ElephantROBOTICS

The RoboFlowScript Programming Language

Catalog

1. The RoboFlowScript Programming Language

The RoboFlowScript is the programming language that controls the robot. The RoboFlowScript includes variables, types, and the flow control statements. There are also built-in variables and functions that monitor and control I/O and robot movements.

1.1. Numbers, Variables, and Types

In RoboFlowScript arithmetic expression syntax is standard:

```
1+4-5
5*1/2
(4+5)*3/(7-6)
```

In boolean expressions, boolean operators are spelled out:

```
true or false and (2 == 5)
52 > 19 or 5 != -62 and -7 < 24
not 46 >= 73 and 39 <= 92
```

Variable assignment is done using the equal sign =:

```
num = 68
test = false or true and not false
num_expr = 39-90/3.19024
string = “This is string text”
p = [ 1, 2, 3, 4, 5, 6 ]
p[Z] = p[0] + 5
```

The type of a variable is deduced from the assignment of the variable. In the example above num is a Number, test is a Boolean, num_expression is a Number, string is a String, p is Array.

The RoboFlowScript types are:

- Number (floating-point number or integer)
- Boolean
- String

- Array

Array is a collection of 6 numbers: X, Y, Z coordinates of the robot and RX, RY, RZ rotation values of the toolhead.

1.2. Flow of Control

The flow of control of a program is controlled by if and switch statements:

```
if (cond1) {
    stmt1
    stmt2
} else if (cond2) {
    stmt3
    stmt4
} else {
    stmt5
    stmt6
}

switch (var) {
case 1:
    stmt1;
case2:
    stmt2;
default:
    def_stmt;
}
```

and while-loops:

```
while (num < 5) {
    num = num + 1
    stmt
}
```

1.3. Keywords

- **return** – returns from a function
- **call** – calls user-defined function
- **function** – defines user-defined function
- **import** – imports OS variable to the script scope
- **export** – exports script variable to the OS variables list
- **global** – directly access variables from the OS

1.4. Function

A function is declared as follows:

```
function add(a, b)
{
    return a + b
}
```

The function can then be called like this:

```
sum = call add(1, 4)
```

All the function arguments are passed-by-value. This means that any modification done to the content of the argument within the scope of the function will not be reflected outside that scope.

2. Built-in Functions

2.1. GetAngles

GetAngles()

Returns current joints angle values of the robot.

Parameters:

None.

Example Code:

```
angles = GetAngles()
```

2.2. GetCoords

GetCoords()

Returns current coordinates & rotation angle value of the toolhead of the robot.

Parameters:

None.

Example Code:

```
coords = GetCoords()
```

2.3. GetDigitalIn

GetDigitalIn(pin_number)

Returns requested input pin signal (0 or 1).

Parameters:

pin_number – Integer between 0-31 of input pin number

Example Code:

```
if (GetDigitalIn(5) == 1) {
    pin_7 = GetDigitalIn(7)
}
```

2.4. GetDigitalOut

GetDigitalOut(pin_number)

Returns requested output pin signal (0 or 1).

Parameters:

pin_number – Integer between 0-31 of output pin number

Example Code:

```
if (GetDigitalOut(9) == 1) {
    pin_7 = GetDigitalOut(8)
}
```

2.5. ReceiveData

ReceiveData(IP_address, port, timeout)

Receives data from the specified IP address and port and returns received data. Timeout specifies time to wait in seconds. If timeout is reached, string “connect timeout” returned.

Parameters:

IP_address – IP address as string

port – port number

timeout – time to wait for data in seconds

Example Code:

```
ReceiveData("192.168.1.154", 3001, 10.5)
```

2.6. SendData

SendData(IP_address, port, data, is_string)

Sends given data to the specified IP address and port. If is_string is true, sends data as string.

Parameters:

IP_address – IP address as string

port – port number

data – buffer with the data

is_string – boolean value. If true, sends data as string.

Example Code:

```
SendData("192.168.1.154", 3000, buffer, true)
```

2.7. SetAnalogOut

SetAnalogOut(pin_number, pin_value)

Sets requested analog output pin to the specified value.

Parameters:

pin_number – Integer between 0-31 of output analog pin number

pin_value – Floating-point number

Example Code:

```
SetAnalogOut(2, 3.1415)
```

2.8. SetAngle

SetAngle(joint, angle, speed)

Moves requested joint to the specified angle with the given speed.

Parameters:

joint – J1-J6 or integer 0-5 representing joint number

angle – angle of the joint to move to

speed – speed of moving of the joint

Example Code:

```
SetAngle(J1, 30.33, 3000)
```

2.9. SetAngles

SetAngles(angles, speed)

Moves robot joints to the specified position with the given speed. Position is specified with angle values of each joint

Parameters:

angles – angle values of each joint

speed – speed of moving of the robot

Example Code:

```
SetAngles(angles, 3000)
```

2.10. SetCoord

SetCoord(axis, coord, speed)

Moves robot along requested axis to the specified coordinate with the given speed.

Parameters:

axis – any of X, Y, Z, RX, RY, RZ or integer 0-5 representing axis by number

coord – coordinate of the specified axis where to move the robot

speed – speed of moving along the requested axis

Example Code:

```
SetCoord(X, 250.22, 3000)
```

2.11. SetCoords

SetCoords(coords, speed)

Moves robot to the specified position with the given speed. Position is specified with coordinates of each axis.

Parameters:

coords – array of coordinates for each axis

speed – speed of moving of the robot

Example Code:

```
SetCoords(coords, 3000)
```

2.12. SetDigitalOut

SetDigitalOut(pin_number, pin_signal)

Sets requested digital output pin to the specified value.

Parameters:

pin_number – Integer between 0-31 of output digital pin number

pin_signal– Integer 0 or 1 value

Example Code:

```
SetDigitalOut(12, 1)
```

3. Built-in Constants

3.1. true & false

Represent boolean values true and false and can be used in variable assignments, conditions and expressions.

3.2. X, Y, Z, RX, RY, RZ

Can be used to access and set individual elements of array using respective coordinates and rotation values.

3.3. J1, J2, J3, J4, J5, J6

Can be used to access and set individual elements of array using angle values of each joint.