leJOS

For motors / actuators the port has to be one of A, B, C.

For sensors, the port has to be on of S1, S2, S3, S4.

Snippet Name and Parameters	Description	Use on	Variable type	Remark	Example
Clock, milliseconds	Sets a variable to true for one tick if the time in milliseconds passed since the last time it was set to true.	input	bool	See also ResetClock.	input bool @macro "Clock", "1000" second
ResetClock, clockVariableName, autoFalse	Resets a clock, such that the full time interval of the clock has to elapse, before the clock will be set to true again. If autoFalse is true, the reset variable will be set to false automatically.	output	bool	autoFalse is true per default.	output bool @macro "ResetClock", "second", "true" reset
Time	Reads the elapsed time since program start (milliseconds)	input	int		input int @macro "Time" timeInMs
TickLoopDuration, targetInMilliseconds	Delays the execution until the tick loop takes at least as long as the given target duration. The input variable is set to the actual tick loop duration.	input	int	Should be used on the very first input variable in the model, such that waiting is the last action in the tick loop. In case the actual tick loop duration is longer than the target duration, the modeler can provide some error handling.	input int @macro "TickLoopDuration", "1000000" duration
TickWakeUp	Sets the input variable to the current system time (milliseconds). The model can add to this variable to get a new value. This is the next system time the tick function will be called. In other words, the next tick function call is delayed until the wake up time has been reached. For instance the statement **nextTickWakeUp += 500** could be used to call the tick function again in 500 milliseconds, if nextTickWakeUp is an input with the corresponding annotation.	input	int	Should be used on the very last input variable in the model, such that waiting and settings the system time is the last action done, before the tick function call.	input int @macro "TickWakeUp" nextTickWakeUp
TickCount	Counts the ticks. First tick is 0, the following are 1, 2, 3, ...	input	int		input int @macro "TickCount" ticks
Sleep	Lets the current thread sleep the time in milliseconds of the variable value.	output	int		output int @macro "Sleep" sleep
Print, autoReset	Prints a string variable if the string is not empty. If autoReset is true then the string variable is set to the empty string after it has been printed	output	string	autoReset is true per default.	output string @macro "Print", "true" text
DrawString, x, y	Prints a string to the given x and y coordinate on the LCD.	output	string		output string @macro "DrawString", "1", "1" text
DrawInt, x, y	Prints a interger to the given x and y coordinate on the LCD.	outptu	int		output int @macro "DrawInt", "1", "1" number
Button, button Id	Sets a variable to true iff the button on the Mindstorms device is pressed.	input	bool	The buttonId has to be one of ENTER, LEFT, RIGHT	input bool @macro "Button", "ENTER" enterButtonDown
TouchSensor, port	Sets a variable to true iff the touch sensor on the given port is pressed.	input	bool		input bool @macro "TouchSensor", "S4" touch

LightSensor, port, percentValue	Reads the value of a light sensor. If percentValue is true, the a percent value is returned, based on the light sensor calibration.	input	int	percentValue is not available on EV3	input int @macro "LightSensor", "S1" lightLevel
CalibrateLightSensor, port, signal	Calibrates a light sensors high or low values. This means if the variable is true, the current value of the light sensor is taken as its reference high / low value. Note that this should be used together with the LightSensor, because this macro introduces the actual variable.	output	bool	signal has to be one of High, Low	output bool @macro "CalibrateLightSensor", "S1", "Low" calibrateLow
Floodlight, port	Sets the state of the red lamp of the light sensor.	output	bool		output bool @macro "Floodlight", "S1" setFloodlight
GetFloodlight, port	Reads the state of the red lamp of the light sensor.	input	bool		input bool @macro "GetFloodlight", "S1" getFloodlight
RCXLamp, port	Turns an RCX lamp on (variable is true) or off (variable is false)	output	bool		output bool @macro "RCXLamp", "A" lamp
MotorSpeed, port, brake	Sets the speed of the motor in degrees per second. Maximum is 900. If the speed value is negative, the motor will drive backwards. If the speed is zero, the motor will actively brake until it stops (brake is true) or remove all power and rollout (brake is false).	output	int	brake is true per default.	output int @macro "MotorSpeed", "A", "false" motorA
GetMotorSpeed, port	Allows to get the current motor speed in degrees.	input	int		input int @macro "GetMotorSpeed", "A" getMotorA
MotorIsMoving, port	Sets a variable to true iff the motor on the given port is moving.	input	bool		input bool @macro "MotorIsMoving", "A" isMotorAMoving
MotorRotation, port	Lets a motor rotate the variable value in degrees. This is only done if the value is unequal zero. If the value is negative, the motor rotates backwards. The variable is set to zero afterwards, such that setting the variable once to a value *X*, will let the motor rotate *X* degrees.	output	int		output int @macro "MotorRotation", "A" rotateInDegrees
GetMotorRotation, port	Allows to get the current motor rotation in degress.	input	int		input int @macro "GetMotorRotation", "A" rotationA
MotorRotationSpeed, port	Sets the speed of a motor without rotation it. This speed will be used by MotorRotation.	output	int		output int @macro "MotorRotationSpeed", "A" rotationSpeed
Beep, volume	Plays a beep sound as long as the variable is true.	output	bool	default volume is 10	output bool @macro "Beep", "10" beep
Buzz, volume	Plays a buzz sound as long as the variable is true.	output	bool	default volume is 10	output bool @macro "Buzz", "10" buzz
BeepSequence, direction, volume	Plays a sequence of tones in either ascending or descending tone frequency if the variable is true. The variable is set to false automatically.	output	bool	direction has to be one of Up, Down default volume is 10	output bool @macro "BeepSequence", "10" beepSequence

PlayTone, frequency, duration, volume	Plays the tone specified by frequency (e.g. 440 is an A), duration and volume. Duration and volume are optional and will be set to 100 if not defined.	output	bool	Default duration and volume is 100	ouput bool @macro "PlayTone", "440", "100", "10" A4
PlayFrequency, duration, volume	Plays the tone specified by frequency (the value of the varibale e.g. 440 is an A), duration and volume. Duration and volume are optional and will be set to 100 if not defined. Used frequency will be used as default frequency for next tone	output	int	Default duration and volume is 100	ouput int @macro "PlayFrequency", "100" A4
Volume	Sets the volume for sound output.	output	int		ouput int @macro "Volume" volume
UltrasonicSensor, port	Reads the distance that an ultrasonic sensor measures.	input	int		input int @macro "UltrasonicSensor", "S3" ultraSonic
Gyro, port, mode	Reads the value of a gyroscope.	input	int	Not available on NXT mode hat to be one of Angle, Rate	input int @macro "Gyro", "S3", "Angle" gyro
CalibrateGyro, port, autoReset	Resets a gyroscope if the variable is true. If autoReset is true, the variable is set to false automatically.	output	bool	autoReset is true per default	output bool @macro "CalibrateGyro", "S3", "true" calibrateGyro