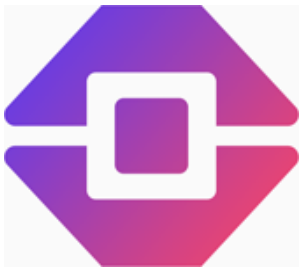

EV3 Swift Playground Command Reference

There is some documentation about the API in the playgrounds glossary. You can find the explanation of the commands there and I copied those into this document for convenience.

The parameters that you can pass to the functions aren't documented there but as soon as you start typing the command within the playground quick type lists the possible commands and all possible options.

Compared to the Sphere R2D2 - or Parrot Mambo API the EV3 API is much more complex. The reason is that there are more sensors and more actors than with other „robots".

Brick Commands

Light Commands

```
ev3.brickLightOn(withColor: BrickLightColor.[green, orange, red], inMode:
BrickLightMode.[flashing, on, pulsating])
```

Definition

The *brickLightOn* function controls the Brick Status Light. The Brick Status Light surrounds the Brick Buttons on the face of the EV3 Brick. You can turn the Brick Status Light on in green, orange, or red; turn it off; or make it pulse on and off.

Example

```
ev3.brickLightOn(withColor: BrickLightColor.red, inMode: BrickLightMode.flashing)
```

```
ev3.brickLightOff()
```

Definition

The *brickLightOff* function turns the Brick Status Light off.

Sound Commands

```
ev3.playSound(file: SoundFile.[arm, blip, errorAlarm, fanfare, goodby,
hello, laser, object, ouch, snap, start, stop], atVolume: Float,
withStyle: SoundStyle.[playOnce, playRepeat, waitForCompletion])
ev3.playSound(note: Note.<a Note*>, forSeconds: Float, atVolume: Float)
ev3.playSound(note: Note.<a Note*>, forSeconds: Float, atVolume: Float,
waitForCompletion: Bool)
ev3.playSound(frequency: Float, forSeconds: Float, atVolume: Float)
ev3.playSound(frequency: Float, forSeconds: Float, atVolume: Float,
waitForCompletion: Bool)
```

Definition

The *playSound* function plays sound using the speaker inside the EV3 Brick. You can play pre-recorded sound files or specify a musical note or tone.

Example

```
ev3.playSound(file: SoundFile.arm, atVolume: 100, withStyle: SoundStyle.waitForCompletion)
ev3.playSound(note: Note.a4, forSeconds: 0.1, atVolume: 100)
ev3.playSound(note: Note.a4, forSeconds: 0.1, atVolume: 100, waitForCompletion: true)
ev3.playSound(frequency: 1000, forSeconds: 1, atVolume: 100)
ev3.playSound(frequency: 1000, forSeconds: 1, atVolume: 100, waitForCompletion: true)
```

```
ev3.stopSound()
```

Definition

The *stopSound* function stops any sound that is currently being played by the EV3 Brick. This is usually used to stop a sound that was started earlier in the program by a 'playSound' function that didn't wait for the sound to complete.

Display Commands

```
ev3.display(text: String)
ev3.display(text: String atX: Int, atY:Int, withColor: DisplayColor.
[black, white], withFont: DisplayFont.[bold, large, normal], ClearScreen:
Bool)
```

Definition

The *display* function can display text anywhere on the EV3 Brick Display.

Examples

```
ev3.display(text: „HELLO")
ev3.display(text: "HELLO", atX: 10, atY: 10, withColor: DisplayColor.black, withFont:
DisplayFont.normal, clearScreen: true)
```

```
ev3.displayPoint(atX: Int, atY: Int, withColor: DisplayColor.[black, white], clearScreen: Bool)
```

Definition

The *displayPoint* function draws a single pixel on the Display.

Example

```
ev3.displayPoint(atX: 10, atY: 10, withColor: DisplayColor.black, clearScreen: true)
```

```
ev3.displayLine(fromX: Int, fromY: Int, toX: Int, toY: Int, withColor: DisplayColor.[black, white], clearScreen: Bool)
```

Definition

The *displayLine* function draws a straight line between any two points on the Display.

Example

```
ev3.displayLine(fromX: 10, fromY: 10, toX: 100, toY: 100, withColor: DisplayColor.black, clearScreen: true)
```

```
ev3.displayCircle(centerX: Int, centerY: Int, withRadius: Float, withFill: Bool, withColor: DisplayColor.[black, white], clearScreen: Bool)
```

Definition

The *displayCircle* function draws a circle on the Display.

Example

```
ev3.displayCircle(centerX: 100, centerY: 100, withRadius: 10, withFill: true, withColor: DisplayColor.black, clearScreen: true)
```

```
ev3.displayRectangle(atX: Int, atY: Int, length: Int, height: Int, withFill: Bool, withColor: DisplayColor.[black, white], clearScreen: Bool)
```

Definition

Die Funktion *displayRectangle* zeichnet auf dem Display ein Rechteck.

Example

```
ev3.displayRectangle(atX: 10, atY: 10, length: 100, height: 100, withFill: true, withColor: DisplayColor.black, clearScreen: true)
```

```
ev3.displayImage(named: ImageName.[accept, awake, boom, decline, ev4icon, hurt, neutral, pinchRight, pirate, questionMark, stop, warning])
ev3.displayImage(named: ImageName.[accept, awake, boom, decline, ev4icon, hurt, neutral, pinchRight, pirate, questionMark, stop, warning], atX: Int, atY: Int)
```

Definition

The *displayImage* function draws a graphic image file. It lets you choose from the list of image files that are available on the EV3 Brick.

Example

```
ev3.displayImage(named: ImageName.accept)
ev3.displayImage(named: ImageName.accept, atX: 10, atY: 10, clearScreen: true)
```

Motor Commands

Motor Motion Commands

```
ev3.motorOn(on: OutputPort.[a, b, c, d], withPower: Float)
ev3.motorOn(forDegrees: Float, on: OutputPort.[a, b, c, d], withPower:
Float)
ev3.motorOn(forDegrees: Float, on: OutputPort.[a, b, c, d], withPower:
Float, brakeAtEnd: Bool)
ev3.motorOn(forSeconds: Float, on: OutputPort.[a, b, c, d], withPower:
Float)
ev3.motorOn(forSeconds: Float, on: OutputPort.[a, b, c, d], withPower:
Float, brakeAtEnd: Bool)
ev3.motorOn(forRotations: Float, on: OutputPort.[a, b, c, d], withPower:
Float)
ev3.motorOn(forRotations: Float, on: OutputPort.[a, b, c, d], withPower:
Float, brakeAtEnd: Bool)
```

Definition

The *motorOn* function controls a motor. You can turn a motor on or off, control its power level, or turn the motor on for a specified length of time or number of rotations. The 'motorOn' function turns the motor on, then immediately continues to the next block in the program unless the duration is specified by using the forSeconds, forDegrees or forRotations parameters. You can control the speed and direction of the motor using the withPower parameters. The motor will run until it is stopped or changed by another instruction later in the program, or until the program ends.

```
ev3.motorOff(on: OutputPort.[a, b, c, d])
ev3.motorOff(on: OutputPort.[a, b, c, d], brakeAtEnd: Bool)
```

Definition

The *motorOff* function turns the motor off. It is usually used to stop a motor that was started by the 'motorOn' function earlier in the program. If withBrake is "True", the motor is stopped immediately. The motor will be held in its stopped position until another function starts it, or until the program ends. If withBrake is "False", power to the motor is simply turned off. The motor will coast using any remaining momentum until it stops, or until another function starts.

```
ev3.waitForMotorDegrees(on: OutputPort.[a, b, c, d], lessThanOrEqualTo:
Float)
ev3.waitForMotorDegrees(on: OutputPort.[a, b, c, d],
greaterThanOrEqualTo: Float)
```

Definition

The *waitForMotorDegrees* function lets your program wait for the rotation of your motor, measured in degrees, to become either greater than or less than a desired Threshold Value, before it moves on to the next instruction in the program.

```
ev3.waitForMotorDegreesChange(on: OutputPort.[a, b, c, d])
```

Definition

This function is not documented in EV3 Playground.

```
ev3.waitForMotorPower(on: OutputPort.[a, b, c, d], lessThanOrEqualTo:
Float)
ev3.waitForMotorPower(on: OutputPort.[a, b, c, d], greaterThanOrEqualTo:
Float)
```

Definition

The *waitForMotorPower* function lets your program wait for the current power given to a motor to become either greater than or less than a desired Threshold Value, before it moves on to the next instruction in the program.

```
ev3.waitForMotorRotations(on: OutputPort.[a, b, c, d], lessThanOrEqualTo:
Float)
ev3.waitForMotorRotations(on: OutputPort.[a, b, c, d],
greaterThanOrEqualTo: Float)
```

Definition

The *waitForMotorRotations* function lets your program wait for the rotation of your motor, measured in number of rotations, to become either greater than or less than a desired Threshold Value, before it moves on to the next instruction in the program.

```
ev3.waitForMotorRotationsChange(on: OutputPort.[a, b, c, d])
```

Definition

This function is not documented in EV3 Playground.

```
ev3.move(leftPort: OutputPort.[a, b, c, d], rightPort: OutputPort.[a, b,
c, d], leftPower: Float, rightPower: Float)
ev3.move(forDegrees: Float, leftPort: OutputPort.[a, b, c, d], rightPort:
OutputPort.[a, b, c, d], leftPower: Float, rightPower: Float)
ev3.move(forDegrees: Float, leftPort: OutputPort.[a, b, c, d], rightPort:
OutputPort.[a, b, c, d], leftPower: Float, rightPower: Float, brakeAtEnd:
Bool)
ev3.move(forSeconds: Float, leftPort: OutputPort.[a, b, c, d], rightPort:
OutputPort.[a, b, c, d], leftPower: Float, rightPower: Float)
ev3.move(forSeconds: Float, leftPort: OutputPort.[a, b, c, d], rightPort:
OutputPort.[a, b, c, d], leftPower: Float, rightPower: Float, brakeAtEnd:
Bool)
ev3.move(forRotations: Float, leftPort: OutputPort.[a, b, c, d],
rightPort: OutputPort.[a, b, c, d], leftPower: Float, rightPower: Float)
ev3.move(forRotations: Float, leftPort: OutputPort.[a, b, c, d],
rightPort: OutputPort.[a, b, c, d], leftPower: Float, rightPower: Float,
brakeAtEnd: Bool)
```

Definition

The *move* function can make a robot drive forward, backward, turn, or stop. Use the 'move' function for robot vehicles that have two Large Motors, with one motor driving the left side of the vehicle and the other driving the right side. You can make the two motors move at different speeds or in different directions to make your robot turn. The 'move' function turns both motors on, then immediately continues to the next instruction in the program unless the duration is specified by using the forSeconds, forDegrees or forRotations parameters. You can control the speed and direction of the motors using the leftPower and rightPower parameters. The motors will run until they are stopped or changed by another instruction later in the program, or until the program ends.

```
ev3.stopMove(leftPort: OutputPort.[a, b, c, d], rightPort: OutputPort.[a,
b, c, d])
ev3.stopMove(leftPort: OutputPort.[a, b, c, d], rightPort: OutputPort.[a,
b, c, d], withBrake: Bool)
```

Definition

The *stopMove* function turns both motors off. Use the 'stopMove' function to stop a robot that was started by the 'move' function earlier in the program. If withBrake is "True", the motors are stopped immediately. The motors will be held in their stopped position until another function starts them, or until the program ends. If withBrake is "False", power to the motors is simply turned off. The motors will coast using any remaining momentum until they stop, or until another function starts.

Motor Sensor Commands

```
ev3.measureMotorDegrees(on: OutputPort.[a, b, c, d])
```

Definition

The *measureMotorDegrees* function measures the current motor rotation in degrees. The motor rotation is measured in degrees, relative to its position at the beginning of the program or since the sensor was last reset.

```
ev3.measureMotorPower(on: OutputPort.[a, b, c, d])
```

Definition

The *measureMotorPower* function measures the current power level of the motor.

```
ev3.measureMotorRotations(on: OutputPort.[a, b, c, d])
```

Definition

The *measureMotorRotations* function measures the current motor rotation in number of rotations. The motor rotation is measured in number of rotations, relative to the beginning of the program or since the sensor was last reset.

```
ev3.resetMotor(on: OutputPort.[a, b, c, d])
```

Definition

The *resetMotor* function allows you to reset the motor rotation counter.

Gyroskope Sensor Commands

```
ev3.measureGyroAngle(on: InputPort.[one, two, three, four])
```

Definition

The *measureGyroAngle* function measures the angle using the Gyro Sensor. The angle is measured relative to the last time the sensor was reset.

```
ev3.measureGyroRate(on: InputPort.[one, two, three, four])
```

Definition

The *measureGyroRate* function measures the rotation rate in degrees per seconds using the Gyro Sensor.

```
ev3.waitForGyroAngleChange(on: InputPort.[one, two, three, four])
```

Definition

This function is not documented in EV3 Playground.

```
ev3.waitForGyroAngle(on: InputPort.[one, two, three, four],
lessThanOrEqualTo: Float)
ev3.waitForGyroAngle(on: InputPort.[one, two, three, four],
greaterThanOrEqualTo: Float)
```

Definition

The *waitForGyroAngle* function lets your program wait for the angle to become either greater than or less than a desired Threshold Value before it moves on to the next instruction in the program.

```
ev3.waitForGyroRateChange(on: InputPort.[one, two, three, four])
```

Definition

This function is not documented in EV3 Playground.

```
ev3.waitForGyroRate(on: InputPort.[one, two, three, four],
lessThanOrEqualTo: Float)
ev3.waitForGyroRate(on: InputPort.[one, two, three, four],
greaterThanOrEqualTo: Float)
```

Definition

The *waitForGyroRate* function lets your program wait for the rotation rate, measured in degrees per second, to become either greater than or less than a desired Threshold Value, before it moves on to the next instruction in the program.

```
ev3.waitForGyroRateChange(on: InputPort.[one, two, three, four])
```

Definition

This function is not documented in EV3 Playground.

```
ev3.resetGyro(on: InputPort.[one, two, three, four])
```

Definition

The *resetGyro* function allows you to reset the gyro counter.

Light Sensor Commands

```
ev3.measureLightAmbient(on: InputPort.[one, two, three, four])
```

Definition

The *measureLightAmbient* function measures the ambient light intensity using the Color Sensor.

```
ev3.measureLightColor(on: InputPort.[one, two, three, four])
```

Definition

The *measureLightColor* function detects colors using the Color Sensor.

```
ev3.measureLightReflection(on: InputPort.[one, two, three, four])
```

Definition

The *measureLightReflection* function measures the reflected light intensity using the Color Sensor.

```
ev3.waitForLightAmbient(on: InputPort.[one, two, three, four],
lessThanOrEqualTo: Float)
ev3.waitForLightAmbient(on: InputPort.[one, two, three, four],
greaterThanOrEqualTo: Float)
```

Definition

The *waitForLightAmbient* function lets your program wait for the measured ambient light intensity to become either greater than or less than a desired Threshold Value, before it moves on to the next instruction in the program.

```
ev3.waitForLightAmbientChange(on: InputPort.[one, two, three, four])
```

Definition

This function is not documented in EV3 Playground.

```
ev3.waitForLightColor(on: InputPort.[one, two, three, four], color:
ColorValue.[black, blue, brown, green, red, white, yellow])
```

Definition

The *waitForLightColor* function lets your program wait until the Color Sensor detects a specified color, before it moves on to the next instruction in the program.

```
ev3.waitForLightColorChange(on: InputPort.[one, two, three, four])
```

Definition

This function is not documented in EV3 Playground.

```
ev3.waitForLightReflection(on: InputPort.[one, two, three, four],
llessThanOrEqualTo: Float)
ev3.waitForLightReflection(on: InputPort.[one, two, three, four],
grgreaterThanOrEqualTo: Float)
```

Definition

The *waitForLightReflection* function lets your program wait for the measured reflected light intensity to become either greater than or less than a desired Threshold Value, before it moves on to the next instruction in the program.

```
ev3.waitForLightReflectionChange(on: InputPort.[one, two, three, four])
```

Definition

This function is not documented in EV3 Playground.

Touch Sensor

```
ev3.measureTouch(on: InputPort.[one, two, three, four])
```

Definition

The *measureTouch* function measures the state of the Touch Sensor. The Touch Sensor can either be Pressed or Released

```
ev3.measureTouchCount(on: InputPort.[one, two, three, four])
```

Definition

The *measureTouchCount* function measures the number of times the Touch Sensor has been bumped since the start of the program or since it was last reset.

```
ev3.resetTouchCount(on: InputPort.[one, two, three, four])
```

Definition

The *resetTouchCount* function allows you to reset the touch counter.

```
ev3.waitForTouch(on: InputPort.[one, two, three, four])
```

Definition

The *waitForTouch* function lets your program wait until the Touch Sensor is pressed, before it moves on to the next instruction in the program.

```
ev3.waitForTouchCount(on: InputPort.[one, two, three, four],
greaterThanOrEqualTo: Float)
```

Definition

The *waitForTouchCount* function lets your program wait until the Touch Sensor has been bumped for a specified number of times, before it moves on to the next instruction in the program.

```
ev3.waitForTouchReleased(on: InputPort.[one, two, three, four])
```

Definition

The *waitForTouchReleased* function lets your program wait until the Touch Sensor is released, before it moves on to the next instruction in the program.

Ultrasonic Sensor

```
ev3.measureUltrasonicCentimeters(on: InputPort.[one, two, three, four])
```

Definition

The *measureUltrasonicCentimeters* function measures the distance to an object, in centimeters, using the Ultrasonic Sensor.

```
ev3.measureUltrasonicInches(on: InputPort.[one, two, three, four])
```

Definition

The *measureUltrasonicInches* function measures the distance to an object, in inches, using the Ultrasonic Sensor.

```
ev3.waitForUltrasonicCentimeters(on: InputPort.[one, two, three, four],
lessThanOrEqualTo: Float)
ev3.waitForUltrasonicCentimeters(on: InputPort.[one, two, three, four],
greaterThanOrEqualTo: Float)
```

Definition

The *waitForUltrasonicCentimeters* function lets your program wait for the distance, measured in centimeters, to become either greater than or less than a desired Threshold Value, before it moves on to the next instruction in the program.

```
ev3.waitForUltrasonicInches(on: InputPort.[one, two, three, four],
lessThanOrEqualTo: Float)
ev3.waitForUltrasonicInches(on: InputPort.[one, two, three, four],
greaterThanOrEqualTo: Float)
```

Definition

The *waitForUltrasonicInches* function lets your program wait for the distance, measured in inches, to become either greater than or less than a desired Threshold Value, before it moves on to the next instruction in the program.

```
ev3.waitForUltrasonicDecrease(on: InputPort.[one, two, three, four])
```

Definition

The *waitForUltrasonicDecrease* function lets your program wait for the measured distance to decrease, before it moves on to the next instruction in the program.

```
ev3.waitForUltrasonicIncrease(on: InputPort.[one, two, three, four])
```

Definition

The *waitForUltrasonicIncrease* function lets your program wait for the measured distance to increase, before it moves on to the next instruction in the program.

General Commands

```
ev3.waitFor(seconds: Float)
```

Definition

The `waitFor` function lets your program wait for an amount of time, specified in seconds, before it moves on to the next instruction in the program.