

Ev3Dev

0.1.1

Generated by Doxygen 1.8.11

Contents

1	Hierarchical Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	File Index	7
3.1	File List	7
4	Class Documentation	9
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	setEndCondition(EndCondition condition)	13
4.1.5	Member Data Documentation	13
4.1.5.1	_endCondition	13
4.2	ev3::ActionDriveDistance Class Reference	13
4.2.1	Detailed Description	14
4.2.2	Constructor & Destructor Documentation	14
4.2.2.1	ActionDriveDistance(int distance)	14
4.2.2.2	ActionDriveDistance(CommandsVector commands, int distance)	14
4.2.3	Member Function Documentation	15
4.2.3.1	getActionPrototype()	15
4.2.3.2	getDistance()	15
4.2.3.3	getString() override	15
4.3	ev3::ActionDriveForever Class Reference	15
4.3.1	Detailed Description	16
4.3.2	Constructor & Destructor Documentation	16
4.3.2.1	ActionDriveForever(bool forward=true)	16
4.3.2.2	ActionDriveForever(CommandsVector commands, bool forward=true)	16
4.3.3	Member Function Documentation	17
4.3.3.1	getActionPrototype()	17
4.3.3.2	getString() override	17
4.3.3.3	isForward()	17
4.4	ev3::ActionRepeat Class Reference	17
4.4.1	Detailed Description	18
4.4.2	Constructor & Destructor Documentation	18
4.4.2.1	ActionRepeat(StoredActions actions, unsigned int n)	18
4.4.3	Member Function Documentation	18
4.4.3.1	getString()	18
4.5	ev3::ActionRotate Class Reference	19
4.5.1	Detailed Description	20
4.5.2	Constructor & Destructor Documentation	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	setId(const unsigned int id)	26
4.8.2.7	setMeasurement(Measurements measurements)	26
4.8.2.8	updateLastMessage(Message *message)	26
4.8.3	Member Data Documentation	26
4.8.3.1	_measurements	27
4.9	ev3::Behaviour Class Reference	27
4.9.1	Detailed Description	28
4.9.2	Member Enumeration Documentation	28
4.9.2.1	BehaviourType	28
4.9.3	Constructor & Destructor Documentation	29
4.9.3.1	Behaviour(BehaviourType type, BehaviourStates states)	29
4.9.3.2	Behaviour(BehaviourType type)	29
4.9.4	Member Function Documentation	29
4.9.4.1	getPrototype()	29
4.9.4.2	getString()	29
4.9.4.3	react(Event::EventType type)	29
4.9.4.4	setMeasurements(Measurements measurements)	30
4.9.4.5	setReactionStates(BehaviourStates reactionStates)	30
4.9.4.6	setStates(BehaviourStates states)	30
4.9.4.7	setStopState(BehaviourState state)	30
4.10	ev3::BehaviourDriveOnSquare Class Reference	31
4.10.1	Detailed Description	31
4.10.2	Constructor & Destructor Documentation	31
4.10.2.1	BehaviourDriveOnSquare(unsigned int side, bool turningRight)	31
4.10.2.2	BehaviourDriveOnSquare(BehaviourStates states, unsigned int side, bool turningRight)	32
4.10.3	Member Function Documentation	32
4.10.3.1	getPrototype()	32
4.10.3.2	getString()	32
4.11	ev3::BehaviourExploreRandom Class Reference	33
4.11.1	Detailed Description	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 &)=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
4.16	ev3dev::color_sensor Class Reference	39
4.17	ev3::ColorUtils Class Reference	40
4.18	ev3::Command Class Reference	40
4.18.1	Detailed Description	41
4.18.2	Member Function Documentation	41
4.18.2.1	getString()	41
4.19	ev3::CommandMotor Class Reference	41

4.19.1 Detailed Description	42
4.19.2 Constructor & Destructor Documentation	42
4.19.2.1 CommandMotor(Motor &motor)	42
4.19.3 Member Function Documentation	42
4.19.3.1 getMotor()	42
4.20 ev3::CommandMotorReset Class Reference	43
4.20.1 Detailed Description	43
4.20.2 Constructor & Destructor Documentation	43
4.20.2.1 CommandMotorReset(Motor &motor)	43
4.21 ev3::CommandMotorRunForever Class Reference	44
4.21.1 Detailed Description	44
4.21.2 Constructor & Destructor Documentation	44
4.21.2.1 CommandMotorRunForever(Motor &motor)	44
4.22 ev3::CommandMotorSetSpeed Class Reference	45
4.22.1 Detailed Description	45
4.22.2 Constructor & Destructor Documentation	45
4.22.2.1 CommandMotorSetSpeed(Motor &motor, int value)	45
4.23 ev3::CommandMotorSetSpeedRegEnabled Class Reference	46
4.23.1 Detailed Description	46
4.23.2 Constructor & Destructor Documentation	46
4.23.2.1 CommandMotorSetSpeedRegEnabled(Motor &motor, bool value)	46
4.24 ev3::CommandMotorSetStopMode Class Reference	47
4.24.1 Detailed Description	48
4.24.2 Member Enumeration Documentation	48
4.24.2.1 StopMode	48
4.24.3 Constructor & Destructor Documentation	48
4.24.3.1 CommandMotorSetStopMode(Motor &motor, StopMode mode)	48
4.25 ev3::CommandMotorStop Class Reference	48
4.25.1 Detailed Description	49
4.25.2 Constructor & Destructor Documentation	49

4.25.2.1	CommandMotorStop(Motor &motor)	49
4.26	ev3::CommandSensor Class Reference	49
4.26.1	Detailed Description	50
4.26.2	Constructor & Destructor Documentation	50
4.26.2.1	CommandSensor(Sensor &sensor)	50
4.26.3	Member Function Documentation	50
4.26.3.1	getSensor()	50
4.27	ev3::Communication Class Reference	50
4.27.1	Detailed Description	51
4.27.2	Member Function Documentation	51
4.27.2.1	createThread(Queue< Message > *sendQueue, Queue< Message > *receiveQueue, bool isMaster=false)	51
4.27.2.2	run(Queue< Message > *sendQueue, Queue< Message > *receiveQueue, bool isMaster=false)	52
4.28	ev3::CommUtils Class Reference	52
4.28.1	Detailed Description	53
4.28.2	Member Function Documentation	53
4.28.2.1	getBufferFromString(const std::string message)	53
4.28.2.2	getStringFromBuffer(const Buffer buffer)	53
4.28.2.3	makeSockAddr(std::string ipAddress, int portNumber, struct sockaddr_↵ in *sockaddr)	54
4.28.2.4	preparePassiveSocket(unsigned int portNumber)	54
4.28.2.5	receiveMessage(unsigned int socket, Message &message, NetworkNode &sender)	54
4.28.2.6	receiveMessageDelay(unsigned int socket, Message &message, NetworkNode &sender, unsigned int msDelay=DEFAULT_RECEIVE_DELAY)	55
4.28.2.7	sendBroadcastMessage(unsigned int socket, unsigned int port, std::string message)	55
4.28.2.8	sendMessage(unsigned int socket, unsigned int port, Message &message, std::↵::string &proto, bool isMaster, unsigned int repeat=SENT_MESSAGE_COPIES)	55
4.28.2.9	sendMessageTo(unsigned int socket, std::string ipAddress, unsigned int destinationPort, std::string message)	56
4.28.3	Member Data Documentation	56
4.28.3.1	_packetBuffer	56

4.29	ev3dev::dc_motor Class Reference	56
4.30	ev3dev::device Class Reference	58
4.31	ev3::Devices Class Reference	58
4.31.1	Detailed Description	60
4.31.2	Constructor & Destructor Documentation	60
4.31.2.1	Devices(const Devices &other)	60
4.31.3	Member Function Documentation	61
4.31.3.1	addListener(Sensor::SensorType type)	61
4.31.3.2	checkDevices(RequiredDevices &devices)	61
4.31.3.3	getInstance()	61
4.31.3.4	getMotor(ev3dev::port_type port)	61
4.31.3.5	getSensor(ev3dev::port_type port)	61
4.31.3.6	operator=(const Devices &other)	62
4.31.3.7	removeListener(Sensor::SensorType type)	62
4.31.3.8	setProximitySensor(ev3dev::port_type port)	62
4.31.3.9	setSafetyTouchSensor(ev3dev::port_type port)	62
4.32	ev3::Event Class Reference	63
4.32.1	Detailed Description	63
4.32.2	Member Enumeration Documentation	64
4.32.2.1	EventType	64
4.32.3	Constructor & Destructor Documentation	64
4.32.3.1	Event(EventType type)	64
4.32.4	Member Function Documentation	64
4.32.4.1	getStringType()	64
4.32.4.2	getType()	64
4.33	ev3::EventAction Class Reference	65
4.33.1	Detailed Description	65
4.33.2	Constructor & Destructor Documentation	65
4.33.2.1	EventAction(EventType eventType, Action::ActionType actionType)	65
4.33.3	Member Function Documentation	65

4.33.3.1	getActionType()	66
4.34	ev3::EventQueue Class Reference	66
4.35	ev3::EventSensorWatch Class Reference	67
4.35.1	Detailed Description	67
4.35.2	Constructor & Destructor Documentation	67
4.35.2.1	EventSensorWatch(Sensor::SensorType type, SensorValue value)	67
4.35.3	Member Function Documentation	68
4.35.3.1	getType()	68
4.35.3.2	getValue()	68
4.36	ev3dev::button::file_descriptor Struct Reference	68
4.37	ev3dev::gyro_sensor Class Reference	68
4.38	ev3dev::i2c_sensor Class Reference	69
4.39	ev3dev::infrared_sensor Class Reference	70
4.40	ev3dev::large_motor Class Reference	70
4.41	ev3dev::lcd Class Reference	71
4.42	ev3dev::led Class Reference	71
4.43	ev3::LedControl Class Reference	73
4.43.1	Detailed Description	74
4.43.2	Member Enumeration Documentation	74
4.43.2.1	LedColors	74
4.43.2.2	LedType	74
4.43.3	Member Function Documentation	74
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)	74
4.43.3.2	flashColor(LedColors color, unsigned int msInterval, unsigned int repeat=1)	75
4.43.3.3	off(unsigned int leds=LedType::ALL)	75
4.43.3.4	on(unsigned int leds=LedType::ALL, unsigned int brightness=MAX_BRIGHTNESS)	75
4.43.3.5	onExclusive(unsigned int leds=LedType::ALL, unsigned int brightness=MAX_BRIGHTNESS)	75
4.43.3.6	setColor(LedColors color)	75

4.44	ev3dev::lego_port Class Reference	76
4.45	ev3dev::light_sensor Class Reference	76
4.46	ev3::Logger Class Reference	77
4.47	ev3::Master Class Reference	78
4.47.1	Detailed Description	79
4.47.2	Member Function Documentation	79
4.47.2.1	createThread(Queue< Message > *sendQueue, Queue< Message > *receiveQueue)	79
4.47.2.2	run(Queue< Message > *sendQueue, Queue< Message > *receiveQueue)	79
4.47.2.3	send(Message message, bool recordMessage=true)	79
4.48	ev3dev::medium_motor Class Reference	80
4.49	ev3::Message Class Reference	80
4.49.1	Detailed Description	82
4.49.2	Member Enumeration Documentation	82
4.49.2.1	MessageType	82
4.49.3	Constructor & Destructor Documentation	82
4.49.3.1	Message(unsigned int senderId, unsigned int receiverId, unsigned int messageId, MessageType type, StringVector parameters={})	82
4.49.4	Member Function Documentation	83
4.49.4.1	decodeMessage(const std::string message)	83
4.49.4.2	empty()	83
4.49.4.3	encodeMessage(Message &message)	83
4.49.4.4	getMessageId()	84
4.49.4.5	getParameters()	84
4.49.4.6	getReceiverId()	84
4.49.4.7	getSenderId()	84
4.49.4.8	getString()	84
4.49.4.9	getStringType()	84
4.49.4.10	getType()	85
4.49.4.11	setMessageId(unsigned int id)	85
4.49.4.12	setParameters(StringVector parameters)	85

4.49.4.13	setReceiverId(unsigned int id)	85
4.49.4.14	setSenderId(unsigned int id)	85
4.49.4.15	setType(MessageType type)	85
4.50	ev3::Motor Class Reference	86
4.50.1	Detailed Description	86
4.50.2	Constructor & Destructor Documentation	86
4.50.2.1	Motor(ev3dev::motor motor)	86
4.50.3	Member Function Documentation	87
4.50.3.1	getMotor()	87
4.51	ev3dev::motor Class Reference	87
4.52	ev3::CommUtils::NetworkNode Struct Reference	89
4.52.1	Detailed Description	89
4.53	ev3dev::power_supply Class Reference	90
4.54	ev3::Queue< T > Class Template Reference	90
4.55	ev3dev::remote_control Class Reference	91
4.56	ev3::Robot Class Reference	92
4.56.1	Detailed Description	93
4.56.2	Constructor & Destructor Documentation	94
4.56.2.1	Robot(Devices::RequiredDevices devices, AvailableActions actions)	94
4.56.3	Member Function Documentation	95
4.56.3.1	createThread(Queue< Message > *sendQueue, Queue< Message > *receiveQueue)	95
4.56.3.2	generateBehaviour(Behaviour::BehaviourType type, StringVector parameters)	95
4.56.3.3	getString()	95
4.56.3.4	run(Queue< Message > *sendQueue, Queue< Message > *receiveQueue)	95
4.56.3.5	send(Message message)	96
4.56.4	Member Data Documentation	96
4.56.4.1	_behaviourSet	96
4.56.4.2	_pulsePerUnitRatio	96
4.57	ev3::RobotModelA Class Reference	96
4.57.1	Detailed Description	97

4.57.2	Member Function Documentation	97
4.57.2.1	generateAction(SharedPtrAction action, Action::ActionType type)	97
4.57.2.2	generateBehaviour(Behaviour::BehaviourType type, StringVector parameters) override	97
4.57.2.3	getString() override	98
4.58	ev3::RobotState Class Reference	98
4.58.1	Detailed Description	100
4.58.2	Member Enumeration Documentation	100
4.58.2.1	States	100
4.58.3	Constructor & Destructor Documentation	100
4.58.3.1	RobotState(ChangeMap changes, LedControl *led)	100
4.58.4	Member Function Documentation	100
4.58.4.1	changeState(States state)	100
4.58.4.2	getBehaviour()	101
4.58.4.3	getPendingMessage()	101
4.58.4.4	isPendingEnabled()	101
4.58.4.5	process(Message msg)	101
4.58.4.6	setBehaviour(SharedPtrBehaviour behaviour)	101
4.58.4.7	switchState(Message::MessageType type)	102
4.59	ev3::RobotStateActive Class Reference	102
4.59.1	Constructor & Destructor Documentation	102
4.59.1.1	RobotStateActive(LedControl *led)	102
4.59.2	Member Function Documentation	103
4.59.2.1	process(Message msg)	103
4.60	ev3::RobotStateIdle Class Reference	103
4.60.1	Constructor & Destructor Documentation	103
4.60.1.1	RobotStateIdle(LedControl *led)	103
4.60.2	Member Function Documentation	104
4.60.2.1	process(Message msg)	104
4.61	ev3::RobotStatePanic Class Reference	104
4.61.1	Constructor & Destructor Documentation	104

4.61.1.1	RobotStatePanic(LedControl *led)	104
4.61.2	Member Function Documentation	105
4.61.2.1	process(Message msg)	105
4.62	ev3::RobotStatePaused Class Reference	105
4.62.1	Constructor & Destructor Documentation	105
4.62.1.1	RobotStatePaused(LedControl *led)	105
4.62.2	Member Function Documentation	106
4.62.2.1	process(Message msg)	106
4.63	ev3::RobotStateWorking Class Reference	106
4.63.1	Constructor & Destructor Documentation	106
4.63.1.1	RobotStateWorking(LedControl *led)	106
4.63.2	Member Function Documentation	107
4.63.2.1	process(Message msg)	107
4.64	ev3dev::sensor Class Reference	107
4.65	ev3::Sensor Class Reference	109
4.65.1	Detailed Description	109
4.65.2	Member Enumeration Documentation	109
4.65.2.1	SensorType	109
4.65.3	Constructor & Destructor Documentation	110
4.65.3.1	Sensor(ev3dev::sensor sensor, SensorType type)	110
4.65.4	Member Function Documentation	110
4.65.4.1	getDecimals()	110
4.65.4.2	getNumValues()	110
4.65.4.3	getSensor()	110
4.65.4.4	getType()	110
4.65.4.5	getValue(unsigned int n)	110
4.65.4.6	getValueF(unsigned int n)	111
4.65.4.7	prepareMessage(SensorValue value, SensorType type)	111
4.66	ev3dev::servo_motor Class Reference	111
4.67	ev3::SignalHandler Class Reference	112
4.68	ev3dev::sound Class Reference	113
4.69	ev3dev::sound_sensor Class Reference	113
4.70	ev3dev::touch_sensor Class Reference	114
4.71	ev3dev::ultrasonic_sensor Class Reference	114

5	File Documentation	117
5.1	/home/panda/Dokumenty/Repos/Ev3Dev/include/action/Action.h File Reference	117
5.1.1	Detailed Description	118
5.1.2	Typedef Documentation	118
5.1.2.1	StoredActions	118
5.2	/home/panda/Dokumenty/Repos/Ev3Dev/include/action/Behaviour.h File Reference	118
5.2.1	Detailed Description	119
5.3	/home/panda/Dokumenty/Repos/Ev3Dev/include/action/BehaviourState.h File Reference	119
5.3.1	Detailed Description	119
5.4	/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/Communication.h File Reference	119
5.4.1	Detailed Description	120
5.5	/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/CommUtils.h File Reference	120
5.5.1	Detailed Description	121
5.6	/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/Message.h File Reference	121
5.6.1	Detailed Description	121
5.7	/home/panda/Dokumenty/Repos/Ev3Dev/include/robot/Devices.h File Reference	121
5.7.1	Detailed Description	122
Index		123

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	40
ev3::Command	40
ev3::CommandMotor	41
ev3::CommandMotorReset	43
ev3::CommandMotorRunForever	44
ev3::CommandMotorSetSpeed	45
ev3::CommandMotorSetSpeedRegEnabled	46
ev3::CommandMotorSetStopMode	47
ev3::CommandMotorStop	48
ev3::CommandSensor	49
ev3::Communication	50
ev3::CommUtils	52
ev3dev::device	58
ev3dev::dc_motor	56
ev3dev::led	71
ev3dev::lego_port	76
ev3dev::motor	87
ev3dev::large_motor	70

ev3dev::medium_motor	80
ev3dev::power_supply	90
ev3dev::sensor	107
ev3dev::color_sensor	39
ev3dev::gyro_sensor	68
ev3dev::i2c_sensor	69
ev3dev::infrared_sensor	70
ev3dev::light_sensor	76
ev3dev::sound_sensor	113
ev3dev::touch_sensor	114
ev3dev::ultrasonic_sensor	114
ev3dev::servo_motor	111
ev3::Devices	58
ev3::Event	63
ev3::EventAction	65
ev3::EventSensorWatch	67
ev3::EventQueue	66
ev3dev::button::file_descriptor	68
ev3dev::lcd	71
ev3::LedControl	73
ev3::Logger	77
ev3::Master	78
ev3::Message	80
ev3::Motor	86
ev3::CommUtils::NetworkNode	89
ev3::Queue< T >	90
ev3::Queue< ev3::Message >	90
ev3dev::remote_control	91
ev3::Robot	92
ev3::RobotModelA	96
ev3::RobotState	98
ev3::RobotStateActive	102
ev3::RobotStateIdle	103
ev3::RobotStatePanic	104
ev3::RobotStatePaused	105
ev3::RobotStateWorking	106
ev3::Sensor	109
ev3::SignalHandler	112
ev3dev::sound	113

Chapter 2

Class Index

2.1 Class List

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

ev3::Action	Base class for all Action controlling classes	9
ev3::ActionDriveDistance	Implements Robot simple task to drive straight for a given distance	13
ev3::ActionDriveForever	Implements Robot simple task to drive straight forever	15
ev3::ActionRepeat	Stores many Actions in a vector and executes them in loop	17
ev3::ActionRotate	Implements Robot simple task to rotate a given angle, while not driving	19
ev3::ActionRotateRandDirection	Implements Robot simple task to rotate a random angle	21
ev3::ActionStop	Implements Robot simple task to stop all active motors	23
ev3::Agent	Master-side representative of a robot unit	24
ev3::Behaviour	Base class for all defined behaviours	27
ev3::BehaviourDriveOnSquare	Implements complex behaviour of driving on a square-shaped route	31
ev3::BehaviourExploreRandom	Implements complex behaviour of exploring the surrounding with random rotation	33
ev3::BehaviourState	Encapsulates action and other information in a form of a state	34
ev3::CommUtils::Buffer	Contains buffer and its size	37
ev3dev::button	37
ev3::CircularBuffer< T >	38
ev3dev::color_sensor	39
ev3::ColorUtils	40
ev3::Command	Base class for all command controlling classes	40
ev3::CommandMotor	Base class for all motor controlling commands	41
ev3::CommandMotorReset	Calls <code>reset ()</code> method of containing Motor	43

ev3::CommandMotorRunForever	
Calls <code>run_forever()</code> method of containing Motor	44
ev3::CommandMotorSetSpeed	
Call <code>set_speed_sp()</code> method of containing Motor	45
ev3::CommandMotorSetSpeedRegEnabled	
Calls <code>set_speed_regulation_enabled()</code> method of containing Motor	46
ev3::CommandMotorSetStopMode	
Calls <code>set_stop_command()</code> method of containing Motor	47
ev3::CommandMotorStop	
Calls <code>stop()</code> method of containing Motor	48
ev3::CommandSensor	
Base class for all sensor controlling commands	49
ev3::Communication	
Encapsulates low-level communication and adds logic concerning sending and receiving Message queueing	50
ev3::CommUtils	
Responsible for low-level communication	52
ev3dev::dc_motor	56
ev3dev::device	58
ev3::Devices	
Singleton class responsible for managing devices connected to the robot	58
ev3::Event	
Base class for all Event classes	63
ev3::EventAction	
Event class triggered when something happened with Action	65
ev3::EventQueue	66
ev3::EventSensorWatch	
Triggered when measurement of certain Sensor occurred	67
ev3dev::button::file_descriptor	68
ev3dev::gyro_sensor	68
ev3dev::i2c_sensor	69
ev3dev::infrared_sensor	70
ev3dev::large_motor	70
ev3dev::lcd	71
ev3dev::led	71
ev3::LedControl	
Class specifically designed to eliminate <code>ev3dev</code> library limitations of controlling LED panel	73
ev3dev::lego_port	76
ev3dev::light_sensor	76
ev3::Logger	77
ev3::Master	
Controls the whole system and knows about every Agent	78
ev3dev::medium_motor	80
ev3::Message	
Stores information passed between physical system units (another robots or master)	80
ev3::Motor	
Encapsulates ev3dev::motor	86
ev3dev::motor	87
ev3::CommUtils::NetworkNode	
Stores information about a particular node in the network	89
ev3dev::power_supply	90
ev3::Queue< T >	90
ev3dev::remote_control	91
ev3::Robot	
Main class representing actual robot	92
ev3::RobotModelA	
Describes particular Robot construction and its way of implementing actions and running behaviours	96

ev3::RobotState	
Base class for all Robot states	98
ev3::RobotStateActive	102
ev3::RobotStateIdle	103
ev3::RobotStatePanic	104
ev3::RobotStatePaused	105
ev3::RobotStateWorking	106
ev3dev::sensor	107
ev3::Sensor	
Encapsulates ev3dev::sensor	109
ev3dev::servo_motor	111
ev3::SignalHandler	112
ev3dev::sound	113
ev3dev::sound_sensor	113
ev3dev::touch_sensor	114
ev3dev::ultrasonic_sensor	114

Chapter 3

File Index

3.1 File List

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

/home/panda/Dokumenty/Repos/Ev3Dev/include/action/ Action.h	
Contains all Action classes	117
/home/panda/Dokumenty/Repos/Ev3Dev/include/action/ Behaviour.h	
Contains all Behaviour classes	118
/home/panda/Dokumenty/Repos/Ev3Dev/include/action/ BehaviourState.h	
Contains BehaviourState class	119
/home/panda/Dokumenty/Repos/Ev3Dev/include/action/ Command.h	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/action/ CommandMotor.h	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/action/ CommandSensor.h	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/ Communication.h	
Contains Communication class	119
/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/ CommUtils.h	
Contains CommUtils class	120
/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/ Event.h	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/ Message.h	
Contains Message class	121
/home/panda/Dokumenty/Repos/Ev3Dev/include/control/ LedControl.h	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ ev3dev.h	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/master/ Agent.h	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/master/ Master.h	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/robot/ Devices.h	
Contains Devices classes	121
/home/panda/Dokumenty/Repos/Ev3Dev/include/robot/ Motor.h	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/robot/ Robot.h	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/robot/ RobotModelA.h	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/robot/ RobotState.h	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/robot/ Sensor.h	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/utils/ CircularBuffer.h	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/utils/ ColorUtils.h	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/utils/ EventQueue.h	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/utils/ Logger.h	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/utils/ Queue.h	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/utils/ SignalHandler.h	??
/home/panda/Dokumenty/Repos/Ev3Dev/include/utils/ Utils.h	??

Chapter 4

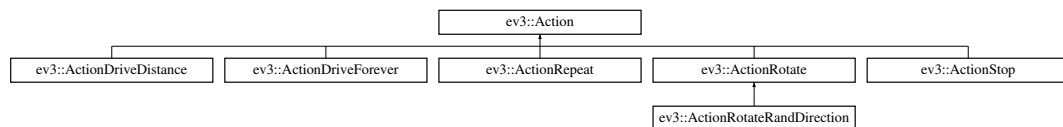
Class Documentation

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>	Commands stored within this Action .
<i>type</i>	Type of Action used.

4.1.3.2 Action::Action (CommandsVector *commands*)

Constructor with CommandsVector parameter.

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

Parameters

<i>commands</i>	Commands stored within this Action .
-----------------	--

4.1.3.3 Action::Action (ActionType *type*)

Constructor with ActionType parameter.

Parameters

<i>type</i>	Type of Action 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:

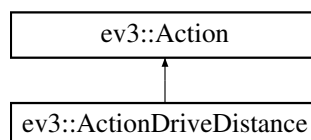
- /home/panda/Dokumenty/Repos/Ev3Dev/include/action/[Action.h](#)
- /home/panda/Dokumenty/Repos/Ev3Dev/src/action/Action.cpp

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 Robot 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 Robot 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:

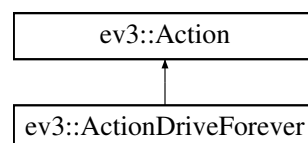
- `/home/panda/Dokumenty/Repos/Ev3Dev/include/action/Action.h`
- `/home/panda/Dokumenty/Repos/Ev3Dev/src/action/Action.cpp`

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:

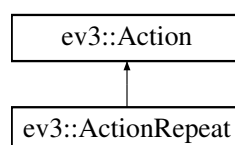
- `/home/panda/Dokumenty/Repos/Ev3Dev/include/action/Action.h`
- `/home/panda/Dokumenty/Repos/Ev3Dev/src/action/Action.cpp`

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:

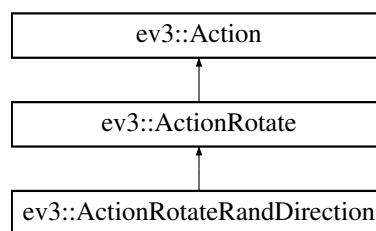
- [/home/panda/Dokumenty/Repos/Ev3Dev/include/action/Action.h](#)
- [/home/panda/Dokumenty/Repos/Ev3Dev/src/action/Action.cpp](#)

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 Robot 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:

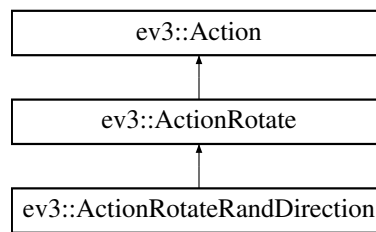
- `/home/panda/Dokumenty/Repos/Ev3Dev/include/action/Action.h`
- `/home/panda/Dokumenty/Repos/Ev3Dev/src/action/Action.cpp`

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:

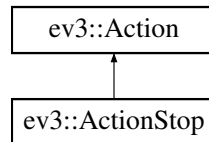
- /home/panda/Dokumenty/Repos/Ev3Dev/include/action/[Action.h](#)
- /home/panda/Dokumenty/Repos/Ev3Dev/src/action/[Action.cpp](#)

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>	Message to be analyzed.
<i>retMessage</i>	Modified Message to be sent to Robot .

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

Set currently executing [Behaviour](#).

Parameters

<i>behaviour</i>	Behaviour 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 Agent .
-----------	---

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

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

Parameters

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

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

Update data concerning last message sent.

Parameters

<i>message</i>	Last Message sent to corresponding Robot .
----------------	--

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:

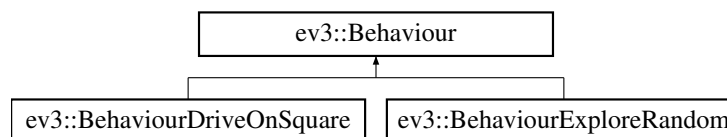
- /home/panda/Dokumenty/Repos/Ev3Dev/include/master/Agent.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/master/Agent.cpp

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>	Behaviour type.
<i>states</i>	Vector of available Behaviour states.

4.9.3.2 Behaviour::Behaviour (BehaviourType *type*)

Constructor with behaviour type.

Parameters

<i>type</i>	Behaviour 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>	Event 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 Behaviour .
-----------------------	--

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

Available states setter.

Parameters

<i>states</i>	Vector of states for this Behaviour .
---------------	---

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

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

Parameters

<i>state</i>	BehaviourState object for stop state.
--------------	---

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

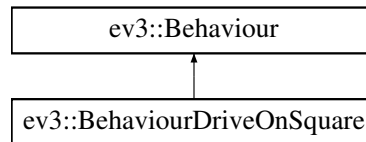
- /home/panda/Dokumenty/Repos/Ev3Dev/include/action/[Behaviour.h](#)
- /home/panda/Dokumenty/Repos/Ev3Dev/src/action/Behaviour.cpp

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 Behaviour 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:

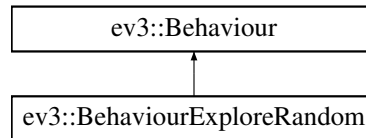
- [/home/panda/Dokumenty/Repos/Ev3Dev/include/action/Behaviour.h](#)
- [/home/panda/Dokumenty/Repos/Ev3Dev/src/action/Behaviour.cpp](#)

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

<i>states</i>	Vector of available Behaviour 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](#) &)=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 &)` [default]

Default copy constructor.

Parameters

<i>Other</i>	BehaviourState 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>	Action 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>	Action 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

Public Member Functions

- **CircularBuffer** (unsigned int limit)
- void **push** (T object)
- bool **contain** (T object)

Private Attributes

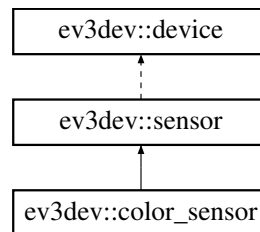
- `std::vector< T > _buffer`
- `unsigned int _index = 0`
- `unsigned int _limit`

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

- `/home/panda/Dokumenty/Repos/Ev3Dev/include/utls/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

Public Types

- typedef std::string **colorCode**

Static Public Member Functions

- static void **printColorTest** ()

Static Public Attributes

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

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:

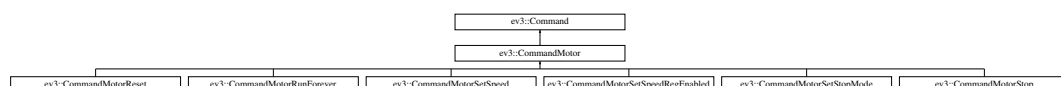
- /home/panda/Dokumenty/Repos/Ev3Dev/include/action/Command.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/action/Command.cpp

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>	Motor to execute CommandMotor 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:

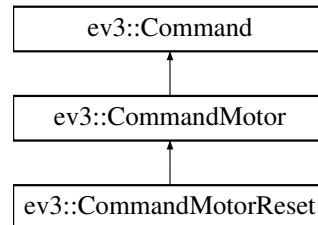
- [/home/panda/Dokumenty/Repos/Ev3Dev/include/action/CommandMotor.h](#)
- [/home/panda/Dokumenty/Repos/Ev3Dev/src/action/CommandMotor.cpp](#)

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

<code>motor</code>	Motor to execute CommandMotor on.
--------------------	---

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

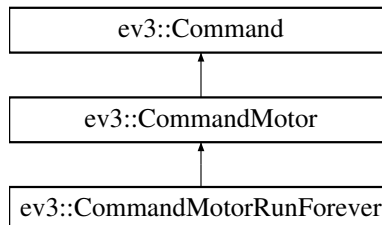
- `/home/panda/Dokumenty/Repos/Ev3Dev/include/action/CommandMotor.h`
- `/home/panda/Dokumenty/Repos/Ev3Dev/src/action/CommandMotor.cpp`

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>	Motor to execute CommandMotor on.
--------------	---

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

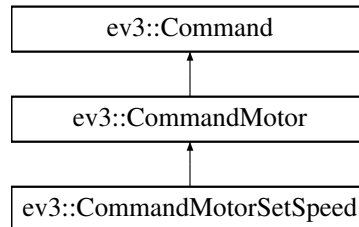
- `/home/panda/Dokumenty/Repos/Ev3Dev/include/action/CommandMotor.h`
- `/home/panda/Dokumenty/Repos/Ev3Dev/src/action/CommandMotor.cpp`

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>	Motor to execute CommandMotor on.
<i>value</i>	Speed value in tachometer 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:

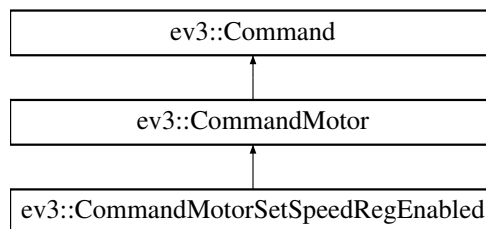
- /home/panda/Dokumenty/Repos/Ev3Dev/include/action/CommandMotor.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/action/CommandMotor.cpp

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>	Motor to execute CommandMotor 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:

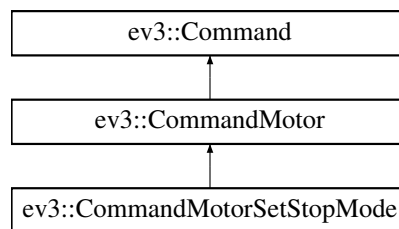
- /home/panda/Dokumenty/Repos/Ev3Dev/include/action/CommandMotor.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/action/CommandMotor.cpp

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>	Motor to execute CommandMotor on.
<i>mode</i>	Stop mode chosen from <code>StopMode</code> .

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

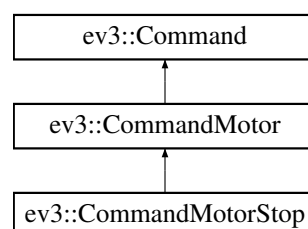
- `/home/panda/Dokumenty/Repos/Ev3Dev/include/action/CommandMotor.h`
- `/home/panda/Dokumenty/Repos/Ev3Dev/src/action/CommandMotor.cpp`

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>	Motor to execute CommandMotor on.
--------------	---

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

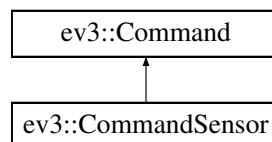
- /home/panda/Dokumenty/Repos/Ev3Dev/include/action/CommandMotor.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/action/CommandMotor.cpp

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>	Sensor to execute CommandSensor 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 Message queue.
<i>receiveQueue</i>	In Message 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 Message queue.
<i>receiveQueue</i>	In Message 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>	Message reference to be set after receiving.
<i>sender</i>	NetworkNode 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>	Message reference to be set after receiving.
<i>sender</i>	NetworkNode 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>	Message 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>	Message to be sent.
<i>proto</i>	Message prototype passed to avoid its multiple encoding.
<i>isMaster</i>	Flag from Communication 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>	Message 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]`

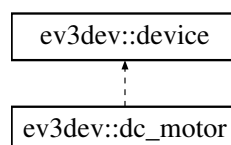
Circular buffer used to store limited number of previous [Message](#) prototypes received.

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**

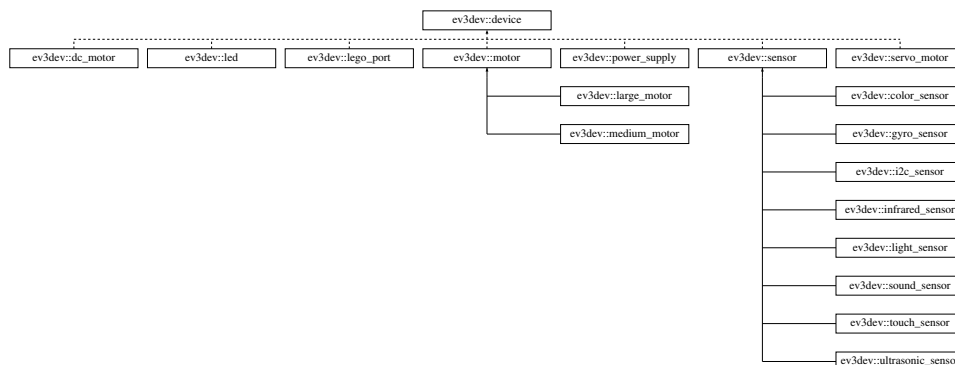
Additional Inherited Members

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 private constructor (preventing object construction).
- [Devices](#) (const [Devices](#) &other)
Default private copy constructor (preventing object construction by copying).
- [Devices](#) & operator= (const [Devices](#) &other)
Private assignment operator (preventing object assignment).
- [~Devices](#) ()
Default private 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 private copy constructor (preventing object construction by copying).

Parameters

<i>other</i>	Other Devices 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 Sensor 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 Motor 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 Sensor is.
-------------	---

Returns

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

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

Private assignment operator (preventing object assignment).

Parameters

<i>other</i>	Other Devices 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 Sensor 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:

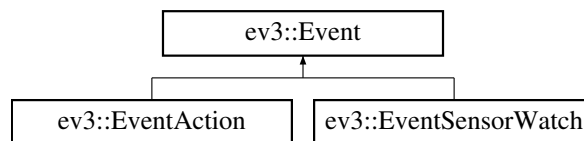
- /home/panda/Dokumenty/Repos/Ev3Dev/include/robot/Devices.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/robot/Devices.cpp

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:

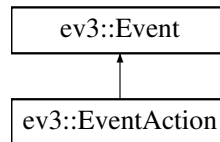
- /home/panda/Dokumenty/Repos/Ev3Dev/include/communication/Event.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/communication/Event.cpp

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 Event types concerning actions.
<i>actionType</i>	Type of Action 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

Public Member Functions

- void **push** (SharedPtrEvent message)
- SharedPtrEvent **pop** ()
- bool **empty** ()
- unsigned int **size** ()

Static Public Member Functions

- static EventQueue * **getInstance** ()
- static void **destroy** ()

Protected Member Functions

- **EventQueue** (const EventQueue &)
- EventQueue & **operator=** (const EventQueue &)

Protected Attributes

- std::queue< SharedPtrEvent > **_queue**
- std::mutex **_mutex**

Static Protected Attributes

- static EventQueue * **_instance** = nullptr

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

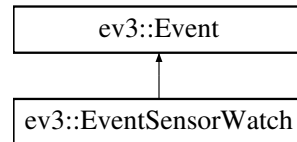
- /home/panda/Dokumenty/Repos/Ev3Dev/include/utils/EventQueue.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/utils/EventQueue.cpp

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

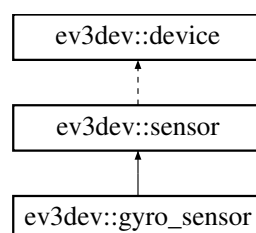
- `int _fd`

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" }

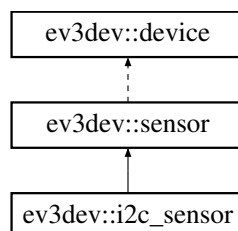
Additional Inherited Members

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)

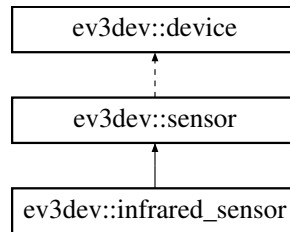
Additional Inherited Members

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" }

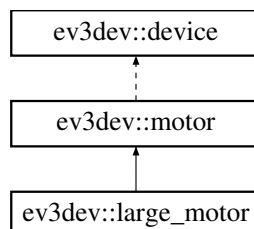
Additional Inherited Members

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

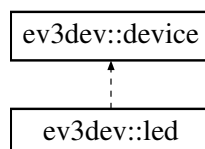
- unsigned char * **_fb**
- uint32_t **_fbsize**
- uint32_t **_llength**
- uint32_t **_xres**
- uint32_t **_yres**
- uint32_t **_bpp**

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 LedControl::LedType 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 LedControl::LedType 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 LedControl::LedType 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 LedControl::LedType 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

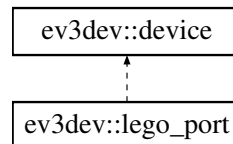
<i>color</i>	Type of to be displayed.
--------------	--------------------------

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

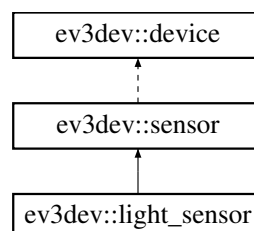
Additional Inherited Members

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

Public Types

- enum **LogLevel** {
DEBUG = 1, **VERBOSE** = 1 << 1, **INFO** = 1 << 2, **WARNING** = 1 << 3,
ERROR = 1 << 4 }
- enum **LogOutput** { **STD_OUT** = 1, **STD_ERR** = 1 << 1, **FILE** = 1 << 2 }

Public Member Functions

- void **log** (std::string message, LogLevel level, LogOutput output=STD_OUT)
- void **setLogLevel** (LogLevel level)
- void **setLogLevel** (std::string level)
- void **setLogOutput** (LogOutput output)

Static Public Member Functions

- static [Logger](#) * **getInstance** ()
- static void **destroy** ()

Private Member Functions

- **Logger** (const [Logger](#) &)
- [Logger](#) & **operator=** (const [Logger](#) &)
- std::string **getLabel** (LogLevel level, LogOutput output)
- std::string **getColor** (LogLevel level, LogOutput output)

Private Attributes

- LogLevel **_level** = ERROR
- LogOutput **_output**
- bool **_loggerForced** = false

Static Private Attributes

- static [Logger](#) * **_instance** = nullptr

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

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

<i>sendQueue</i>	Out Message queue.
<i>receiveQueue</i>	In Message 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

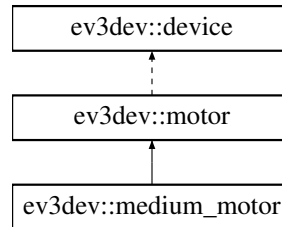
<i>message</i>	Message to be passed to Communication thread via <code>sendQueue</code> .
<i>recordMessage</i>	True if information about message should be saved for further purposes, false otherwise.

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 }

Messge 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 Message 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 Message 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 Message 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>	ev3dev::Motor object.
--------------	---------------------------------------

4.50.3 Member Function Documentation

4.50.3.1 ev3dev::motor Motor::getMotor ()

[Motor](#) getter.

Returns

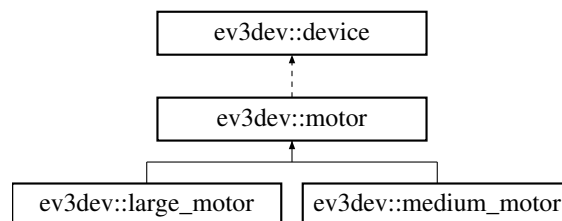
Stored [ev3dev::motor](#) object.

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

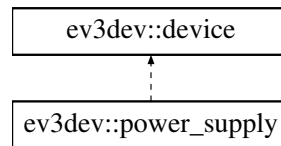
Stores information about a particular node in the network.

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

Public Member Functions

- void **push** (T message)
- T **pop** ()
- bool **empty** ()

Private Attributes

- std::queue< T > **_messages**
- std::mutex **_mutex**

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:

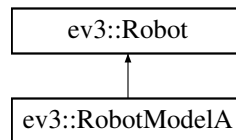
- /home/panda/Dokumenty/Repos/Ev3Dev/include/ev3dev/ev3dev.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/ev3dev/ev3dev.cpp

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 Action 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 Message queue.
<i>receiveQueue</i>	In Message 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>	Behaviour type.
<i>parameters</i>	Additional parameters required by a particular Behaviour .

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 Message queue.
<i>receiveQueue</i>	In Message queue.

4.56.3.5 void Robot::send ([Message](#) *message*)

General sending method for logging and assigning id.

Parameters

<i>message</i>	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:

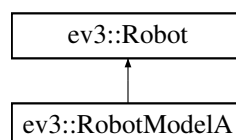
- /home/panda/Dokumenty/Repos/Ev3Dev/include/robot/Robot.h
- /home/panda/Dokumenty/Repos/Ev3Dev/src/robot/Robot.cpp

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 Action to be constructed.
<i>type</i>	Action type.

Returns

Copy of the [Action](#) object with new data.

4.57.2.2 [SharedPtrBehaviour](#) [RobotModelA::generateBehaviour](#) ([Behaviour::BehaviourType](#) type, [StringVector](#) parameters) [override], [private], [virtual]

Overrides [Robot](#) method of [Behaviour](#) creation.

See also

[Robot::generateBehaviour](#)

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

4.57.2.3 `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:

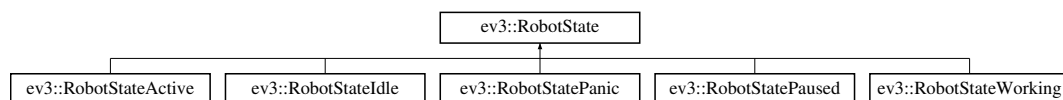
- `/home/panda/Dokumenty/Repos/Ev3Dev/include/robot/RobotModelA.h`
- `/home/panda/Dokumenty/Repos/Ev3Dev/src/robot/RobotModelA.cpp`

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.
- [Message::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 LedControl 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>	Message 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>	Behaviour 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

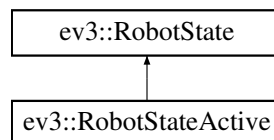
Pointer to created state.

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

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 Constructor & Destructor Documentation

4.59.1.1 RobotStateActive::RobotStateActive (LedControl * led)

Constructor with LED controller.

Parameters

<i>led</i>	LedControl pointer.
------------	-------------------------------------

4.59.2 Member Function Documentation

4.59.2.1 RobotState * RobotStateActive::process (Message msg) [virtual]

Overriden process method.

See also

[RobotState::process](#)

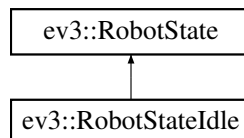
Reimplemented from [ev3::RobotState](#).

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

Inheritance diagram for ev3::RobotStateIdle:



Public Member Functions

- [RobotStateIdle](#) ([LedControl](#) *led)
Constructor with LED controller.
- [RobotState](#) * [process](#) ([Message](#) msg)

Additional Inherited Members

4.60.1 Constructor & Destructor Documentation

4.60.1.1 RobotStateIdle::RobotStateIdle (LedControl * led)

Constructor with LED controller.

Parameters

<i>led</i>	LedControl pointer.
------------	-------------------------------------

4.60.2 Member Function Documentation

4.60.2.1 `RobotState * RobotStateIdle::process (Message msg)` [virtual]

Parameters

<i>msg</i>	
------------	--

Returns

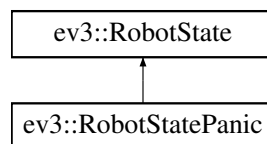
Reimplemented from [ev3::RobotState](#).

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

Inheritance diagram for `ev3::RobotStatePanic`:



Public Member Functions

- [RobotStatePanic](#) ([LedControl](#) *led)
Constructor with LED controller.
- [RobotState](#) * [process](#) ([Message](#) msg)

Additional Inherited Members

4.61.1 Constructor & Destructor Documentation

4.61.1.1 `RobotStatePanic::RobotStatePanic (LedControl * led)`

Constructor with LED controller.

Parameters

<i>led</i>	LedControl pointer.
------------	-------------------------------------

4.61.2 Member Function Documentation

4.61.2.1 RobotState * RobotStatePanic::process (Message msg) [virtual]

Parameters

<i>msg</i>	
------------	--

Returns

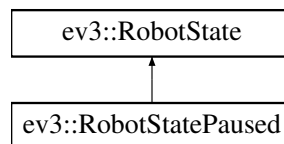
Reimplemented from [ev3::RobotState](#).

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

Inheritance diagram for ev3::RobotStatePaused:



Public Member Functions

- [RobotStatePaused](#) ([LedControl](#) *led)
Constructor with LED controller.
- [RobotState](#) * [process](#) ([Message](#) msg)

Additional Inherited Members

4.62.1 Constructor & Destructor Documentation

4.62.1.1 RobotStatePaused::RobotStatePaused (LedControl * led)

Constructor with LED controller.

Parameters

<i>led</i>	LedControl pointer.
------------	-------------------------------------

4.62.2 Member Function Documentation

4.62.2.1 `RobotState * RobotStatePaused::process (Message msg)` `[virtual]`

Parameters

<i>msg</i>	
------------	--

Returns

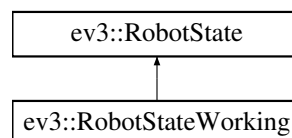
Reimplemented from [ev3::RobotState](#).

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

Inheritance diagram for `ev3::RobotStateWorking`:



Public Member Functions

- [RobotStateWorking](#) ([LedControl](#) *led)
Constructor with LED controller.
- [RobotState](#) * [process](#) ([Message](#) msg)

Additional Inherited Members

4.63.1 Constructor & Destructor Documentation

4.63.1.1 `RobotStateWorking::RobotStateWorking (LedControl * led)`

Constructor with LED controller.

Parameters

<i>led</i>	LedControl pointer.
------------	-------------------------------------

4.63.2 Member Function Documentation

4.63.2.1 RobotState * RobotStateWorking::process (Message msg) [virtual]

Parameters

<i>msg</i>	
------------	--

Returns

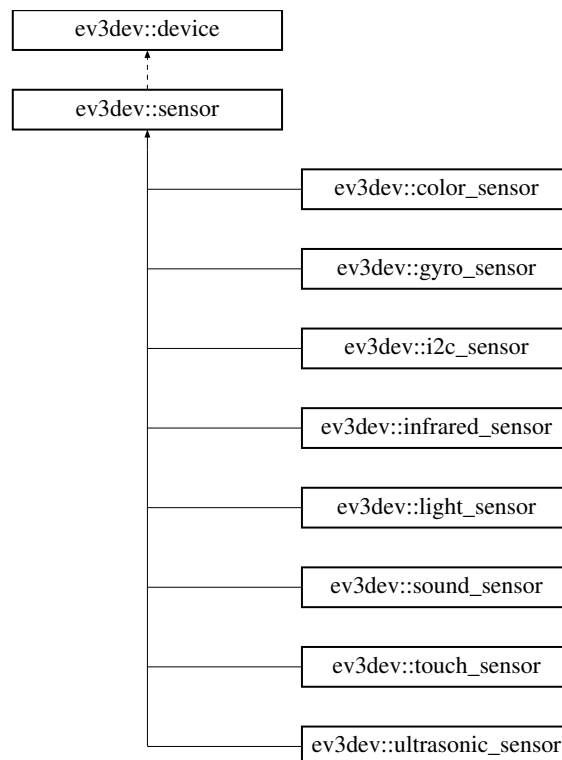
Reimplemented from [ev3::RobotState](#).

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)
- [ev3dev::sensor](#) [getSensor](#) ()
- int [getValue](#) (unsigned int n)
- float [getValueF](#) (unsigned int n)
- int [getDecimals](#) ()
- unsigned int [getNumValues](#) ()
- [SensorType](#) [getType](#) ()

Static Public Member Functions

- static StringVector [prepareMessage](#) (SensorValue value, [SensorType](#) type)

Private Attributes

- [SensorType](#) [_type](#)
- [ev3dev::sensor](#) [_sensor](#)

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)`

Parameters

<i>sensor</i>	
<i>type</i>	

4.65.4 Member Function Documentation

4.65.4.1 `int Sensor::getDecimals ()`

Returns

4.65.4.2 `unsigned int Sensor::getNumValues ()`

Returns

4.65.4.3 `ev3dev::sensor Sensor::getSensor ()`

Returns

4.65.4.4 `Sensor::SensorType Sensor::getType ()`

Returns

4.65.4.5 `int Sensor::getValue (unsigned int n)`

Parameters

<i>n</i>	
----------	--

Returns

4.65.4.6 float Sensor::getValueF (unsigned int *n*)

Parameters

<i>n</i>	
----------	--

Returns

4.65.4.7 StringVector Sensor::prepareMessage (SensorValue *value*, SensorType *type*) [static]

Parameters

<i>value</i>	
<i>type</i>	

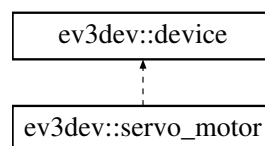
Returns

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

Static Public Member Functions

- `static void HandleSignal (int signum)`

Static Public Attributes

- `static Robot * robot = nullptr`
- `static Master * master = nullptr`

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

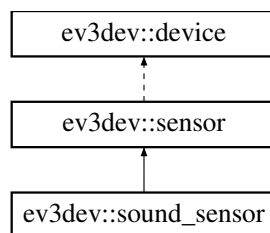
- static void **beep** (const std::string &args="", bool bSynchronous=false)
- static void **tone** (float frequency, float ms, bool bSynchronous=false)
- static void **tone** (const std::vector< std::vector< float > > &sequence, bool bSynchronous=false)
- static void **play** (const std::string &soundfile, 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" }

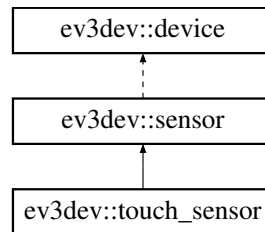
Additional Inherited Members

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" }

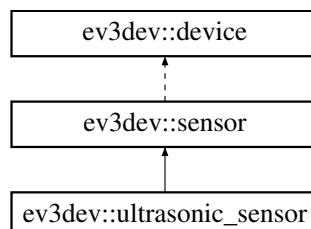
Additional Inherited Members

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](#) outputs.

5.7.1 Detailed Description

Contains Devices classes.

Index

/home/panda/Dokumenty/Repos/Ev3Dev/include/action/↔ ActionDriveDistance
Action.h, 117 ev3::ActionDriveDistance, 14

/home/panda/Dokumenty/Repos/Ev3Dev/include/action/↔ ActionDriveForever
Behaviour.h, 118 ev3::ActionDriveForever, 16

/home/panda/Dokumenty/Repos/Ev3Dev/include/action/↔ ActionRepeat
BehaviourState.h, 119 ev3::ActionRepeat, 18

/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/↔ ActionRotate
CommUtils.h, 120 ev3::ActionRotate, 20

/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/↔ ActionRotateRandDirection
Communication.h, 119 ev3::ActionRotateRandDirection, 21, 22

/home/panda/Dokumenty/Repos/Ev3Dev/include/communication/↔ ActionStop
Message.h, 121 ev3::ActionStop, 23

/home/panda/Dokumenty/Repos/Ev3Dev/include/robot/↔ ActionType
Devices.h, 121 ev3::Action, 11

_behaviourSet
ev3::Robot, 96

_endCondition
ev3::Action, 13

_measurements
ev3::Agent, 26

_packetBuffer
ev3::CommUtils, 56

_pulsePerUnitRatio
ev3::Robot, 96

ABORT
ev3::Message, 82

ACTION_FINISHED
ev3::Event, 64

ACTION_INTERR
ev3::Event, 64
ev3::Message, 82

ACTION_OK
ev3::Message, 82

ACTIVE
ev3::RobotState, 100

ACK
ev3::Message, 82

AGENT_OVER
ev3::Message, 82

AGENT
ev3::Message, 82

ALL
ev3::LedControl, 74

AMBER
ev3::LedControl, 74

Action
ev3::Action, 11

Action.h
StoredActions, 118

addListener
ev3::Devices, 61

BEHAVIOUR_START
ev3::Event, 64

BEHAVIOUR_STOP
ev3::Event, 64

BEHAVIOUR
ev3::Message, 82

BRAKE
ev3::CommandMotorSetStopMode, 48

Behaviour
ev3::Behaviour, 29

BehaviourDriveOnSquare
ev3::BehaviourDriveOnSquare, 31, 32

BehaviourExploreRandom
ev3::BehaviourExploreRandom, 33

BehaviourState
ev3::BehaviourState, 35

BehaviourType
ev3::Behaviour, 28

COAST
ev3::CommandMotorSetStopMode, 48

COLOR
ev3::Sensor, 109

CUSTOM
ev3::Behaviour, 28

changeState
ev3::RobotState, 100

checkDevices
ev3::Devices, 61

CommandMotor
ev3::CommandMotor, 42

CommandMotorReset
ev3::CommandMotorReset, 43

- CommandMotorRunForever
 - ev3::CommandMotorRunForever, [44](#)
- CommandMotorSetSpeed
 - ev3::CommandMotorSetSpeed, [45](#)
- CommandMotorSetSpeedRegEnabled
 - ev3::CommandMotorSetSpeedRegEnabled, [46](#)
- CommandMotorSetStopMode
 - ev3::CommandMotorSetStopMode, [48](#)
- CommandMotorStop
 - ev3::CommandMotorStop, [49](#)
- CommandSensor
 - ev3::CommandSensor, [50](#)
- createThread
 - ev3::Communication, [51](#)
 - ev3::Master, [79](#)
 - ev3::Robot, [95](#)
- DRIVE_DISTANCE
 - ev3::Action, [11](#)
- DRIVE_FOREVER
 - ev3::Action, [11](#)
- DRIVE_ON_SQUARE
 - ev3::Behaviour, [28](#)
- decodeMessage
 - ev3::Message, [83](#)
- Devices
 - ev3::Devices, [60](#)
- EMPTY
 - ev3::Event, [64](#)
 - ev3::Message, [82](#)
- EXPLORE_RANDOM
 - ev3::Behaviour, [28](#)
- empty
 - ev3::Message, [83](#)
- encodeMessage
 - ev3::Message, [83](#)
- 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< T >, 38
- ev3::ColorUtils, 40
- ev3::CommUtils, 52
 - _packetBuffer, 56
 - getBufferFromString, 53
 - getStringFromBuffer, 53
 - makeSockAddr, 54
 - preparePassiveSocket, 54
 - receiveMessage, 54
 - receiveMessageDelay, 55
 - sendBroadcastMessage, 55
 - sendMessage, 55
 - sendMessageTo, 56
- ev3::CommUtils::Buffer, 37
- ev3::CommUtils::NetworkNode, 89
- ev3::Command, 40
 - getString, 41
- ev3::CommandMotor, 41
 - CommandMotor, 42
 - getMotor, 42
- ev3::CommandMotorReset, 43
 - CommandMotorReset, 43
- ev3::CommandMotorRunForever, 44
 - CommandMotorRunForever, 44
- ev3::CommandMotorSetSpeed, 45
 - CommandMotorSetSpeed, 45
- ev3::CommandMotorSetSpeedRegEnabled, 46
 - CommandMotorSetSpeedRegEnabled, 46
- ev3::CommandMotorSetStopMode, 47
 - BRAKE, 48
 - COAST, 48
 - CommandMotorSetStopMode, 48
 - HOLD, 48
 - StopMode, 48
- ev3::CommandMotorStop, 48
 - CommandMotorStop, 49
- ev3::CommandSensor, 49
 - CommandSensor, 50
 - getSensor, 50
- ev3::Communication, 50
 - createThread, 51
 - run, 52
- ev3::Devices, 58
 - addListener, 61
 - checkDevices, 61
 - Devices, 60
 - getInstance, 61
 - getMotor, 61
 - getSensor, 61
 - operator=, 62
 - removeListener, 62
 - setProximitySensor, 62
 - setSafetyTouchSensor, 62
- ev3::Event, 63
 - ACTION_FINISHED, 64
 - ACTION_INTERR, 64
 - BEHAVIOUR_START, 64
 - BEHAVIOUR_STOP, 64
 - EMPTY, 64
 - Event, 64
 - EventType, 64
 - getStringType, 64
 - getType, 64
 - OBSTACLE_DETECTED, 64
 - PROXIMITY_ALERT, 64
 - SENSOR_WATCH, 64
- ev3::EventAction, 65
 - EventAction, 65
 - getActionType, 65
- ev3::EventQueue, 66
- ev3::EventSensorWatch, 67
 - EventSensorWatch, 67
 - getType, 68
 - getValue, 68
- ev3::LedControl, 73
 - ALL, 74
 - AMBER, 74
 - flash, 74
 - flashColor, 75
 - GREEN_ALL, 74
 - GREEN_L, 74
 - GREEN_R, 74
 - GREEN, 74
 - LedColors, 74
 - LedType, 74
 - off, 75
 - on, 75
 - onExclusive, 75
 - RED_ALL, 74
 - RED_L, 74
 - RED_R, 74
 - RED, 74
 - setColor, 75
 - YELLOW, 74
- ev3::Logger, 77
- ev3::Master, 78
 - createThread, 79
 - run, 79
 - send, 79
- ev3::Message, 80
 - ABORT, 82
 - ACTION_INTERR, 82
 - ACTION_OK, 82
 - ACK, 82
 - AGENT_OVER, 82
 - AGENT, 82
 - BEHAVIOUR, 82
 - decodeMessage, 83
 - EMPTY, 82
 - empty, 83
 - encodeMessage, 83
 - getMessageId, 83

- getParameters, [84](#)
- getReceiverId, [84](#)
- getSenderId, [84](#)
- getString, [84](#)
- getStringType, [84](#)
- getType, [84](#)
- MASTER_OVER, [82](#)
- MASTER, [82](#)
- MEASURE, [82](#)
- Message, [82](#)
- MessageType, [82](#)
- NOT, [82](#)
- PAUSE, [82](#)
- PING, [82](#)
- PONG, [82](#)
- RESUME, [82](#)
- SENSOR_VALUE, [82](#)
- START, [82](#)
- setMessageId, [85](#)
- setParameters, [85](#)
- setReceiverId, [85](#)
- setSenderId, [85](#)
- setType, [85](#)
- ev3::Motor, [86](#)
 - getMotor, [87](#)
 - Motor, [86](#)
- ev3::Queue< T >, [90](#)
- ev3::Robot, [92](#)
 - _behaviourSet, [96](#)
 - _pulsePerUnitRatio, [96](#)
 - createThread, [95](#)
 - generateBehaviour, [95](#)
 - getString, [95](#)
 - Robot, [94](#)
 - run, [95](#)
 - send, [96](#)
- ev3::RobotModelA, [96](#)
 - generateAction, [97](#)
 - generateBehaviour, [97](#)
 - getString, [98](#)
- ev3::RobotState, [98](#)
 - ACTIVE, [100](#)
 - changeState, [100](#)
 - getBehaviour, [100](#)
 - getPendingMessage, [101](#)
 - IDLE, [100](#)
 - isPendingEnabled, [101](#)
 - PANIC, [100](#)
 - PAUSED, [100](#)
 - process, [101](#)
 - RobotState, [100](#)
 - setBehaviour, [101](#)
 - States, [100](#)
 - switchState, [102](#)
 - WORKING, [100](#)
- ev3::RobotStateActive, [102](#)
 - process, [103](#)
 - RobotStateActive, [102](#)
- ev3::RobotStateIdle, [103](#)
 - process, [104](#)
 - RobotStateIdle, [103](#)
- ev3::RobotStatePanic, [104](#)
 - process, [105](#)
 - RobotStatePanic, [104](#)
- ev3::RobotStatePaused, [105](#)
 - process, [106](#)
 - RobotStatePaused, [105](#)
- ev3::RobotStateWorking, [106](#)
 - process, [107](#)
 - RobotStateWorking, [106](#)
- ev3::Sensor, [109](#)
 - COLOR, [109](#)
 - GYRO, [109](#)
 - getDecimals, [110](#)
 - getNumValues, [110](#)
 - getSensor, [110](#)
 - getType, [110](#)
 - getValue, [110](#)
 - getValueF, [110](#)
 - INFRARED, [109](#)
 - LIGHT, [109](#)
 - prepareMessage, [111](#)
 - SOUND, [109](#)
 - Sensor, [110](#)
 - SensorType, [109](#)
 - TOUCH, [109](#)
 - ULTRASONIC, [109](#)
- ev3::SignalHandler, [112](#)
- ev3dev::button, [37](#)
- ev3dev::button::file_descriptor, [68](#)
- ev3dev::color_sensor, [39](#)
- ev3dev::dc_motor, [56](#)
- ev3dev::device, [58](#)
- ev3dev::gyro_sensor, [68](#)
- ev3dev::i2c_sensor, [69](#)
- ev3dev::infrared_sensor, [70](#)
- ev3dev::large_motor, [70](#)
- ev3dev::lcd, [71](#)
- ev3dev::led, [71](#)
- ev3dev::lego_port, [76](#)
- ev3dev::light_sensor, [76](#)
- ev3dev::medium_motor, [80](#)
- ev3dev::motor, [87](#)
- ev3dev::power_supply, [90](#)
- ev3dev::remote_control, [91](#)
- ev3dev::sensor, [107](#)
- ev3dev::servo_motor, [111](#)
- ev3dev::sound, [113](#)
- ev3dev::sound_sensor, [113](#)
- ev3dev::touch_sensor, [114](#)
- ev3dev::ultrasonic_sensor, [114](#)
- Event
 - ev3::Event, [64](#)
- EventAction
 - ev3::EventAction, [65](#)
- EventSensorWatch

- ev3::EventSensorWatch, 67
- EventType
 - ev3::Event, 64
- execute
 - ev3::ActionRotateRandDirection, 22
- flash
 - ev3::LedControl, 74
- flashColor
 - ev3::LedControl, 75
- GREEN_ALL
 - ev3::LedControl, 74
- GREEN_L
 - ev3::LedControl, 74
- GREEN_R
 - ev3::LedControl, 74
- GREEN
 - ev3::LedControl, 74
- GYRO
 - ev3::Sensor, 109
- generateAction
 - ev3::RobotModelA, 97
- generateBehaviour
 - ev3::Robot, 95
 - ev3::RobotModelA, 97
- 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, 65
- getBehaviour
 - ev3::RobotState, 100
- getBufferFromString
 - ev3::CommUtils, 53
- getCommId
 - ev3::Agent, 25
- getDecimals
 - ev3::Sensor, 110
- getDistance
 - ev3::ActionDriveDistance, 15
- getId
 - ev3::Agent, 25
- getInstance
 - ev3::Devices, 61
- getMessageId
 - ev3::Message, 83
- getMotor
 - ev3::CommandMotor, 42
 - ev3::Devices, 61
 - ev3::Motor, 87
- getNumValues
 - ev3::Sensor, 110
- getParameters
 - ev3::Message, 84
- getPendingMessage
 - ev3::RobotState, 101
- getPrototype
 - ev3::Behaviour, 29
 - ev3::BehaviourDriveOnSquare, 32
 - ev3::BehaviourExploreRandom, 33
- getReaction
 - ev3::BehaviourState, 36
- getReceiverId
 - ev3::Message, 84
- getRotation
 - ev3::ActionRotate, 20
- getSenderId
 - ev3::Message, 84
- getSensor
 - ev3::CommandSensor, 50
 - ev3::Devices, 61
 - ev3::Sensor, 110
- 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, 41
 - ev3::Message, 84
 - ev3::Robot, 95
 - ev3::RobotModelA, 98
- getStringFromBuffer
 - ev3::CommUtils, 53
- getStringType
 - ev3::Event, 64
 - ev3::Message, 84
- getType
 - ev3::Action, 12
 - ev3::Event, 64
 - ev3::EventSensorWatch, 68
 - ev3::Message, 84
 - ev3::Sensor, 110
- getValue
 - ev3::EventSensorWatch, 68
 - ev3::Sensor, 110
- getValueF
 - ev3::Sensor, 110
- HOLD
 - ev3::CommandMotorSetStopMode, 48
- IDLE
 - ev3::RobotState, 100
- INFRARED
 - ev3::Sensor, 109

- isExecuted
 - ev3::Action, [12](#)
- isFinished
 - ev3::Action, [12](#)
- isForward
 - ev3::ActionDriveForever, [17](#)
- isPendingEnabled
 - ev3::RobotState, [101](#)
- isStopState
 - ev3::BehaviourState, [36](#)
- LIGHT
 - ev3::Sensor, [109](#)
- LedColors
 - ev3::LedControl, [74](#)
- LedType
 - ev3::LedControl, [74](#)
- MASTER_OVER
 - ev3::Message, [82](#)
- MASTER
 - ev3::Message, [82](#)
- MEASURE
 - ev3::Message, [82](#)
- makeSockAddr
 - ev3::CommUtils, [54](#)
- Message
 - ev3::Message, [82](#)
- MessageType
 - ev3::Message, [82](#)
- Motor
 - ev3::Motor, [86](#)
- NOP
 - ev3::Action, [11](#)
- NOT
 - ev3::Message, [82](#)
- OBSTACLE_DETECTED
 - ev3::Event, [64](#)
- off
 - ev3::LedControl, [75](#)
- on
 - ev3::LedControl, [75](#)
- onExclusive
 - ev3::LedControl, [75](#)
- operator=
 - ev3::Devices, [62](#)
- PANIC
 - ev3::RobotState, [100](#)
- PAUSED
 - ev3::RobotState, [100](#)
- PAUSE
 - ev3::Message, [82](#)
- PING
 - ev3::Message, [82](#)
- PONG
 - ev3::Message, [82](#)
- PROXIMITY_ALERT
 - ev3::Event, [64](#)
- prepareMessage
 - ev3::Sensor, [111](#)
- preparePassiveSocket
 - ev3::CommUtils, [54](#)
- process
 - ev3::BehaviourState, [36](#)
 - ev3::RobotState, [101](#)
 - ev3::RobotStateActive, [103](#)
 - ev3::RobotStateIdle, [104](#)
 - ev3::RobotStatePanic, [105](#)
 - ev3::RobotStatePaused, [106](#)
 - ev3::RobotStateWorking, [107](#)
- processMessage
 - ev3::Agent, [25](#)
- RED_ALL
 - ev3::LedControl, [74](#)
- RED_L
 - ev3::LedControl, [74](#)
- RED_R
 - ev3::LedControl, [74](#)
- REPEAT
 - ev3::Action, [11](#)
- RESUME
 - ev3::Message, [82](#)
- RED
 - ev3::LedControl, [74](#)
- ROTATE_RANDOM_DIR
 - ev3::Action, [11](#)
- ROTATE
 - ev3::Action, [11](#)
- react
 - ev3::Behaviour, [29](#)
- receiveMessage
 - ev3::CommUtils, [54](#)
- receiveMessageDelay
 - ev3::CommUtils, [55](#)
- removeListener
 - ev3::Devices, [62](#)
- Robot
 - ev3::Robot, [94](#)
- RobotState
 - ev3::RobotState, [100](#)
- RobotStateActive
 - ev3::RobotStateActive, [102](#)
- RobotStateIdle
 - ev3::RobotStateIdle, [103](#)
- RobotStatePanic
 - ev3::RobotStatePanic, [104](#)
- RobotStatePaused
 - ev3::RobotStatePaused, [105](#)
- RobotStateWorking
 - ev3::RobotStateWorking, [106](#)
- run
 - ev3::Communication, [52](#)
 - ev3::Master, [79](#)
 - ev3::Robot, [95](#)

SENSOR_VALUE
 ev3::Message, [82](#)

SENSOR_WATCH
 ev3::Event, [64](#)

SOUND
 ev3::Sensor, [109](#)

START
 ev3::Message, [82](#)

STOP
 ev3::Action, [11](#)

send
 ev3::Master, [79](#)
 ev3::Robot, [96](#)

sendBroadcastMessage
 ev3::CommUtils, [55](#)

sendMessage
 ev3::CommUtils, [55](#)

sendMessageTo
 ev3::CommUtils, [56](#)

Sensor
 ev3::Sensor, [110](#)

SensorType
 ev3::Sensor, [109](#)

setBehaviour
 ev3::Agent, [26](#)
 ev3::RobotState, [101](#)

setColor
 ev3::LedControl, [75](#)

setCommId
 ev3::Agent, [26](#)

setCommands
 ev3::Action, [12](#)

setEndCondition
 ev3::Action, [13](#)

setId
 ev3::Agent, [26](#)

setMeasurement
 ev3::Agent, [26](#)

setMeasurements
 ev3::Behaviour, [30](#)

setMessageId
 ev3::Message, [85](#)

setNextState
 ev3::BehaviourState, [36](#)

setParameters
 ev3::Message, [85](#)

setProximitySensor
 ev3::Devices, [62](#)

setReactionStates
 ev3::Behaviour, [30](#)

setReactions
 ev3::BehaviourState, [37](#)

setReceiverId
 ev3::Message, [85](#)

setSafetyTouchSensor
 ev3::Devices, [62](#)

setSenderId
 ev3::Message, [85](#)

setStates
 ev3::Behaviour, [30](#)

setStopState
 ev3::Behaviour, [30](#)

setType
 ev3::Message, [85](#)

States
 ev3::RobotState, [100](#)

StopMode
 ev3::CommandMotorSetStopMode, [48](#)

StoredActions
 Action.h, [118](#)

switchState
 ev3::RobotState, [102](#)

TOUCH
 ev3::Sensor, [109](#)

ULTRASONIC
 ev3::Sensor, [109](#)

updateLastMessage
 ev3::Agent, [26](#)

WORKING
 ev3::RobotState, [100](#)

YELLOW
 ev3::LedControl, [74](#)