Overview

ARC provides the JavaScript programming language for custom code that interconnects skills. The ARC JavaScript API provides access to standard ECMA 5.1 Script specifications and a library of additional functions. The other functions are defined in this section.

JavaScript is not JAVA. JavaScript, often abbreviated as JS, is a programming language that conforms to the ECMA 5.1 Script specification. JavaScript is high-level, just-in-time compiled, and multi-paradigm. It has curly-bracket syntax, dynamic typing, prototype-based object orientation, and first-class functions. JavaScript engines were initially used only in web browsers, but they are now core components of other software systems, such as Synthiam ARC. Although there are similarities between JavaScript and Java, including language name, syntax, and respective standard libraries, the two languages are distinct and differ significantly in design.

The ARC JavaScript engine exposes several objects with methods for interacting with hardware, the framework, and other robot skills. Information on creating global user-defined custom objects and methods is also provided at the end of this document.

ARC JavaScript Functions

In addition to the built-in ECMA 5.1 JavaScript classes (outlined below), ARC includes several types with methods to control your robot, read sensors, interact with the operating system, and more. To view built-in ARC JavaScript methodsâ€¦

1) Check the intellisense checkbox.
2) Type one of the following class names, press â€˜.â€™, and scroll through the available methods.

ADC â€“ EZB analog to digital converter functions

Audio â€“ streaming audio functions

COM â€“ local serial COM port functions

Digital â€“ EZB digital I/O functions

EZB â€“ EZB-specific functions for getting voltage, checking the connection, etc.

File â€“ writing and reading local files

I2C â€“ EZB I2C helper functions

Movement â€“ movement panel control of directions

Net â€“ networking functions

Ping â€“ EZB ultrasonic distance sensor functions

PWM â€“ EZB pulse width modulation on digital I/O pins

Servo â€“ EZB servo movement functions

UART â€“ EZB hardware UART and software Serial functions

Utility â€“ various utility functions

Supported JavaScript ECMA 5.1 Classes

The JavaScript ECMA 5.1 includes several classes for working with strings, math, and various features. For detailed information and examples, the following classes can be researched online at developer.mozilla.org and w3schools.com.

- Array
- Boolean
- Console (support for time, count, error, info, log)
- Date
- Error
- Iterator
- JSON
- Map
- Math
- Number
- Object
- Proxy
- Reflect
- RegExp
- Set
- String
- Symbol

Root Commands

Root commands do not belong to a class and are included for convenience to access global variables and send commands to other robot skills.

getVar(variableName, [default value])

Retrieves the value from the Arcâ€™s public global variable storage. These are variables published by robot skills, such as the Camera, Auto Position, Speech Recognition, etc... (Read More)

{*variableName*} - The name of the global variable as a string
{default value} â€“ [Optional] If specified, this value is returned if the global variable doesnâ€™t exist.
{*return*} â€“ The value of the global variable

Example:

```
// Get the current direction the robot is moving
var direction = getVar(â€œ$Directionâ€);

// Get the value of $test, and if it doesnâ€™t exist, return false
var testVar = getVar(â€œ$testâ€, false);
```

setVar(variableName, value)

Sets the value from Arcâ€™s public global variable storage. This allows the variable to be available to other robot skills or scripts using getVar(). (Read More)

{*variableName*} - The name of the global variable as a string
{*value*} â€“ The value that you wish to store in the global variable

Example:

```
// Set a value of 5 to be accessible by other robot skills
setVar(â€œ$MyValueâ€, 5);
```

setVarObject(variableName, value)

Sets the value from Arcâ€™s public global variable storage of the object. This allows the variable to be available to other robot skills or scripts using getVar(). (Read More). This differs from setVar() because it sets the value as an object without any translation. So you can pass entire objects, such as functions, classes, or multidimensional arrays.

{*variableName*} - The name of the global variable as a string
{*value*} â€“ The object value that you wish to store in the global variable

Example:

```
// Set the multidimensional array to be accessible by other robot skills with getVar()
x = [ [0, 1, 2], [3, 4, 5] ];
setVar(â€œ$MyValueâ€, x);
```

varExists(variableName)

Check if the global variable exists. (Read More)

{*variableName*} - The name of the global variable as a string
{*return*} â€“ true/false Boolean if the variable exists

Example:

```
// print if the variable exists
print(varExists(â€œ$MyValueâ€));
```

print(txt)

Writes the value to the console output.

{*txt*} â€“ The value/data to print

Example:

```
// Print a string to the console output
print(â€œHello Worldâ€);
```

sleep(timeMs)

Pauses the execution of the program for the specified time in milliseconds

{*timeMS*} â€“ The time to pause

Example:

```
// Print a string, pause for 5 seconds, and print another string
print(â€œHelloâ€);
sleep(5000);
print(â€œ Worldâ€);
```

sleepRandom(minTimeMS, maxTimeMS)

Pauses the program's execution for random milliseconds between the min and max.

{*minTimeMS*} â€“ The minimum time to pause
{*maxTimeMS*} â€“ The maximum time to pause

Example:

```
// Print a string, pause for a random time, and print another string
print(â€œHelloâ€);
sleepRandom(1000, 5000);
print(â€œ Worldâ€);
```

controlCommand(control name, command, [parameters])

Sends the command to the control name with the optional parameters. The cheat sheet displays all of the interrogated control commands supported by roobt skills in the project.

Example:

```
// Start the camera
controlCommand(â€œCameraâ€, â€œStartCameraâ€);
print(â€œCamera has startedâ€);
```

showControl(controlName)

Shows the user interface control. This also adds the control to the stack, which can be

Example:

```
// Start the camera
controlCommand(â€œCameraâ€, â€œStartCameraâ€);
print(â€œCamera has startedâ€);
```

closeControl(controlName)

Closed the user interface control and displayed the previous control in the display stack.

Example:

```
CloseControl();
```

Variables

JavaScript variables are private to the namespace of each script engine. That means a variable created in a JavaScript script does not exist in other JavaScript scripts. For example, if you have variables declared in the JavaScript of the Camera control, the variables are not accessible in the JavaScript of the WiiMote control. However, you can share variables across scripts using global variable storage.

Public variables use ARCâ€™s global variable storage manager, which is global across all robot skills and compilers. Public variables are accessed using the getVar(), setVar(), and setVarObject() JavaScript commands.

Blockly generates JavaScript, which means the variables will be private by default. The variables can start with a $ (dollar sign) to be public.

JavaScript Constants

A constant is a global variable available throughout the JavaScript environment within ARC. The variable value cannot be changed and is declared behind the scenes when your code is executed. These constant variables are also unique because they do not need to be surrounded by "quotes" when referenced. This is because their actual value is a number. Numbers (such as integers and decimal floats) do not need to be surrounded by quotes (although you may surround them with quotes if you prefer). Only string values need to be wrapped in quotes.

D0 ... D23
V0 ... V99
ADC0 ... ADC7

Custom JavaScript Extension

The ARC JavaScript engine exposes the ability to extend the built-in objects with user-defined custom objects. These custom objects can be either methods, variables, or entire classes. An example of doing this is in the Create Robot Skill manual. Access it by Clicking Here.

.Net CLR

For more advanced users, the underlying .Net CLR is available to the ARC JavaScript engine. This means you can access the entire underlying ARC platform and operating system from the JavaScript engine.

What Is CLR?

The .NET Framework provides a run-time environment, called the common language runtime, that runs the code and provides services that make the development process easier. Compilers and tools expose the common language runtime's functionality and enable you to write code that benefits from this managed execution environment. Code that you develop with a language compiler that targets the runtime is called managed code. Managed code benefits from features such as cross-language integration, cross-language exception handling, enhanced security, versioning and deployment support, a simplified model for component interaction, and debugging and profiling services.

.NET Framework Example

This example will use the System.IO.StreamWriter to write some text to a log file on the C:\. The ARC JavaScript engine already does provide file writing capabilities, this is used as an example to demonstrate accessing the raw CLR commands and namespaces directly.

```
// Create an instance of the stream writer
var file = new System.IO.StreamWriter('c:\\log.txt');

// Write some text to the file
file.WriteLine('Hello World !');

// Close the file and dispose the object
file.Dispose();
```

EZ_B Driver Example

The ARC JavaScript engine exposes the raw EZ_B driver for each EZB connection. This is the same as the ARC.EZBManager.EZBs array when creating a robot skill.

```
// Move the servo on port D2 to position 20 on EZB #0
EZBs[0].Servo.SetServoPosition(2, 20);
```

Exception Handling

When executing JavaScript code, different errors can occur. JavaScript will normally stop and generate an error message when an error occurs. These can be coding errors made by the programmer, errors due to wrong input, and other unforeseeable things such as ControlCommand() exceptions. If executing a ControlCommand() to instruct another robot skill and it fails, a Try Catch can handle the exception error.

Throw, and Try...Catch...Finally

- The *try* statement defines a code block to run (to try). This allows you to define a block of code to be tested for errors while executing it.
- The *catch* statement defines a code block to handle any error. This defines a block of code to be executed if an error occurs in the try block.
- The *finally* statement defines a code block to run regardless of the result. This statement lets you execute code after try and catch, regardless of the result:
- The *throw* statement defines a custom error.

The Error Object

JavaScript has a built-in error object that provides error information when an error occurs. The error object provides two useful properties: name and message.

Error Object Properties

Property	Description
name	Sets or returns an error name
message	Sets or returns an error message (a string)

Error Name Values

The error name property can return six different values:

Error Name	Description
EvalError	An error has occurred in the eval() function
RangeError	A number "out of range" has occurred
ReferenceError	An illegal reference has occurred
SyntaxError	A syntax error has occurred
TypeError	A type error has occurred
URIError	An error in encodeURI() has occurred

Example #1

This example forces an error using the *throw* statement.

```
try {

  throw "this is an error";
} catch(err) {

  print(err);
}
```

Examples

Parse JSON

The JSON javascript class is included in the ARC javascript compiler. You can parse the JSON in javascript directly in ARC.

Here's an example parsing JSON from a string...

Code:

```
var resp = '{ "glossary": { "title": "example glossary" } }';
var myObj = JSON.parse(resp);
print (myObj.glossary.title);
```

And here's an example of getting JSON from an HTTP get request...

Code:

```
var resp = Net.hTTPGet("http://ip.jsontest.com/");
var myObj = JSON.parse(resp);
print (myObj.ip);
```

ADC

get12Bit

```
ADC.get12Bit(port, [ezbIndex])
```

Parameters

port	Analog port to read from.
ezbIndex (optional)	Board index of the EZB to read from.

Returns

The value read from the specified ADC port as a 12 bit number.

Description

Returns the value read from the specified ADC port as a 12 bit number which ranges between 0 and 4095.

waitForBetween

```
ADC.waitForBetween(port, low, high, [frequencyMs], [ezbIndex], [timeoutMS])
```

Parameters

port	Analog port to read from.
low	Inclusive lower bound on value to wait for.
high	Inclusive upper bound on value to wait for.
frequencyMs (optional)	How often the ADC port is checked in milliseconds.
ezbIndex (optional)	Board index of the EZB to read from.
timeoutMS (optional)	Number of milliseconds to wait before timeout.

Returns

The value read from the specified ADC port that is between or low and high. Returns -1 if timeout.

Description

Suspends execution of the script until the value read from the specified ADC port is between low (inclusive) and high (inclusive). The value read from the ADC port ranges between 0 and 255. If frequencyMs is provided the port is checked every frequencyMs milliseconds. Otherwise the port is checked as often as possible.

waitForEquals

```
ADC.waitForEquals(port, value, [frequencyMs], [ezbIndex], [timeoutMS])
```

Parameters

port	Analog port to read from.
value	Value to wait for.
frequencyMs (optional)	How often the port is checked in milliseconds.
ezbIndex (optional)	Board index of the EZB to read from.
timeoutMS (optional)	Number of milliseconds to wait before timeout.

Returns

The value read from the specified ADC port that is equal to value. Returns -1 if timeout.

Description

Suspends execution of the script until the value read from the specified ADC port is equal to value. The value read from the ADC port ranges between 0 and 255. If frequencyMs is provided the port is checked every frequencyMs milliseconds. Otherwise the port is checked as often as possible.

waitForHigher

```
ADC.waitForHigher(port, value, [frequencyMs], [ezbIndex], [timeoutMS])
```

Parameters

port	Analog port to read from.
value	Exclusive lower bound on value to wait for.
frequencyMs (optional)	How often the port is checked in milliseconds.
ezbIndex (optional)	Board index of the EZB to read from.
timeoutMS (optional)	Number of milliseconds to wait before timeout.

Returns

The value read from the specified ADC port that higher than value. Returns -1 if timeout

Description

Suspends execution of the script until the value read from the specified ADC port is greater than value. The value read from the ADC port ranges between 0 and 255. If frequencyMs is provided the port is checked every frequencyMs milliseconds. Otherwise the port is checked as

often as possible.

waitForLower

```
ADC.waitForLower(port, value, [frequencyMs], [ezbIndex], [timeoutMS])
```

Parameters

port	Analog port to read from.
value	Exclusive upper bound on value to wait for.
frequencyMs (optional)	How often the port is checked in milliseconds.
ezbIndex (optional)	Board index of the EZB to read from.
timeoutMS (optional)	Number of milliseconds to wait before timeout.

Returns

The value read from the specified ADC port that is lower than value. Returns -1 if timeout

Description

Suspends execution of the script until the value read from the specified ADC port is lower than value. The value read from the ADC port ranges between 0 and 255. If frequencyMs is provided the port is checked every frequencyMs milliseconds. Otherwise the port is checked as often as possible.

get

```
ADC.get(port, [ezbIndex])
```

Parameters

port	Analog port to read from.
ezbIndex (optional)	Board index of the EZB to read from.

Returns

The value read from the specified ADC port.

Description

Returns the value read from the specified ADC port. The value read from the ADC port ranges between 0 and 255.

Audio

getVolume

```
Audio.getVolume([ezbIndex])
```

Parameters

ezbIndex (optional)	Board index of the EZB to get volume level from.

Returns

The volume level of the specified EZB.

Description

Returns the volume level of the specified EZB. The volume level ranges between 0 (quite), 100 (loud), 200 (2x over drive).

isConnected

```
Audio.isConnected([ezbIndex])
```

Parameters

ezbIndex (optional)	Board index of the EZB to check if audio is connected.

Returns

True if audio is connected to the EZB. False otherwise.

Description

Returns true if audio is connected to the EZB. Returns false otherwise.

playAudioFile

```
Audio.playAudioFile(filename)
```

Parameters

filename	Filename of the audio file to play.

Returns

Nothing

Description

Plays the audio file at the location given by filename through the computerâ€™s audio system.

say

```
Audio.say(txt)
```

Parameters

txt	Text to say.

Returns

Nothing

Description

Speak the text given in the string txt using text-to-speech through the computerâ€™s audio system. Execution of the script will continue while the audio is playing.

saySSMLEZB

```
Audio.saySSMLEZB(ssml, [ezbIndex])
```

Parameters

ssml	Text to say.
ezbIndex (optional)	Board index of the EZB to output the audio from.

Returns

Nothing

Description

Speak the SSML document given in the string using text-to-speech through the EZBâ€™s audio system. Execution of the script will continue while the audio is playing.

Read about SSML in this support document here: https://www.w3.org/standards/history/speech-synthesis/

Example

```
// This is an example using the say-as to specify an input format and speak the date
```

```
var str = "<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">" +
          "<say-as type="date:mdy"> 1/29/2009 </say-as>" +
          "</speak>";

Audio.saySSMLEZB(str);

// This example includes a pause of 500ms between the two words hello and there
var str = "<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">" +
          "hello <break time="500ms"> there" +
          "</break></speak>";

Audio.saySSMLEZB(str);
```

saySSMLEZBWait

```
Audio.saySSMLEZBWait(ssml, [ezbIndex])
```

Parameters

ssml	Text to say.
ezbIndex (optional)	Board index of the EZB to output the audio from.

Returns

Nothing

Description

Speak the SSML document given in the string using text-to-speech through the EZBâ€™s audio system. Execution of the script will be suspended until the audio is finished playing.

Read about SSML formatted speech here: https://www.w3.org/standards/history/speech-synthesis/

Example

```
// This is an example using the say-as to specify an input format and speak the date
var str = "<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">" +
          "<say-as type="date:mdy"> 1/29/2009 </say-as>" +
          "</speak>";

Audio.saySSMLEZBWait(str);

// This example includes a pause of 500ms between the two words hello and there
var str = "<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">" +
          "hello <break time="500ms"> there" +
          "</break></speak>";

Audio.saySSMLEZBWait(str);
```

saySSMLWait

```
Audio.saySSMLWait(ssml)
```

Parameters

ssml	Text to say.

Returns

Nothing

Description

Speak the text given in the provided SSML using text-to-speech through the computerâ€™s audio system. Execution of the script will be suspended until the audio is finished playing.

Read about how to format SSML in this document here: https://www.w3.org/standards/history/speech-synthesis/

Example

```
// This is an example using the say-as to specify an input format and speak the date
var str = "<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">" +
          "<say-as type="date:mdy"> 1/29/2009 </say-as>" +
          "</speak>";

Audio.saySSMLWait(str);

// This example includes a pause of 500ms between the two words hello and there
var str = "<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">" +
          "hello <break time="500ms"> there" +
          "</break></speak>";

Audio.saySSMLWait(str);
```

setVolume

```
Audio.setVolume(volume, [ezbIndex])
```

Parameters

volume	Level to set the volume to.
ezbIndex (optional)	Board index of the EZB to set the volume level for.

Returns

Nothing

Description

Set the volume level of the EZB to volume. The volume level ranges between 0 (quite), 100 (loud), 200 (2x over drive).

soundNote

```
Audio.soundNote(note, length, [signalType], [ezbIndex])
```

Parameters

note	Note to play as a string.
length	How long to play the note for in milliseconds.
signalType (optional)	Type of signal to use when playing the note as a string.
ezbIndex (optional)	Board index of the EZB to output the audio from.

Returns

Nothing

Description

Plays the specified note for length milliseconds. Notes are inputted as strings which range from "C1" to "Bb3". The note is played using the signal specified by signalType. If signalType is not provided a sine wave is used. Accepted signal types are "Sine", "Square", "Triangle", "Pulse", "Sawtooth", "WhiteNoise", "GaussNoise", "DigitalNoise". This function does not wait for the note to end to continue executing the script. In case of playing multiple notes in sequence, call the sleep() function to ensure the notes don't play over previous notes.

speakVolume

```
Audio.speakVolume(volume)
```

Parameters

volume	Level to set the text-to-speech volume to.

Returns

Nothing

Description

Sets the text-to-speech volume level of the computer to volume. The volume level ranges between 0 and 100.

stop

```
Audio.stop([ezbIndex])
```

Parameters

ezbIndex (optional)	Board index of the EZB to stop playing audio.

Returns

Nothing

Description

Stops all audio currently playing on the EZB.

stopAudioFile

```
Audio.stopFile()
```

Returns

Nothing

Description

Stops the playback of the audio file started by playAudioFile(filename) that is playing through the computerâ€™s audio system.

waitForSpeech

```
Audio.waitForSpeech(timeout, s1, ...)
```

Parameters

timeout	Time to wait in seconds before stopping waiting for speech.
s1...	Strings to wait for that are detected in the microphone.

Returns

The phrase heard if the timeout did not occur in a lowercase string.

Description

Suspends execution of the script until one of the strings specified by s1,.... is detected in the microphone, or a timeout occurs after timeout

seconds. If a timeout occurs, the method will return "timeout".

***Note: the returned string is in lowercase**

Example

```
// Specify the words in the function
response = Audio.waitForSpeech(10, "One", "two", "Three");

print("Heard: " + response);

// Use an array of words to listen for
var words = ["One", "two", "Three"];

response = Audio.waitForSpeech(10, words);

print("Heard: " + response);
```

setVolumePC

```
Audio.setVolumePC(volume)
```

Parameters

volume	Level to set the PC speaker volume to between 0-255

Returns

Nothing

Description

Set the volume level of the default audio output device (Soundcard). The volume level ranges between 0 and 255. With 0 (mute) and 255 is loudest

waitForSpeechRange

```
Audio.waitForSpeechRange(timeout, start, end, [increment])
```

Parameters

timeout	Time to wait in seconds before stopping waiting for speech.
start	numeric start value of the range
end	numeric end value of the range
increment	the increment of the range (optional)

Returns

The value within the range as a number or the word "timeout" if timeout is reached.

Description

Suspends execution of the script until one of the numbers within the range (start, end) is detected in the microphone by speech, or a timeout occurs after timeout seconds. If a timeout occurs, the method will return "timeout." The returned value is a number (i.e., 23, 10, 5) and not a string (i.e., twenty-three, ten, five).

Each option will be displayed if the number of options within the range is less than 10. For example, if the start is 10 and the end is 20, there will be 10 options for each option to be displayed.

If there are more than ten options, the range will be displayed.

Large Number Ranges

This style of speech recognition populates the pre-defined responses it expects to hear. In this case, a range of numbers. This dramatically increases the accuracy of the speech recognition system by limiting the detected phrase/words to the list. By providing the list, the speech recognition system will generate definitions of the sound waveforms it is looking for.

While this method dramatically improves detection accuracy, it also can consume a lot of PC resources if an extensive number range is used. For example, a range higher than 100 is considered an extensive range. You may notice a significant delay every time this function is called, and that is due to the vast range.

A recommended solution is to use the waitForAnyNumberSpeech() command.

An additional solution is to prompt for multiple digits that will generate a more significant number when added together with simple math. For example, if you require 1000 positions, consider prompting for each digit.

Example

```
// Get a number from the user within the range between 10 and 20
response = Audio.waitForSpeechRange(10, 10, 20);
Audio.say("You selected " + response);

// Get a number from the user within the range between 10 and 20 incremented by 2 (every even number)
response = Audio.waitForSpeechRange(10, 10, 20, 2);
Audio.say("You selected " + response);
```

waitForAnySpeech

```
Audio.waitForAnySpeech(timeout, prompt)
```

Parameters

timeout	Time to wait in seconds before stopping waiting for speech.
prompt	The text prompt to display to the user while listening for speech

Returns

The phrase heard if the timeout did not occur in a lowercase string.

Description

Suspends execution of the script until any speech is detected in the microphone or a timeout occurs after timeout seconds. If a timeout occurs, the method will return "timeout."

***Note: the returned string is in lowercase**

This command uses the entire dictionary of speech recognition words. Generally, using this dictionary size will have a very low recognition quality. This may work okay if you are feeding a chatbot (such as OpenAI or PandoraBot). But, if you are looking for specific phrases, this will most likely not work, and you should use the waitForSpeech() command instead. Using this command for open conversations is discouraged.

Example

```
// Listen for any response from the user
response = Audio.waitForAnySpeech(30, "Say anything to me");

print("Heard: " + response);
```

waitForAnyNumberSpeech

```
Audio.waitForAnyNumberSpeech(timeout, prompt)
```

Parameters

timeout	Time to wait in seconds before stopping waiting for speech.
prompt	The text prompt to display to the user while listening for speech

Returns

The number heard if the timeout did not occur in a lowercase string.

Description

Suspends execution of the script until any number is detected in the microphone or a timeout occurs after timeout seconds. If a timeout occurs, the method will return "timeout" in lowercase. The number can include a minus sign and a decimal point.

Example

```
// Listen for any response from the user
response = Audio.waitForAnyNumberSpeech(30, "Give me a number");

print("Heard: " + response);
```

saySSML

```
Audio.saySSML(ssml)
```

Parameters

ssml	Text to say.

Returns

Nothing

Description

> Speak the formatted SSML given using text-to-speech through the computerâ€™s audio system. Execution of the script will continue while the audio is playing.

This support document explains SSML here: https://www.w3.org/standards/history/speech-synthesis/

Example

```
// This is an example using the say-as to specify an input format and speak the date
var str = "<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">" +
          "<say-as type="date:mdy"> 1/29/2009 </say-as>" +
          "</speak>";

Audio.saySSML(str);

// This example includes a pause of 500ms between the two words hello and there
var str = "<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">" +
          "hello <break time="500ms"> there" +
          "</break></speak>";

Audio.saySSML(str);
```

sayEZB

```
Audio.sayEZB(txt, [ezbIndex])
```

Parameters

txt	Text to say.

ezbIndex (optional)	Board index of the EZB to output the audio from.

Returns

Nothing

Description

Speak the text given in the string txt using text-to-speech through the EZBâ€™s audio system. Execution of the script will continue while the audio is playing.

sayEZBWait

```
Audio.sayEZBWait(txt, [ezbIndex])
```

Parameters

txt	Text to say.
ezbIndex(optional)	Board index of the EZB to output the audio from.

Returns

Nothing

Description

Speak the text given in the string txt using text-to-speech through the EZBâ€™s audio system. Execution of the script will be suspended until the audio is finished playing.

COM

available

`COM.available(port)`

Parameters

port	COM port name as a string.

Returns

True if the specified COM port is available. False otherwise.

Description

Returns true if the specified COM port is available. Returns false otherwise. port is provided as a string (e.g. â€œCOM3â€).

availablePorts

`COM.availablePorts()`

Returns

Array of available COM ports.

Description

Returns an array of available COM ports. Available COM ports are represented in the array as strings (e.g. â€œCOM3â€).

clearInputBuffer

`COM.clearInputBuffer(port)`

Parameters

port	COM port name as a string.

Returns

Nothing

Description

Clears all data from the input buffer of the specified COM port.

close

`COM.close(port)`

Parameters

port	COM port name as a string.

Returns

Nothing

Description

Closes the specified COM port.

isPortOpen

`COM.isPortOpen(port)`

Parameters

port	COM port name as a string.

Returns

True if the COM port is open. False otherwise.

Description

Returns true if the specified COM port is open. Returns false otherwise.

open

`COM.open(port, baud)`

Parameters

port	COM port name as a string.
baud	Baud rate of the port

Returns

Nothing

Description

Opens the specified COM port with a baud rate of baud. This function must be called before calling any other read or write functions to the COM port.

readAllText

`COM.readAllText(port)`

Parameters

port	COM port name as a string.

Returns

All data in the COM port as a string.

Description

Reads all data from the specified COM port and returns it as a string.

readByte

`COM.readByte(port)`

Parameters

port	COM port name as a string.

Returns

The first byte read from the COM port.

Description

Reads and returns one byte from the specified COM port.

readLine

`COM.readLine(port)`

Parameters

port	COM port name as a string.

Returns

One line of data from the COM port as a string.

Description

Reads all data from the specified COM port until an endline character is reached. Value is returned as a string.

write

`COM.write(port, byte/byteArray)`

Parameters

port	COM port name as a string.
byte/byteArray	Byte or byte array to write to the COM port.

Returns

Nothing

Description

Writes one byte given by byte to the specified COM port. If a byte array is provided instead, the byte array given by byteArray will be written to the specified COM port.

writeString

`COM.writeString(port, str)`

Parameters

port	COM port name as a string.
str	String to write to the COM port.

Returns

Nothing

Description

Writes the string given by str to the specified COM port.

writeStringNewLine

```
COM.writeStringNewLine(port, str)
```

Parameters

port	COM port name as a string.
str	String to write to the COM port.

Description

Writes the string given by str with a newline character appended to the end to the specified COM port.

read

```
COM.read(port, bytesToRead)
```

Parameters

port	COM port name as a string.
bytesToRead	Number of bytes to read from the COM port.

Returns

A byte array of length bytesToRead containing the data read from the COM port.

Description

Reads bytesToRead bytes from the COM port and returns it in a byte array.

Digital

get

```
Digital.get(port, [ezbIndex])
```

Parameters

port	Digital port to read from.
ezbIndex (optional)	Board index of the EZB to read from.

Returns

True if the specified digital port is 1 (high). False if the digital port is 0 (low).

Description

Reads and returns the digital value from the specified digital port.

set

```
Digital.set(port, value, [ezbIndex])
```

Parameters

port	Digital port to set the value of.
value	Value to set the digital port to as a Boolean or integer.
ezbIndex (optional)	Board index of the EZB to use.

Returns

Nothing

Description

Sets the value of the specified digital port to value.

setRandom

```
Digital.setRandom(port, [ezbIndex])
```

Parameters

port	Digital port to set the value of.
ezbIndex (optional)	Board index of the EZB to use.

Returns

The random value that the digital port was set to.

Description

Randomly sets the value of the specified digital port to either 0 or 1.

toggle

```
Digital.toggle(port, [ezbIndex])
```

Parameters

port	Digital port to toggle.
ezbIndex (optional)	Board index of the EZB to use.

Returns

The value toggled to.

Description

Toggles the digital port. If the digital port was 0 then it is set to 1. If the digital port was 1 then it is set to 0.

wait

```
Digital.wait(port, valueToWaitFor, [frequencyMs], [ezbIndex], [timeoutMS])
```

Parameters

port	Digital port to set the value of.
valueToWaitFor	Digital value to wait for as boolean or integer.
frequencyMs (optional)	How often the port is checked in milliseconds.
ezbIndex (optional)	Board index of the EZB to read from.
timeoutMS (optional)	Number of milliseconds to wait before timeout.

Returns

Nothing

Description

Suspends execution of the script until the value read from the specified digital port is equal to valueToWaitFor. If frequencyMs is provided the port is checked every frequencyMs milliseconds. Otherwise the port is checked as often as possible.

EZB

getCPUTemp

```
EZB.getCPUTemp([ezbIndex])
```

Parameters

ezbIndex (optional)	Board index of the EZB to read from.

Returns

CPU temperature of the EZB in degrees celsius.

Description

Returns the CPU temperature of the EZB in degrees Celsius.

getVoltage

```
EZB.getCPUTemp([ezbIndex])
```

Parameters

ezbIndex (optional)	Board index of the EZB to read from.

Returns

Input voltage of the EZB in volts.

Description

Returns the input voltage of the EZB in volts.

isConnected

```
EZB.isConnected(ezbIndex)
```

Parameters

ezbIndex	Board index of the EZB to check if connected.

Returns

True if the EZB at board index ezbIndex is connected. False otherwise.

Description

Checks if the EZB at board index ezbIndex is connected or not.

Example

```
if (EZB.isConnected(0)) {
  print("EZB is connected.");
} else {
  print("EZB is not connected.");
}
```

File

append

```
File.append(filename, byte/byteArray)
```

Parameters

filename	Path of file to append to.
byte/byteArray	Byte or byte array to append to the file.

Returns

Nothing

Description

Appends the byte given by byte to the file. If a byte array is provided instead, the byte array given by byteArray will be appended to the file.

appendString

```
File.appendString(filename, str)
```

Parameters

filename	Path of file to append to.
str	String to append to the file.

Returns

Nothing

Description

Appends the string given by str to the file.

appendStringLine

```
File.appendStringLine(filename, str)
```

Parameters

filename	Path of file to append to.
str	String to append to the file.

Returns

Nothing

Description

Appends the string given by str with a newline character appended to the end to the file.

delete

```
File.delete(filename)
```

Parameters

filename	Path of file to delete.

Returns

Nothing

Description

Deletes the file at the path specified by filename.

exists

```
File.exists(filename)
```

Parameters

filename	Path of file to check if exists.

Returns

True if the file at path filename exists. False otherwise.

Description

Checks if the file at path filename exists.

getLength

```
File.getLength(filename)
```

Parameters

filename	Path of file to get length of.

Returns

Number of characters in the file.

Description

Returns the number of characters contained in the file.

getReadPosition

```
File.getReadPosition(filename)
```

Parameters

filename	Path of file to get read position of.

Returns

Read position of the file as the number of characters that have been read since it was opened.

Description

Returns the read position of the file as the number of characters that have been read since it was opened.

isFileOpenForReading

```
File.isFileOpenForReading(filename)
```

Parameters

filename	Path of file to check if open for reading.

Returns

True if file is open for reading. False otherwise.

Description

Checks if the file at filename is open for reading or not. If a read function has been called on the file this will return true. If the file has not been opened for reading, or the file is closed using the readClose funciton then this will return false. A file will remain open for reading even after the script finishes executing unless the file is closed using the readClose function.

isReadEnd

```
File.isReadEnd(filename)
```

Parameters

filename	Path of file to check if at end.

Returns

True if the read position of the file is as at the end of the file. False otherwise.

Description

Checks if the read position of the file at filename is at the end of the file.

readAllText

```
File.readAllText(filename)
```

Parameters

filename	Path of file to read.

Returns

All text contained within the file as a string.

Description

Reads and returns all the text contained with the file at filename as a string.

readByte

```
File.readByte(filename)
```

Parameters

filename	Path of file to read from.

Returns

The ASCII code representation for the character at the current read position of the file.

Description

Reads and returns the ASCII code representation (one byte of data) of the character located at the current read position of the file, and then advances the read position by one character.

readChar

```
File.readChar(filename)
```

Parameters

filename	Path of file to read from.

Returns

The character at the current read position of the file.

Description

Reads and returns the character located at the current read position of the file, and then advances the read position by one character.

readClose

```
File.readClose(filename)
```

Parameters

filename	Path of file to close.

Returns

Nothing

Description

Closes the file at path filename that was being read from.

readLine

```
File.readLine(filename)
```

Parameters

filename	Path of file to read from.

Returns

The string from the current read position to the next endline character.

Description

Reads and returns all characters as a string from the current read position to the next endline character. The returned string does not contain the endline character. The read position advances to be immediately after the endline character.

readRandomLine

```
File.readRandomLine(filename)
```

Parameters

filename	Path of file to read from.

Returns

A random line from the file as a string.

Description

Reads and returns a random line from the file, which starts from the end of one line and ends at the next endline character. This function does not affect the read position of the file.

readReset

```
File.readReset(filename)
```

Parameters

filename	Path of file to reset read position of.

Returns

Nothing

Description

Resets the read position of the file to 0.

setReadPosition

```
File.setReadPosition(filename, position)
```

Parameters

filename	Path of file to set read position of.

position	Read position to set to.

Returns

Nothing

Description

Sets the read position of the file to position.

I2C

read

```
I2C.read(address, bytesToExpect, [ezbIndex])
```

Parameters

address	Address of the device to read from as a hexadecimal number (e.g.0x5e).
bytesToExpect	Number of bytes to read from the device.
ezbIndex (optional)	Board index of the EZB to read from.

Returns

A byte array of length bytesToExpect containing the bytes read from the device.

Description

Reads bytesToExpect bytes from the device at address and returns them as a byte array.

readByte

```
I2C.readByte(address, [ezbIndex])
```

Parameters

address	Address of the device to read from as a hexadecimal number (e.g.0x5e).
ezbIndex (optional)	Board index of the EZB to read from.

Returns

One byte read from the device.

Description

Reads one byte of data from the device at address.

readString

```
I2C.readString(address, bytesToExpect, [ezbIndex])
```

Parameters

address	Address of the device to read from as a hexadecimal number (e.g.0x5e).
bytesToExpect	Number of bytes to read from the device (1 byte = 1 character).
ezbIndex (optional)	Board index of the EZB to read from.

Returns

A string of length bytesToExpect.

Description

Reads a string of data of length bytesToExpect from the device at address.

setClockSpeed

```
I2C.setClockSpeed(speed, [ezbIndex])
```

Parameters

speed	Clock speed to set the I2C interface to.
ezbIndex (optional)	Board index of the EZB to set the I2C clock speed of.

Returns

Nothing

Description

Sets the clock speed of the I2C interface. The default speed is 100000, which is 100khz. Many devices support faster speeds, up to 400000 (400khz).

write

```
I2C.write(address, byte/byteArray, [ezbIndex])
```

Parameters

address	Address of the device to read from as a hexadecimal number (e.g.0x5e).
byte/byteArray	Byte or byte array to write to the device.
ezbIndex (optional)	Board index of the EZB to write to.

Returns

Nothing

Description

Writes byte to the device at address. A byte array can instead be provided to write to the device at address.

writeString

```
I2C.write(address, str, [ezbIndex])
```

Parameters

address	Address of the device to read from as a hexadecimal number (e.g.0x5e).
str	String to write to the device.
ezbIndex (optional)	Board index of the EZB to write to.

Returns

Nothing

Description

Writes the string given by str to the device at address.

Movement

down

```
Movement.down([timeOut])
```

Parameters

timeOut (optional)	Duration to move down for in milliseconds.

Returns

Nothing

Description

Moves the robot down using the project's Movement Panel skill. The Movement Panel skill must have the capability to move the robot down. If timeOut is provided the robot will move down for timeOut milliseconds. Otherwise it will continue to move down until a different Movement function is called.

forward

```
Movement.forward([speed], [timeOut])
```

Parameters

speed (optional)	Speed to move at.
timeOut (optional)	Duration to move forward for in milliseconds.

Returns

Nothing

Description

Moves the robot forward using the project's Movement Panel skill. The speed can be specified by providing speed as a value between 0 and 255. If timeOut is provided the robot will move forward for timeOut milliseconds. Otherwise it will continue to move forward until a different Movement function is called.

land

```
Movement.land()
```

Returns

Nothing

Description

Lands the robot using the project's Movement Panel skill. The Movement Panel skill must have the capability to land the robot.

getSpeed

```
Movement.getSpeed()
```

Returns

The global movement speed value as a value between 0 and 255.

Description

Returns the global movement speed value as a value between 0 and 255. The global movement speed value is the speed used for the left and right wheels by the project's Movement Panel skill if it has the capability to do so.

getSpeedLeft

```
Movement.getSpeedLeft()
```

Returns

The global movement speed value of the left wheel as a value between 0 and 255.

Description

Returns the global movement speed value of the left wheel as a value between 0 and 255. The global movement speed value of the left wheel is the speed used for the left wheel by the project's Movement Panel skill if it has the capability to do so.

getSpeedRight

```
Movement.getSpeedRight()
```

Returns

The global movement speed value of the right wheel as a value between 0 and 255.

Description

Returns the global movement speed value of the right wheel as a value between 0 and 255. The global movement speed value of the right wheel is the speed used for the right wheel by the project's Movement Panel skill if it has the capability to do so.

left

```
Movement.left([speed], [timeOut])
```

Parameters

speed (optional)	Speed to move at.
timeOut (optional)	Duration to move left for in milliseconds.

Returns

Nothing

Description

Moves the robot left using the project's Movement Panel skill. The speed can be specified by providing speed as a value between 0 and 255. If timeOut is provided the robot will move left for timeOut milliseconds. Otherwise it will continue to move left until a different Movement function is called.

reverse

```
Movement.reverse([speed], [timeOut])
```

Parameters

speed (optional)	Speed to move at.
timeOut (optional)	Duration to move reverse for in milliseconds.

Returns

Nothing

Description

Moves the robot reverse using the project's Movement Panel skill. The speed can be specified by providing speed as a value between 0 and 255. If timeOut is provided the robot will move reverse for timeOut milliseconds. Otherwise it will continue to move reverse until a different Movement function is called.

right

```
Movement.right([speed], [timeOut])
```

Parameters

speed (optional)	Speed to move at.
timeOut (optional)	Duration to move right for in milliseconds.

Returns

Nothing

Description

Moves the robot right using the project's Movement Panel skill. The speed can be specified by providing speed as a value between 0 and 255. If timeOut is provided the robot will move right for timeOut milliseconds. Otherwise it will continue to move right until a different Movement function is called.

rollLeft

```
Movement.rollLeft([timeOut])
```

Parameters

timeOut (optional)	Duration to roll left for in milliseconds.

Returns

Nothing

Description

Rolls the robot left using the project's Movement Panel skill. The Movement Panel skill must have the capability to roll the robot left. If timeOut is provided the robot will roll left for timeOut milliseconds. Otherwise it will continue to roll left until a different Movement function is called.

rollRight

```
Movement.rollRight([timeOut])
```

Parameters

timeOut (optional)	Duration to roll right for in milliseconds.

Returns

Nothing

Description

Rolls the robot right using the project's Movement Panel skill. The Movement Panel skill must have the capability to roll the robot right. If timeOut is provided the robot will roll right for timeOut milliseconds. Otherwise it will continue to roll right until a different Movement function is called.

setSpeed

```
Movement.setSpeed(speed/speedLeft, [speedRight])
```

Parameters

speed/speedLeft	Global Movement Speed value for left and right wheel. If speedRight is provided this value will be used for the global Movement Speed value of the left wheel only.
speedRight (optional)	Global movement speed value for the right wheel.

Returns

Nothing

Description

Sets the global movement speed value for the left and right wheels. If speedLeft and speedRight are provided the values for the left and right wheels will be set respectively.

Example

```
// Set both left and right speeds to 127 (half speed)
Movement.setSpeed(127);

// Set the left speed to 100 and right speed to 200 (slightly turn left)
Movement.setSpeed(100, 200);
```

setSpeedLeft

```
Movement.setSpeedLeft(speed)
```

Parameters

speed	Global movement speed value for the left wheel.

Returns

Nothing

Description

Sets the global movement speed value for the left wheel. **speed** is a value between 0 and 255.

setSpeedRight

```
Movement.setSpeedRight(speed)
```

Parameters

speed	Global movement speed value for the right wheel.

Returns

Nothing

Description

Sets the global movement speed value for the right wheel. speed is a value between 0 and 255.

stop

```
Movement.stop()
```

Returns

Nothing

Description

Stops the robot using the project's Movement Panel skill.

takeoff

```
Movement.takeoff()
```

Returns

Nothing

Description

Tells the robot to takeoff using project's Movement Panel skill. The Movement Panel skill must have takeoff capability.

up

```
Movement.up([timeOut])
```

Parameters

timeOut (optional)	Duration to move up for in milliseconds.

Returns

Nothing

Description

Moves the robot up using the project's Movement Panel skill. The Movement Panel skill must have the capability to move the robot up. If timeOut is provided the robot will move up for timeOut milliseconds. Otherwise it will continue to move up until a different Movement function is called.

waitForDown

```
Movement.waitForDown()
```

Returns

Nothing

Description

Suspends execution of the script until down is executed from the project's movement panel either by the user or from another script.

waitForForward

```
Movement.waitForForward()
```

Returns

Nothing

Description

Suspends execution of the script until forward is executed from the project's movement panel either by the user or from another script.

waitForLeft

```
Movement.waitForLeft()
```

Returns

Nothing

Description

Suspends execution of the script until left is executed from the project's movement panel either by the user or from another script.

waitForReverse

```
Movement.waitForReverse()
```

Returns

Nothing

Description

Suspends execution of the script until reverse is executed from the project's movement panel either by the user or from another script.

waitForRight

```
Movement.waitForRight()
```

Returns

Nothing

Description

Suspends execution of the script until right is executed from the project's movement panel either by the user or from another script.

waitForStop

```
Movement.waitForStop()
```

Returns

Nothing

Description

Suspends execution of the script until stop is executed from the project's movement panel either by the user or from another script.

waitForUp

```
Movement.waitForUp()
```

Returns

Nothing

Description

Suspends execution of the script until up is executed from the project's movement panel either by the user or from another script.

fromString

This is a slug for fromString. Can use custom movement names

Net

hTTPGet

```
Net.hTTPGet(url, [timeout])
```

Parameters

url	URL to send HTTP Get request to.
timeout (optional)	Timeout in milliseconds.

Returns

HTTP contents from the provided URL as a string.

Description

Sends an HTTP Get request to url. If timeout is specified the request will timeout after timeout milliseconds. Suspends script execution until the request completes, or the request times out.

hTTPPost

```
Net.hTTPPost(url, postData, [timeout])
```

Parameters

url	URL to send HTTP POST request to.
postData	Data to send as part of POST request as a string.
timeout (optional)	Timeout in milliseconds.
headerNames (optional)	Array string of header names to append to the post request
headerValues (optional)	Array string of header values to append to the post request (must match count of headerNames)

Returns

The HTTP POST request response as a string.

Description

Sends an HTTP POST request to url with postData stored in the request body. If timeout is specified the request will timeout after timeout milliseconds. Suspends script execution until the request completes, or the request times out.

Example

```
var resp = Net.hTTPPost(
 "https://postman-echo.com/post",
 "Some text");

 print(resp);

// post to an echo server
var resp = Net.hTTPPost(
 "https://postman-echo.com/post",
 "banana",
 1000,
 ["customHeaderTitle"],
 ["customHeaderValue"]);

 print(resp);
```

isInternetAvailable

```
Net.isInternetAvailable()
```

Returns

True if the computer is connected to the internet. False otherwise.

Description

Checks the internet connection of the computer.

sendUDP

```
Net.sendUDP(hostname, port, byteArray)
```

Parameters

hostname	Hostname of receiver as a string.
port	Port to use.
byteArray	Data to send as a byte array.

Returns

The length of the byteArray being sent.

Description

Sends byteArray over the specified port to the hostname using UDP.

Example

```
// 1. This corresponds to the ASCII codes for "Hello"
```

```
var data = [72, 101, 108, 108, 111];

// 2. Call Net.sendUDP with the hostname, port, and your byteArray
Net.sendUDP("10.0.0.10", 8870, data);

// 1. Call Net.sendUDP with the hostname, port, and your byteArray(hello)
Net.sendUDP("10.0.0.10", 8870, [72, 101, 108, 108, 111]);
```

Ping

get

```
Ping.get(triggerPort, echoPort, [ezbIndex])
```

Parameters

triggerPort	Trigger port used by the ultrasonic distance sensor.
echoPort	Echo port used by the ultrasonic distance sensor.
ezbIndex (optional)	Board index of the EZB to use.

Returns

The value measured from the ultrasonic distance sensor as a number between 0 and 255.

Description

Pings the ultrasonic distance sensor once and returns the value measured.

waitForBetween

```
Ping.waitForBetween(triggerPort, echoPort, low, high, [ezbIndex])
```

Parameters

triggerPort	Trigger port used by the ultrasonic distance sensor.
echoPort	Echo port used by the ultrasonic distance sensor.
low	Inclusive lower bound on value to wait for.
high	Inclusive upper bound on value to wait for.
ezbIndex (optional)	Board index of the EZB to use.

Returns

The value measured from the ultrasonic distance sensor that is between low and high as a number between 0 and 255.

Description

Suspends execution of the script until the value read from the ultrasonic distance sensor is between low (inclusvie) and high (inclusive). The value read from the ultrasonic distance sensor is returned when the script resumes execution.

waitForEquals

```
Ping.waitForEquals(triggerPort, echoPort, distance, [ezbIndex])
```

Parameters

triggerPort	Trigger port used by the ultrasonic distance sensor.
echoPort	Echo port used by the ultrasonic distance sensor.
distance	Value to wait for.
ezbIndex (optional)	Board index of the EZB to use.

Returns

The value measured from the ultrasonic distance sensor that is equal to distance as a number between 0 and 255.

Description

Suspends execution of the script until the value read from the ultrasonic distance sensor is equal to distance. The value read from the ultrasonic distance sensor is returned when the script resumes execution.

waitForHigher

```
Ping.waitForHigher(triggerPort, echoPort, distance, [ezbIndex])
```

Parameters

triggerPort	Trigger port used by the ultrasonic distance sensor.
echoPort	Echo port used by the ultrasonic distance sensor.
distance	Exclusive lower bound on value to wait for
ezbIndex (optional)	Board index of the EZB to use.

Returns

The value measured from the ultrasonic distance sensor that is higher than distance as a number between 0 and 255.

Description

Suspends execution of the script until the value read from the ultrasonic distance sensor is higher than distance. The value read from the ultrasonic distance sensor is returned when the script resumes execution.

waitForLower

```
Ping.waitForLower(triggerPort, echoPort, distance, [ezbIndex])
```

Parameters

triggerPort	Trigger port used by the ultrasonic distance sensor.
echoPort	Echo port used by the ultrasonic distance sensor.
distance	Exclusive upper bound on value to wait for.
ezbIndex (optional)	Board index of the EZB to use.

Returns

The value measured from the ultrasonic distance sensor that is lower than distance as a number between 0 and 255.

Description

Suspends execution of the script until the value read from the ultrasonic distance sensor is lower than distance. The value read from the ultrasonic distance sensor is returned when the script resumes execution.

PWM

get

```
PWM.get(port, [ezbIndex])
```

Parameters

port	Port to get PWM duty cycle from.
ezbIndex (optional)	Board index of the EZB to get PWM duty cycle from.

Returns

The PWM duty cycle percentage of the specified port as a value between 0 and 100.

Description

Returns the PWM duty cycle percentage of the specified port. Duty cycle percentage is returned as a value between 0 and 100.

set

```
PWM.set(port, dutyCycle, [ezbIndex])
```

Parameters

port	Port to use.
dutyCycle	Duty cycle percentage to set PWM duty cycle to.
ezbIndex (optional)	Board index of the EZB to get PWM duty cycle from.

Returns

Nothing

Description

Sets the PWM duty cycle percentage of the specified port to dutyCycle which is a value between 0 and 100.

setRandom

```
PWM.setRandom(port, lowCycle, highCycle, [ezbIndex])
```

Parameters

port	Port to use.
lowCycle	Inclusive lower bound on duty cycle percentage to set PWM duty cycle to.
highCycle	Exclusive upper bound on duty cycle percentage to set PWM duty cycle to.
ezbIndex (optional)	Board index of the EZB to get PWM duty cycle from.

Returns

The random value that the PWM was set to.

Description

Sets the PWM duty cycle percentage of the specified port to a value randomly chosen between lowCycle (inclusive) and highCycle (exclusive). lowCycle and highCycle are values between 0 and 100.

Servo

decrement

```
Servo.decrement(port, count, [ezbIndex])
```

Parameters

port	Servo port to use.
count	Degrees to decrement by.
ezbIndex (optional)	Board index of the EZB to use.

Returns

Nothing

Description

Decrements the position of the servo at the specified port by count. Servo position is between 0 and 180.

getPosition

```
Servo.getPosition(port, [ezbIndex])
```

Parameters

port	Servo port to use.
ezbIndex (optional)	Board index of the EZB to use.

Returns

The commanded position of the servo at the specified port as a value between 1 and 180.

Description

Returns the position of the servo at the specified port as a value between 1 and 180. This function only returns the position the servo was commanded to move to, not the actual position of the servo. This does not query a smart servo to get the position. It only returns the last position the servo was moved to.

If you wish to get the servo position of a bi-directional servo, use the getPositionRealtime() command

getPositionRealtime

```
Servo.getPositionRealtime(port, [ezbIndex])
```

Parameters

port	Servo port to use.
ezbIndex (optional)	Board index of the EZB to use.

Returns

The actual position of the servo at the specified port as a value between 1 and 180.

Description

Returns the position of the servo at the specified port. This function returns the physical position of the servo returned by the servo. This function was meant to work with servos which accept bi-directional communication (e.g. Dynamixel, LewanSoul).

- If this function is called on a pwm servo or a servo that does not support bi-direction communication, the last set servo position will be returned
- If the servo being queried has a communication issue, the last set position will be returned

getSpeed

```
Servo.getSpeed(port, [ezbIndex])
```

Parameters

port	Servo port to get speed from.
ezbIndex (optional)	Board index of the EZB to use.

Returns

The speed that the specified port is set to as a value

Description

Returns the speed that the specified port is set to as a value

increment

```
Servo.increment(port, count, [ezbIndex])
```

Parameters

port	Servo port to use.
count	Degrees to increment by.

ezbIndex (optional)	Board index of the EZB to use.

Returns

Nothing

Description

Increments the position of the servo at the specified port by count. Servo position is between 0 and 180.

release

```
Servo.release(port, [ezbIndex])
```

Parameters

port	Servo port to release.
ezbIndex (optional)	Board index of the EZB to use.

Returns

Nothing

Description

Stops the servo at the specified port from holding its position.

releaseAll

```
Servo.releaseAll([ezbIndex])
```

Parameters

ezbIndex (optional)	Board index of the EZB to use.

Returns

Nothing

Description

Stops all servos at all ports from holding their position.

*Warning: this will affect all ports and reset their state. That means if you using some ports for digital i/o or UART, those ports will be reset. This resets ALL ports, not some. If you wish to only release servos on specific ports, create a script using Servo.release(port) instead of this.

i.e.

```
Servo.release(d0);
Servo.release(d1);
Servo.release(d3);
```

setMaxPositionLimit

```
Servo.setMaxPositionLimit(port, position, [ezbIndex])
```

Parameters

port	Servo port to set max position limit of.
position	Max position limit.
ezbIndex (optional)	Board index of the EZB to use.

Returns

Nothing

Description

Sets the global maximum position that the servo at the specified port can move to the position. The position is a value between 0 and 180 (or the global max servo position if overridden)

While robot skills may have their own Min/Max servo options, this sets the global value. This is used for safety to ensure servos don't move outside the specified range.

setMinPositionLimit

```
Servo.setMinPositionLimit(port, position, [ezbIndex])
```

Parameters

port	Servo port to set min position limit of.
position	Min position limit.
ezbIndex (optional)	Board index of the EZB to use.

Returns

Nothing

Description

Sets the global minimum position that the servo at the specified port can move to the position. The position is a value between 0 and 180 (or the global max servo position if overridden).

While robot skills may have their own Min/Max servo options, this sets the global value. This is used for safety to ensure servos don't move outside the specified range.

setPosition

```
Servo.setPosition(port, position, [ezbIndex])
```

Parameters

port	Servo port to set position of.
position	Position to move the servo to.
ezbIndex (optional)	Board index of the EZB to use.

Returns

Nothing

Description

Commands the servo to move to position. position is a value between 1 and 180.

setPositionRandom

```
Servo.setPositionRandom(port, lowPosition, highPosition, [ezbIndex])
```

Parameters

port	Servo port to set position of.
lowPosition	Inclusive lower bound on random position value.
highPosition	Exclusive upper bound on random position value.
ezbIndex (optional)	Board index of the EZB to use.

Returns

The random position that the servo was commanded to move to.

Description

Commands the servo to move to a random position between lowPosition (inclusive) and highPosition (exclusive). lowPosition and highPositionis are values between 1 and 180.

setSpeed

```
Servo.setSpeed(port, speed, [ezbIndex])
```

Parameters

port	Servo port to use.
speed	Speed to set the servo port to.
ezbIndex (optional)	Board index of the EZB to use.

Returns

Nothing

Description

Sets the speed of the servo port to speed. speed is a value between 0 (fastest) and 10 (slowest).

This tutorial explains how to re-initialize the servo speeds during startup: Setting Servo speeds and Initialization Script Tutorial - Tutorials - Community - Synthiam

waitForMove

```
Servo.waitForMove(port, [ezbIndex], [timeoutMS])
```

Parameters

port	Servo port to wait for movement.
ezbIndex (optional)	Board index of the EZB to use.
timeoutMS (optional)	Number of milliseconds to wait before timeout.

Returns

The position of the servo at the specified port that has changed. Returns -1 if timeout.

Description

Suspends execution of the script until the commanded position of the servo at the specified port changes. The commanded position can be changed either by the user from another skill, or another script.

waitForPositionEquals

```
Servo.waitForPositionEquals(port, position, [ezbIndex], [timeoutMS])
```

Parameters

port	Servo port to use.
position	Position to wait for.
ezbIndex (optional)	Board index of the EZB to use.
timeoutMS (optional)	Number of milliseconds to wait before timeout.

Returns

Current servo position. Returns -1 if timeout

Description

Suspends execution of the script until the commanded position of the servo at the specified port is equal to position. The commanded position can be changed either by the user from another skill, or another script.

waitForPositionHigher

```
Servo.waitForPositionHigher(port, position, [ezbIndex], [timeoutMS])
```

Parameters

port	Servo port to use.
position	Exclusive lower bound on the position to wait for.
ezbIndex (optional)	Board index of the EZB to use.
timeoutMS (optional)	Number of milliseconds to wait before timeout.

Returns

The position of the servo at the specified port that is higher than position. Returns -1 if timeout.

Description

Suspends execution of the script until the commanded position of the servo at the specified port is higher than position. The commanded position can be changed either by the user from another skill, or another script.

waitForPositionLower

```
Servo.waitForPositionLower(port, position, [ezbIndex], [timeoutMS])
```

Parameters

port	Servo port to use.
position	Exclusive upper bound on the position to wait for.
ezbIndex (optional)	Board index of the EZB to use.
timeoutMS (optional)	Number of milliseconds to wait before timeout.

Returns

The position of the servo at the specified port that is lower than position. Returns -1 if timeout

Description

Suspends execution of the script until the commanded position of the servo at the specified port is lower than position. The commanded position can be changed either by the user from another skill, or another script.

setVelocity

```
setVelocity(port, velocity, ezb index);
```

Parameters

port	The servo port
velocity	The velocity value
ezb index (optional)	The ezb index

Description

Set the velocity of a servo. The servo and servo driver must support this feature. For example, check the Dynamixel robot skill because some of their servos support the Velocity feature.

Example

```
// Set the velocity of the servo v1 to 10. By default, this will use EZB #0 because no ezb is specified
Servo.setVelocity(v1, 10);

// Specify the velocity of servo V1 on ezb index #3
Servo.setVelocity(v1, 10, 3);
```

setAcceleration

```
setAcceleration(port, value, ezb index);
```

Parameters

port	The servo port
value	The Acceleration value
ezb index (optional)	The ezb index

Description

Set the acceleration of a servo. The servo and servo driver must support this feature. For example, check the Dynamixel robot skill because some of their servos support the Acceleration feature.

Example

```
// Set the Acceleration of the servo v1 to 10. By default, this will use EZB #0 because no ezb is specified
```

```
Servo.setAcceleration(v1, 10);

// Specify the Acceleration of servo V1 on ezb index #3
Servo.setAcceleration(v1, 10, 3);
```

IsReleased

```
IsReleased(port, ezbIndex)
```

Parameters

port	The servo port
ezbIndex (optional)	The ezb index

Returns

boolean (true of false)

Description

Returns the released state of the servo (torque off or on)

Example

```
if (IsReleased(d0))
  print("Servo d0 is released");
```

setFineTuneOffset

```
setFineTuneOffset(port, offset, ezbIndex)
```

Parameters

port - Servo port (i.e. d0	v2
offset - Value of the offset. Can be a positive or negative value	
ezbIndex - The ezb to apply this offset value (optional)	

Description

Sets the global fine-tune offset in positions of the specified servo. For example, if you set the offset to +10, every servo position set by other robot skills or script commands will add 10 positions. If you query the servo position, it'll return 10, not 20.

One of the uses for this could be used to maintain servos upright from an IMU, such as in this post.

Example

```
// Set the servo D0 to an offset of +20 positions
setFineTuneOffset(D0, 20);

// Set the servo D0 to an offset of -15 positions
setFineTuneOffset(D0, -15);
```

getFineTuneOffset

```
getFineTuneOffset(port, ezbIndex)
```

Parameters

port - Servo port (i.e. d0	v2
ezbIndex - The ezb to apply this offset value (optional)	

Description

Gets the global fine-tuned offset in positions of the specified servo. The value is set by either loading a fine tune servo profile or calling setServoFileTune()

One of the uses for this could be used to maintain servos upright from an IMU, such as in this post.

Example

```
// Get the servo D0 offset and print it to the output
print("The offset of D0 is " + getFineTuneOffset(D0));
```

getVelocity

```
Servo.getVelocity(port, [ezbIndex])
```

Parameters

port	Servo port to get velocity from.
ezbIndex (optional)	Board index of the EZB to use.

Returns

The velocity that the specified port is set to as a value

Description

Returns the velocity that the specified port is set to as a value

getAcceleration

```
Servo.getAcceleration(port, [ezbIndex])
```

Parameters

port	Servo port to get acceleration from.
ezbIndex (optional)	Board index of the EZB to use.

Returns

The acceleration that the specified port is set to as a value

Description

Returns the acceleration that the specified port is set to as a value

UART

hardwareUartAvailable

```
UART.hardwareUartAvailable(uartIndex, [ezbIndex])
```

Parameters

uartIndex	Index of UART to check.
ezbIndex (optional)	Board index of the EZB to use.

Returns

Number of bytes available in the UART receive buffer at the specified UART.

Description

Returns the number of bytes available in the UART receive buffer at the specified UART given by uartIndex. The UART must be initialized first.

hardwareUartRead

```
UART.hardwareUartRead(uartIndex, bytesToRead, [ezbIndex])
```

Parameters

uartIndex	Index of UART to read from.
bytesToRead	Number of bytes to read from the UART.
ezbIndex (optional)	Board index of the EZB to use.

Returns

A byte array of length bytesToRead containing data received from the specified UART.

Description

Returns a byte array of length bytesToRead containing data received from the specified UART. The UART must be initialized first.

hardwareUartReadAvailable

```
UART.hardwareUartReadAvailable(uartIndex, [ezbIndex])
```

Parameters

uartIndex	Index of UART to read from.
ezbIndex (optional)	Board index of the EZB to use.

Returns

A byte array containing all available data received from the specified UART.

Description

Returns a byte array containing all available data received from the specified UART. The UART must be initialized first.

hardwareUartReadString

```
UART.hardwareUartReadString(uartIndex, bytesToRead, [ezbIndex])
```

Parameters

uartIndex	Index of UART to read from.
bytesToRead	Number of bytes to read from the UART.
ezbIndex (optional)	Board index of the EZB to use.

Returns

A string of length bytesToRead containing data received from the specified UART.

Description

Returns a string of length bytesToRead containing data received from the specified UART. The UART must be initialized first.

hardwareUartReadStringAvailable

```
UART.hardwareUartReadStringAvailable(uartIndex, [ezbIndex])
```

Parameters

uartIndex	Index of UART to read from.
ezbIndex (optional)	Board index of the EZB to use.

Returns

A string containing all available data received from the specified UART.

Description

Returns a string containing all available data received from the specified UART. The UART must be initialized first.

hardwareUartWrite

```
UART.hardwareUartWrite(uartIndex, byte/byteArray, [ezbIndex])
```

Parameters

uartIndex	Index of UART to write to.
byte/byteArray	Byte or byte array to write to the UART.
ezbIndex (optional)	Board index of the EZB to use.

Returns

Nothing

Description

Writes byte to the specified UART. A byte array can instead be provided to write to the specified UART. The UART must be initialized first.

Example

```
// Initialize the first uart (#0) to 9600 baud
UART.initHardwareUart(0, 9600);

// Write the array of bytes to the first UART
UART.hardwareUartWrite(0, [0xff, 0xf0, 0x30]);

// Initialize the first uart (#0) to 9600 baud
UART.initHardwareUart(0, 9600);

// Write a single byte to the first UART
UART.hardwareUartWrite(0, 0xff);

var str = [0xff, 0xfe, 0xfa, 0x35, 0xaf];

// write the amount of data we're gonna send
UART.hardwareUartWrite(0, str.length);

// now send the data
UART.hardwareUartWrite(0, str);
```

hardwareUartWriteString

```
UART.hardwareUartWriteString(uartIndex, str, [ezbIndex])
```

Parameters

uartIndex	Index of UART to write to.
str	String to write to the UART.
ezbIndex (optional)	Board index of the EZB to use.

Returns

Nothing

Description

Writes string given by str to the specified UART. The UART must be initialized first.

Example

```
var str = "some random amount of data";

// send the data to UART #0 on the first EZB
UART.hardwareUartWriteString(0, str);
```

initHardwareUart

```
UART.initHardwareUart(uartIndex, baud, [ezbIndex])
```

Parameters

uartIndex	Index of UART to initialize.
baud	Baud rate to set the UART to.
ezbIndex (optional)	Board index of the EZB to use.

Returns

Nothing

Description

Initializes the UART with the specified baud rate. The UART will stay initialized until the EZB is power cycled. The UART must be initialized using this command before you can READ or WRITE to the interface.

Once you have initialized the UART interface with this command, you may now use the other UART commands to read and write to the interface.

sendSerial

```
UART.sendSerial(port, baud, byte/byteArray, [ezbIndex])
```

Parameters

port	Digital port to send data over.
baud	Baud rate to send data at.

byte/byteArray	Byte or byte array to send.
ezbIndex (optional)	Board index of the EZB to use.

Returns

Nothing

Description

Sends a byte or a byte array of data over the specified digital port with a baud rate of baud.

sendSerialString

```
UART.sendSerialString(port, baud, str, [ezbIndex])
```

Parameters

port	Digital port to send data over.
baud	Baud rate to send data at.
str	String to send.
ezbIndex (optional)	Board index of the EZB to use.

Returns

Nothing

Description

Sends a string of data over the specified digital port with a baud rate of baud.

Utility

browser

```
Utility.browser(url)
```

Parameters

url	URL to open.

Returns

Nothing

Description

Opens the provided URL in the computer's default browser.

checkForUpdate

```
Utility.checkForUpdate()
```

Returns

True if an ARC update is available. False otherwise.

Description

Checks if an ARC update is available.

clearGlobalVariable

```
Utility.clearGlobalVariable(globalVariableName)
```

Parameters

globalVariableName	Name of the global variable to clear as a string.

Returns

Nothing

Description

Deletes the specified global variable.

closeControl

```
Utility.closeControl()
```

Returns

Nothing

Description

Used for mobile devices and the Interface Builder only, this function will close the current control, the same as pressing the BACK button on your device.

defineGlobalVariable

```
Utility.defineGlobalVariable(globalVariableName, size, [defaultValue])
```

Parameters

globalVariableName	Name of the global variable to define as a string. Must start with '$'
size	Size of the array to create.
defaultValue (optional)	Default value to fill the array.

Returns

Nothing

Description

Creates a global variable array with name globalVariableName and size size. If defaultValue is provided the array will be filled with defaultValue. Otherwise it will be filled with empty strings.

exec

```
Utility.exec(filename, [arguments])
```

Parameters

filename	File path of the executable file to execute.
arguments (optional)	Optional arguments to pass to the executable file as a string.

Returns

Nothing

Description

Execute the executable file at file path filename with optional arguments arguments.

fillGlobalArray

```
Utility.fillGlobalArray(globalVariableName, defaultValue)
```

Parameters

globalVariableName	Name of the global array as a string.
defaultValue	Value to fill the array with.

Returns

Nothing

Description

Fills the global array with name globalVariableName with the value defaultValue.

getBit

```
Utility.getBit(value, bitPosition)
```

Parameters

value	Number to get bit from.
bitPosition	Position to get bit from.

Returns

The bit (1 or 0) at position bitPosition in the number value.

Description

Returns the bit at position bitPosition in the number value. Position is 0 indexed starting from the least significant bit.

getGlobalArraySize

```
Utility.getGlobalArraySize(globalVariableName)
```

Parameters

globalVariableName	Name of the global array as a string.

Returns

Returns The size of the global array.

Description

Returns the size of the global array with the name globalVariableName.

getRandom

```
Utility.getRandom(min, max)
```

Parameters

min	Inclusive lower bound on the random integer.
max	Exclusive upper bound on the random integer.

Returns

Random integer between min (inclusive) and max (exclusive).

Description

Returns a random integer between min (inclusive) and max (exclusive).

getRandomUnique

```
Utility.getRandomUnique(min, max)
```

Parameters

min	Inclusive lower bound on the random integer.
max	Exclusive upper bound on the random integer.

Returns

Random integer between min (inclusive) and max (exclusive) and unique from the last returned integer.

Description

Returns a random integer between min (inclusive) and max (exclusive). This function will never return the same integer twice.

loadProject

```
Utility.loadProject(filename)
```

Parameters

filename	File path of project to load.

Returns

Nothing

Description

Loads the project at file path filename.

map

```
Utility.map(input, inputMin, inputMax, outputMin, outputMax)
```

Parameters

input	Value to map.
inputMin	Minimum value the input can be.
inputMax	Maximum value the input can be.
outputMin	Minimum value the output can be.
outputMax	Maximum value the output can be.

Returns

input mapped onto the provided range of outputs.

Description

Maps input onto the provided range of outputs. (e.g. if input is halfway between inputMin and inputMax, then this function will return the value halfway between ouputMin and outputMax).

The formula for calculating the return value is shown below.

setBits

```
Utility.setBits(b7, b6, b5, b4, b3, b2, b1, b0)
```

Parameters

b7	The bit in the 8th position (most significant bit).
b6	The bit in the 7th position.
b5	The bit in the 6th position.
b4	The bit in the 5th position.
b3	The bit in the 4th position.
b2	The bit in the 3rd position.
b1	The bit in the 2nd position.
b0	The bit in the 1st position (least significant bit).

Returns

The number in decimal that is equal to the binary number (b7 b6 b5 b4 b3 b2 b1 b0).

Description

Converts the bits given by b7, b6, b5, b4, b3, b2, b1, b0 into decimal and returns the number.

showControl

```
Utility.showControl(controlName)
```

Parameters

controlName	Name of the control to show.

Returns

Nothing

Description

Used for mobile devices and the Interface Builder only, this function will open the control with name controlName into the foreground.

showDesktop

```
Utility.showDesktop(desktopNumber)
```

Parameters

desktopNumber	Virtual desktop to show.

Returns

Nothing

Description

Switches to virtual desktop desktopNumber. desktopNumber can be either 1, 2 or 3.

sleepPCHibernate

```
Utility.sleepPCHibernate(force, wake)
```

Parameters

force	Boolean to force the computer to hibernate or not. If true other applications have no say in the decision.
wake	Boolean to allow the computer will wake up on Wake events or not.

Returns

Nothing

Description

Puts the computer in hibernation mode. If force is true, other applications will have no say in the decision to hibernate. If wake is true, the computer will be allowed to wake up on Wake events.

sleepPCSuspend

```
Utility.sleepPCSuspend(force, wake)
```

Parameters

force	Boolean to force the computer to sleep or not. If true other applications have no say in the decision.
wake	Boolean to allow the computer will wake up on Wake events or not.

Returns

Nothing

Description

Puts the computer in sleep mode. If force is true, other applications will have no say in the decision to sleep. If wake is true, the computer will be allowed to wake up on Wake events.

waitUntilTime

```
Utility.waitUntilTime(hour, minute)
```

Parameters

hour	Hour to wait until in 24 hour format.
minute	Minute to wait until.

Returns

Nothing

Description

Suspends execution of the script until the time (from the computer's clock) reaches the specified time.

shutdownPC

```
Utility.shutdownPC();
```

Description

Shuts down the PC. Does not prompt to save the project. Forces windows to shut down all applications.

Navigation

updateLocation

```
Navigation.updateLocation(x, y, degreesHeading, [confidence]);
```

Parameters

x	Cartesian X (cm) coordinate of the robot
y	Cartesian Y (cm) coordinate of the robot
degreesHeading	Direction in degrees (0-359) that the robot is facing relative to the starting position
confidence	[optional] the confidence (0-255) of accuracy of the coordinate location

Description

Send the cartesian coordinate of the robot to the NMS (navigation messaging system).

Example

```
// Send to the NMS that the robot is moving forward 30 cm over 30 seconds

for (var x = 0; x < 30; x++) {

  print("Position X:" + x);
  Navigation.updateLocation(x, y, degreesHeading);
  sleep(1000);
}
```

updateScan

```
updateScan(distance, confidence, degree)
```

Parameters

distance	the distance in CM of the detected obstacle
confidence	the confidence (0-255) of the detected obstacle
degree	the degree (0-359) of the obstacle

Description

Programmatically send the detected obstacle distance based on the current cartesian coordinate of the robot to the NMS (navigation messaging system).

SetNavigationStatusToStopCancel

```
SetNavigationStatusToStopCancel()
```

Description

Instruct the navigator in NMS Level #1 to stop/cancel navigating. The variable $NavigationStatus will also update to StoppedCancelled.

*Note: the NMS Level #1 robot skill must support this function. Verify with its manual that this feature is supported. There should also be an option in the respective robot skill for support pausing with close distances.

Example NMS Level #1 controls are:

- EZ-Slam
- The Navigator
- The Better Navigator
- Other navigation robot skills

Example

```
// Stop navigating if the ping sensor detects an object less than 100 distance
if (Ping.get(d0, d0) < 100)
  Navigation.SetNavigationStatusToStopCancel();
```

SetNavigationStatusToPause

```
SetNavigationStatusToPause()
```

Description

Instruct the navigator in NMS Level #1 to pause navigating. The variable $NavigationStatus will also update to Paused.

*Note: the NMS Level #1 robot skill must support this function. Verify with its manual that this feature is supported. There should also be an option in the respective robot skill for support pausing with close distances.

Example NMS Level #1 controls are:

- EZ-Slam
- The Navigator
- The Better Navigator
- Other navigation robot skills

Example

```
// Pause navigating if the ping sensor detects an object less than 100 distance and continue if greater
if (Ping.get(d0, d0) < 100)
  Navigation.SetNavigationStatusToPause();
else
  Navigation.SetNavigationStatusToNavigating();
```

SetNavigationStatusToNavigating

SetNavigationStatusToNavigating()

Description

Instruct the navigator in NMS Level #1 to continue navigating if previously paused. The variable $NavigationStatus will also update to Navigating.

*Note: the NMS Level #1 robot skill must support this function. Verify with its manual that this feature is supported. There should also be an option in the respective robot skill for support pausing with close distances.

Example NMS Level #1 controls are:

- EZ-Slam
- The Navigator
- The Better Navigator
- Other navigation robot skills

Example

```
// Pause navigating if the ping sensor detects an object less than 100 distance and continue if greater
if (Ping.get(d0, d0) < 100)
  Navigation.SetNavigationStatusToPause();
else
  Navigation.SetNavigationStatusToNavigating();
```