensor](#) ids that will be measured.*
- bool `_active` = false  
*Specified whether [Behaviour](#) is currently active or not.*

### 4.9.1 Detailed Description

Base class for all defined behaviours.

It's responsible for maintaining active actions in a form of a state machine as well as keep track of sensors' measurements.

### 4.9.2 Member Enumeration Documentation

#### 4.9.2.1 enum `ev3::Behaviour::BehaviourType`

Type of [Behaviour](#).

Enumerator

- CUSTOM*** Custom, user-defined behaviour.
- DRIVE\_ON\_SQUARE*** Follow square-shaped route.
- EXPLORE\_RANDOM*** Drive in a direction and rotate randomly.

### 4.9.3 Constructor & Destructor Documentation

#### 4.9.3.1 Behaviour::Behaviour ( BehaviourType *type*, BehaviourStates *states* )

Constructor with type and states vector parameters.

Parameters

<i>type</i>	<a href="#">Behaviour</a> type.
<i>states</i>	Vector of available <a href="#">Behaviour</a> states.

#### 4.9.3.2 Behaviour::Behaviour ( BehaviourType *type* )

Constructor with behaviour type.

Parameters

<i>type</i>	<a href="#">Behaviour</a> type.
-------------	---------------------------------

### 4.9.4 Member Function Documentation

#### 4.9.4.1 StringVector Behaviour::getPrototype ( ) [virtual]

Get [Behaviour](#) encoded name and its parameters.

Returns

StringVector with encoded name and parameters as its members.

Reimplemented in [ev3::BehaviourExploreRandom](#), and [ev3::BehaviourDriveOnSquare](#).

#### 4.9.4.2 std::string Behaviour::getString ( ) [virtual]

Get [Behaviour](#) human-readable name.

Returns

String with name and parameters

Reimplemented in [ev3::BehaviourExploreRandom](#), and [ev3::BehaviourDriveOnSquare](#).

#### 4.9.4.3 void Behaviour::react ( Event::EventType *type* )

Performs special actions based on [Event](#) passed.

## Parameters

<i>type</i>	<a href="#">Event</a> type that will be processed.
-------------	--

4.9.4.4 void Behaviour::setMeasurements ( **Measurements** *measurements* )

[Sensor](#) which measurements will be required.

## Parameters

<i>measurements</i>	Vector of sensor types.
---------------------	-------------------------

4.9.4.5 void Behaviour::setReactionStates ( **BehaviourStates** *reactionStates* )

Special reaction states which occur when event is fired.

## Parameters

<i>reactionStates</i>	Vector of reaction states for this <a href="#">Behaviour</a> .
-----------------------	--

4.9.4.6 void Behaviour::setStates ( **BehaviourStates** *states* )

Available states setter.

## Parameters

<i>states</i>	Vector of states for this <a href="#">Behaviour</a> .
---------------	---

4.9.4.7 void Behaviour::setStopState ( **BehaviourState** *state* )

Special stop state, used mainly to get precise sensor measurements.

## Parameters

<i>state</i>	<a href="#">BehaviourState</a> object for stop state.
--------------	---

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/action/[Behaviour.h](#)
- /home/panda/Dokumenty/Repos/Ev3Dev/src/action/Behaviour.cpp

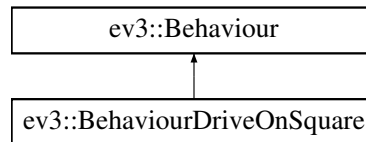


## 4.10 ev3::BehaviourDriveOnSquare Class Reference

Implements complex behaviour of driving on a square-shaped route.

```
#include <Behaviour.h>
```

Inheritance diagram for ev3::BehaviourDriveOnSquare:



### Public Member Functions

- [BehaviourDriveOnSquare](#) (unsigned int side, bool turningRight)  
*Constructor with square side and direction (either left or right).*
- [BehaviourDriveOnSquare](#) ([BehaviourStates](#) states, unsigned int side, bool turningRight)  
*Constructor with [Behaviour](#) states and driving parameters.*
- virtual StringVector [getPrototype](#) ()  
*Get [BehaviourDriveOnSquare](#) encoded name and its parameters.*
- virtual std::string [getString](#) ()  
*Get [BehaviourDriveOnSquare](#) human-readable name.*

### Private Attributes

- unsigned int [\\_squareSide](#)  
*Length of square side in units.*
- bool [\\_isTurningRight](#)  
*Drive direction. True for turning right, false otherwise.*

### Additional Inherited Members

#### 4.10.1 Detailed Description

Implements complex behaviour of driving on a square-shaped route.

Square side and direction (right/left) can be implicitly defined.

#### 4.10.2 Constructor & Destructor Documentation

##### 4.10.2.1 BehaviourDriveOnSquare::BehaviourDriveOnSquare ( unsigned int side, bool turningRight )

Constructor with square side and direction (either left or right).

## Parameters

<i>side</i>	Length of square side in units.
<i>turningRight</i>	True for turning right, false otherwise.

#### 4.10.2.2 BehaviourDriveOnSquare::BehaviourDriveOnSquare ( BehaviourStates *states*, unsigned int *side*, bool *turningRight* )

Constructor with [Behaviour](#) states and driving parameters.

## Parameters

<i>states</i>	Vector of <a href="#">Behaviour</a> states to be processed.
<i>side</i>	Length of square side in units.
<i>turningRight</i>	True for turning right, false otherwise.

### 4.10.3 Member Function Documentation

#### 4.10.3.1 StringVector BehaviourDriveOnSquare::getPrototype ( ) [virtual]

Get [BehaviourDriveOnSquare](#) encoded name and its parameters.

## Returns

StringVector with encoded name and parameters as its members.

Reimplemented from [ev3::Behaviour](#).

#### 4.10.3.2 std::string BehaviourDriveOnSquare::getString ( ) [virtual]

Get [BehaviourDriveOnSquare](#) human-readable name.

## Returns

String with name and parameters

Reimplemented from [ev3::Behaviour](#).

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

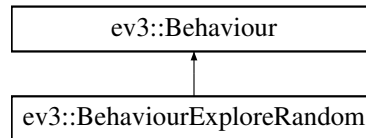
- [/home/panda/Dokumenty/Repos/Ev3Dev/include/action/Behaviour.h](#)
- [/home/panda/Dokumenty/Repos/Ev3Dev/src/action/Behaviour.cpp](#)

## 4.11 ev3::BehaviourExploreRandom Class Reference

Implements complex behaviour of exploring the surrounding with random rotation.

```
#include <Behaviour.h>
```

Inheritance diagram for ev3::BehaviourExploreRandom:



### Public Member Functions

- [BehaviourExploreRandom](#) ()  
*Default constructor.*
- [BehaviourExploreRandom](#) ([BehaviourStates](#) states)  
*Constructor with [Behaviour](#) states parameter.*
- virtual [StringVector](#) [getPrototype](#) ()  
*Get [BehaviourExploreRandom](#) encoded name and its parameters.*
- virtual [std::string](#) [getString](#) ()  
*Get [BehaviourExploreRandom](#) human-readable name.*

### Additional Inherited Members

#### 4.11.1 Detailed Description

Implements complex behaviour of exploring the surrounding with random rotation.

#### 4.11.2 Constructor & Destructor Documentation

##### 4.11.2.1 BehaviourExploreRandom::BehaviourExploreRandom ( BehaviourStates states )

Constructor with [Behaviour](#) states parameter.

Parameters

<a href="#">states</a>	Vector of available <a href="#">Behaviour</a> states.
------------------------	---

#### 4.11.3 Member Function Documentation

##### 4.11.3.1 StringVector BehaviourExploreRandom::getPrototype ( ) [virtual]

Get [BehaviourExploreRandom](#) encoded name and its parameters.

**Returns**

StringVector with encoded name and parameters as its members.

Reimplemented from [ev3::Behaviour](#).

#### 4.11.3.2 `std::string BehaviourExploreRandom::getString ( )` [virtual]

Get [BehaviourExploreRandom](#) human-readable name.

**Returns**

String with name and parameters

Reimplemented from [ev3::Behaviour](#).

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

- `/home/panda/Dokumenty/Repos/Ev3Dev/include/action/Behaviour.h`
- `/home/panda/Dokumenty/Repos/Ev3Dev/src/action/Behaviour.cpp`

## 4.12 [ev3::BehaviourState](#) Class Reference

Encapsulates action and other information in a form of a state.

```
#include <BehaviourState.h>
```

**Public Member Functions**

- [BehaviourState](#) ()=default  
*Default constructor.*
- [BehaviourState](#) (const [BehaviourState](#) &other)=default  
*Default copy constructor.*
- [BehaviourState](#) ([SharedPtrAction](#) action, unsigned int nextState, bool [isStopState](#)=false)  
*Constructor with action, next state id and stop state flag.*
- [BehaviourState](#) ([SharedPtrAction](#) action, unsigned int nextState, [ReactionsTransitions](#) reactions)  
*Constructor with action, next state id and event-state map.*
- unsigned int [process](#) ()  
*Process state in every iteration.*
- [SharedPtrAction](#) [getAction](#) ()  
*State's Action getter.*
- void [setNextState](#) (const unsigned int next)  
*Next state id setter.*
- bool [isStopState](#) ()  
*Stop flag getter.*
- void [setReactions](#) ([ReactionsTransitions](#) reactions)  
*Reactions setter.*
- int [getReaction](#) ([Event::EventType](#) type)  
*Reaction getter.*

## Private Attributes

- [SharedPtrAction \\_action](#) = nullptr  
*Encapsulated action.*
- bool [\\_isExecuted](#) = false  
*True if state was executed, false otherwise.*
- bool [\\_isStopState](#) = false  
*Stop flag.*
- unsigned int [\\_nextStateId](#)  
*Id of the next state.*
- [ReactionsTransitions \\_reactions](#)  
*Map of event-triggered transitions.*

### 4.12.1 Detailed Description

Encapsulates action and other information in a form of a state.

It can contain reactions to different events.

### 4.12.2 Constructor & Destructor Documentation

#### 4.12.2.1 ev3::BehaviourState::BehaviourState ( const BehaviourState & *other* ) [default]

Default copy constructor.

##### Parameters

<i>other</i>	Other <a href="#">BehaviourState</a> object.
--------------	--

#### 4.12.2.2 BehaviourState::BehaviourState ( SharedPtrAction *action*, unsigned int *nextState*, bool *isStopState* = false )

Constructor with action, next state id and stop state flag.

##### Parameters

<i>action</i>	<a href="#">Action</a> object to be executed within this state.
<i>nextState</i>	Id of the next state that will replace this one.
<i>isStopState</i>	Flag defining this state as a in-between, stopping state.

#### 4.12.2.3 BehaviourState::BehaviourState ( SharedPtrAction *action*, unsigned int *nextState*, [ReactionsTransitions](#) *reactions* )

Constructor with action, next state id and event-state map.

## Parameters

<i>action</i>	<a href="#">Action</a> object to be executed within this state,
<i>nextState</i>	Id of the next state that will replace this one.
<i>reactions</i>	Map containing event-state pairs describing reactions.

## 4.12.3 Member Function Documentation

4.12.3.1 `SharedPtrAction BehaviourState::getAction ( )`

State's [Action](#) getter.

## Returns

[Action](#) shared\_ptr object.

4.12.3.2 `int BehaviourState::getReaction ( Event::EventType type )`

Reaction getter.

## Parameters

<i>type</i>	EventType to which reaction occurs.
-------------	-------------------------------------

## Returns

Id of the reaction state.

4.12.3.3 `bool BehaviourState::isStopState ( )`

Stop flag getter.

## Returns

True if state is flagged as a stop state, false otherwise.

4.12.3.4 `unsigned int BehaviourState::process ( )`

Process state in every iteration.

## Returns

Id of the next state.

4.12.3.5 `void BehaviourState::setNextState ( const unsigned int next )`

Next state id setter.

## Parameters

<i>next</i>	Integer defining next state id.
-------------	---------------------------------

4.12.3.6 void BehaviourState::setReactions ( ReactionsTransitions *reactions* )

Reactions setter.

## Parameters

<i>reactions</i>	Map with Event-State pair.
------------------	----------------------------

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/action/[BehaviourState.h](#)
- /home/panda/Dokumenty/Repos/Ev3Dev/src/action/BehaviourState.cpp

## 4.13 ev3::CommUtils::Buffer Struct Reference

Contains buffer and its size.

## Public Attributes

- void \* [buffer](#)  
*Pointer to allocated buffer.*
- size\_t [size](#)  
*Size of bytes allocated.*

## 4.13.1 Detailed Description

Contains buffer and its size.

Used by low-level methods.

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/communication/[CommUtils.h](#)

## 4.14 ev3dev::button Class Reference

## Classes

- struct [file\\_descriptor](#)

## Public Member Functions

- **button** (int bit)
- bool **pressed** () const
- bool **process** ()

## Static Public Member Functions

- static bool **process\_all** ()

## Public Attributes

- std::function< void(bool)> **onclick**

## Static Public Attributes

- static [button](#) **back**
- static [button](#) **left**
- static [button](#) **right**
- static [button](#) **up**
- static [button](#) **down**
- static [button](#) **enter**

## Private Attributes

- int **\_bit**
- bool **\_state** = false
- std::vector< unsigned long > **\_buf**
- std::shared\_ptr< [file\\_descriptor](#) > **\_fd**

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp

## 4.15 `ev3::CircularBuffer< T >` Class Template Reference

Template class for storing N objects of a particular class.

```
#include <CircularBuffer.h>
```



## Public Member Functions

- [CircularBuffer](#) ()=delete  
*No default constructor.*
- [CircularBuffer](#) (unsigned int limit)  
*Constructor with limit parameter.*
- void [push](#) (T object)  
*Put object into the buffer.*
- bool [contain](#) (T object)  
*Get information about certain object.*

## Private Attributes

- std::vector< T > [\\_buffer](#)  
*The actual buffer implemented as a vector.*
- unsigned int [\\_index](#) = 0  
*Current insertion position.*
- unsigned int [\\_limit](#)  
*Upper limit for the buffer.*

### 4.15.1 Detailed Description

```
template<class T>
class ev3::CircularBuffer< T >
```

Template class for storing N objects of a particular class.

Replaces old objects if limit is exceeded.

### 4.15.2 Constructor & Destructor Documentation

4.15.2.1 `template<class T > ev3::CircularBuffer< T >::CircularBuffer ( unsigned int limit )`

Constructor with limit parameter.

Parameters

<i>limit</i>	Positive integer defining upper buffer limit.
--------------	---

### 4.15.3 Member Function Documentation

4.15.3.1 `template<class T> bool ev3::CircularBuffer< T >::contain ( T object )`

Get information about certain object.

## Parameters

<i>object</i>	Object to be found in buffer.
---------------	-------------------------------

## Returns

True if found, false otherwise.

#### 4.15.3.2 `template<class T> void ev3dev::CircularBuffer< T >::push ( T object )`

Put object into the buffer.

Replace old ones if limit is reached.

## Parameters

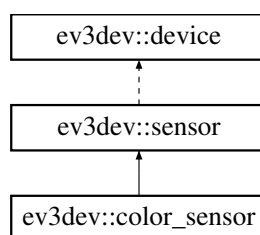
<i>object</i>	Template object to be put into the buffer.
---------------	--

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

- `/home/panda/Dokumenty/Repos/Ev3Dev/include/utils/CircularBuffer.h`

## 4.16 `ev3dev::color_sensor` Class Reference

Inheritance diagram for `ev3dev::color_sensor`:



## Public Member Functions

- **`color_sensor`** (`address_type address=INPUT_AUTO`)
- `int reflected_light_intensity ()`
- `int ambient_light_intensity ()`
- `int color ()`
- `int red ()`
- `int green ()`
- `int blue ()`

## Static Public Attributes

- static const std::string **mode\_col\_reflect** { "COL-REFLECT" }
- static const std::string **mode\_col\_ambient** { "COL-AMBIENT" }
- static const std::string **mode\_col\_color** { "COL-COLOR" }
- static const std::string **mode\_ref\_raw** { "REF-RAW" }
- static const std::string **mode\_rgb\_raw** { "RGB-RAW" }

## Additional Inherited Members

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp

## 4.17 ev3::ColorUtils Class Reference

Stores all available color escape codes.

```
#include <ColorUtils.h>
```

## Public Types

- typedef std::string **colorCode**  
*Type for storing color names.*

## Static Public Member Functions

- static void **printColorTest** ()  
*Print "TEST" in all available colors.*

## Static Public Attributes

- static const **colorCode** **BLACK** {"\033[30m"}  
*BLACK color code.*
- static const **colorCode** **RED** {"\033[31m"}  
*RED color code.*
- static const **colorCode** **GREEN** {"\033[32m"}  
*GREEN color code.*
- static const **colorCode** **YELLOW** {"\033[33m"}  
*YELLOW color code.*
- static const **colorCode** **BLUE** {"\033[34m"}  
*BLUE color code.*
- static const **colorCode** **MAGENTA** {"\033[35m"}  
*MAGENTA color code.*
- static const **colorCode** **CYAN** {"\033[36m"}

- *CYAN color code.*
- static const `colorCode WHITE` {"\033[37m"}
- *WHITE color code.*
- static const `colorCode BLACK_BOLD` {"\033[30;1m"}
- *BLACK\_BOLD color code.*
- static const `colorCode RED_BOLD` {"\033[31;1m"}
- *RED\_BOLD color code.*
- static const `colorCode GREEN_BOLD` {"\033[32;1m"}
- *GREEN\_BOLD color code.*
- static const `colorCode YELLOW_BOLD` {"\033[33;1m"}
- *YELLOW\_BOLD color code.*
- static const `colorCode BLUE_BOLD` {"\033[34;1m"}
- *BLUE\_BOLD color code.*
- static const `colorCode MAGENTA_BOLD` {"\033[35;1m"}
- *MAGENTA\_BOLD color code.*
- static const `colorCode CYAN_BOLD` {"\033[36;1m"}
- *CYAN\_BOLD color code.*
- static const `colorCode WHITE_BOLD` {"\033[37;1m"}
- *WHITE\_BOLD color code.*
- static const `colorCode BLACK_FAINT` {"\033[30;2m"}
- *BLACK\_FAINT color code.*
- static const `colorCode RED_FAINT` {"\033[31;2m"}
- *RED\_FAINT color code.*
- static const `colorCode GREEN_FAINT` {"\033[32;2m"}
- *GREEN\_FAINT color code.*
- static const `colorCode YELLOW_FAINT` {"\033[33;2m"}
- *YELLOW\_FAINT color code.*
- static const `colorCode BLUE_FAINT` {"\033[34;2m"}
- *BLUE\_FAINT color code.*
- static const `colorCode MAGENTA_FAINT` {"\033[35;2m"}
- *MAGENTA\_FAINT color code.*
- static const `colorCode CYAN_FAINT` {"\033[36;2m"}
- *CYAN\_FAINT color code.*
- static const `colorCode WHITE_FAINT` {"\033[37;2m"}
- *WHITE\_FAINT color code.*
- static const `colorCode RESET` {"\033[39;0m"}
- *RESET color.*

#### 4.17.1 Detailed Description

Stores all available color escape codes.

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

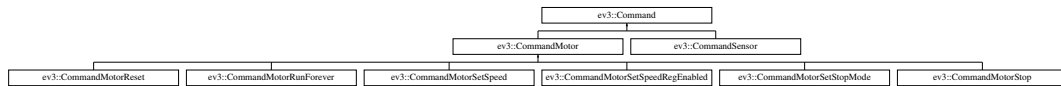
- /home/panda/Dokumenty/Repos/Ev3Dev/include/utils/ColorUtils.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/utils/ColorUtils.cpp

## 4.18 ev3::Command Class Reference

Base class for all command controlling classes.

```
#include <Command.h>
```

Inheritance diagram for ev3::Command:



### Public Member Functions

- [Command](#) ()  
*Default constructor.*
- virtual void [execute](#) ()  
*Execute device specific command.*
- virtual std::string [getString](#) ()  
*Return [Command](#)'s name.*

### Protected Attributes

- std::string [\\_debugInfo](#) = ""  
*String containing [Command](#)'s name.*

#### 4.18.1 Detailed Description

Base class for all command controlling classes.

Each [Command](#) class encapsulates basic motor or sensor operation.

#### 4.18.2 Member Function Documentation

##### 4.18.2.1 std::string Command::getString ( ) [virtual]

Return [Command](#)'s name.

Returns

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

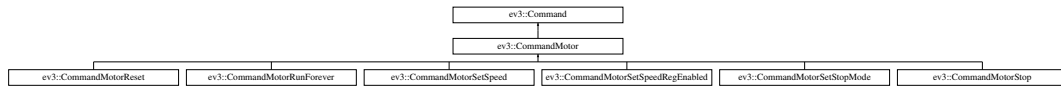
- /home/panda/Dokumenty/Repos/Ev3Dev/include/action/Command.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/action/Command.cpp

## 4.19 ev3::CommandMotor Class Reference

Base class for all motor controlling commands.

```
#include <CommandMotor.h>
```

Inheritance diagram for ev3::CommandMotor:



### Public Member Functions

- [CommandMotor](#) ([Motor](#) &motor)  
*Constructor with [ev3dev::motor](#) parameter.*
- [Motor](#) [getMotor](#) ()  
*Get motor associated with [Command](#).*

### Protected Attributes

- const std::string [SPEED\\_REGULATION\\_ON](#) = "on"  
*[Command](#) parameter to turn speed regulation on a [Motor](#) on.*
- const std::string [SPEED\\_REGULATION\\_OFF](#) = "off"  
*[Command](#) parameter to turn speed regulation on a [Motor](#) off.*
- [Motor](#) [\\_motor](#)  
*[Motor](#) on which this [CommandMotor](#) will be executed.*

### 4.19.1 Detailed Description

Base class for all motor controlling commands.

See also

[ev3dev::motor](#)

### 4.19.2 Constructor & Destructor Documentation

#### 4.19.2.1 CommandMotor::CommandMotor ( [Motor](#) & *motor* )

Constructor with [ev3dev::motor](#) parameter.

Parameters

<i>motor</i>	<a href="#">Motor</a> to execute <a href="#">CommandMotor</a> on.
--------------	---

### 4.19.3 Member Function Documentation

#### 4.19.3.1 Motor CommandMotor::getMotor ( )

Get motor associated with [Command](#).

Returns

[Motor](#) class object.

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

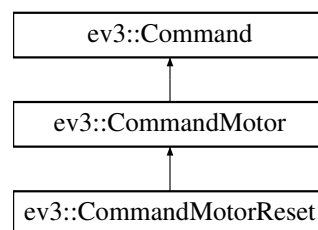
- /home/panda/Dokumenty/Repos/Ev3Dev/include/action/CommandMotor.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/action/CommandMotor.cpp

## 4.20 ev3::CommandMotorReset Class Reference

Calls `reset ( )` method of containing [Motor](#).

```
#include <CommandMotor.h>
```

Inheritance diagram for `ev3::CommandMotorReset`:



### Public Member Functions

- [CommandMotorReset](#) ([Motor](#) &motor)  
*Constructor with `ev3dev::motor` parameter.*
- void [execute](#) ( ) override  
*Perform `reset ( )` method on [Motor](#).*

### Additional Inherited Members

#### 4.20.1 Detailed Description

Calls `reset ( )` method of containing [Motor](#).

#### 4.20.2 Constructor & Destructor Documentation

##### 4.20.2.1 CommandMotorReset::CommandMotorReset ( [Motor](#) & motor )

Constructor with `ev3dev::motor` parameter.

## Parameters

<i>motor</i>	<a href="#">Motor</a> to execute <a href="#">CommandMotor</a> on.
--------------	---

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

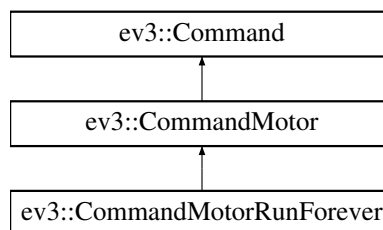
- /home/panda/Dokumenty/Repos/Ev3Dev/include/action/CommandMotor.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/action/CommandMotor.cpp

## 4.21 ev3::CommandMotorRunForever Class Reference

Calls `run_forever()` method of containing [Motor](#).

```
#include <CommandMotor.h>
```

Inheritance diagram for `ev3::CommandMotorRunForever`:



### Public Member Functions

- [CommandMotorRunForever](#) ([Motor](#) &motor)  
*Constructor with `ev3dev::motor` parameter.*
- void [execute](#) () override  
*Perform `run_forever()` method on [Motor](#).*

### Additional Inherited Members

#### 4.21.1 Detailed Description

Calls `run_forever()` method of containing [Motor](#).

#### 4.21.2 Constructor & Destructor Documentation

##### 4.21.2.1 CommandMotorRunForever::CommandMotorRunForever ( [Motor](#) & *motor* )

Constructor with `ev3dev::motor` parameter.



## Parameters

<i>motor</i>	<a href="#">Motor</a> to execute <a href="#">CommandMotor</a> on.
--------------	---

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

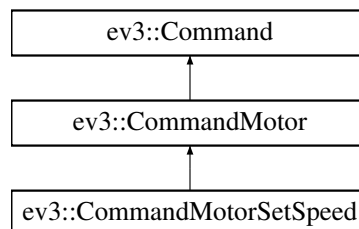
- /home/panda/Dokumenty/Repos/Ev3Dev/include/action/CommandMotor.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/action/CommandMotor.cpp

## 4.22 ev3::CommandMotorSetSpeed Class Reference

Call `set_speed_sp()` method of containing [Motor](#).

```
#include <CommandMotor.h>
```

Inheritance diagram for `ev3::CommandMotorSetSpeed`:



### Public Member Functions

- [CommandMotorSetSpeed](#) ([Motor](#) &motor, int value)  
*Constructor with [ev3dev::motor](#) parameter.*
- void [execute](#) () override  
*Perform `set_speed_sp()` method on [Motor](#).*

### Private Attributes

- int [\\_value](#)  
*Speed value in tachometer pulses per second.*

### Additional Inherited Members

#### 4.22.1 Detailed Description

Call `set_speed_sp()` method of containing [Motor](#).

#### 4.22.2 Constructor & Destructor Documentation

##### 4.22.2.1 CommandMotorSetSpeed::CommandMotorSetSpeed ( [Motor](#) & motor, int value )

Constructor with [ev3dev::motor](#) parameter.

## Parameters

<i>motor</i>	<a href="#">Motor</a> to execute <a href="#">CommandMotor</a> on.
<i>value</i>	Speed value in tacho pulses per second.

## Warning

Speed regulation must be turned on for this to take effect.

## See also

[CommandMotorSetSpeedRegEnabled](#)

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

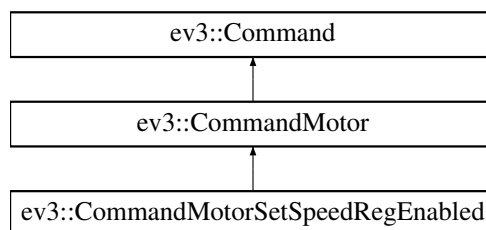
- `/home/panda/Dokumenty/Repos/Ev3Dev/include/action/CommandMotor.h`
- `/home/panda/Dokumenty/Repos/Ev3Dev/src/action/CommandMotor.cpp`

## 4.23 `ev3::CommandMotorSetSpeedRegEnabled` Class Reference

Calls `set_speed_regulation_enabled()` method of containing [Motor](#).

```
#include <CommandMotor.h>
```

Inheritance diagram for `ev3::CommandMotorSetSpeedRegEnabled`:



## Public Member Functions

- [CommandMotorSetSpeedRegEnabled](#) ([Motor](#) &motor, bool value)  
*Constructor with `ev3dev::motor` parameter.*
- void [execute](#) () override  
*Perform `set_speed_regulation_enabled()` on [Motor](#).*

## Private Attributes

- bool [\\_value](#)  
*True value sets speed regulation enabled, false disables it.*

## Additional Inherited Members

### 4.23.1 Detailed Description

Calls `set_speed_regulation_enabled()` method of containing [Motor](#).

### 4.23.2 Constructor & Destructor Documentation

#### 4.23.2.1 CommandMotorSetSpeedRegEnabled::CommandMotorSetSpeedRegEnabled ( [Motor](#) & *motor*, *bool value* )

Constructor with [ev3dev::motor](#) parameter.

##### Parameters

<i>motor</i>	<a href="#">Motor</a> to execute <a href="#">CommandMotor</a> on.
<i>value</i>	If true, turn speed regulation on, false to turn it off.

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

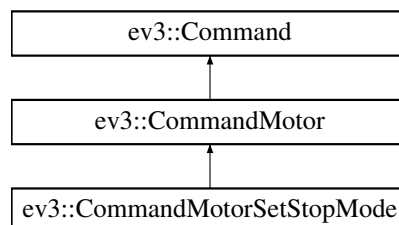
- `/home/panda/Dokumenty/Repos/Ev3Dev/include/action/CommandMotor.h`
- `/home/panda/Dokumenty/Repos/Ev3Dev/src/action/CommandMotor.cpp`

## 4.24 ev3::CommandMotorSetStopMode Class Reference

Calls `set_stop_command()` method of containing [Motor](#).

```
#include <CommandMotor.h>
```

Inheritance diagram for `ev3::CommandMotorSetStopMode`:



## Public Types

- enum [StopMode](#) { [COAST](#), [BRAKE](#), [HOLD](#) }
- Stop modes for motors.*

## Public Member Functions

- [CommandMotorSetStopMode](#) ([Motor](#) &motor, [StopMode](#) mode)  
*Constructor with [ev3dev::motor](#) parameter.*
- void [execute](#) () override  
*Perform [set\\_stop\\_command\(\)](#) method on [Motor](#).*

## Private Attributes

- [StopMode \\_mode](#)  
*Mode chosen to be selected on [Motor](#) when exeuted.*

## Additional Inherited Members

### 4.24.1 Detailed Description

Calls [set\\_stop\\_command\(\)](#) method of containing [Motor](#).

### 4.24.2 Member Enumeration Documentation

#### 4.24.2.1 enum [ev3::CommandMotorSetStopMode::StopMode](#)

Stop modes for motors.

#### Enumerator

**COAST** No voltage. [Motor](#) slowly stops.

**BRAKE** Passive braking. [Motor](#) stops faster.

**HOLD** Active braking. Hardly prevent motor from any movement.

### 4.24.3 Constructor & Destructor Documentation

#### 4.24.3.1 [CommandMotorSetStopMode::CommandMotorSetStopMode](#) ( [Motor](#) & motor, [StopMode](#) mode )

Constructor with [ev3dev::motor](#) parameter.

#### Parameters

<i>motor</i>	<a href="#">Motor</a> to execute <a href="#">CommandMotor</a> on.
<i>mode</i>	Stop mode chosen from <a href="#">StopMode</a> .

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

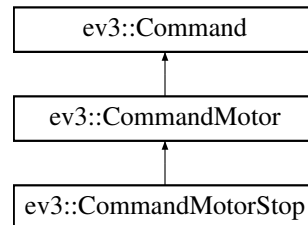
- [/home/panda/Dokumenty/Repos/Ev3Dev/include/action/CommandMotor.h](#)
- [/home/panda/Dokumenty/Repos/Ev3Dev/src/action/CommandMotor.cpp](#)

## 4.25 ev3::CommandMotorStop Class Reference

Calls `stop()` method of containing [Motor](#).

```
#include <CommandMotor.h>
```

Inheritance diagram for `ev3::CommandMotorStop`:



### Public Member Functions

- [CommandMotorStop](#) ([Motor](#) &motor)  
*Constructor with [ev3dev::motor](#) parameter.*
- void [execute](#) () override  
*Perform `stop()` method on [Motor](#).*

### Additional Inherited Members

#### 4.25.1 Detailed Description

Calls `stop()` method of containing [Motor](#).

#### 4.25.2 Constructor & Destructor Documentation

##### 4.25.2.1 CommandMotorStop::CommandMotorStop ( [Motor](#) & motor )

Constructor with [ev3dev::motor](#) parameter.

##### Parameters

<i>motor</i>	<a href="#">Motor</a> to execute <a href="#">CommandMotor</a> on.
--------------	---

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

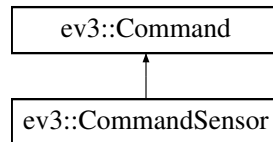
- `/home/panda/Dokumenty/Repos/Ev3Dev/include/action/CommandMotor.h`
- `/home/panda/Dokumenty/Repos/Ev3Dev/src/action/CommandMotor.cpp`

## 4.26 ev3::CommandSensor Class Reference

Base class for all sensor controlling commands.

```
#include <CommandSensor.h>
```

Inheritance diagram for ev3::CommandSensor:



### Public Member Functions

- [CommandSensor](#) ([Sensor](#) &sensor)  
*Constructor with [ev3dev::sensor](#) parameter.*
- [Sensor](#) [getSensor](#) ()  
*Get sensor associated with [Command](#).*

### Protected Attributes

- [Sensor](#) [\\_sensor](#)  
*[Sensor](#) on which this [CommandSensor](#) will be executed.*

### 4.26.1 Detailed Description

Base class for all sensor controlling commands.

See also

[ev3dev::sensor](#)

### 4.26.2 Constructor & Destructor Documentation

#### 4.26.2.1 CommandSensor::CommandSensor ( [Sensor](#) & *sensor* )

Constructor with [ev3dev::sensor](#) parameter.

Parameters

<i>sensor</i>	<a href="#">Sensor</a> to execute <a href="#">CommandSensor</a> on.
---------------	---

### 4.26.3 Member Function Documentation

## 4.26.3.1 Sensor CommandSensor::getSensor ( )

Get sensor associated with [Command](#).

## Returns

[Sensor](#) class object.

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/action/CommandSensor.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/action/CommandSensor.cpp

## 4.27 ev3::Communication Class Reference

Encapsulates low-level communication and adds logic concerning sending and receiving [Message](#) queueing.

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

## Public Member Functions

- [Communication](#) ()  
*Default constructor.*
- std::thread [createThread](#) (Queue< [Message](#) > \*sendQueue, Queue< [Message](#) > \*receiveQueue, bool isMaster=false)  
*Thread creation method (insted of running [Communication](#) in the main thread).*
- void [run](#) (Queue< [Message](#) > \*sendQueue, Queue< [Message](#) > \*receiveQueue, bool isMaster=false)  
*Starts [Communication](#) procedures.*

## Private Member Functions

- void [receive](#) ()  
*Looped [Message](#) receiving.*
- void [send](#) ()  
*Looped [Message](#) sending.*

## Private Attributes

- bool [\\_isMaster](#) = false  
*True if [Communication](#) is synchronized with master, false otherwise.*
- Queue< [Message](#) > \* [\\_sendQueue](#)  
*Out [Message](#) queue.*
- Queue< [Message](#) > \* [\\_receiveQueue](#)  
*In [Message](#) queue.*
- [CommUtils](#) [\\_commUtils](#)  
*Low-level object performing the actual sending/receiving.*
- unsigned int [\\_socket](#)  
*Assigned socket id.*
- unsigned int [\\_port](#) = DEFAULT\_PORT  
*Chosen port number.*

### 4.27.1 Detailed Description

Encapsulates low-level communication and adds logic concerning sending and receiving [Message](#) queueing.

### 4.27.2 Member Function Documentation

**4.27.2.1** `std::thread Communication::createThread ( Queue< Message > * sendQueue, Queue< Message > * receiveQueue, bool isMaster = false )`

Thread creation method (insted of running [Communication](#) in the main thread).

#### Parameters

<i>sendQueue</i>	Out <a href="#">Message</a> queue.
<i>receiveQueue</i>	In <a href="#">Message</a> queue.
<i>isMaster</i>	True if queue is synchronized with master, false otherwise.

#### Returns

New `std::thread` object with [Communication](#) class active.

**4.27.2.2** `void Communication::run ( Queue< Message > * sendQueue, Queue< Message > * receiveQueue, bool isMaster = false )`

Starts [Communication](#) procedures.

#### Parameters

<i>sendQueue</i>	Out <a href="#">Message</a> queue.
<i>receiveQueue</i>	In <a href="#">Message</a> queue.
<i>isMaster</i>	True if queue is synchronized with master, false otherwise.

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

- `/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/Communication.h`
- `/home/panda/Dokumenty/Repos/Ev3Dev/src/communication/Communication.cpp`

## 4.28 ev3::CommUtils Class Reference

Responsible for low-level communication.

```
#include <CommUtils.h>
```



## Classes

- struct [Buffer](#)  
*Contains buffer and its size.*
- struct [NetworkNode](#)  
*Stores information about a particular node in the network.*

## Public Member Functions

- [CommUtils](#) ()  
*Default constructor.*
- int [preparePassiveSocket](#) (unsigned int portNumber)  
*Prepares socket for transmission on given port.*
- int [sendMessage](#) (unsigned int socket, unsigned int port, [Message](#) &message, std::string &proto, bool isMaster, unsigned int repeat=SENT\_MESSAGE\_COPIES)  
*General method for sending messages.*
- int [receiveMessage](#) (unsigned int socket, [Message](#) &message, [NetworkNode](#) &sender)  
*General receive method.*
- int [receiveMessageDelay](#) (unsigned int socket, [Message](#) &message, [NetworkNode](#) &sender, unsigned int msDelay=DEFAULT\_RECEIVE\_DELAY)  
*General receive method with waiting delay.*

## Private Member Functions

- int [sendBroadcastMessage](#) (unsigned int socket, unsigned int port, std::string message)  
*Send message to all recipients in current network.*
- int [sendMessageTo](#) (unsigned int socket, std::string ipAddress, unsigned int destinationPort, std::string message)  
*Send message to specific ipv4 address.*
- int [makeSockAddr](#) (std::string ipAddress, int portNumber, struct sockaddr\_in \*sockaddr)  
*Prepares sockaddr\_in structure.*
- [Buffer](#) [getBufferFromString](#) (const std::string message)  
*Converts Message prototype to Buffer structure.*
- std::string [getStringFromBuffer](#) (const [Buffer](#) buffer)  
*Converts Buffer structure into Message prototype.*

## Private Attributes

- std::map< unsigned int, [NetworkNode](#) > [\\_remotes](#)  
*Map used to register all acquired nodes in the network.*
- std::queue< [NetworkNode](#) > [\\_unregisteredRemotes](#)  
*Queue storing temporal information about not yet registered remotes (agents).*
- [CircularBuffer](#)< std::string > [\\_packetBuffer](#)  
*Circular buffer used to store limited number of previous Message prototypes received.*

### 4.28.1 Detailed Description

Responsible for low-level communication.

Uses socket API and UNIX sending and receiving methods.

## 4.28.2 Member Function Documentation

### 4.28.2.1 CommUtils::Buffer CommUtils::getBufferFromString ( const std::string *message* ) [private]

Converts [Message](#) prototype to [Buffer](#) structure.

#### Parameters

<i>message</i>	String prototype to be converted.
----------------	-----------------------------------

#### Returns

[Buffer](#) object after memory allocation.

### 4.28.2.2 std::string CommUtils::getStringFromBuffer ( const Buffer *buffer* ) [private]

Converts [Buffer](#) structure into [Message](#) prototype.

#### Parameters

<i>buffer</i>	Structure with allocated memory with data.
---------------	--

#### Returns

String with [Message](#) prototype.

### 4.28.2.3 int CommUtils::makeSockAddr ( std::string *ipAddress*, int *portNumber*, struct sockaddr\_in \* *sockaddr* ) [private]

Prepares sockaddr\_in structure.

#### Parameters

<i>ipAddress</i>	String containing ipv4 address.
<i>portNumber</i>	Number of port to communicate.
<i>sockaddr</i>	Structure to be set after calling.

#### Returns

Error code.

### 4.28.2.4 int CommUtils::preparePassiveSocket ( unsigned int *portNumber* )

Prepares socket for transmission on given port.

## Parameters

<i>portNumber</i>	Port number to assign socket to.
-------------------	----------------------------------

## Returns

Id of the socket assigned.

4.28.2.5 `int CommUtils::receiveMessage ( unsigned int socket, Message & message, NetworkNode & sender )`

General receive method.

## Parameters

<i>socket</i>	Previously prepared socket.
<i>message</i>	<a href="#">Message</a> reference to be set after receiving.
<i>sender</i>	<a href="#">NetworkNode</a> to be set after receiving.

## Returns

Error code or positive integer with number of bytes received.

4.28.2.6 `int CommUtils::receiveMessageDelay ( unsigned int socket, Message & message, NetworkNode & sender, unsigned int msDelay = DEFAULT_RECEIVE_DELAY )`

General receive method with waiting delay.

## Parameters

<i>socket</i>	Previously prepared socket.
<i>message</i>	<a href="#">Message</a> reference to be set after receiving.
<i>sender</i>	<a href="#">NetworkNode</a> to be set after receiving.
<i>msDelay</i>	Maximum time in milliseconds to wait for message.

## Returns

Error code or positive integer with number of bytes received.

4.28.2.7 `int CommUtils::sendBroadcastMessage ( unsigned int socket, unsigned int port, std::string message )`  
[private]

Send message to all recipients in current network.

## Parameters

<i>socket</i>	Previously prepared socket.
<i>port</i>	Number of port to communicate through.
<i>message</i>	<a href="#">Message</a> to be sent.

**Returns**

Error code or positive integer with number of bytes sent.

**4.28.2.8** `int CommUtils::sendMessage ( unsigned int socket, unsigned int port, Message & message, std::string & proto, bool isMaster, unsigned int repeat = SENT_MESSAGE_COPIES )`

General method for sending messages.

**Parameters**

<i>socket</i>	Previously prepared socket.
<i>port</i>	Number of port to communicate through.
<i>message</i>	<a href="#">Message</a> to be sent.
<i>proto</i>	<a href="#">Message</a> prototype passed to avoid its multiple encoding.
<i>isMaster</i>	Flag from <a href="#">Communication</a> class. True if master is the sender.
<i>repeat</i>	Number of copies to be sent.

**Returns**

Error code or positive integer with number of bytes sent.

**4.28.2.9** `int CommUtils::sendMessageTo ( unsigned int socket, std::string ipAddress, unsigned int destinationPort, std::string message ) [private]`

Send message to specific ipv4 address.

**Parameters**

<i>socket</i>	Previously prepared socket.
<i>ipAddress</i>	String containing ipv4 address.
<i>destinationPort</i>	Number of recipient port.
<i>message</i>	<a href="#">Message</a> to be sent.

**Returns**

Error code or positive integer with number of bytes sent.

**4.28.3 Member Data Documentation**

**4.28.3.1** `CircularBuffer<std::string> ev3::CommUtils::_packetBuffer [private]`

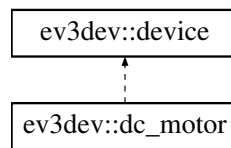
Circular buffer used to store limited number of previous [Message](#) prototypes received.

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

- `/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/CommUtils.h`
- `/home/panda/Dokumenty/Repos/Ev3Dev/src/communication/CommUtils.cpp`

## 4.29 ev3dev::dc\_motor Class Reference

Inheritance diagram for ev3dev::dc\_motor:



### Public Member Functions

- **dc\_motor** (address\_type address=OUTPUT\_AUTO)
- auto **set\_command** (std::string v) -> decltype(\*this)
- mode\_set **commands** () const
- std::string **driver\_name** () const
- int **duty\_cycle** () const
- int **duty\_cycle\_sp** () const
- auto **set\_duty\_cycle\_sp** (int v) -> decltype(\*this)
- std::string **polarity** () const
- auto **set\_polarity** (std::string v) -> decltype(\*this)
- std::string **address** () const
- int **ramp\_down\_sp** () const
- auto **set\_ramp\_down\_sp** (int v) -> decltype(\*this)
- int **ramp\_up\_sp** () const
- auto **set\_ramp\_up\_sp** (int v) -> decltype(\*this)
- mode\_set **state** () const
- auto **set\_stop\_command** (std::string v) -> decltype(\*this)
- mode\_set **stop\_commands** () const
- int **time\_sp** () const
- auto **set\_time\_sp** (int v) -> decltype(\*this)
- void **run\_forever** ()
- void **run\_timed** ()
- void **run\_direct** ()
- void **stop** ()

### Static Public Attributes

- static const std::string **command\_run\_forever** { "run-forever" }
- static const std::string **command\_run\_timed** { "run-timed" }
- static const std::string **command\_run\_direct** { "run-direct" }
- static const std::string **command\_stop** { "stop" }
- static const std::string **polarity\_normal** { "normal" }
- static const std::string **polarity\_inversed** { "inversed" }
- static const std::string **stop\_command\_coast** { "coast" }
- static const std::string **stop\_command\_brake** { "brake" }

### Protected Attributes

- std::string **\_port\_name**

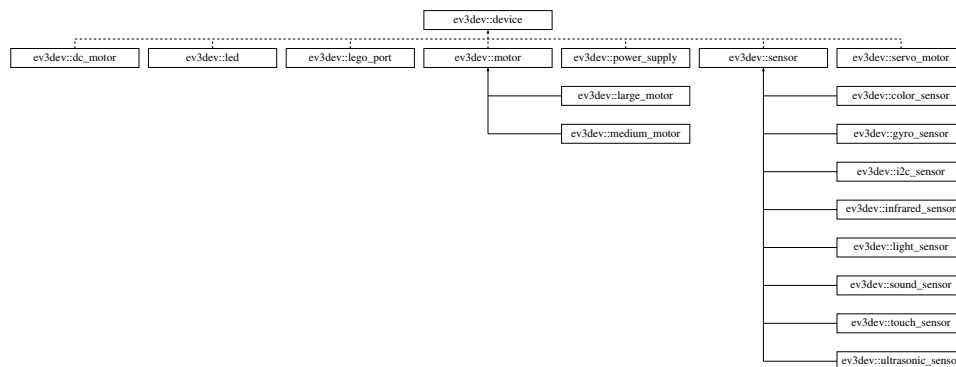
## Additional Inherited Members

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp

## 4.30 ev3dev::device Class Reference

Inheritance diagram for ev3dev::device:



## Public Member Functions

- bool **connect** (const std::string &dir, const std::string &pattern, const std::map< std::string, std::set< std::string >> &match) noexcept
- bool **connected** () const
- int **device\_index** () const
- int **get\_attr\_int** (const std::string &name) const
- void **set\_attr\_int** (const std::string &name, int value)
- std::string **get\_attr\_string** (const std::string &name) const
- void **set\_attr\_string** (const std::string &name, const std::string &value)
- std::string **get\_attr\_line** (const std::string &name) const
- mode\_set **get\_attr\_set** (const std::string &name, std::string \*pCur=nullptr) const
- std::string **get\_attr\_from\_set** (const std::string &name) const

## Protected Attributes

- std::string **\_path**
- int **\_device\_index** = -1

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp

## 4.31 ev3::Devices Class Reference

Singleton class responsible for managing devices connected to the robot.

```
#include <Devices.h>
```

### Public Types

- typedef std::map< ev3dev::port\_type, [Motor](#) > [MotorsVector](#)  
*Type for mapping [Motor](#) objects to their assigned ports.*
- typedef std::map< ev3dev::port\_type, [Sensor](#) > [SensorsVector](#)  
*Type for mapping [Sensor](#) objects to their assigned ports.*
- typedef std::vector< std::pair< ev3dev::port\_type, ev3dev::device\_type > > [RequiredDevices](#)  
*Vector of pairs mapping port to required device.*
- typedef std::map< ev3dev::port\_type, SensorValue > [SensorStatus](#)  
*Map containing pairs port-values for all sensors.*

### Public Member Functions

- bool [checkDevices](#) ([RequiredDevices](#) &devices)  
*Check connected devices and requirements.*
- void [update](#) ()  
*Performs update on measuring values.*
- void [addListener](#) ([Sensor::SensorType](#) type)  
*Add listener for given [Sensor](#) type.*
- void [removeListener](#) ([Sensor::SensorType](#) type)  
*Remove listener for given [Sensor](#) type.*
- [Motor](#) [getMotor](#) (ev3dev::port\_type port)  
*[Motor](#) getter.*
- [Sensor](#) [getSensor](#) (ev3dev::port\_type port)  
*[Sensor](#) getter.*
- void [setSafetyTouchSensor](#) (ev3dev::port\_type port)  
*Specify port on which touch sensor that detects collisions is.*
- void [setProximitySensor](#) (ev3dev::port\_type port)  
*Specify port on which proximity sensor that detects obstacles is.*
- void [stopAllDevices](#) ()  
*Stops all [Motors](#).*

### Static Public Member Functions

- static [Devices](#) \* [getInstance](#) ()  
*Instance getter.*
- static void [destroy](#) ()  
*Deallocate instance.*

### Static Public Attributes

- static const ev3dev::port\_type [PORT\\_ANY](#) {"any"}  
*Can be used to define that device port is irrelevant.*

## Protected Member Functions

- [Devices](#) ()  
*Default protected constructor (preventing object construction).*
- [Devices](#) (const [Devices](#) &other)  
*Default protected copy constructor (preventing object construction by copying).*
- [Devices](#) & operator= (const [Devices](#) &other)  
*Protected assignment operator (preventing object assignment).*
- [~Devices](#) ()  
*Default protected destructor (preventing object unwanted destruction).*

## Protected Attributes

- std::map< [Sensor::SensorType](#), bool > [\\_listeners](#)  
*[Sensor](#) listeners.*
- std::map< ev3dev::port\_type, int > [\\_safetyTouchSensors](#)  
*Touch sensor for detecting collisions.*
- std::map< ev3dev::port\_type, int > [\\_proximitySensors](#)  
*Proximity sensors for detecting obstacles.*
- [MotorsVector](#) [\\_motors](#)  
*Stored [Motor](#) objects.*
- [SensorsVector](#) [\\_sensors](#)  
*Stored [Sensor](#) objects.*
- [SensorStatus](#) [\\_status](#)  
*Sensors' status with all values.*

## Static Protected Attributes

- static [Devices](#) \* [\\_instance](#) = nullptr  
*Instance of [Devices](#) singleton class.*

### 4.31.1 Detailed Description

Singleton class responsible for managing devices connected to the robot.

### 4.31.2 Constructor & Destructor Documentation

#### 4.31.2.1 `ev3::Devices::Devices ( const Devices & other )` [protected]

Default protected copy constructor (preventing object construction by copying).

#### Parameters

<i>other</i>	Other <a href="#">Devices</a> object.
--------------	---------------------------------------



### 4.31.3 Member Function Documentation

#### 4.31.3.1 void Devices::addListener ( [Sensor::SensorType](#) *type* )

Add listener for given [Sensor](#) type.

Parameters

<i>type</i>	Type of <a href="#">Sensor</a> for which value to watch.
-------------	--

#### 4.31.3.2 bool Devices::checkDevices ( [RequiredDevices](#) & *devices* )

Check connected devices and requirements.

Parameters

<i>devices</i>	Vector of required devices.
----------------	-----------------------------

Returns

True if everything is connected properly, false otherwise.

#### 4.31.3.3 [Devices](#) \* Devices::getInstance ( ) [static]

Instance getter.

Returns

Create previously or new instance of class [Devices](#).

#### 4.31.3.4 [Motor](#) Devices::getMotor ( [ev3dev::port\\_type](#) *port* )

[Motor](#) getter.

Parameters

<i>port</i>	Port id on which the <a href="#">Motor</a> is.
-------------	--

Returns

[Motor](#) object assigned to specified port.

#### 4.31.3.5 [Sensor](#) Devices::getSensor ( [ev3dev::port\\_type](#) *port* )

[Sensor](#) getter.

## Parameters

<i>port</i>	Port id on which the <a href="#">Sensor</a> is.
-------------	---

## Returns

[Sensor](#) object assigned to specified port.

#### 4.31.3.6 **Devices& ev3::Devices::operator= ( const Devices & *other* )** [protected]

Protected assignment operator (preventing object assignment).

## Parameters

<i>other</i>	Other <a href="#">Devices</a> object.
--------------	---------------------------------------

## Returns

Copy of passed object.

#### 4.31.3.7 **void Devices::removeListener ( Sensor::SensorType *type* )**

Remove listener for given [Sensor](#) type.

## Parameters

<i>type</i>	Type of <a href="#">Sensor</a> for which value not to watch anymore.
-------------	--

#### 4.31.3.8 **void Devices::setProximitySensor ( ev3dev::port\_type *port* )**

Specify port on which proximity sensor that detects obstacles is.

## Parameters

<i>port</i>	Port for proximity sensor.
-------------	----------------------------

#### 4.31.3.9 **void Devices::setSafetyTouchSensor ( ev3dev::port\_type *port* )**

Specify port on which touch sensor that detects collisions is.

## Parameters

<i>port</i>	Port for safety touch sensor.
-------------	-------------------------------

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

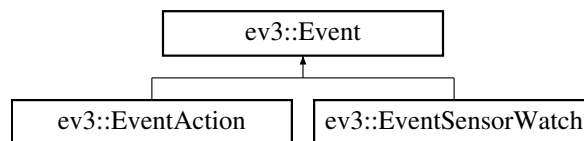
- /home/panda/Dokumenty/Repos/Ev3Dev/include/robot/Devices.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/robot/Devices.cpp

## 4.32 ev3::Event Class Reference

Base class for all [Event](#) classes.

```
#include <Event.h>
```

Inheritance diagram for ev3::Event:



### Public Types

- enum [EventType](#) {  
[EMPTY](#), [BEHAVIOUR\\_START](#), [BEHAVIOUR\\_STOP](#), [SENSOR\\_WATCH](#),  
[OBSTACLE\\_DETECTED](#), [PROXIMITY\\_ALERT](#), [ACTION\\_FINISHED](#), [ACTION\\_INTERR](#) }  
*Event type.*

### Public Member Functions

- [Event](#) ()  
*Default constructor.*
- [Event](#) ([EventType](#) type)  
*Constructor with Event type parameter.*
- [EventType](#) [getType](#) ()  
*Event type getter.*
- std::string [getStringType](#) ()  
*Get human-readable Event name.*

### Private Attributes

- [EventType](#) \_type  
*Event type value.*

#### 4.32.1 Detailed Description

Base class for all [Event](#) classes.

Triggered when certain events occur during the robot's main loop execution.

## 4.32.2 Member Enumeration Documentation

### 4.32.2.1 enum ev3::Event::EventType

[Event](#) type.

Enumerator

**EMPTY** Empty event, no meaning.  
**BEHAVIOUR\_START** [Behaviour](#) was started.  
**BEHAVIOUR\_STOP** [Behaviour](#) was stopped.  
**SENSOR\_WATCH** Value was measured from sensor.  
**OBSTACLE\_DETECTED** [Robot](#) hit an obstacle.  
**PROXIMITY\_ALERT** Distance sensor triggered alert.  
**ACTION\_FINISHED** Triggered when action was properly executed.  
**ACTION\_INTERR** Triggered when action was interrupted.

## 4.32.3 Constructor & Destructor Documentation

### 4.32.3.1 Event::Event ( EventType type )

Constructor with [Event](#) type parameter.

Parameters

<i>type</i>	Type of the event triggered.
-------------	------------------------------

## 4.32.4 Member Function Documentation

### 4.32.4.1 std::string Event::getStringType ( )

Get human-readable [Event](#) name.

Returns

String with [Event](#) name.

### 4.32.4.2 Event::EventType Event::getType ( )

[Event](#) type getter.

Returns

EventType value.

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

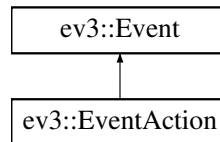
- /home/panda/Dokumenty/Repos/Ev3Dev/include/communication/Event.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/communication/Event.cpp

## 4.33 ev3::EventAction Class Reference

[Event](#) class triggered when something happened with [Action](#).

```
#include <Event.h>
```

Inheritance diagram for ev3::EventAction:



### Public Member Functions

- [EventAction](#) ([EventType](#) eventType, [Action::ActionType](#) actionType)  
*Constructor with [Event](#) type and [Action](#) type.*
- [Action::ActionType](#) getActionType ()  
*[Action](#) type getter.*

### Private Attributes

- [Action::ActionType](#) \_actionType  
*Stored [Action](#) type.*

### Additional Inherited Members

#### 4.33.1 Detailed Description

[Event](#) class triggered when something happened with [Action](#).

#### 4.33.2 Constructor & Destructor Documentation

##### 4.33.2.1 EventAction::EventAction ( EventType eventType, Action::ActionType actionType )

Constructor with [Event](#) type and [Action](#) type.

##### Parameters

<i>eventType</i>	One of <a href="#">Event</a> types concerning actions.
<i>actionType</i>	Type of <a href="#">Action</a> this event concerns.

#### 4.33.3 Member Function Documentation

#### 4.33.3.1 Action::ActionType EventAction::getActionType ( )

Action type getter.

##### Returns

Stored type of Action.

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/communication/Event.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/communication/Event.cpp

## 4.34 ev3::EventQueue Class Reference

Singleton class responsible for managing Event objects.

```
#include <EventQueue.h>
```

### Public Member Functions

- void push (SharedPtrEvent event)  
*Insert new Event object to the queue.*
- SharedPtrEvent pop ()  
*Removes first object from the queue.*
- bool empty ()  
*Check whether queue is empty.*
- unsigned int size ()  
*Queue size getter.*

### Static Public Member Functions

- static EventQueue \* getInstance ()  
*Instance getter.*
- static void destroy ()  
*Deallocate instance.*

### Protected Member Functions

- EventQueue ()  
*Default protected constructor (preventing object construction).*
- EventQueue (const EventQueue &other)  
*Default protected copy constructor (preventing object construction by copying).*
- EventQueue & operator= (const EventQueue &other)  
*Protected assignment operator (preventing object assignment).*
- ~EventQueue ()  
*Default protected destructor (preventing object unwanted destruction).*

## Protected Attributes

- `std::queue< SharedPtrEvent > _queue`  
*The actual queue implemented as `std::queue`.*
- `std::mutex _mutex`  
*Synchronization mutex.*

## Static Protected Attributes

- static `EventQueue * _instance = nullptr`  
*Instance of [EventQueue](#) singleton class.*

### 4.34.1 Detailed Description

Singleton class responsible for managing [Event](#) objects.

Instance is shared between many classes and threads.

### 4.34.2 Constructor & Destructor Documentation

#### 4.34.2.1 `ev3::EventQueue::EventQueue ( const EventQueue & other )` `[protected]`

Default protected copy constructor (preventing object construction by copying).

#### Parameters

<i>other</i>	Other <a href="#">EventQueue</a> object.
--------------	--

### 4.34.3 Member Function Documentation

#### 4.34.3.1 `bool EventQueue::empty ( )`

Check whether queue is empty.

#### Returns

True if queue is empty, false otherwise.

#### 4.34.3.2 `EventQueue * EventQueue::getInstance ( )` `[static]`

Instance getter.

#### Returns

Create previously or new instance of class [EventQueue](#).

#### 4.34.3.3 `EventQueue& ev3::EventQueue::operator= ( const EventQueue & other )` `[protected]`

Protected assignment operator (preventing object assignment).



## Parameters

<i>other</i>	Other <a href="#">EventQueue</a> object.
--------------	--

## Returns

Copy of passed object.

4.34.3.4 **SharedPtrEvent** EventQueue::pop ( )

Removes first object from the queue.

## Returns

Copy of removed object.

4.34.3.5 **void** EventQueue::push ( **SharedPtrEvent** *event* )

Insert new [Event](#) object to the queue.

## Parameters

<i>event</i>	<a href="#">Event</a> object to be inserted.
--------------	--

4.34.3.6 **unsigned int** EventQueue::size ( )

[Queue](#) size getter.

## Returns

Number of elements stored in queue.

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

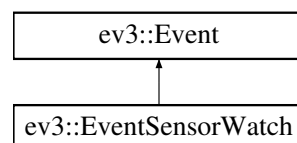
- /home/panda/Dokumenty/Repos/Ev3Dev/include/utills/[EventQueue.h](#)
- /home/panda/Dokumenty/Repos/Ev3Dev/src/utills/EventQueue.cpp

4.35 **ev3::EventSensorWatch** Class Reference

Triggered when measurement of certain [Sensor](#) occurred.

```
#include <Event.h>
```

Inheritance diagram for ev3::EventSensorWatch:



## Public Member Functions

- [EventSensorWatch](#) ([Sensor::SensorType](#) type, [SensorValue](#) value)  
*Constructor with sensor type and measured value.*
- [SensorValue](#) [getValue](#) ()  
*Stored sensor value getter.*
- [Sensor::SensorType](#) [getType](#) ()  
*Stored [Sensor](#) type getter.*

## Private Attributes

- [Sensor::SensorType](#) [\\_sensorType](#)  
*[Sensor](#) type this event concerns.*
- [SensorValue](#) [\\_sensorValue](#)  
*Measured values.*

## Additional Inherited Members

### 4.35.1 Detailed Description

Triggered when measurement of certain [Sensor](#) occurred.

### 4.35.2 Constructor & Destructor Documentation

#### 4.35.2.1 [EventSensorWatch::EventSensorWatch](#) ( [Sensor::SensorType](#) type, [SensorValue](#) value )

Constructor with sensor type and measured value.

#### Parameters

<i>type</i>	Value identifying sensor type.
<i>value</i>	Vector with all measurements.

### 4.35.3 Member Function Documentation

#### 4.35.3.1 [Sensor::SensorType](#) [EventSensorWatch::getType](#) ( )

Stored [Sensor](#) type getter.

#### Returns

[Sensor](#) type value.

## 4.35.3.2 SensorValue EventSensorWatch::getValue ( )

Stored sensor value getter.

## Returns

Vector with certain [Sensor](#) measurements.

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/communication/Event.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/communication/Event.cpp

## 4.36 ev3dev::button::file\_descriptor Struct Reference

## Public Member Functions

- **file\_descriptor** (const char \*path, int flags)
- **operator int** ()

## Public Attributes

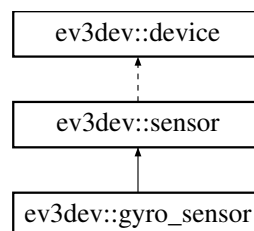
- int **\_fd**

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp

## 4.37 ev3dev::gyro\_sensor Class Reference

Inheritance diagram for ev3dev::gyro\_sensor:



## Public Member Functions

- **gyro\_sensor** (address\_type address=INPUT\_AUTO)
- int **angle** ()
- int **rate** ()

### Static Public Attributes

- static const std::string **mode\_gyro\_ang** { "GYRO-ANG" }
- static const std::string **mode\_gyro\_rate** { "GYRO-RATE" }
- static const std::string **mode\_gyro\_fas** { "GYRO-FAS" }
- static const std::string **mode\_gyro\_g\_a** { "GYRO-G&A" }
- static const std::string **mode\_gyro\_cal** { "GYRO-CAL" }

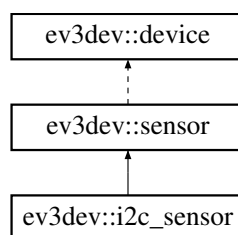
### Additional Inherited Members

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp

## 4.38 ev3dev::i2c\_sensor Class Reference

Inheritance diagram for ev3dev::i2c\_sensor:



### Public Member Functions

- **i2c\_sensor** (address\_type address=INPUT\_AUTO)
- std::string **fw\_version** () const
- int **poll\_ms** () const
- auto **set\_poll\_ms** (int v) -> decltype(\*this)

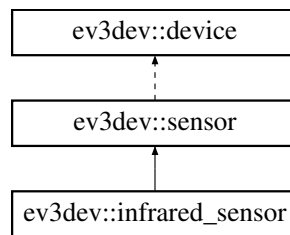
### Additional Inherited Members

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp

## 4.39 ev3dev::infrared\_sensor Class Reference

Inheritance diagram for ev3dev::infrared\_sensor:



### Public Member Functions

- **infrared\_sensor** (address\_type address=INPUT\_AUTO)
- int **proximity** ()

### Static Public Attributes

- static const std::string **mode\_ir\_prox** { "IR-PROX" }
- static const std::string **mode\_ir\_seek** { "IR-SEEK" }
- static const std::string **mode\_ir\_remote** { "IR-REMOTE" }
- static const std::string **mode\_ir\_rem\_a** { "IR-REM-A" }
- static const std::string **mode\_ir\_cal** { "IR-CAL" }

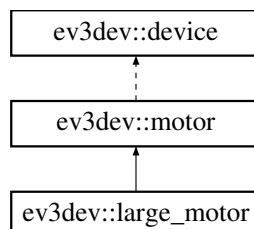
### Additional Inherited Members

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp

## 4.40 ev3dev::large\_motor Class Reference

Inheritance diagram for ev3dev::large\_motor:



### Public Member Functions

- **large\_motor** (address\_type address=OUTPUT\_AUTO)

## Additional Inherited Members

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp

## 4.41 ev3dev::lcd Class Reference

### Public Member Functions

- bool **available** () const
- uint32\_t **resolution\_x** () const
- uint32\_t **resolution\_y** () const
- uint32\_t **bits\_per\_pixel** () const
- uint32\_t **frame\_buffer\_size** () const
- uint32\_t **line\_length** () const
- unsigned char \* **frame\_buffer** ()
- void **fill** (unsigned char pixel)

### Protected Member Functions

- void **init** ()
- void **deinit** ()

### Private Attributes

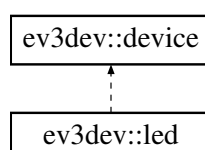
- unsigned char \* **\_fb**
- uint32\_t **\_fbsize**
- uint32\_t **\_llength**
- uint32\_t **\_xres**
- uint32\_t **\_yres**
- uint32\_t **\_bpp**

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp

## 4.42 ev3dev::led Class Reference

Inheritance diagram for ev3dev::led:



## Public Member Functions

- **led** (std::string name)
- int **max\_brightness** () const
- int **brightness** () const
- auto **set\_brightness** (int v) -> decltype(\*this)
- mode\_set **triggers** () const
- std::string **trigger** () const
- auto **set\_trigger** (std::string v) -> decltype(\*this)
- int **delay\_on** () const
- auto **set\_delay\_on** (int v) -> decltype(\*this)
- int **delay\_off** () const
- auto **set\_delay\_off** (int v) -> decltype(\*this)
- float **brightness\_pct** () const
- auto **set\_brightness\_pct** (float v) -> decltype(\*this)
- void **on** ()
- void **off** ()
- void **flash** (unsigned on\_ms, unsigned off\_ms)

## Static Public Member Functions

- static void **set\_color** (const std::vector< [led](#) \* > &group, const std::vector< float > &color)
- static void **all\_off** ()

## Static Public Attributes

- static [led](#) **red\_left** {"ev3:left:red:ev3dev"}
- static [led](#) **red\_right** {"ev3:right:red:ev3dev"}
- static [led](#) **green\_left** {"ev3:left:green:ev3dev"}
- static [led](#) **green\_right** {"ev3:right:green:ev3dev"}
- static std::vector< [led](#) \* > **left** { &led::red\_left, &led::green\_left }
- static std::vector< [led](#) \* > **right** { &led::red\_right, &led::green\_right }
- static std::vector< float > **red** { static\_cast<float>(1), static\_cast<float>(0) }
- static std::vector< float > **green** { static\_cast<float>(0), static\_cast<float>(1) }
- static std::vector< float > **amber** { static\_cast<float>(1), static\_cast<float>(1) }
- static std::vector< float > **orange** { static\_cast<float>(1), static\_cast<float>(0.5) }
- static std::vector< float > **yellow** { static\_cast<float>(0.5), static\_cast<float>(1) }

## Protected Attributes

- int **\_max\_brightness** = 0

## Additional Inherited Members

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp

## 4.43 ev3::LedControl Class Reference

Class specifically designed to eliminate ev3dev library limitations of controlling LED panel.

```
#include <LedControl.h>
```

### Public Types

- enum `LedType` {  
`RED_L` = 1, `RED_R` = 1 << 1, `GREEN_L` = 1 << 2, `GREEN_R` = 1 << 3,  
`RED_ALL` = `RED_L` | `RED_R`, `GREEN_ALL` = `GREEN_L` | `GREEN_R`, `ALL` = `RED_ALL` | `GREEN_ALL` }  
*Type of LED diode.*
- enum `LedColors` { `RED`, `AMBER`, `YELLOW`, `GREEN` }  
*Predefined colors, that particular combination of diodes can represent.*

### Public Member Functions

- virtual `~LedControl` ()  
*Default destructor.*
- void `on` (unsigned int leds=`LedType::ALL`, unsigned int brightness=`MAX_BRIGHTNESS`)  
*Turn the specified diodes on.*
- void `onExclusive` (unsigned int leds=`LedType::ALL`, unsigned int brightness=`MAX_BRIGHTNESS`)  
*Turn the specified diodes on and also turn off the other ones.*
- void `off` (unsigned int leds=`LedType::ALL`)  
*Turn the specified diodes off.*
- void `setColor` (`LedColors` color)  
*Set diodes to match particular color.*
- void `reset` ()  
*Ends flashing and turns all diodes off.*
- void `flash` (unsigned int leds, unsigned int msInterval, unsigned int repeat=1, unsigned int brightnessRed=`MAX_BRIGHTNESS`, unsigned int brightnessGreen=`MAX_BRIGHTNESS`)  
*Orders diodes to flash with given interval.*
- void `flashColor` (`LedColors` color, unsigned int msInterval, unsigned int repeat=1)  
*Orders diodes to flash a particular color with given interval.*
- void `endFlashing` ()  
*Stops flashing.*

### Static Public Attributes

- static const unsigned int `MAX_BRIGHTNESS` = 255  
*Maximum value of brightness.*

### Private Attributes

- std::thread `_flashThread`  
*Parallel thread responsible for flashing.*
- bool `_isFlashingEnded`  
*Synchronization variable indicating, when the flash has to end.*



### 4.43.1 Detailed Description

Class specifically designed to eliminate ev3dev library limitations of controlling LED panel.

### 4.43.2 Member Enumeration Documentation

#### 4.43.2.1 enum ev3::LedControl::LedColors

Predefined colors, that particular combination of diodes can represent.

Enumerator

**RED** Only red diode.  
**AMBER** Red with a little bit of green.  
**YELLOW** Little red and full green.  
**GREEN** Only green diode.

#### 4.43.2.2 enum ev3::LedControl::LedType

Type of LED diode.

Enumerator

**RED\_L** Red left diode.  
**RED\_R** Red right diode.  
**GREEN\_L** Green left diode.  
**GREEN\_R** Green right diode.  
**RED\_ALL** Both red diodes.  
**GREEN\_ALL** Both green diodes.  
**ALL** All four diodes.

### 4.43.3 Member Function Documentation

4.43.3.1 void LedControl::flash ( unsigned int *leds*, unsigned int *msInterval*, unsigned int *repeat* = 1, unsigned int *brightnessRed* = MAX\_BRIGHTNESS, unsigned int *brightnessGreen* = MAX\_BRIGHTNESS )

Orders diodes to flash with given interval.

Parameters

<i>leds</i>	Combination of <a href="#">LedControl::LedType</a> values.
<i>msInterval</i>	Flash interval in milliseconds.
<i>repeat</i>	Number of iterations or 0 for infinite flashing.
<i>brightnessRed</i>	Brightness of the red diodes.
<i>brightnessGreen</i>	Brightness of the green diodes.

#### 4.43.3.2 void LedControl::flashColor ( LedColors color, unsigned int msInterval, unsigned int repeat = 1 )

Orders diodes to flash a particular color with given interval.

##### Parameters

<i>color</i>	Type of color to be displayed.
<i>msInterval</i>	Flash interval in milliseconds.
<i>repeat</i>	Number of iterations or 0 for infinite flashing.

#### 4.43.3.3 void LedControl::off ( unsigned int leds = LedType : : ALL )

Turn the specified diodes off.

##### Parameters

<i>leds</i>	Combination of <a href="#">LedControl::LedType</a> values.
-------------	--

#### 4.43.3.4 void LedControl::on ( unsigned int leds = LedType : : ALL, unsigned int brightness = MAX\_BRIGHTNESS )

Turn the specified diodes on.

##### Parameters

<i>leds</i>	Combination of <a href="#">LedControl::LedType</a> values.
<i>brightness</i>	Value of brightness to be set.

#### 4.43.3.5 void LedControl::onExclusive ( unsigned int leds = LedType : : ALL, unsigned int brightness = MAX\_BRIGHTNESS )

Turn the specified diodes on and also turn off the other ones.

##### Parameters

<i>leds</i>	Combination of <a href="#">LedControl::LedType</a> values.
<i>brightness</i>	Value of brightness to be set.

#### 4.43.3.6 void LedControl::setColor ( LedColors color )

Set diodes to match particular color.

##### Parameters

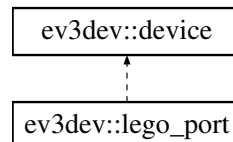
<i>color</i>	Type of to be displayed.
--------------	--------------------------

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/control/LedControl.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/control/LedControl.cpp

## 4.44 ev3dev::lego\_port Class Reference

Inheritance diagram for ev3dev::lego\_port:



### Public Member Functions

- **lego\_port** (address\_type)
- std::string **driver\_name** () const
- mode\_set **modes** () const
- std::string **mode** () const
- auto **set\_mode** (std::string v) -> decltype(\*this)
- std::string **address** () const
- auto **set\_set\_device** (std::string v) -> decltype(\*this)
- std::string **status** () const

### Protected Member Functions

- bool **connect** (const std::map< std::string, std::set< std::string >> &) noexcept

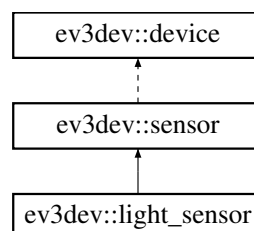
### Additional Inherited Members

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp

## 4.45 ev3dev::light\_sensor Class Reference

Inheritance diagram for ev3dev::light\_sensor:



## Public Member Functions

- **light\_sensor** (address\_type address=INPUT\_AUTO)
- float **reflected\_light\_intensity** ()
- float **ambient\_light\_intensity** ()

## Static Public Attributes

- static const std::string **mode\_reflect** { "REFLECT" }
- static const std::string **mode\_ambient** { "AMBIENT" }

## Additional Inherited Members

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp

## 4.46 ev3::Logger Class Reference

Singleton class responsible for displaying information about events, messages, exceptions and executed methods.

```
#include <Logger.h>
```

## Public Types

- enum [LogLevel](#) {  
[DEBUG](#) = 1, [VERBOSE](#) = 1 << 1, [INFO](#) = 1 << 2, [WARNING](#) = 1 << 3,  
[ERROR](#) = 1 << 4 }  
*Default logging complexity.*
- enum [LogOutput](#) { [STD\\_OUT](#) = 1, [STD\\_ERR](#) = 1 << 1, [FILE](#) = 1 << 2 }  
*Desired logging output.*

## Public Member Functions

- void [log](#) (std::string message, [LogLevel](#) level, [LogOutput](#) output=STD\_OUT)  
*Print message to a specified output.*
- void [setLogLevel](#) ([LogLevel](#) level)  
*Logger level setter.*
- void [setLogLevel](#) (std::string level)  
*Logger level setter by name.*
- void [setLogOutput](#) ([LogOutput](#) output)  
*Logger output setter.*

## Static Public Member Functions

- static [Logger](#) \* [getInstance](#) ()  
*Instance getter.*
- static void [destroy](#) ()  
*Deallocate instance.*

## Private Member Functions

- [Logger](#) ()  
*Default protected constructor (preventing object construction).*
- [Logger](#) (const [Logger](#) &other)  
*Default protected copy constructor (preventing object construction by copying).*
- [Logger](#) & [operator=](#) (const [Logger](#) &other)  
*Protected assignment operator (preventing object assignment).*
- [~Logger](#) ()  
*Default protected destructor (preventing object unwanted destruction).*
- std::string [getLabel](#) ([LogLevel](#) level, [LogOutput](#) output)  
*Get level label.*
- std::string [getColor](#) ([LogLevel](#) level, [LogOutput](#) output)  
*Get color for logging level.*

## Private Attributes

- [LogLevel](#) [\\_level](#) = [ERROR](#)  
*Current [Logger](#) level.*
- [LogOutput](#) [\\_output](#)  
*Current [Logger](#) output.*
- bool [\\_loggerForced](#) = false  
*Control flag.*

## Static Private Attributes

- static [Logger](#) \* [\\_instance](#) = nullptr  
*Instance of [Logger](#) singleton class.*

### 4.46.1 Detailed Description

Singleton class responsible for displaying information about events, messages, exceptions and executed methods.

## 4.46.2 Member Enumeration Documentation

### 4.46.2.1 enum `ev3::Logger::LogLevel`

Default logging complexity.

For a particular level, everything above will be printed as well.

Enumerator

- DEBUG** Print objects and methods information.
- VERBOSE** Print communication and states specifics.
- INFO** Print information for the user.
- WARNING** Print all warnings.
- ERROR** Print all errors.

### 4.46.2.2 enum `ev3::Logger::LogOutput`

Desired logging output.

Enumerator

- STD\_OUT** Standard output.
- STD\_ERR** Standard error output.
- FILE** File output.

## 4.46.3 Constructor & Destructor Documentation

### 4.46.3.1 `ev3::Logger::Logger ( const Logger & other )` `[private]`

Default protected copy constructor (preventing object construction by copying).

Parameters

<i>other</i>	Other <a href="#">Logger</a> object.
--------------	--------------------------------------

## 4.46.4 Member Function Documentation

### 4.46.4.1 `std::string Logger::getColor ( LogLevel level, LogOutput output )` `[private]`

Get color for logging level.

Parameters

<i>level</i>	Logging level to get color for.
<i>output</i>	Logging output to match color on.

**Returns**

String with color code.

**4.46.4.2 `Logger * Logger::getInstance ( )` [static]**

Instance getter.

**Returns**

Create previously or new instance of class [Logger](#).

**4.46.4.3 `std::string Logger::getLabel ( LogLevel level, LogOutput output )` [private]**

Get level label.

**Parameters**

<i>level</i>	Logging level to get label for.
<i>output</i>	Desired message output.

**Returns**

String with formatted label.

**4.46.4.4 `void Logger::log ( std::string message, LogLevel level, LogOutput output = STD_OUT )`**

Print message to a specified output.

**Parameters**

<i>message</i>	String containing message.
<i>level</i>	Logging level (used to choose color for the level label).
<i>output</i>	Type of output to be used,

**4.46.4.5 `Logger& ev3::Logger::operator= ( const Logger & other )` [private]**

Protected assignment operator (preventing object assignment).

**Parameters**

<i>other</i>	Other <a href="#">Logger</a> object.
--------------	--------------------------------------

**Returns**

Copy of passed object.

**4.46.4.6 void Logger::setLogLevel ( LogLevel *level* )**

[Logger](#) level setter.

**Parameters**

<i>level</i>	LogLevel enum value.
--------------	----------------------

**4.46.4.7 void Logger::setLogLevel ( std::string *level* )**

[Logger](#) level setter by name.

**Parameters**

<i>level</i>	String with name of the level.
--------------	--------------------------------

**4.46.4.8 void Logger::setLogOutput ( LogOutput *output* )**

[Logger](#) output setter.

**Parameters**

<i>output</i>	Type of output to be used as default.
---------------	---------------------------------------

**4.46.5 Member Data Documentation****4.46.5.1 bool ev3::Logger::\_loggerForced = false [private]**

Control flag.

Checked when device should not produce any [Logger](#) output.

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/utls/Logger.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/utls/Logger.cpp

**4.47 ev3::Master Class Reference**

Controls the whole system and knows about every [Agent](#).

```
#include <Master.h>
```



## Public Types

- typedef std::map< unsigned int, [Agent](#) > [AgentMap](#)  
*Type for mapping Agents to their ids.*

## Public Member Functions

- std::thread [createThread](#) ([Queue](#)< [Message](#) > \*sendQueue, [Queue](#)< [Message](#) > \*receiveQueue)  
*Creates thread instead of running [Master](#) in the main thread.*
- void [run](#) ([Queue](#)< [Message](#) > \*sendQueue, [Queue](#)< [Message](#) > \*receiveQueue)  
*Starts [Master](#) procedures.*
- void [send](#) ([Message](#) message, bool recordMessage=true)  
*Sending method assigning id to the message.*
- void [stop](#) ()  
*Stop [Master](#) main loop and exit.*

## Private Attributes

- [AgentMap](#) \_agents  
*Map of all active Agents.*
- [Queue](#)< [Message](#) > \* \_sendQueue  
*Out [Message](#) Queue.*
- [Queue](#)< [Message](#) > \* \_receiveQueue  
*In [Message](#) Queue.*
- [SharedPtrBehaviour](#) \_currentBehaviour  
*Currently active [Behaviour](#) for all Agents.*
- unsigned int \_agentId = MASTER\_ID  
*Incremented variable used to assign ids to new Agents.*
- [Measurements](#) \_measurements  
*Types of Sensors which values are interesting and must be gathered.*

### 4.47.1 Detailed Description

Controls the whole system and knows about every [Agent](#).

Initiates [Behaviour](#) and receives values from sensor.

### 4.47.2 Member Function Documentation

#### 4.47.2.1 std::thread Master::createThread ( [Queue](#)< [Message](#) > \* [sendQueue](#), [Queue](#)< [Message](#) > \* [receiveQueue](#) )

Creates thread instead of running [Master](#) in the main thread.

#### Parameters

<a href="#">sendQueue</a>	Out <a href="#">Message</a> queue.
<a href="#">receiveQueue</a>	In <a href="#">Message</a> queue.

**Returns**

New `std::thread` object with active [Master](#) class.

**4.47.2.2** `void Master::run ( Queue< Message > * sendQueue, Queue< Message > * receiveQueue )`

Starts [Master](#) procedures.

**Parameters**

<i>sendQueue</i>	
<i>receiveQueue</i>	

**4.47.2.3** `void Master::send ( Message message, bool recordMessage = true )`

Sending method assigning id to the message.

**Parameters**

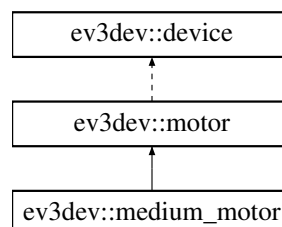
<i>message</i>	<a href="#">Message</a> to be passed to <a href="#">Communication</a> thread via <code>sendQueue</code> .
<i>recordMessage</i>	True if information about message should be saved for further purposes, false otherwise.

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

- `/home/panda/Dokumenty/Repos/Ev3Dev/include/master/Master.h`
- `/home/panda/Dokumenty/Repos/Ev3Dev/src/master/Master.cpp`

## 4.48 `ev3dev::medium_motor` Class Reference

Inheritance diagram for `ev3dev::medium_motor`:

**Public Member Functions**

- **`medium_motor`** (`address_type` `address`=`OUTPUT_AUTO`)

## Additional Inherited Members

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp

## 4.49 ev3::Message Class Reference

Stores information passed between physical system units (another robots or master).

```
#include <Message.h>
```

### Public Types

- enum [MessageType](#) {  
EMPTY, ACK, NOT, AGENT,  
MASTER, MASTER\_OVER, PING, PONG,  
AGENT\_OVER, ABORT, BEHAVIOUR, START,  
RESUME, PAUSE, ACTION\_OK, ACTION\_INTERR,  
SENSOR\_VALUE, MEASURE }

*Message Type.*

### Public Member Functions

- [Message](#) ()  
*Default constructor.*
- [Message](#) (unsigned int senderId, unsigned int receiverId, unsigned int messageId, [MessageType](#) type, StringVector parameters={})  
*Full message constructor.*
- unsigned int [getSenderId](#) ()  
*Sender id getter.*
- unsigned int [getReceiverId](#) ()  
*Receiver id getter.*
- unsigned int [getMessageId](#) ()  
*Consequently incremented integer id getter.*
- [MessageType](#) [getType](#) ()  
*Message type getter.*
- StringVector [getParameters](#) ()  
*Message parameters getter.*
- void [setSenderId](#) (unsigned int id)  
*Sender id setter.*
- void [setReceiverId](#) (unsigned int id)  
*Receiver id setter.*
- void [setMessageId](#) (unsigned int id)  
*Consequently incremented integer id setter.*
- void [setType](#) ([MessageType](#) type)  
*Message type setter.*

- void `setParameters` (StringVector parameters)  
*Message parameters setter.*
- bool `empty` ()  
*Tell whether Message type is EMPTY.*
- std::string `getString` ()  
*Human-readable name getter.*
- void `reset` ()  
*Reset all values to default ones and type to EMPTY.*

### Static Public Member Functions

- static std::string `encodeMessage` (Message &message)  
*Encode message data into string.*
- static Message `decodeMessage` (const std::string message)  
*Decode string into Message object.*

### Private Member Functions

- std::string `getStringType` ()  
*Human-readable Message type name (mainly for logging).*

### Private Attributes

- unsigned int `_id`  
*Message id.*
- unsigned int `_sender`  
*Message sender id.*
- unsigned int `_receiver`  
*Message receiver id.*
- MessageType `_type` = EMPTY  
*Message type.*
- StringVector `_parameters`  
*Vector with all optional parameters.*

#### 4.49.1 Detailed Description

Stores information passed between physical system units (another robots or master).

## 4.49.2 Member Enumeration Documentation

### 4.49.2.1 enum ev3::Message::MessageType

Message Type.

Enumerator

**EMPTY** Empty message, no meaning.

**ACK** Accept previously received request.

**NOT** Deny previously received request.

**AGENT** [Agent](#) side synchronization.

**MASTER** [Master](#) side synchronization.

**MASTER\_OVER** [Master](#) work finished.

**PING** Connection sustain request.

**PONG** Connection sustain answer.

**AGENT\_OVER** [Agent](#) work finished.

**ABORT** Exit processing now.

**BEHAVIOUR** [Behaviour](#) definition received.

**START** [Behaviour](#) start.

**RESUME** [Behaviour](#) resume.

**PAUSE** [Behaviour](#) pause.

**ACTION\_OK** [Action](#) finished correctly.

**ACTION\_INTERR** [Action](#) interrupted.

**SENSOR\_VALUE** [Sensor](#) measurement occurred.

**MEASURE** Instructions what to measure.

## 4.49.3 Constructor & Destructor Documentation

### 4.49.3.1 Message::Message ( unsigned int *senderId*, unsigned int *receiverId*, unsigned int *messageId*, MessageType *type*, StringVector *parameters* = { } )

Full message constructor.

Parameters

<i>senderId</i>	Id of the sender (given by master).
<i>receiverId</i>	Id of the receiver.
<i>messageId</i>	Consequently incremented message id.
<i>type</i>	Predefined <a href="#">Message</a> type.
<i>parameters</i>	Vector of additional, optional string parameters.

## 4.49.4 Member Function Documentation

#### 4.49.4.1 **Message** Message::decodeMessage ( const std::string *message* ) [static]

Decode string into [Message](#) object.

##### Parameters

<i>message</i>	String value to be decoded.
----------------	-----------------------------

##### Returns

[Message](#) object decoded, if processed successfully.

#### 4.49.4.2 bool Message::empty ( )

Tell whether [Message](#) type is EMPTY.

##### Returns

True if Messge is EMPTY, false otherwise.

#### 4.49.4.3 std::string Message::encodeMessage ( Message & *message* ) [static]

Encode message data into string.

##### Parameters

<i>message</i>	Reference to message object to be encoded.
----------------	--

##### Returns

String with encoded data of the message.

#### 4.49.4.4 unsigned int Message::getMessageId ( )

Consequently incremented integer id getter.

##### Returns

Id of the message.

#### 4.49.4.5 StringVector Message::getParameters ( )

[Message](#) parameters getter.

##### Returns

String vector with all optional parameters.

#### 4.49.4.6 `unsigned int Message::getReceiverId ( )`

Receiver id getter.

##### Returns

Id of the message receiver.

#### 4.49.4.7 `unsigned int Message::getSenderId ( )`

Sender id getter.

##### Returns

Id of the message sender (should be set to the value of the main class executing this method).

#### 4.49.4.8 `std::string Message::getString ( )`

Human-readable name getter.

##### Returns

Formatted string containing name and all parameters.

#### 4.49.4.9 `std::string Message::getStringType ( )` `[private]`

Human-readable `Message` type name (mainly for logging).

##### Returns

String with `Message` type name.

#### 4.49.4.10 `Message::MessageType Message::getType ( )`

`Message` type getter.

##### Returns

Enum value with `Message` type.

#### 4.49.4.11 `void Message::setMessageId ( unsigned int id )`

Consequently incremented integer id setter.

## Parameters

<i>id</i>	Id of the message.
-----------	--------------------

4.49.4.12 void Message::setParameters ( StringVector *parameters* )

[Message](#) parameters setter.

## Parameters

<i>parameters</i>	String vector with all optional parameters.
-------------------	---

4.49.4.13 void Message::setReceiverId ( unsigned int *id* )

Receiver id setter.

## Parameters

<i>id</i>	Id of the message receiver.
-----------	-----------------------------

4.49.4.14 void Message::setSenderId ( unsigned int *id* )

Sender id setter.

## Parameters

<i>id</i>	Id of the message sender (should be set to the value of the main class executing this method).
-----------	--

4.49.4.15 void Message::setType ( MessageType *type* )

[Message](#) type setter.

## Parameters

<i>type</i>	Enum value with <a href="#">Message</a> type.
-------------	---

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/communication/[Message.h](#)
- /home/panda/Dokumenty/Repos/Ev3Dev/src/communication/Message.cpp



## 4.50 ev3::Motor Class Reference

Encapsulates [ev3dev::motor](#).

```
#include <Motor.h>
```

### Public Member Functions

- [Motor](#) ([ev3dev::motor](#) motor)  
*Constructor with [Motor](#).*
- [ev3dev::motor](#) getMotor ()  
*[Motor](#) getter.*

### Private Attributes

- [ev3dev::motor](#) \_motor  
*Stored motor.*

#### 4.50.1 Detailed Description

Encapsulates [ev3dev::motor](#).

Can provide additional logic.

#### 4.50.2 Constructor & Destructor Documentation

##### 4.50.2.1 Motor::Motor ( [ev3dev::motor](#) motor )

Constructor with [Motor](#).

Parameters

<i>motor</i>	<a href="#">ev3dev::Motor</a> object.
--------------	---------------------------------------

#### 4.50.3 Member Function Documentation

##### 4.50.3.1 [ev3dev::motor](#) Motor::getMotor ( )

[Motor](#) getter.

Returns

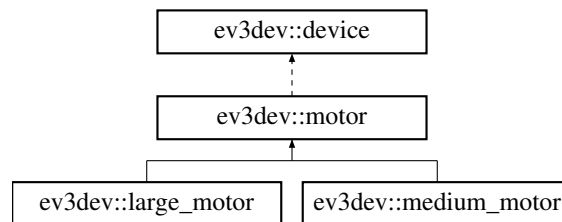
Stored [ev3dev::motor](#) object.

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/robot/Motor.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/robot/Motor.cpp

## 4.51 ev3dev::motor Class Reference

Inheritance diagram for ev3dev::motor:



### Public Types

- typedef device\_type **motor\_type**

### Public Member Functions

- **motor** (address\_type)
- **motor** (address\_type, const motor\_type &)
- auto **set\_command** (std::string v) -> decltype(\*this)
- mode\_set **commands** () const
- int **count\_per\_rot** () const
- std::string **driver\_name** () const
- int **duty\_cycle** () const
- int **duty\_cycle\_sp** () const
- auto **set\_duty\_cycle\_sp** (int v) -> decltype(\*this)
- std::string **encoder\_polarity** () const
- auto **set\_encoder\_polarity** (std::string v) -> decltype(\*this)
- std::string **polarity** () const
- auto **set\_polarity** (std::string v) -> decltype(\*this)
- std::string **address** () const
- int **position** () const
- auto **set\_position** (int v) -> decltype(\*this)
- int **position\_p** () const
- auto **set\_position\_p** (int v) -> decltype(\*this)
- int **position\_i** () const
- auto **set\_position\_i** (int v) -> decltype(\*this)
- int **position\_d** () const
- auto **set\_position\_d** (int v) -> decltype(\*this)
- int **position\_sp** () const
- auto **set\_position\_sp** (int v) -> decltype(\*this)
- int **speed** () const
- int **speed\_sp** () const
- auto **set\_speed\_sp** (int v) -> decltype(\*this)
- int **ramp\_up\_sp** () const
- auto **set\_ramp\_up\_sp** (int v) -> decltype(\*this)
- int **ramp\_down\_sp** () const
- auto **set\_ramp\_down\_sp** (int v) -> decltype(\*this)
- std::string **speed\_regulation\_enabled** () const
- auto **set\_speed\_regulation\_enabled** (std::string v) -> decltype(\*this)

- int **speed\_regulation\_p** () const
- auto **set\_speed\_regulation\_p** (int v) -> decltype(\*this)
- int **speed\_regulation\_i** () const
- auto **set\_speed\_regulation\_i** (int v) -> decltype(\*this)
- int **speed\_regulation\_d** () const
- auto **set\_speed\_regulation\_d** (int v) -> decltype(\*this)
- mode\_set **state** () const
- std::string **stop\_command** () const
- auto **set\_stop\_command** (std::string v) -> decltype(\*this)
- mode\_set **stop\_commands** () const
- int **time\_sp** () const
- auto **set\_time\_sp** (int v) -> decltype(\*this)
- void **run\_forever** ()
- void **run\_to\_abs\_pos** ()
- void **run\_to\_rel\_pos** ()
- void **run\_timed** ()
- void **run\_direct** ()
- void **stop** ()
- void **reset** ()
- motor\_type **type\_name** ()

### Static Public Attributes

- static const motor\_type **motor\_large** { "lego-ev3-l-motor" }
- static const motor\_type **motor\_medium** { "lego-ev3-m-motor" }
- static const std::string **command\_run\_forever** { "run-forever" }
- static const std::string **command\_run\_to\_abs\_pos** { "run-to-abs-pos" }
- static const std::string **command\_run\_to\_rel\_pos** { "run-to-rel-pos" }
- static const std::string **command\_run\_timed** { "run-timed" }
- static const std::string **command\_run\_direct** { "run-direct" }
- static const std::string **command\_stop** { "stop" }
- static const std::string **command\_reset** { "reset" }
- static const std::string **encoder\_polarity\_normal** { "normal" }
- static const std::string **encoder\_polarity\_inversed** { "inversed" }
- static const std::string **polarity\_normal** { "normal" }
- static const std::string **polarity\_inversed** { "inversed" }
- static const std::string **speed\_regulation\_on** { "on" }
- static const std::string **speed\_regulation\_off** { "off" }
- static const std::string **stop\_command\_coast** { "coast" }
- static const std::string **stop\_command\_brake** { "brake" }
- static const std::string **stop\_command\_hold** { "hold" }

### Protected Member Functions

- bool **connect** (const std::map< std::string, std::set< std::string >> &) noexcept

### Private Attributes

- motor\_type **\_type**

## Additional Inherited Members

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

- `/home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h`
- `/home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp`

## 4.52 `ev3::CommUtils::NetworkNode` Struct Reference

Stores information about a particular node in the network.

```
#include <CommUtils.h>
```

### Public Attributes

- unsigned int `port`  
*Port number.*
- `std::string` `ipAddress`  
*Node's ipv4 address.*

### 4.52.1 Detailed Description

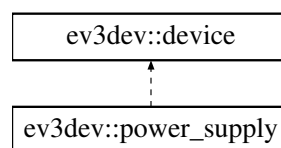
Stores information about a particular node in the network.

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

- `/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/CommUtils.h`

## 4.53 `ev3dev::power_supply` Class Reference

Inheritance diagram for `ev3dev::power_supply`:



### Public Member Functions

- **`power_supply`** (`std::string` name)
- int **`measured_current`** () const
- int **`measured_voltage`** () const
- int **`max_voltage`** () const
- int **`min_voltage`** () const
- `std::string` **`technology`** () const
- `std::string` **`type`** () const
- float **`measured_amps`** () const
- float **`measured_volts`** () const

### Static Public Attributes

- static `power_supply` `battery` { "" }

### Additional Inherited Members

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp

## 4.54 ev3::Queue< T > Class Template Reference

Template class implementing synchronized queue.

```
#include <Queue.h>
```

### Public Member Functions

- void `push` (T element)  
*Insert new object to the queue.*
- T `pop` ()  
*Removes first object from the queue.*
- bool `empty` ()  
*Check whether queue is empty.*

### Private Attributes

- std::queue< T > `_elements`  
*The actual queue implemented as std::queue.*
- std::mutex `_mutex`  
*Synchronization mutex.*

#### 4.54.1 Detailed Description

```
template<class T>  
class ev3::Queue< T >
```

Template class implementing synchronized queue.

All method are guarded by mutex.

## 4.54.2 Member Function Documentation

### 4.54.2.1 `template<class T> bool ev3::Queue< T>::empty ( )`

Check whether queue is empty.

#### Returns

True if queue is empty, false otherwise.

### 4.54.2.2 `template<class T> T ev3::Queue< T>::pop ( )`

Removes first object from the queue.

#### Returns

Copy of removed object.

### 4.54.2.3 `template<class T> void ev3::Queue< T>::push ( T element )`

Insert new object to the queue.

#### Parameters

<i>element</i>	Inserted object.
----------------	------------------

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

- `/home/panda/Dokumenty/Repos/Ev3Dev/include/utils/Queue.h`

## 4.55 `ev3dev::remote_control` Class Reference

### Public Types

- enum **buttons** {  
**red\_up** = (1 << 0), **red\_down** = (1 << 1), **blue\_up** = (1 << 2), **blue\_down** = (1 << 3),  
**beacon** = (1 << 4) }

### Public Member Functions

- **remote\_control** (unsigned channel=1)
- **remote\_control** ([infrared\\_sensor](#) &, unsigned channel=1)
- bool **connected** () const
- unsigned **channel** () const
- bool **process** ()

## Public Attributes

- `std::function< void(bool)>` **on\_red\_up**
- `std::function< void(bool)>` **on\_red\_down**
- `std::function< void(bool)>` **on\_blue\_up**
- `std::function< void(bool)>` **on\_blue\_down**
- `std::function< void(bool)>` **on\_beacon**
- `std::function< void(int)>` **on\_state\_change**

## Protected Member Functions

- virtual void **on\_value\_changed** (int value)

## Protected Attributes

- `infrared_sensor * _sensor` = nullptr
- `bool _owns_sensor` = false
- `unsigned _channel` = 0
- `int _value` = 0
- `int _state` = 0

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

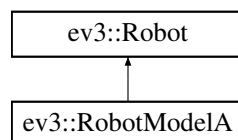
- `/home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h`
- `/home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp`

## 4.56 ev3::Robot Class Reference

Main class representing actual robot.

```
#include <Robot.h>
```

Inheritance diagram for ev3::Robot:



## Public Types

- `typedef std::vector< Action::ActionType > AvailableActions`  
*Type for specifying all available actions for given [Robot](#) model.*

## Public Member Functions

- [Robot](#) ()  
*Default constructor.*
- [Robot](#) ([Devices::RequiredDevices](#) devices, [AvailableActions](#) actions)  
*Constructor with required devices and actions parameters.*
- virtual [~Robot](#) ()  
*Default destructor.*
- std::thread [createThread](#) ([Queue< Message > \\*sendQueue](#), [Queue< Message > \\*receiveQueue](#))  
*Thread creation method (instead of running [Robot](#) in main thread).*
- virtual void [run](#) ([Queue< Message > \\*sendQueue](#), [Queue< Message > \\*receiveQueue](#))  
*Starts [Robot](#) procedures.*
- void [stop](#) ()  
*Immediately stop [Robot](#) object and all assigned motors.*
- void [send](#) ([Message](#) message)  
*General sending method for logging and assigning id.*
- virtual std::string [getString](#) ()  
*Human-readable [Robot](#) name getter.*

## Protected Member Functions

- virtual [SharedPtrBehaviour](#) [generateBehaviour](#) ([Behaviour::BehaviourType](#) type, [StringVector](#) parameters)  
*Generate behaviour based on its type and parameters.*

## Protected Attributes

- unsigned int [\\_id](#) = 0  
*This [Robot](#)'s id assigned by [Master](#).*
- unsigned int [\\_commId](#) = 0  
*[Communication](#) id (assigned to messages).*
- float [\\_pulsePerUnitRatio](#) = 1.f  
*Number of rotation pulses per one distance unit.*
- [Devices::RequiredDevices](#) [\\_requiredDevices](#)  
*Vector of mapped ports and devices that are required.*
- [AvailableActions](#) [\\_availableActions](#)  
*Vector of executable [Action](#) types.*
- [Queue< Message > \\* \\_sendQueue](#)  
*Out [Message](#) queue.*
- [Queue< Message > \\* \\_receiveQueue](#)  
*In [Message](#) queue.*
- [LedControl](#) [\\_ledControl](#)  
*Object controlling behaviour of LED diodes.*
- [RobotState](#) \* [\\_state](#) = new [RobotStateIdle](#)(&[\\_ledControl](#))  
*Current [Robot](#) state.*



## Private Member Functions

- void `processState` ()  
*Process current [Robot](#)'s state (which processes [Behaviour](#)).*
- void `processEvents` ()  
*Process all [Event](#) objects from [EventQueue](#).*
- void `processMessage` ()  
*Interprets and process received Messages.*
- void `ping` ()  
*Sends PING [Message](#) to master.*

## Private Attributes

- bool `_behaviourSet` = false  
*Control flag.*
- [Message](#) `_currentMessage`  
*Last received [Message](#).*
- `HighResClock::time_point` `_masterPingTime` = `HighResClock::now()`  
*Time since last PONG [Message](#) from [Master](#).*
- unsigned int `_score`  
*Score of the [Robot](#).*

### 4.56.1 Detailed Description

Main class representing actual robot.

Base class for all different [Robot](#) models. Aggregates [RobotState](#), messages and [Behaviour](#) processing as well as information exchange with [Communication](#) thread.

### 4.56.2 Constructor & Destructor Documentation

#### 4.56.2.1 `Robot::Robot ( Devices::RequiredDevices devices, AvailableActions actions )`

Constructor with required devices and actions parameters.

##### Parameters

<i>devices</i>	Vector of mapped ports and devices types.
<i>actions</i>	Vector with <a href="#">Action</a> types executable by a particular robot.

### 4.56.3 Member Function Documentation

#### 4.56.3.1 `std::thread Robot::createThread ( Queue< Message > * sendQueue, Queue< Message > * receiveQueue )`

Thread creation method (instead of running [Robot](#) in main thread).

## Parameters

<i>sendQueue</i>	Out <a href="#">Message</a> queue.
<i>receiveQueue</i>	In <a href="#">Message</a> queue.

## Returns

New std::thread object with [Robot](#) class active.

**4.56.3.2** `SharedPtrBehaviour Robot::generateBehaviour ( Behaviour::BehaviourType type, StringVector parameters )` `[protected]`, `[virtual]`

Generate behaviour based on its type and parameters.

## Parameters

<i>type</i>	<a href="#">Behaviour</a> type.
<i>parameters</i>	Additional parameters required by a particular <a href="#">Behaviour</a> .

## Returns

New shared pointer with generated [Behaviour](#) object.

Reimplemented in [ev3::RobotModelA](#).

**4.56.3.3** `std::string Robot::getString ( )` `[virtual]`

Human-readable [Robot](#) name getter.

## Returns

String with [Robot](#) name.

Reimplemented in [ev3::RobotModelA](#).

**4.56.3.4** `void Robot::run ( Queue< Message > * sendQueue, Queue< Message > * receiveQueue )` `[virtual]`

Starts [Robot](#) procedures.

## Parameters

<i>sendQueue</i>	Out <a href="#">Message</a> queue.
<i>receiveQueue</i>	In <a href="#">Message</a> queue.

## 4.56.3.5 void Robot::send ( Message message )

General sending method for logging and assigning id.

## Parameters

message	Message to be sent to Communication thread.
---------	---

## 4.56.4 Member Data Documentation

## 4.56.4.1 bool ev3::Robot::\_behaviourSet = false [private]

Control flag.

True if Robot has any Behaviour assigned, false otherwise.

## 4.56.4.2 float ev3::Robot::\_pulsePerUnitRatio = 1.f [protected]

Number of rotation pulses per one distance unit.

Calculated based on attached wheel circumference.

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

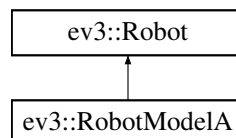
- /home/panda/Dokumenty/Repos/Ev3Dev/include/robot/Robot.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/robot/Robot.cpp

## 4.57 ev3::RobotModelA Class Reference

Describes particular Robot construction and its way of implementing actions and running behaviours.

```
#include <RobotModelA.h>
```

Inheritance diagram for ev3::RobotModelA:



## Public Member Functions

- RobotModelA ()  
*Default constructor.*
- virtual std::string getString () override  
*Human-readable name getter.*

## Private Member Functions

- virtual [SharedPtrBehaviour](#) [generateBehaviour](#) ([Behaviour::BehaviourType](#) type, [StringVector](#) parameters) override  
*Overrides [Robot](#) method of [Behaviour](#) creation.*
- [SharedPtrAction](#) [generateAction](#) ([SharedPtrAction](#) action, [Action::ActionType](#) type)  
*Generate [Action](#) based on its type.*

## Private Attributes

- float [\\_wheelRadius](#) = 5.75 / 2.f  
*This model's wheel radius.*

## Additional Inherited Members

### 4.57.1 Detailed Description

Describes particular [Robot](#) construction and its way of implementing actions and running behaviours.

### 4.57.2 Member Function Documentation

#### 4.57.2.1 [SharedPtrAction](#) [RobotModelA::generateAction](#) ( [SharedPtrAction](#) action, [Action::ActionType](#) type ) [private]

Generate [Action](#) based on its type.

#### Parameters

<i>action</i>	Shared pointer object with <a href="#">Action</a> to be constructed.
<i>type</i>	<a href="#">Action</a> type.

#### Returns

Copy of the [Action](#) object with new data.

#### 4.57.2.2 [std::string](#) [RobotModelA::getString](#) ( ) [override],[virtual]

Human-readable name getter.

#### Returns

String with [Robot](#) model name.

Reimplemented from [ev3::Robot](#).

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

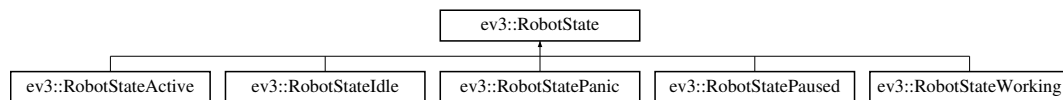
- /home/panda/Dokumenty/Repos/Ev3Dev/include/robot/RobotModelA.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/robot/RobotModelA.cpp

## 4.58 ev3::RobotState Class Reference

Base class for all [Robot](#) states.

```
#include <RobotState.h>
```

Inheritance diagram for ev3::RobotState:



### Public Types

- enum [States](#) {  
[IDLE](#), [ACTIVE](#), [WORKING](#), [PAUSED](#),  
[PANIC](#) }  
*State names (types).*
- typedef std::map< [MessageType](#), [States](#) > [ChangeMap](#)  
*Type for defining transitions when particular Messages occur.*

### Public Member Functions

- [RobotState](#) ([ChangeMap](#) changes, [LedControl](#) \*led)  
*Constructor with transitions map and LED control pointer.*
- virtual [RobotState](#) \* [process](#) ([Message](#) msg)  
*Processes currently assigned state.*
- [MessageType](#) [getPendingMessage](#) ()  
*Get Message to be sent to Master.*
- void [updateTimer](#) ()  
*Updates timeouts and pings.*
- bool [isPendingEnabled](#) ()  
*Get information whether state is waiting for response.*
- void [setBehaviour](#) ([SharedPtrBehaviour](#) behaviour)  
*Set new Behaviour for this state.*
- [SharedPtrBehaviour](#) [getBehaviour](#) ()  
*Behaviour getter.*

### Static Public Attributes

- static const float [MASTER\\_TIMEOUT](#) = 10.f \* 1000  
*Default time to enter PANIC state.*
- static const float [MASTER\\_PING\\_TIME](#) = 3.f \* 1000  
*Time interval for PING-PONG Message exchange.*

## Protected Member Functions

- `RobotState * switchState (Message::MessageType type)`  
*Normal state changing method.*
- `RobotState * changeState (States state)`  
*Force state changing method.*

## Protected Attributes

- `SharedPtrBehaviour _currentBehaviour`  
*Currently processed [Behaviour](#).*
- `States _state`  
*Current state type.*
- `ChangeMap _changes`  
*Map of state transitions.*
- `LedControl * _led`  
*LED diodes controlling pointer.*
- `Message::MessageType _pendingMessage = Message::EMPTY`  
*Type of [Message](#) that's going to be forwarded.*
- `float _pendingTimeout = 0.f`  
*Time to wait for response.*
- `HighResClock::time_point _masterTimeout = HighResClock::now()`  
*Time for measuring master PING response.*
- `HighResClock::time_point _messageDelay = HighResClock::now()`  
*Time for measuring master response for a particular [Message](#).*

### 4.58.1 Detailed Description

Base class for all [Robot](#) states.

Contains of transitions, timing methods and [Behaviour](#) processing.

### 4.58.2 Member Enumeration Documentation

#### 4.58.2.1 enum `ev3::RobotState::States`

State names (types).

Enumerator

- IDLE** Powered, but not connected.
- ACTIVE** Conected, but no task assigned.
- WORKING** Processing [Behaviour](#).
- PAUSED** [Behaviour](#) processing paused.
- PANIC** Lost connection or no connection at all.

### 4.58.3 Constructor & Destructor Documentation

#### 4.58.3.1 `RobotState::RobotState ( ChangeMap changes, LedControl * led )`

Constructor with transitions map and LED control pointer.

## Parameters

<i>changes</i>	List of available transitions.
<i>led</i>	Pointer to <a href="#">LedControl</a> object for diodes control.

#### 4.58.4 Member Function Documentation

##### 4.58.4.1 `RobotState * RobotState::changeState ( States state )` [protected]

Force state changing method.

## Parameters

<i>state</i>	New state to be assigned.
--------------	---------------------------

## Returns

Pointer to created state.

##### 4.58.4.2 `SharedPtrBehaviour RobotState::getBehaviour ( )`

[Behaviour](#) getter.

## Returns

Shared pointer with stored [Behaviour](#) object.

##### 4.58.4.3 `Message::MessageType RobotState::getPendingMessage ( )`

Get [Message](#) to be sent to [Master](#).

## Returns

Type of [Message](#) that has to be forwarded.

##### 4.58.4.4 `bool RobotState::isPendingEnabled ( )`

Get information whether state is waiting for response.

## Returns

True if new Messages can be sent, false otherwise.

##### 4.58.4.5 `RobotState * RobotState::process ( Message msg )` [virtual]

Processes currently assigned state.

## Parameters

<i>msg</i>	<a href="#">Message</a> to be interpreted withing current state.
------------	--

## Returns

Pointer to new state or 'this'.

Reimplemented in [ev3::RobotStatePanic](#), [ev3::RobotStatePaused](#), [ev3::RobotStateWorking](#), [ev3::RobotStateActive](#), and [ev3::RobotStateIdle](#).

#### 4.58.4.6 void RobotState::setBehaviour ( [SharedPtrBehaviour](#) *behaviour* )

Set new [Behaviour](#) for this state.

## Parameters

<i>behaviour</i>	<a href="#">Behaviour</a> shared pointer object.
------------------	--

#### 4.58.4.7 [RobotState](#) \* [RobotState](#)::switchState ( [Message::MessageType](#) *type* ) [protected]

Normal state changing method.

## Parameters

<i>type</i>	Message type indicating new state to be assigned.
-------------	---

## Returns

Pointer to created state.

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

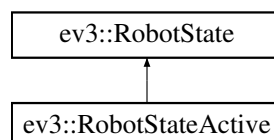
- /home/panda/Dokumenty/Repos/Ev3Dev/include/robot/RobotState.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/robot/RobotState.cpp

## 4.59 [ev3::RobotStateActive](#) Class Reference

State in which [Robot](#) is connected but has no assigned [Behaviour](#).

```
#include <RobotState.h>
```

Inheritance diagram for [ev3::RobotStateActive](#):





## Public Member Functions

- [RobotStateActive](#) ([LedControl](#) \*led)  
*Constructor with LED controller.*
- [RobotState](#) \* [process](#) ([Message](#) msg)  
*Overriden process method.*

## Additional Inherited Members

### 4.59.1 Detailed Description

State in which [Robot](#) is connected but has no assigned [Behaviour](#).

### 4.59.2 Constructor & Destructor Documentation

#### 4.59.2.1 RobotStateActive::RobotStateActive ( [LedControl](#) \* *led* )

Constructor with LED controller.

#### Parameters

<i>led</i>	<a href="#">LedControl</a> pointer.
------------	-------------------------------------

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

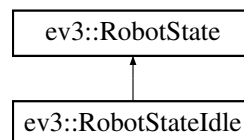
- /home/panda/Dokumenty/Repos/Ev3Dev/include/robot/RobotState.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/robot/RobotState.cpp

## 4.60 ev3::RobotStateIdle Class Reference

State in which [Robot](#) is powered but not connected to [Master](#).

```
#include <RobotState.h>
```

Inheritance diagram for ev3::RobotStateIdle:



## Public Member Functions

- [RobotStateIdle](#) ([LedControl](#) \*led)  
*Constructor with LED controller.*
- [RobotState](#) \* [process](#) ([Message](#) msg)  
*Overriden process method.*

## Additional Inherited Members

### 4.60.1 Detailed Description

State in which [Robot](#) is powered but not connected to [Master](#).

### 4.60.2 Constructor & Destructor Documentation

#### 4.60.2.1 RobotStateIdle::RobotStateIdle ( LedControl \* led )

Constructor with LED controller.

##### Parameters

<i>led</i>	<a href="#">LedControl</a> pointer.
------------	-------------------------------------

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

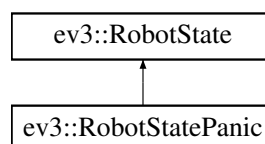
- /home/panda/Dokumenty/Repos/Ev3Dev/include/robot/RobotState.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/robot/RobotState.cpp

## 4.61 ev3::RobotStatePanic Class Reference

State in which [Robot](#) lost connection with [Master](#) or had no connection at all.

```
#include <RobotState.h>
```

Inheritance diagram for ev3::RobotStatePanic:



## Public Member Functions

- [RobotStatePanic](#) ([LedControl](#) \*led)  
*Constructor with LED controller.*
- [RobotState](#) \* [process](#) ([Message](#) msg)  
*Overriden process method.*

## Additional Inherited Members

### 4.61.1 Detailed Description

State in which [Robot](#) lost connection with [Master](#) or had no connection at all.

## 4.61.2 Constructor & Destructor Documentation

### 4.61.2.1 RobotStatePanic::RobotStatePanic ( LedControl \* led )

Constructor with LED controller.

Parameters

<i>led</i>	LedControl pointer.
------------	---------------------

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

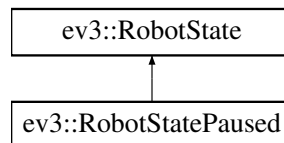
- /home/panda/Dokumenty/Repos/Ev3Dev/include/robot/RobotState.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/robot/RobotState.cpp

## 4.62 ev3::RobotStatePaused Class Reference

State in which Robot's Behaviour processing is paused.

```
#include <RobotState.h>
```

Inheritance diagram for ev3::RobotStatePaused:



### Public Member Functions

- RobotStatePaused (LedControl \*led)  
*Constructor with LED controller.*
- RobotState \* process (Message msg)  
*Overriden process method.*

### Additional Inherited Members

#### 4.62.1 Detailed Description

State in which Robot's Behaviour processing is paused.

#### 4.62.2 Constructor & Destructor Documentation

##### 4.62.2.1 RobotStatePaused::RobotStatePaused ( LedControl \* led )

Constructor with LED controller.

## Parameters

<i>led</i>	<a href="#">LedControl</a> pointer.
------------	-------------------------------------

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

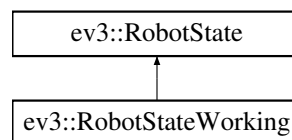
- /home/panda/Dokumenty/Repos/Ev3Dev/include/robot/RobotState.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/robot/RobotState.cpp

## 4.63 ev3::RobotStateWorking Class Reference

State in which [Robot](#) is processing assigned [Behaviour](#).

```
#include <RobotState.h>
```

Inheritance diagram for ev3::RobotStateWorking:



### Public Member Functions

- [RobotStateWorking](#) ([LedControl](#) \*led)  
*Constructor with LED controller.*
- [RobotState](#) \* [process](#) ([Message](#) msg)  
*Overriden process method.*

### Additional Inherited Members

#### 4.63.1 Detailed Description

State in which [Robot](#) is processing assigned [Behaviour](#).

#### 4.63.2 Constructor & Destructor Documentation

##### 4.63.2.1 RobotStateWorking::RobotStateWorking ( [LedControl](#) \* *led* )

Constructor with LED controller.

## Parameters

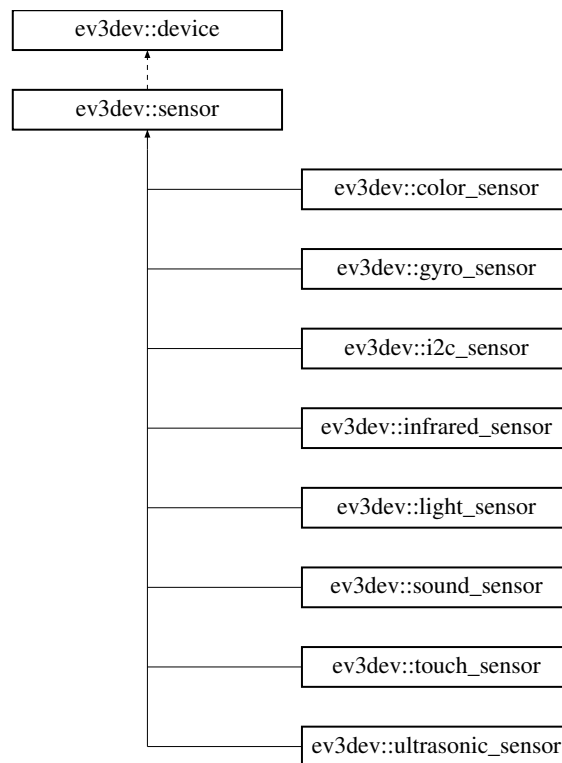
<i>led</i>	<a href="#">LedControl</a> pointer.
------------	-------------------------------------

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/robot/RobotState.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/robot/RobotState.cpp

## 4.64 ev3dev::sensor Class Reference

Inheritance diagram for ev3dev::sensor:



### Public Types

- typedef device\_type **sensor\_type**

### Public Member Functions

- **sensor** (address\_type)
- **sensor** (address\_type, const std::set< sensor\_type > &)
- int **value** (unsigned index=0) const
- float **float\_value** (unsigned index=0) const
- std::string **type\_name** () const
- std::string **bin\_data\_format** () const
- const std::vector< char > & **bin\_data** () const
- template<class T >  
void **bin\_data** (T \*buf) const
- auto **set\_command** (std::string v) -> decltype(\*this)
- mode\_set **commands** () const

- int **decimals** () const
- std::string **driver\_name** () const
- std::string **mode** () const
- auto **set\_mode** (std::string v) -> decltype(\*this)
- mode\_set **modes** () const
- int **num\_values** () const
- std::string **address** () const
- std::string **units** () const

### Static Public Attributes

- static const sensor\_type **ev3\_touch** { "lego-ev3-touch" }
- static const sensor\_type **ev3\_color** { "lego-ev3-color" }
- static const sensor\_type **ev3\_ultrasonic** { "lego-ev3-us" }
- static const sensor\_type **ev3\_gyro** { "lego-ev3-gyro" }
- static const sensor\_type **ev3\_infrared** { "lego-ev3-ir" }
- static const sensor\_type **nxt\_touch** { "lego-nxt-touch" }
- static const sensor\_type **nxt\_light** { "lego-nxt-light" }
- static const sensor\_type **nxt\_sound** { "lego-nxt-sound" }
- static const sensor\_type **nxt\_ultrasonic** { "lego-nxt-us" }
- static const sensor\_type **nxt\_i2c\_sensor** { "nxt-i2c-sensor" }
- static const sensor\_type **nxt\_analog** { "nxt-analog" }

### Protected Member Functions

- bool **connect** (const std::map< std::string, std::set< std::string >> &) noexcept

### Protected Attributes

- std::vector< char > **\_bin\_data**

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp

## 4.65 ev3::Sensor Class Reference

Encapsulates [ev3dev::sensor](#).

```
#include <Sensor.h>
```

### Public Types

- enum [SensorType](#) {  
[TOUCH](#), [COLOR](#), [ULTRASONIC](#), [GYRO](#),  
[INFRARED](#), [SOUND](#), [LIGHT](#) }  
*Sensor type.*

## Public Member Functions

- [Sensor](#) ([ev3dev::sensor](#) sensor, [SensorType](#) type)  
*Constructor with sensor object and type.*
- [ev3dev::sensor](#) [getSensor](#) ()  
*Sensor getter.*
- int [getValue](#) (unsigned int n)  
*Value getter.*
- float [getValueF](#) (unsigned int n)  
*Float value getter.*
- int [getDecimals](#) ()  
*Number of decimal places getter.*
- unsigned int [getNumValues](#) ()  
*Number of different values getter.*
- [SensorType](#) [getType](#) ()  
*Sensor type getter.*

## Static Public Member Functions

- static [StringVector](#) [prepareMessage](#) ([SensorValue](#) value, [SensorType](#) type)  
*Prepare vector of parameters for [Message](#) object.*

## Private Attributes

- [SensorType](#) [\\_type](#)  
*This [Sensor](#) type.*
- [ev3dev::sensor](#) [\\_sensor](#)  
*Stored motor.*

### 4.65.1 Detailed Description

Encapsulates [ev3dev::sensor](#).

Can provide additional logic.

### 4.65.2 Member Enumeration Documentation

#### 4.65.2.1 enum [ev3::Sensor::SensorType](#)

[Sensor](#) type.

Enumerator

- TOUCH** Touch sensor.
- COLOR** Color sensor.
- ULTRASONIC** Ultrasonic sensor.
- GYRO** Gyroscope sensor.
- INFRARED** Infrared sensor.
- SOUND** Sound sensor.
- LIGHT** Light sensor.

### 4.65.3 Constructor & Destructor Documentation

#### 4.65.3.1 `Sensor::Sensor ( ev3dev::sensor sensor, SensorType type )`

Constructor with sensor object and type.



## Parameters

<i>sensor</i>	ev3dev sensor object.
<i>type</i>	<a href="#">Sensor</a> type value.

#### 4.65.4 Member Function Documentation

##### 4.65.4.1 `int Sensor::getDecimals ( )`

Number of decimal places getter.

## Returns

Integer with number of places that a true [Sensor](#) value has.

##### 4.65.4.2 `unsigned int Sensor::getNumValues ( )`

Number of different values getter.

## Returns

Integer with number of different measurements available.

##### 4.65.4.3 `ev3dev::sensor Sensor::getSensor ( )`

[Sensor](#) getter.

## Returns

Stored `ev3dev::sensor` object.

##### 4.65.4.4 `Sensor::SensorType Sensor::getType ( )`

[Sensor](#) type getter.

## Returns

SensorType value.

##### 4.65.4.5 `int Sensor::getValue ( unsigned int n )`

Value getter.

## Parameters

<i>n</i>	Id of the value desired.
----------	--------------------------

## Returns

Integer with [Sensor](#)'s value.

4.65.4.6 float `Sensor::getValueF ( unsigned int n )`

Float value getter.

## Parameters

<i>n</i>	Id of the value desired.
----------	--------------------------

## Returns

Float with [Sensor](#)'s value.

4.65.4.7 `StringVector Sensor::prepareMessage ( SensorValue value, SensorType type )` `[static]`

Prepare vector of parameters for [Message](#) object.

## Parameters

<i>value</i>	Measured values.
<i>type</i>	Used <a href="#">Sensor</a> type.

## Returns

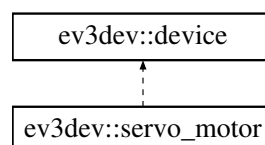
Vector with [Sensor](#) values as strings.

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

- `/home/panda/Dokumenty/Repos/Ev3Dev/include/robot/Sensor.h`
- `/home/panda/Dokumenty/Repos/Ev3Dev/src/robot/Sensor.cpp`

4.66 `ev3dev::servo_motor` Class Reference

Inheritance diagram for `ev3dev::servo_motor`:



## Public Member Functions

- **servo\_motor** (address\_type address=OUTPUT\_AUTO)
- auto **set\_command** (std::string v) -> decltype(\*this)
- std::string **driver\_name** () const
- int **max\_pulse\_sp** () const
- auto **set\_max\_pulse\_sp** (int v) -> decltype(\*this)
- int **mid\_pulse\_sp** () const
- auto **set\_mid\_pulse\_sp** (int v) -> decltype(\*this)
- int **min\_pulse\_sp** () const
- auto **set\_min\_pulse\_sp** (int v) -> decltype(\*this)
- std::string **polarity** () const
- auto **set\_polarity** (std::string v) -> decltype(\*this)
- std::string **address** () const
- int **position\_sp** () const
- auto **set\_position\_sp** (int v) -> decltype(\*this)
- int **rate\_sp** () const
- auto **set\_rate\_sp** (int v) -> decltype(\*this)
- mode\_set **state** () const
- void **run** ()
- void **float\_** ()

## Static Public Attributes

- static const std::string **command\_run** { "run" }
- static const std::string **command\_float** { "float" }
- static const std::string **polarity\_normal** { "normal" }
- static const std::string **polarity\_inversed** { "inversed" }

## Additional Inherited Members

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp

## 4.67 ev3::SignalHandler Class Reference

Simple class catching system signals.

```
#include <SignalHandler.h>
```

## Static Public Member Functions

- static void [HandleSignal](#) (int signum)  
*Main signal catching method.*

## Static Public Attributes

- static [Robot](#) \* [robot](#) = nullptr  
*Pointer to [Robot](#) instance.*
- static [Master](#) \* [master](#) = nullptr  
*Pointer to [Master](#) instance.*

### 4.67.1 Detailed Description

Simple class catching system signals.

Stops [Robot](#) and [Master](#) if break signal is received.

### 4.67.2 Member Function Documentation

#### 4.67.2.1 void SignalHandler::HandleSignal ( int *signum* ) [static]

Main signal catching method.

##### Parameters

<i>signum</i>	Signal code to catch.
---------------	-----------------------

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/Utils/SignalHandler.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/Utils/SignalHandler.cpp

## 4.68 ev3dev::sound Class Reference

### Static Public Member Functions

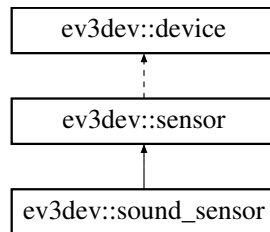
- static void **beep** (const std::string &args="", bool bSynchronous=false)
- static void **tone** (float frequency, float ms, bool bSynchronous=false)
- static void **tone** (const std::vector< std::vector< float > > &sequence, bool bSynchronous=false)
- static void **play** (const std::string &soundfile, bool bSynchronous=false)
- static void **speak** (const std::string &text, bool bSynchronous=false)

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp

## 4.69 ev3dev::sound\_sensor Class Reference

Inheritance diagram for ev3dev::sound\_sensor:



### Public Member Functions

- **sound\_sensor** (address\_type address=INPUT\_AUTO)
- float **sound\_pressure** ()
- float **sound\_pressure\_low** ()

### Static Public Attributes

- static const std::string **mode\_db** { "DB" }
- static const std::string **mode\_dba** { "DBA" }

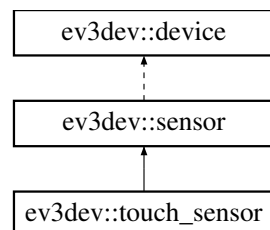
### Additional Inherited Members

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp

## 4.70 ev3dev::touch\_sensor Class Reference

Inheritance diagram for ev3dev::touch\_sensor:



### Public Member Functions

- **touch\_sensor** (address\_type address=INPUT\_AUTO)
- bool **is\_pressed** ()

## Static Public Attributes

- static const std::string **mode\_touch** { "TOUCH" }

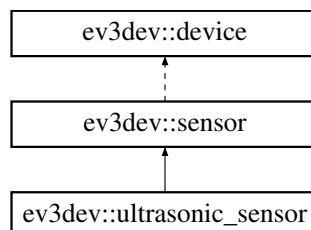
## Additional Inherited Members

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp

## 4.71 ev3dev::ultrasonic\_sensor Class Reference

Inheritance diagram for ev3dev::ultrasonic\_sensor:



## Public Member Functions

- **ultrasonic\_sensor** (address\_type address=INPUT\_AUTO)
- float **distance\_centimeters** ()
- float **distance\_inches** ()
- bool **other\_sensor\_present** ()

## Static Public Attributes

- static const std::string **mode\_us\_dist\_cm** { "US-DIST-CM" }
- static const std::string **mode\_us\_dist\_in** { "US-DIST-IN" }
- static const std::string **mode\_us\_listen** { "US-LISTEN" }
- static const std::string **mode\_us\_si\_cm** { "US-SI-CM" }
- static const std::string **mode\_us\_si\_in** { "US-SI-IN" }

## Additional Inherited Members

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

- /home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp

## Chapter 5

# File Documentation

### 5.1 /home/panda/Dokumenty/Repos/Ev3Dev/include/action/Action.h File Reference

Contains all Action classes.

```
#include "CommandMotor.h"  
#include <memory>
```

#### Classes

- class [ev3::Action](#)  
*Base class for all [Action](#) controlling classes.*
- class [ev3::ActionRepeat](#)  
*Stores many [Actions](#) in a vector and executes them in loop.*
- class [ev3::ActionDriveDistance](#)  
*Implements [Robot](#) simple task to drive straight for a given distance.*
- class [ev3::ActionRotate](#)  
*Implements [Robot](#) simple task to rotate a given angle, while not driving.*
- class [ev3::ActionRotateRandDirection](#)  
*Implements [Robot](#) simple task to rotate a random angle.*
- class [ev3::ActionStop](#)  
*Implements [Robot](#) simple task to stop all active motors.*
- class [ev3::ActionDriveForever](#)  
*Implements [Robot](#) simple task to drive straight forever.*

#### Typedefs

- typedef std::shared\_ptr< Action > [ev3::SharedPtrAction](#)  
*Type for [Action](#) [shared\\_ptr](#).*
- typedef std::vector< SharedPtrAction > [ev3::StoredActions](#)  
*Type for storing many [Actions](#) in one container.*
- typedef std::shared\_ptr< Command > [ev3::SharedPtrCommand](#)  
*Type for [Command](#) [shared\\_ptr](#).*
- typedef std::vector< SharedPtrCommand > [ev3::CommandsVector](#)  
*Type for containing associated [Command](#) pointers.*

### 5.1.1 Detailed Description

Contains all Action classes.

### 5.1.2 Typedef Documentation

#### 5.1.2.1 `typedef std::vector<SharedPtrAction> ev3::StoredActions`

Type for storing many Actions in one container.

See also

[ActionRepeat](#)

## 5.2 /home/panda/Dokumenty/Repos/Ev3Dev/include/action/Behaviour.h File Reference

Contains all Behaviour classes.

```
#include "Action.h"
#include "Utils.h"
#include "Sensor.h"
#include "Event.h"
#include "BehaviourState.h"
#include <unistd.h>
#include <string>
```

### Classes

- class [ev3::Behaviour](#)  
*Base class for all defined behaviours.*
- class [ev3::BehaviourDriveOnSquare](#)  
*Implements complex behaviour of driving on a square-shaped route.*
- class [ev3::BehaviourExploreRandom](#)  
*Implements complex behaviour of exploring the surrounding with random rotation.*

### Typedefs

- typedef `std::shared_ptr< Behaviour >` [ev3::SharedPtrBehaviour](#)  
*Type for [Behaviour](#) `shared_ptr`.*
- typedef `std::vector< BehaviourState >` [ev3::BehaviourStates](#)  
*Type for storing [Behaviour](#) states in one container.*
- typedef `std::vector< Sensor::SensorType >` [ev3::Measurements](#)  
*Type for storing sensors' desired measurements in one container.*



### 5.2.1 Detailed Description

Contains all Behaviour classes.

## 5.3 /home/panda/Dokumenty/Repos/Ev3Dev/include/action/BehaviourState.h File Reference

Contains BehaviourState class.

```
#include "Action.h"  
#include "Event.h"
```

### Classes

- class [ev3::BehaviourState](#)  
*Encapsulates action and other information in a form of a state.*

### Typedefs

- typedef std::map< Event::EventType, unsigned int > [ev3::ReactionsTransitions](#)  
*Type for storing Event-State pairs defining special transitions.*

### 5.3.1 Detailed Description

Contains BehaviourState class.

## 5.4 /home/panda/Dokumenty/Repos/Ev3Dev/include/communication/Communication.h File Reference

Contains Communication class.

```
#include "Queue.h"  
#include "CommUtils.h"  
#include <thread>
```

### Classes

- class [ev3::Communication](#)  
*Encapsulates low-level communication and adds logic concerning sending and receiving [Message](#) queueing.*

## Variables

- static const unsigned int `ev3::MAX_COMM_ITERATIONS` = 10  
*Default maximum number of one time communication thread iterations.*
- static const unsigned int `ev3::SEND_RETRIES` = 3  
*Default number of subsequent attempts to send a message.*

### 5.4.1 Detailed Description

Contains Communication class.

## 5.5 /home/panda/Dokumenty/Repos/Ev3Dev/include/communication/CommUtils.h File Reference

Contains CommUtils class.

```
#include "Message.h"
#include "CircularBuffer.h"
#include <string>
#include <netinet/in.h>
#include <map>
#include <queue>
```

## Classes

- class `ev3::CommUtils`  
*Responsible for low-level communication.*
- struct `ev3::CommUtils::NetworkNode`  
*Stores information about a particular node in the network.*
- struct `ev3::CommUtils::Buffer`  
*Contains buffer and its size.*

## Variables

- static const unsigned int `ev3::DEFAULT_PORT` = 12345  
*Default port number.*
- static const unsigned int `ev3::MAX_PACKET_LENGTH` = 4096  
*Maximum packet size in bytes.*
- static const unsigned int `ev3::DEFAULT_RECEIVE_DELAY` = 1  
*Default time in milliseconds to wait for message (used by non-blocking receive method).*
- static const unsigned int `ev3::MASTER_ID` = 1  
*Default master id.*
- static const unsigned int `ev3::SENT_MESSAGE_COPIES` = 5  
*Default number of copies to be sent every time (preventing packet loss).*
- static const unsigned int `ev3::DEFAULT_PACKET_BUFFER_LIMIT` = 50  
*Maximum number of stored message prototypes (preventing duplicates).*

### 5.5.1 Detailed Description

Contains CommUtils class.

## 5.6 /home/panda/Dokumenty/Repos/Ev3Dev/include/communication/Message.h File Reference

Contains Message class.

```
#include "Utils.h"
#include <vector>
#include <string>
```

### Classes

- class [ev3::Message](#)  
*Stores information passed between physical system units (another robots or master).*

### Variables

- static const char [ev3::MESSAGE\\_DELIM](#) = ':'  
*Default [Message](#) delimiter between parts of encoded message string.*

### 5.6.1 Detailed Description

Contains Message class.

## 5.7 /home/panda/Dokumenty/Repos/Ev3Dev/include/robot/Devices.h File Reference

Contains Devices classes.

```
#include "ev3dev.h"
#include "Motor.h"
#include "Sensor.h"
#include "Utils.h"
```

### Classes

- class [ev3::Devices](#)  
*Singleton class responsible for managing devices connected to the robot.*

## Variables

- `const std::vector< ev3dev::port_type > ev3::INPUTS = {ev3dev::INPUT_1, ev3dev::INPUT_2, ev3dev::INPUT_3, ev3dev::INPUT_4}`  
*Type for storing all available [Sensor](#) inputs.*
- `const std::vector< ev3dev::port_type > ev3::OUTPUTS = {ev3dev::OUTPUT_A, ev3dev::OUTPUT_B, ev3dev::OUTPUT_C, ev3dev::OUTPUT_D}`  
*Type for storing all available [Motor](#) outpus.*

### 5.7.1 Detailed Description

Contains Devices classes.

## 5.8 /home/panda/Dokumenty/Repos/Ev3Dev/include/utils/EventQueue.h File Reference

Contains EventQueue class.

```
#include "Event.h"
#include <queue>
#include <mutex>
```

## Classes

- class [ev3::EventQueue](#)  
*Singleton class responsible for managing [Event](#) objects.*

## Typedefs

- `typedef std::shared_ptr< Event > ev3::SharedPtrEvent`  
*Type for [Event](#) `shared_ptr`.*

### 5.8.1 Detailed Description

Contains EventQueue class.

# Index

/home/panda/Dokumenty/Repos/Ev3Dev/include/action/↔ Action  
Action.h, 125  
ev3::Action, 11

/home/panda/Dokumenty/Repos/Ev3Dev/include/action/↔ Action.h  
Behaviour.h, 126  
StoredActions, 126

/home/panda/Dokumenty/Repos/Ev3Dev/include/action/↔ ActionDriveDistance  
BehaviourState.h, 127  
ev3::ActionDriveDistance, 14

/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/↔ ActionDriveForever  
CommUtils.h, 128  
ev3::ActionDriveForever, 16

/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/↔ ActionRepeat  
Communication.h, 127  
ev3::ActionRepeat, 18

/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/↔ ActionRotate  
Message.h, 129  
ev3::ActionRotate, 20

/home/panda/Dokumenty/Repos/Ev3Dev/include/robot/↔ ActionRotateRandDirection  
Devices.h, 129  
ev3::ActionRotateRandDirection, 21, 22

/home/panda/Dokumenty/Repos/Ev3Dev/include/utils/↔ ActionStop  
EventQueue.h, 130  
ev3::ActionStop, 23

\_behaviourSet  
ev3::Robot, 105

\_endCondition  
ev3::Action, 13

\_loggerForced  
ev3::Logger, 86

\_measurements  
ev3::Agent, 26

\_packetBuffer  
ev3::CommUtils, 58

\_pulsePerUnitRatio  
ev3::Robot, 105

ABORT  
ev3::Message, 91

ACTION\_FINISHED  
ev3::Event, 66

ACTION\_INTERR  
ev3::Event, 66  
ev3::Message, 91

ACTION\_OK  
ev3::Message, 91

ACTIVE  
ev3::RobotState, 108

ACK  
ev3::Message, 91

AGENT\_OVER  
ev3::Message, 91

AGENT  
ev3::Message, 91

ALL  
ev3::LedControl, 79

AMBER  
ev3::LedControl, 79

ActionType  
ev3::Action, 11

addListener  
ev3::Devices, 63

BEHAVIOUR\_START  
ev3::Event, 66

BEHAVIOUR\_STOP  
ev3::Event, 66

BEHAVIOUR  
ev3::Message, 91

BRAKE  
ev3::CommandMotorSetStopMode, 50

Behaviour  
ev3::Behaviour, 29

BehaviourDriveOnSquare  
ev3::BehaviourDriveOnSquare, 31, 32

BehaviourExploreRandom  
ev3::BehaviourExploreRandom, 33

BehaviourState  
ev3::BehaviourState, 35

BehaviourType  
ev3::Behaviour, 28

COAST  
ev3::CommandMotorSetStopMode, 50

COLOR  
ev3::Sensor, 117

CUSTOM  
ev3::Behaviour, 28

changeState  
ev3::RobotState, 109

checkDevices  
ev3::Devices, 63

- CircularBuffer
  - ev3::CircularBuffer, 39
- CommandMotor
  - ev3::CommandMotor, 44
- CommandMotorReset
  - ev3::CommandMotorReset, 45
- CommandMotorRunForever
  - ev3::CommandMotorRunForever, 46
- CommandMotorSetSpeed
  - ev3::CommandMotorSetSpeed, 47
- CommandMotorSetSpeedRegEnabled
  - ev3::CommandMotorSetSpeedRegEnabled, 49
- CommandMotorSetStopMode
  - ev3::CommandMotorSetStopMode, 50
- CommandMotorStop
  - ev3::CommandMotorStop, 51
- CommandSensor
  - ev3::CommandSensor, 52
- contain
  - ev3::CircularBuffer, 39
- createThread
  - ev3::Communication, 54
  - ev3::Master, 87
  - ev3::Robot, 103
- DEBUG
  - ev3::Logger, 84
- DRIVE\_DISTANCE
  - ev3::Action, 11
- DRIVE\_FOREVER
  - ev3::Action, 11
- DRIVE\_ON\_SQUARE
  - ev3::Behaviour, 28
- decodeMessage
  - ev3::Message, 91
- Devices
  - ev3::Devices, 62
- EMPTY
  - ev3::Event, 66
  - ev3::Message, 91
- ERROR
  - ev3::Logger, 84
- EXPLORE\_RANDOM
  - ev3::Behaviour, 28
- empty
  - ev3::EventQueue, 69
  - ev3::Message, 92
  - ev3::Queue, 100
- encodeMessage
  - ev3::Message, 92
- ev3::Action, 9
  - \_endCondition, 13
  - Action, 11
  - ActionType, 11
  - DRIVE\_DISTANCE, 11
  - DRIVE\_FOREVER, 11
  - getActionPrototype, 12
  - getString, 12
  - getType, 12
  - isExecuted, 12
  - isFinished, 12
  - NOP, 11
  - REPEAT, 11
  - ROTATE\_RANDOM\_DIR, 11
  - ROTATE, 11
  - STOP, 11
  - setCommands, 12
  - setEndCondition, 13
- ev3::ActionDriveDistance, 13
  - ActionDriveDistance, 14
  - getActionPrototype, 15
  - getDistance, 15
  - getString, 15
- ev3::ActionDriveForever, 15
  - ActionDriveForever, 16
  - getActionPrototype, 17
  - getString, 17
  - isForward, 17
- ev3::ActionRepeat, 17
  - ActionRepeat, 18
  - getString, 18
- ev3::ActionRotate, 19
  - ActionRotate, 20
  - getActionPrototype, 20
  - getRotation, 20
  - getString, 20
- ev3::ActionRotateRandDirection, 21
  - ActionRotateRandDirection, 21, 22
  - execute, 22
  - getActionPrototype, 22
  - getString, 22
- ev3::ActionStop, 23
  - ActionStop, 23
  - getActionPrototype, 23
  - getString, 24
- ev3::Agent, 24
  - \_measurements, 26
  - getCommId, 25
  - getId, 25
  - processMessage, 25
  - setBehaviour, 26
  - setCommId, 26
  - setId, 26
  - setMeasurement, 26
  - updateLastMessage, 26
- ev3::Behaviour, 27
  - Behaviour, 29
  - BehaviourType, 28
  - CUSTOM, 28
  - DRIVE\_ON\_SQUARE, 28
  - EXPLORE\_RANDOM, 28
  - getPrototype, 29
  - getString, 29
  - react, 29
  - setMeasurements, 30
  - setReactionStates, 30

- setStates, [30](#)
  - setStopState, [30](#)
- ev3::BehaviourDriveOnSquare, [31](#)
  - BehaviourDriveOnSquare, [31](#), [32](#)
  - getPrototype, [32](#)
  - getString, [32](#)
- ev3::BehaviourExploreRandom, [33](#)
  - BehaviourExploreRandom, [33](#)
  - getPrototype, [33](#)
  - getString, [34](#)
- ev3::BehaviourState, [34](#)
  - BehaviourState, [35](#)
  - getAction, [36](#)
  - getReaction, [36](#)
  - isStopState, [36](#)
  - process, [36](#)
  - setNextState, [36](#)
  - setReactions, [37](#)
- ev3::CircularBuffer
  - CircularBuffer, [39](#)
  - contain, [39](#)
  - push, [40](#)
- ev3::CircularBuffer< T >, [38](#)
- ev3::ColorUtils, [41](#)
- ev3::CommUtils, [54](#)
  - \_packetBuffer, [58](#)
  - getBufferFromString, [56](#)
  - getStringFromBuffer, [56](#)
  - makeSockAddr, [56](#)
  - preparePassiveSocket, [56](#)
  - receiveMessage, [57](#)
  - receiveMessageDelay, [57](#)
  - sendBroadcastMessage, [57](#)
  - sendMessage, [58](#)
  - sendMessageTo, [58](#)
- ev3::CommUtils::Buffer, [37](#)
- ev3::CommUtils::NetworkNode, [98](#)
- ev3::Command, [43](#)
  - getString, [43](#)
- ev3::CommandMotor, [44](#)
  - CommandMotor, [44](#)
  - getMotor, [45](#)
- ev3::CommandMotorReset, [45](#)
  - CommandMotorReset, [45](#)
- ev3::CommandMotorRunForever, [46](#)
  - CommandMotorRunForever, [46](#)
- ev3::CommandMotorSetSpeed, [47](#)
  - CommandMotorSetSpeed, [47](#)
- ev3::CommandMotorSetSpeedRegEnabled, [48](#)
  - CommandMotorSetSpeedRegEnabled, [49](#)
- ev3::CommandMotorSetStopMode, [49](#)
  - BRAKE, [50](#)
  - COAST, [50](#)
  - CommandMotorSetStopMode, [50](#)
  - HOLD, [50](#)
  - StopMode, [50](#)
- ev3::CommandMotorStop, [51](#)
  - CommandMotorStop, [51](#)
- ev3::CommandSensor, [52](#)
  - CommandSensor, [52](#)
  - getSensor, [52](#)
- ev3::Communication, [53](#)
  - createThread, [54](#)
  - run, [54](#)
- ev3::Devices, [61](#)
  - addListener, [63](#)
  - checkDevices, [63](#)
  - Devices, [62](#)
  - getInstance, [63](#)
  - getMotor, [63](#)
  - getSensor, [63](#)
  - operator=, [64](#)
  - removeListener, [64](#)
  - setProximitySensor, [64](#)
  - setSafetyTouchSensor, [64](#)
- ev3::Event, [65](#)
  - ACTION\_FINISHED, [66](#)
  - ACTION\_INTERRUPT, [66](#)
  - BEHAVIOUR\_START, [66](#)
  - BEHAVIOUR\_STOP, [66](#)
  - EMPTY, [66](#)
  - Event, [66](#)
  - EventType, [66](#)
  - getStringType, [66](#)
  - getType, [66](#)
  - OBSTACLE\_DETECTED, [66](#)
  - PROXIMITY\_ALERT, [66](#)
  - SENSOR\_WATCH, [66](#)
- ev3::EventAction, [67](#)
  - EventAction, [67](#)
  - getActionType, [67](#)
- ev3::EventQueue, [68](#)
  - empty, [69](#)
  - EventQueue, [69](#)
  - getInstance, [69](#)
  - operator=, [69](#)
  - pop, [71](#)
  - push, [71](#)
  - size, [71](#)
- ev3::EventSensorWatch, [71](#)
  - EventSensorWatch, [72](#)
  - getType, [72](#)
  - getValue, [72](#)
- ev3::LedControl, [78](#)
  - ALL, [79](#)
  - AMBER, [79](#)
  - flash, [79](#)
  - flashColor, [80](#)
  - GREEN\_ALL, [79](#)
  - GREEN\_L, [79](#)
  - GREEN\_R, [79](#)
  - GREEN, [79](#)
  - LedColors, [79](#)
  - LedType, [79](#)
  - off, [80](#)
  - on, [80](#)

- onExclusive, 80
- RED\_ALL, 79
- RED\_L, 79
- RED\_R, 79
- RED, 79
- setColor, 80
- YELLOW, 79
- ev3::Logger, 82
  - \_loggerForced, 86
  - DEBUG, 84
  - ERROR, 84
  - FILE, 84
  - getColor, 84
  - getInstance, 85
  - getLabel, 85
  - INFO, 84
  - log, 85
  - LogLevel, 84
  - LogOutput, 84
  - Logger, 84
  - operator=, 85
  - STD\_ERR, 84
  - STD\_OUT, 84
  - setLogLevel, 86
  - setLogOutput, 86
  - VERBOSE, 84
  - WARNING, 84
- ev3::Master, 86
  - createThread, 87
  - run, 88
  - send, 88
- ev3::Message, 89
  - ABORT, 91
  - ACTION\_INTERR, 91
  - ACTION\_OK, 91
  - ACK, 91
  - AGENT\_OVER, 91
  - AGENT, 91
  - BEHAVIOUR, 91
  - decodeMessage, 91
  - EMPTY, 91
  - empty, 92
  - encodeMessage, 92
  - getMessageId, 92
  - getParameters, 92
  - getReceiverId, 92
  - getSenderId, 93
  - getString, 93
  - getStringType, 93
  - getType, 93
  - MASTER\_OVER, 91
  - MASTER, 91
  - MEASURE, 91
  - Message, 91
  - MessageType, 91
  - NOT, 91
  - PAUSE, 91
  - PING, 91
  - PONG, 91
  - RESUME, 91
  - SENSOR\_VALUE, 91
  - START, 91
  - setMessageId, 93
  - setParameters, 94
  - setReceiverId, 94
  - setSenderId, 94
  - setType, 94
- ev3::Motor, 95
  - getMotor, 95
  - Motor, 95
- ev3::Queue
  - empty, 100
  - pop, 100
  - push, 100
- ev3::Queue< T >, 99
- ev3::Robot, 101
  - \_behaviourSet, 105
  - \_pulsePerUnitRatio, 105
  - createThread, 103
  - generateBehaviour, 104
  - getString, 104
  - Robot, 103
  - run, 104
  - send, 104
- ev3::RobotModelA, 105
  - generateAction, 106
  - getString, 106
- ev3::RobotState, 107
  - ACTIVE, 108
  - changeState, 109
  - getBehaviour, 109
  - getPendingMessage, 109
  - IDLE, 108
  - isPendingEnabled, 109
  - PANIC, 108
  - PAUSED, 108
  - process, 109
  - RobotState, 108
  - setBehaviour, 110
  - States, 108
  - switchState, 110
  - WORKING, 108
- ev3::RobotStateActive, 110
  - RobotStateActive, 111
- ev3::RobotStateIdle, 111
  - RobotStateIdle, 112
- ev3::RobotStatePanic, 112
  - RobotStatePanic, 113
- ev3::RobotStatePaused, 113
  - RobotStatePaused, 113
- ev3::RobotStateWorking, 114
  - RobotStateWorking, 114
- ev3::Sensor, 116
  - COLOR, 117
  - GYRO, 117
  - getDecimals, 119



- getNumValues, [119](#)
- getSensor, [119](#)
- getType, [119](#)
- getValue, [119](#)
- getValueF, [120](#)
- INFRARED, [117](#)
- LIGHT, [117](#)
- prepareMessage, [120](#)
- SOUND, [117](#)
- Sensor, [118](#)
- SensorType, [117](#)
- TOUCH, [117](#)
- ULTRASONIC, [117](#)
- ev3::SignalHandler, [121](#)
  - HandleSignal, [122](#)
- ev3dev::button, [37](#)
- ev3dev::button::file\_descriptor, [73](#)
- ev3dev::color\_sensor, [40](#)
- ev3dev::dc\_motor, [59](#)
- ev3dev::device, [60](#)
- ev3dev::gyro\_sensor, [73](#)
- ev3dev::i2c\_sensor, [74](#)
- ev3dev::infrared\_sensor, [75](#)
- ev3dev::large\_motor, [75](#)
- ev3dev::lcd, [76](#)
- ev3dev::led, [76](#)
- ev3dev::lego\_port, [81](#)
- ev3dev::light\_sensor, [81](#)
- ev3dev::medium\_motor, [88](#)
- ev3dev::motor, [96](#)
- ev3dev::power\_supply, [98](#)
- ev3dev::remote\_control, [100](#)
- ev3dev::sensor, [115](#)
- ev3dev::servo\_motor, [120](#)
- ev3dev::sound, [122](#)
- ev3dev::sound\_sensor, [123](#)
- ev3dev::touch\_sensor, [123](#)
- ev3dev::ultrasonic\_sensor, [124](#)
- Event
  - ev3::Event, [66](#)
- EventAction
  - ev3::EventAction, [67](#)
- EventQueue
  - ev3::EventQueue, [69](#)
- EventSensorWatch
  - ev3::EventSensorWatch, [72](#)
- EventType
  - ev3::Event, [66](#)
- execute
  - ev3::ActionRotateRandDirection, [22](#)
- FILE
  - ev3::Logger, [84](#)
- flash
  - ev3::LedControl, [79](#)
- flashColor
  - ev3::LedControl, [80](#)
- GREEN\_ALL
  - ev3::LedControl, [79](#)
- GREEN\_L
  - ev3::LedControl, [79](#)
- GREEN\_R
  - ev3::LedControl, [79](#)
- GREEN
  - ev3::LedControl, [79](#)
- GYRO
  - ev3::Sensor, [117](#)
- generateAction
  - ev3::RobotModelA, [106](#)
- generateBehaviour
  - ev3::Robot, [104](#)
- getAction
  - ev3::BehaviourState, [36](#)
- getActionPrototype
  - ev3::Action, [12](#)
  - ev3::ActionDriveDistance, [15](#)
  - ev3::ActionDriveForever, [17](#)
  - ev3::ActionRotate, [20](#)
  - ev3::ActionRotateRandDirection, [22](#)
  - ev3::ActionStop, [23](#)
- getActionType
  - ev3::EventAction, [67](#)
- getBehaviour
  - ev3::RobotState, [109](#)
- getBufferFromString
  - ev3::CommUtils, [56](#)
- getColor
  - ev3::Logger, [84](#)
- getCommId
  - ev3::Agent, [25](#)
- getDecimals
  - ev3::Sensor, [119](#)
- getDistance
  - ev3::ActionDriveDistance, [15](#)
- getId
  - ev3::Agent, [25](#)
- getInstance
  - ev3::Devices, [63](#)
  - ev3::EventQueue, [69](#)
  - ev3::Logger, [85](#)
- getLabel
  - ev3::Logger, [85](#)
- getMessageId
  - ev3::Message, [92](#)
- getMotor
  - ev3::CommandMotor, [45](#)
  - ev3::Devices, [63](#)
  - ev3::Motor, [95](#)
- getNumValues
  - ev3::Sensor, [119](#)
- getParameters
  - ev3::Message, [92](#)
- getPendingMessage
  - ev3::RobotState, [109](#)
- getPrototype
  - ev3::Behaviour, [29](#)

- ev3::BehaviourDriveOnSquare, 32
  - ev3::BehaviourExploreRandom, 33
- getReaction
  - ev3::BehaviourState, 36
- getReceiverId
  - ev3::Message, 92
- getRotation
  - ev3::ActionRotate, 20
- getSenderId
  - ev3::Message, 93
- getSensor
  - ev3::CommandSensor, 52
  - ev3::Devices, 63
  - ev3::Sensor, 119
- getString
  - ev3::Action, 12
  - ev3::ActionDriveDistance, 15
  - ev3::ActionDriveForever, 17
  - ev3::ActionRepeat, 18
  - ev3::ActionRotate, 20
  - ev3::ActionRotateRandDirection, 22
  - ev3::ActionStop, 24
  - ev3::Behaviour, 29
  - ev3::BehaviourDriveOnSquare, 32
  - ev3::BehaviourExploreRandom, 34
  - ev3::Command, 43
  - ev3::Message, 93
  - ev3::Robot, 104
  - ev3::RobotModelA, 106
- getStringFromBuffer
  - ev3::CommUtils, 56
- getStringType
  - ev3::Event, 66
  - ev3::Message, 93
- getType
  - ev3::Action, 12
  - ev3::Event, 66
  - ev3::EventSensorWatch, 72
  - ev3::Message, 93
  - ev3::Sensor, 119
- getValue
  - ev3::EventSensorWatch, 72
  - ev3::Sensor, 119
- getValueF
  - ev3::Sensor, 120
- HOLD
  - ev3::CommandMotorSetStopMode, 50
- HandleSignal
  - ev3::SignalHandler, 122
- IDLE
  - ev3::RobotState, 108
- INFRARED
  - ev3::Sensor, 117
- INFO
  - ev3::Logger, 84
- isExecuted
  - ev3::Action, 12
- isFinished
  - ev3::Action, 12
- isForward
  - ev3::ActionDriveForever, 17
- isPendingEnabled
  - ev3::RobotState, 109
- isStopState
  - ev3::BehaviourState, 36
- LIGHT
  - ev3::Sensor, 117
- LedColors
  - ev3::LedControl, 79
- LedType
  - ev3::LedControl, 79
- log
  - ev3::Logger, 85
- LogLevel
  - ev3::Logger, 84
- LogOutput
  - ev3::Logger, 84
- Logger
  - ev3::Logger, 84
- MASTER\_OVER
  - ev3::Message, 91
- MASTER
  - ev3::Message, 91
- MEASURE
  - ev3::Message, 91
- makeSockAddr
  - ev3::CommUtils, 56
- Message
  - ev3::Message, 91
- MessageType
  - ev3::Message, 91
- Motor
  - ev3::Motor, 95
- NOP
  - ev3::Action, 11
- NOT
  - ev3::Message, 91
- OBSTACLE\_DETECTED
  - ev3::Event, 66
- off
  - ev3::LedControl, 80
- on
  - ev3::LedControl, 80
- onExclusive
  - ev3::LedControl, 80
- operator=
  - ev3::Devices, 64
  - ev3::EventQueue, 69
  - ev3::Logger, 85
- PANIC
  - ev3::RobotState, 108

- PAUSED
  - ev3::RobotState, 108
- PAUSE
  - ev3::Message, 91
- PING
  - ev3::Message, 91
- PONG
  - ev3::Message, 91
- PROXIMITY\_ALERT
  - ev3::Event, 66
- pop
  - ev3::EventQueue, 71
  - ev3::Queue, 100
- prepareMessage
  - ev3::Sensor, 120
- preparePassiveSocket
  - ev3::CommUtils, 56
- process
  - ev3::BehaviourState, 36
  - ev3::RobotState, 109
- processMessage
  - ev3::Agent, 25
- push
  - ev3::CircularBuffer, 40
  - ev3::EventQueue, 71
  - ev3::Queue, 100
- RED\_ALL
  - ev3::LedControl, 79
- RED\_L
  - ev3::LedControl, 79
- RED\_R
  - ev3::LedControl, 79
- REPEAT
  - ev3::Action, 11
- RESUME
  - ev3::Message, 91
- RED
  - ev3::LedControl, 79
- ROTATE\_RANDOM\_DIR
  - ev3::Action, 11
- ROTATE
  - ev3::Action, 11
- react
  - ev3::Behaviour, 29
- receiveMessage
  - ev3::CommUtils, 57
- receiveMessageDelay
  - ev3::CommUtils, 57
- removeListener
  - ev3::Devices, 64
- Robot
  - ev3::Robot, 103
- RobotState
  - ev3::RobotState, 108
- RobotStateActive
  - ev3::RobotStateActive, 111
- RobotStateIdle
  - ev3::RobotStateIdle, 112
- RobotStatePanic
  - ev3::RobotStatePanic, 113
- RobotStatePaused
  - ev3::RobotStatePaused, 113
- RobotStateWorking
  - ev3::RobotStateWorking, 114
- run
  - ev3::Communication, 54
  - ev3::Master, 88
  - ev3::Robot, 104
- SENSOR\_VALUE
  - ev3::Message, 91
- SENSOR\_WATCH
  - ev3::Event, 66
- SOUND
  - ev3::Sensor, 117
- START
  - ev3::Message, 91
- STD\_ERR
  - ev3::Logger, 84
- STD\_OUT
  - ev3::Logger, 84
- STOP
  - ev3::Action, 11
- send
  - ev3::Master, 88
  - ev3::Robot, 104
- sendBroadcastMessage
  - ev3::CommUtils, 57
- sendMessage
  - ev3::CommUtils, 58
- sendMessageTo
  - ev3::CommUtils, 58
- Sensor
  - ev3::Sensor, 118
- SensorType
  - ev3::Sensor, 117
- setBehaviour
  - ev3::Agent, 26
  - ev3::RobotState, 110
- setColor
  - ev3::LedControl, 80
- setCommId
  - ev3::Agent, 26
- setCommands
  - ev3::Action, 12
- setEndCondition
  - ev3::Action, 13
- setId
  - ev3::Agent, 26
- setLogLevel
  - ev3::Logger, 86
- setLogOutput
  - ev3::Logger, 86
- setMeasurement
  - ev3::Agent, 26
- setMeasurements
  - ev3::Behaviour, 30

- setMessageId
  - ev3::Message, [93](#)
- setNextState
  - ev3::BehaviourState, [36](#)
- setParameters
  - ev3::Message, [94](#)
- setProximitySensor
  - ev3::Devices, [64](#)
- setReactionStates
  - ev3::Behaviour, [30](#)
- setReactions
  - ev3::BehaviourState, [37](#)
- setReceiverId
  - ev3::Message, [94](#)
- setSafetyTouchSensor
  - ev3::Devices, [64](#)
- setSenderId
  - ev3::Message, [94](#)
- setStates
  - ev3::Behaviour, [30](#)
- setStopState
  - ev3::Behaviour, [30](#)
- setType
  - ev3::Message, [94](#)
- size
  - ev3::EventQueue, [71](#)
- States
  - ev3::RobotState, [108](#)
- StopMode
  - ev3::CommandMotorSetStopMode, [50](#)
- StoredActions
  - Action.h, [126](#)
- switchState
  - ev3::RobotState, [110](#)
- TOUCH
  - ev3::Sensor, [117](#)
- ULTRASONIC
  - ev3::Sensor, [117](#)
- updateLastMessage
  - ev3::Agent, [26](#)
- VERBOSE
  - ev3::Logger, [84](#)
- WARNING
  - ev3::Logger, [84](#)
- WORKING
  - ev3::RobotState, [108](#)
- YELLOW
  - ev3::LedControl, [79](#)