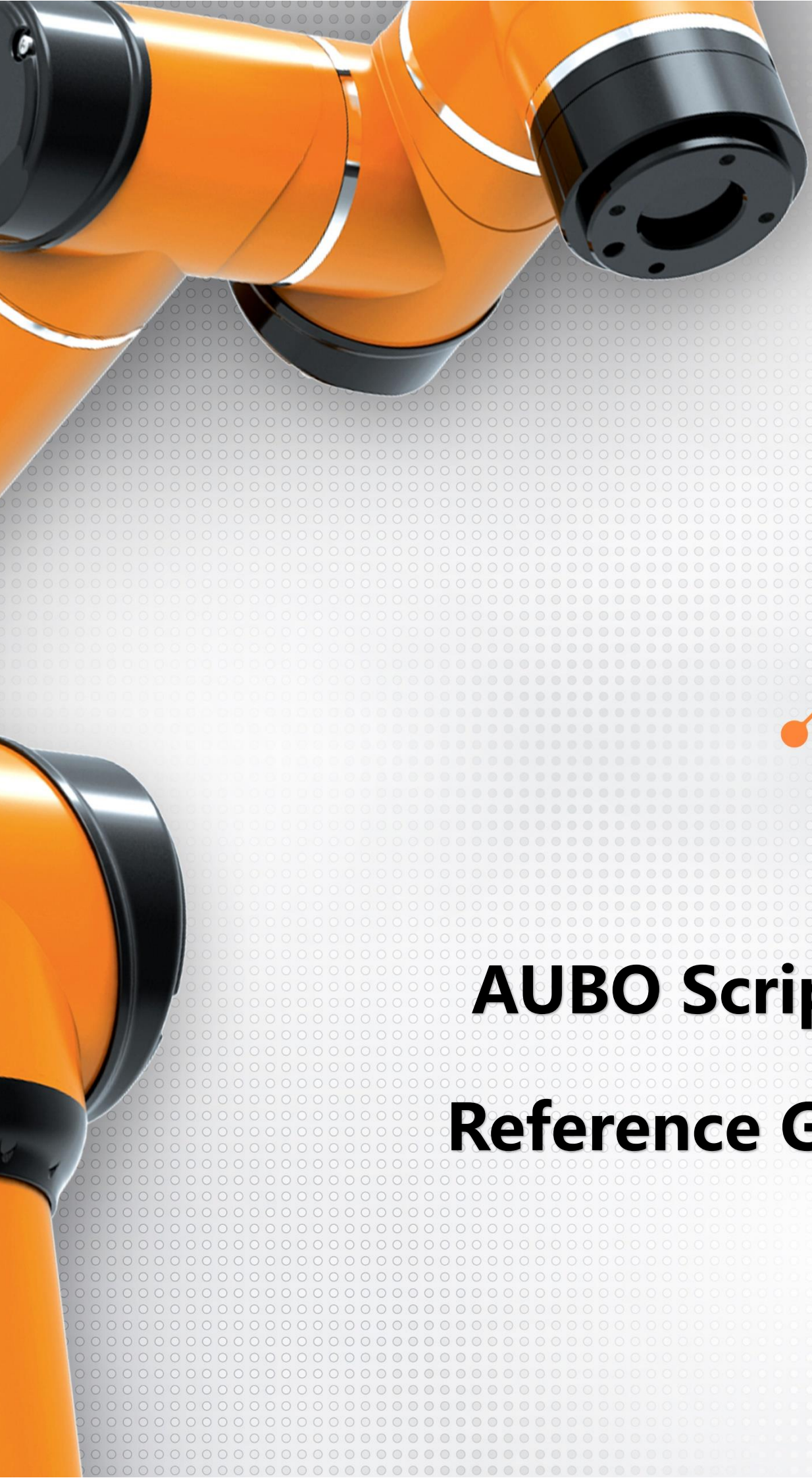


AUBO Scripting Reference Guide

遨博智能
AUBO

AUBO - i series

Scripting Reference Guide

Translation Version 1.0.0

Published by AUBO (Beijing) Robotics Technology Co, Ltd.

This manual is applicable to AUBORPE V 4.5. For details, please refer to the version information section of this manual. Please check the actual product version information carefully before use to ensure consistency.

The user manual will be checked and corrected periodically, and the updated content will appear in the new version. The contents or information in this manual are subject to change without notice.

AUBO (Beijing) Robotics Technology Co., Ltd. shall not be held liable for any errors or omissions that may appear in this manual, or for accidental or consequential damages resulting from the use of this manual and the products described therein.

Please read this manual before installing and using the product.

Please keep this manual in a safe place so that you can read and refer to it at any time.

All pictures in this manual are for illustrative purposes only, please refer to the actual product received.

Contents

1 Lua Foundation

1.1 Lua Introduction

1.1.1 Introduction to Lua Language

Lua was developed in 1993 by a research team at the Pontifical Catholic University of Rio de Janeiro. It is designed to be embedded in the application to provide flexible extension and customization capabilities for the application.

1.1.2 Lua Language Function

Explanatory language: Translate the program into machine language at runtime.

Lightweight: simple, small but powerful.

Scalable: Lua provides very easy to use extension interfaces and mechanisms

1.1.3 Lua Language Application Scenario

Game development

Extensions and database plugins

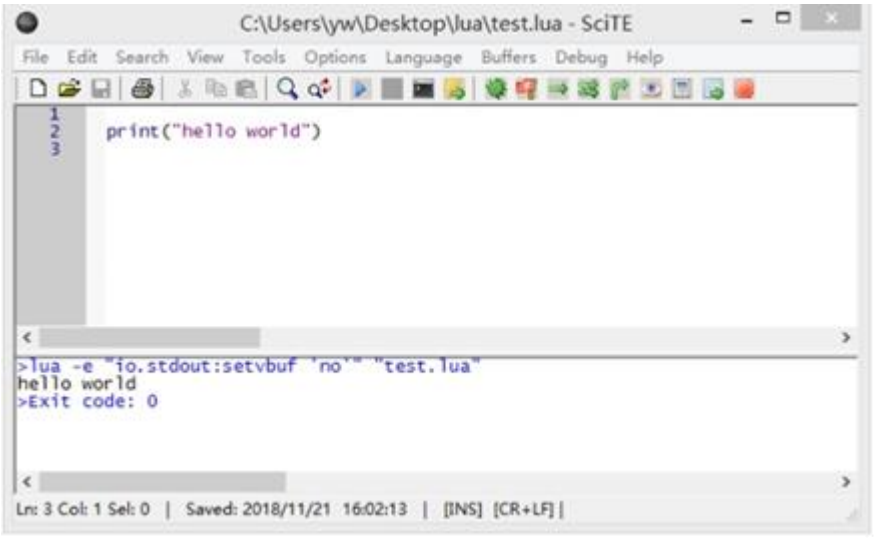
Security system, etc.

1.2 Lua Programming Environment

1.2.1 Programming Environment Construction in Windows

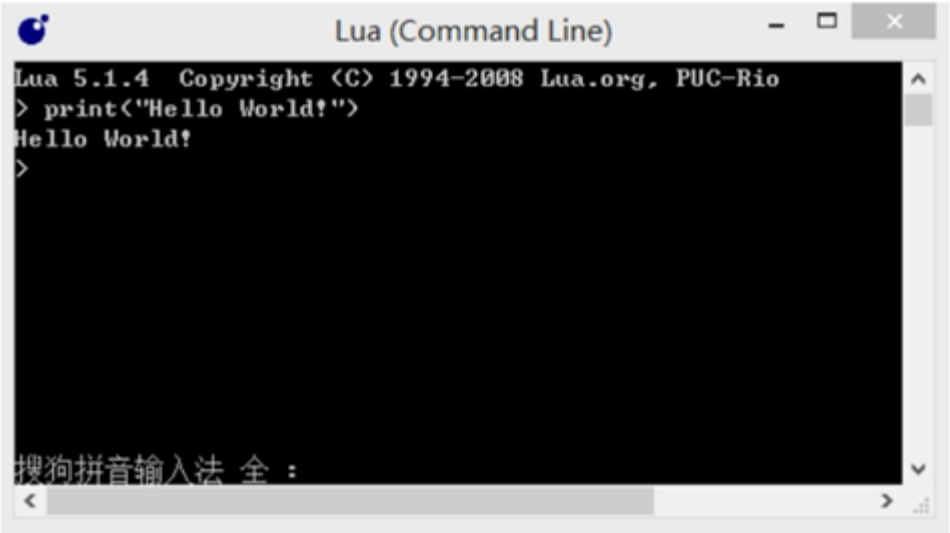
Download and install Lua's IDE: SciTE.

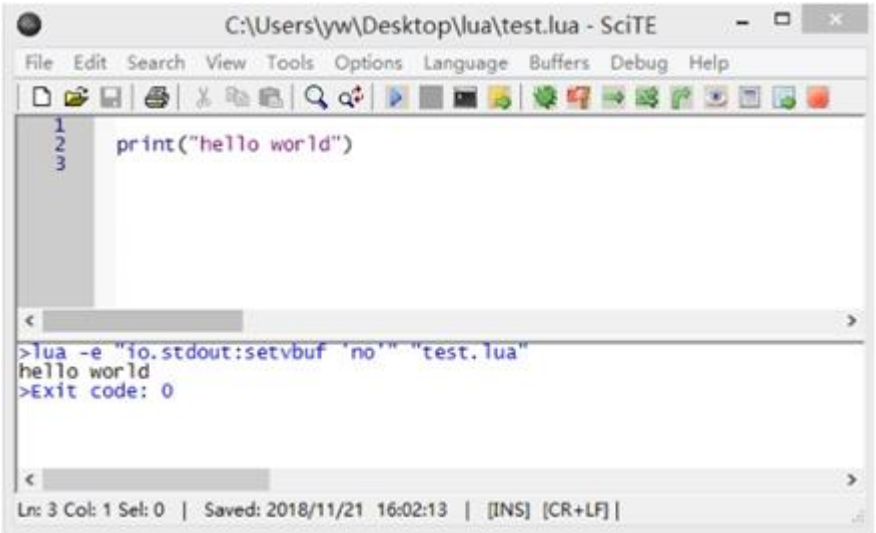
Link: https:/Lua-for-windows.updatestar.com/



1.2.2 Lua Programming Mode

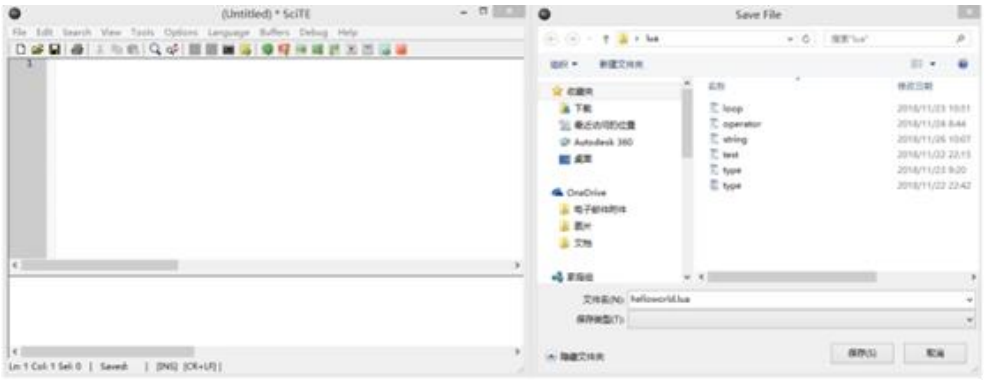
Lua has two programming methods: interactive (command line) and script.

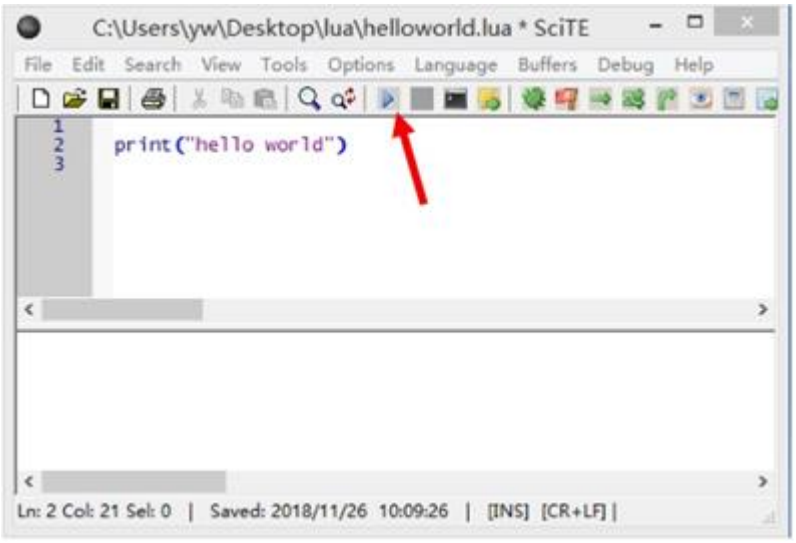




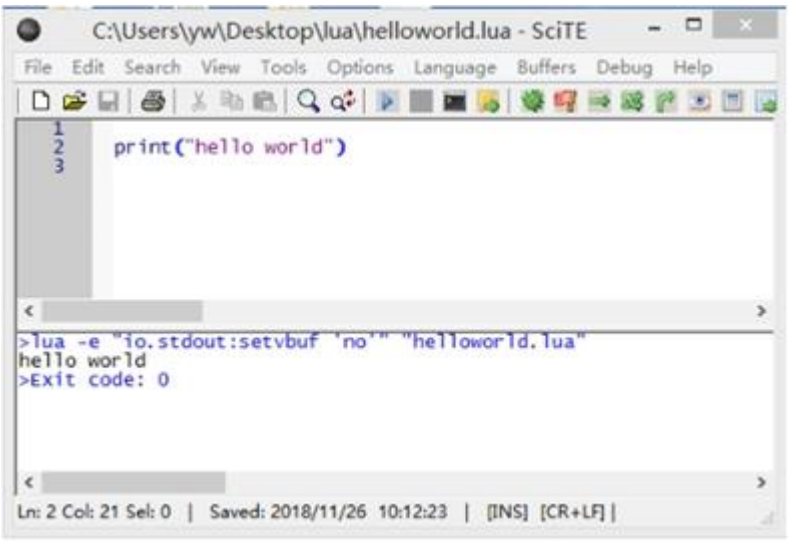
1.2.3 Build a Hello World Program

Create a new project and save it as helloworld.Lua





The output window below prints hello world



1.3 Lua Simple Syntax

1.3.1 Comment

--: Single line comment

--[[ ]]: Multi-line comment

```
--Single line comment
--[[Multi-line comment
    second line
    The third row]]
print("hello world")
```

1.3.2 Identifier

Identifiers are used to define a constant, variable, etc.

Identifiers begin with a letter or an underscore and consist of letters, numbers, and underscores

Lua does not allow special characters such as @, $, % to define identifiers

Lua is a case-sensitive programming language

```
--The following is the use of legal identifiers
abc = 0
Abc = 0
_abc = 0
a_bc = 0
abc1 = 0
abc_1 = 0
--The following is the use of illegal identifiers, will report an error
1abc = 0
abc@ = 0
%abc = 0
```

1.3.3 Reserved Words

Lua has some reserved keywords that cannot be used as identifiers, as follows.

and	**not**	**or**	**break**	**return**	**do**	**then**
if	**else**	**elseif**	**end**	**true**	**false**	**for**

while	repeat	until	in	function	goto	local

1.4 Lua Data Type

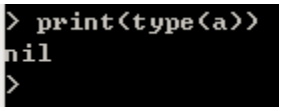
There are 6 types we use.

Use the type () function to see the data type of an object.

Data Type	Description
nil	Represents an invalid value, which is equivalent to false in the conditional expression
boolean	Contains two values, true and false
number	Represents double precision floating point numbers
string	String, represented by "" or ‘’
function	Custom function body
table	Table, represented by { }

1.4.1 Nil Type

1. The nil type indicates that there are no valid values, such as a variable without an assignment.



2. Assigning a value to a global variable or a variable in a table is equivalent to deleting them.


```lua
print("Before assigning nil...")
table = {1,2,3}
for k,v in pairs(table) do
    print(k.."-"..v)
end

print("After assigning nil...")
table[1]=nil
for k,v in pairs(table) do
    print(k.."-"..v)
end
```

```
- Before assigning nil...
- 1-1
- 2-2
- 3-3
- After assigning nil...
- 2-2
- 3-3
```

3. In the command line programming environment, nil should be double quoted when used as a comparison.

```
> print(type(a))
nil
> print(type(a) == nil)
false
> print(type(a) == "nil")
true
```

1.4.2 Boolean Type

Boolean has only two values, true (true) / false (false), Lua treats false and nil as "false", and other values are treated as "true".

```lua
--nil为假
a = nil
if a then
    print("a is true")
else
    print("a is false")
end

--false为假
b = false
if b then
    print("b is true")
else
    print("b is false")
end

--其他为真
c = 1
if c then
    print("c is true")
else
    print("c is false")
end
```

```
>lua -e "io.stdout:setvbuf 'no'" "type.lua"
a is false
b is false
c is true
>Exit code: 0
```


1.4.3 Number Type

Lua has only one type of number, double double-precision floating point.

```
30  print(type(1))
31  print(type(0.5))
32  print(type(1e+2))
```

```
>lua -e "io.stdout:setvbuf 'no'" "type.lua"
number
number
number
>Exit code: 0
```

1.4.4 String Type

1. Use a pair of double quotes "" to represent a string

```
37  str1 = "abc"
38  print(type(str1))
39  print(str1)
```

```
>Exit code: 0
>lua -e "io.stdout:setvbuf
string
abc
>Exit code: 0
```

2. Use [[ ]] to represent a string

```
44  str2 =
45  [[
46      abc
47      def
48      ghi
49  ]]
50  print(type(str2))
51  print(str2)
```

```
>lua -e "io.stdout:setvbuf 'no'"
string
        abc
        def
        ghi

>Exit code: 0
```

3. Measure string length with #

```
57
58  str3 = "abcde"
59  print(#str3)
```

```
>lua -e "io.stdout:
5
>Exit code: 0
```

4. ..Connect two strings


```
63  a = "abc"
64  b = "123"
65  print(a..b)
```

```
>lua -e "io.stdou
abc123
>Exit code: 0
```

1.4.5 Function Type

1. Custom function

```
62  --function
63
64  function add(a,b)
65      return a+b
66  end
67
68  c = add(2,3)
69  print(c)
```

```
>lua -e "io.stdout:setvbuf
5
>Exit code: 0
```

2. Functions can be stored in variables

```
62  --function
63
64  function add(a,b)
65      return a+b
66  end
67
68  c = add(2,3)
69  print("c = "..c)
70
71  tmp = add
72  d = tmp(1,2)
73  print("d = "..d)
```

```
>Exit code: 0
>lua -e "io.stdout:setvbuf '
c = 5
d = 3
>Exit code: 0
```

1.4.6 Table Type

1. Construct an empty table

```
77  --table
78
79  table1 = {}
```

2. Construct a table with elements

```
77  --table
78
79  table1 = {1,2,3,4}
```

3. Read an element in the table

```
77  --table
78
79  table1 = {2,3,4,5}
80  print(table1[1])
```

```
>lua -e "io.stdout:set
2
>Exit code: 0
```

4. Modify an element in the table

```
77  --table
78
79  table1 = {2,3,4,5}
80  print(table1[1])
81
82  table1[1] = 10
83  print(table1[1])
```

```
>lua -e "io.stdout:set
2
10
>Exit code: 0
```

5. Add a new element to the table and insert 0 in the second position of the table.

```
95  table1 = {1,2,3}
96  print("原table:")
97  for k,v in pairs(table1) do
98      print(k.." - "..v)
99  end
100
101 table.insert(table1,2,0)
102 print("新table:")
103 for k,v in pairs(table1) do
104     print(k.." - "..v)
105 end
```

```
>lua -e "io.stdout:se
原table:
1 - 1
2 - 2
3 - 3
新table:
1 - 1
2 - 0
3 - 2
4 - 3
>Exit code: 0
```

6. Delete the second element of the table


```
table1 = {1,2,3}
print("原table:")
for k,v in pairs(table1) do
    print(k.." - "..v)
end

table.remove(table1,2)
print("新table:")
for k,v in pairs(table1) do
    print(k.." - "..v)
end
```

```
>Exit code: 0
>lua -e "io.stdout:se
原table:
1 - 1
2 - 2
3 - 3
新table:
1 - 1
2 - 3
>Exit code: 0
```

1.5 Lua Control Flow Statement

1.5.1 While Loop

```
While (exp) do
    --[[Executed statement]]
end
```

Example: Print 1~9.

```
a = 1
while(a<10) do
    print("a = "..a)
    a = a + 1
end
```

```
>lua -e "io.stc
a = 1
a = 2
a = 3
a = 4
a = 5
a = 6
a = 7
a = 8
a = 9
>Exit code: 0
```

1.5.2 For Loop

Numerical for loop

```
for init, max/min value, increment
do
    --[[Executed statement]]
end
```

Example: Print 1~10.

```
12  for a=1,10,1 do
13      print("a = "..a)
14  end
```

```
>lua -e "io.stdout:setvbu
a = 1
a = 2
a = 3
a = 4
a = 5
a = 6
a = 7
a = 8
a = 9
a = 10
>Exit code: 0
```

Generic for loop

Iterate through all the values through an iterator function.

Example: Print all the elements of the table.

```
18  table1 = {"a","b","c","d"}
19  for k,v in pairs(table1) do
20      print(k.." - "..v)
21  end
```

```
>lua -e "io.stdou
1 - a
2 - b
3 - c
4 - d
>Exit code: 0
```

1.5.3 Repeat... until Loop

```
repeat
    --[[Loop execution of the statement when the exp expression is false]]
until(exp)
```

Example: Print 1~10.

```
27  a = 1
28  repeat
29      print("a = "..a)
30      a = a + 1
31  until(a>10)
```

```
>lua -e "io.stdout:s
a = 1
a = 2
a = 3
a = 4
a = 5
a = 6
a = 7
a = 8
a = 9
a = 10
```

1.5.4 If...else Selection Structure

```
if (exp1) then
    --[[Execute the statement when the exp1 expression is true]]
elseif (exp2) then
    --[[Execute the statement when the exp2 expression is true]]
```



```
else

    --[[Execute the statement when other conditions are met]]

end
```

Example:

```
34  --if...else
35  a = -2
36  if a>0 then
37      print(a.." > 0")
38  elseif a == 0 then
39      print(a.." == 0")
40  else
41      print(a.." < 0")
42  end
```

```
>lua -e "io.stdout:se
-2 < 0
>Exit code: 0
```

1.6 Lua Operator

1.6.1 Arithmetic Operators

+	Addition
*	Multiplication
-	Subtraction
/	Floating point division
//	Rounding down division
%	Modulo
^	Power
-	Take a negative

Example:

```
5   a = 3
6   b = 2
7   print("a + b = "..a+b)
8   print("a - b = "..a-b)
9   print("a * b = "..a*b)
10  print("a / b = "..a/b)
11  print("a % b = "..a%b)
12  print("a ^ b = "..a^b)
13  print("-a = "..-a)
```

```
>lua -e "io.stdout:
a + b = 5
a - b = 1
a * b = 6
a / b = 1.5
a % b = 1
a ^ b = 9
-a = -3
>Exit code: 0
```

1.6.2 Relational Operators

==	**Equal**

~=	**Not equal to**
<	**Less than**
>	**Greater than**
<=	**Less or equal to**
>=	**Greater or equal to**

Example:

```
20  a = 3
21  b = 2
22  print("(a == b) is "..tostring(a==b))
23  print("(a ~= b) is "..tostring(a~=b))
24  print("(a > b) is "..tostring(a>b))
25  print("(a < b) is "..tostring(a<b))
26  print("(a >= b) is "..tostring(a>=b))
27  print("(a <= b) is "..tostring(a<=b))
```

```
>lua -e "io.stdout
(a == b) is false
(a ~= b) is true
(a > b) is true
(a < b) is false
(a >= b) is true
(a <= b) is false
>Exit code: 0
```

1.6.3 Logical Operators

and
or
not

Example:

```
32  a = true
33  b = false
34  print("(a and b) is "..tostring(a and b))
35  print("(a or b) is "..tostring(a or b))
36  print("(not a) is "..tostring(not a))
```

```
>lua -e "io.stdout:set
(a and b) is false
(a or b) is true
(not a) is false
>Exit code: 0
```

1.6.4 Other operators

..	**Connect two characters**
#	**Returns the length of a string or table**

Example

```
a = "abc"
b = "def"
c = a..b
print("c : "..c)
print("c的长度: "..#c)
```

```
>lua -e "io.stdout:se
c : abcdef
c的长度: 6
>Exit code: 0
```

1.7 Lua String

1.7.1 Lua String Representation

The Lua language string can be represented in three ways:

A string of characters between single quotes

A string of characters between double quotes

A string of characters between [[and]]

Example:

```
a = " abc "
b = ' abc '
c = [[ abc ]]

print(a)
print(b)
print(c)
```

```
>lua -e "io.std
 abc
 abc
 abc
>Exit code: 0
```

1.7.2 Commonly Used Escape Characters

The escape character is used to indicate characters that cannot be directly displayed, such as carriage returns.

Commonly used escape characters and meaning:

\n	Line feed (LF)
\r	Enter (CR)
\\	Backslash character \
\’	Represents a single quote ‘
\’’	Represents a double quote ‘’

Example:

```
17
18  a = " abc\n\r"
19  b = " \'abc\' "
20  c = " \"abc\" "
21  d = " \\abc\\ "
22
23  print(a)
24  print(b)
25  print(c)
26  print(d)
```

```
>lua -e "io.stdout:
 abc


 'abc'
 "abc"
 \abc\
>Exit code: 0
```

1.7.3 Common String Operations

1. tonumber(str): The string str is rotated to number, for example, "123" is converted to 123, which facilitates data participation in calculation processing.

```
37  --string to number
38   a = "123"
39   print("type(a) is "..type(a))
40   print(a)
41
42   b = tonumber(a)
43   print("type(b) is "..type(b))
44   print(b)
```

```
>lua -e "io.stdout:s
type(a) is string
123
type(b) is number
123
>Exit code: 0
```

2. tostring(num): The number num is converted into a string, for example, 123 is converted to "123" for data transmission.

```
47   --number to string
48   a = 123
49   print("type(a) is "..type(a))
50   print(a)
51
52   b = tostring(a)
53   print("type(b) is "..type(b))
54   print(b)
```

```
>lua -e "io.stdout:set
type(a) is number
123
type(b) is string
123
>Exit code: 0
```

3. string.format(): Returns a formatted string like printf. Improve code readability.

```
59  --string.format
60
61  print(string.format("%s is %d years old.","Amy",8))
62
63  print(string.format("数字 %d 对应的ASCII字符是 %c",65,65))
64
65  print(string.format("十进制数 %d 对应的 十六进制数为 %X",15,15))
```

```
>lua -e "io.stdout:setvbuf 'no'" "s
Amy is 8 years old.
数字 65 对应的ASCII字符是 A
十进制数 15 对应的 十六进制数为 F
>Exit code: 0
```

The escape code that the format string may contain is:

%c - accepts a number and converts it to the corresponding character in the ASCII table

%d, %i - accept a number and convert it to a signed integer format

%o - accept a number and convert it to octal number format

%u - accept a number and convert it to an unsigned integer format

%x - accept a number and convert it to hexadecimal format, using lowercase letters

%X - accepts a number and converts it to a hexadecimal number format, using upperca letters

%e - accept a number and convert it to scientific notation format, using lowercase e

%E - accept a number and convert it to scientific notation format, using capital letter E

%f - accept a number and convert it to float format

%g(%G) - accepts a number and converts it to a shorter one of %e (%E, correspondi to %G) and %f

%q - accepts a string and converts it to a format that can be safely read by the Lua compil

%s - accepts a string and formats the string according to the given argument

4. string.len(str): Returns the length of the string str.

```
64    --string.len
65
66    a = "abcde"
67    print("the length of string a is "..string.len(a))
```

```
>lua -e "io.stdout:setvbuf 'r
the length of string a is 5
>Exit code: 0
```

5. string.sub(str,i,j): Returns the string between the i-th character and the j-th character of the string str.

```
a = "abcdefg"
b = string.sub(a,3,6)
print(b)
c = string.sub(a,3,-2)
print(c)
```

```
>lua -e "io.stdout:se
cdef
cdef
>Exit code: 0
```

6. string.find(str,substr,[init,[end]]):

Find the specified content substr in the target string str. If found, returns the starting index and ending index of substr, otherwise returns nil. Init is the pass parameter, indicating that the start position of the search is specified. This parameter defaults to 1.

```
--string.find

a = "hello world"

print("find string b:")
b = "world"
start_b,end_b = string.find(a,b)
print(start_b)
print(end_b)

print("find string c:")
c = "worlds"
start_c,end_c = string.find(a,c)
print(start_c)
print(end_c)
```

```
>lua -e "io.stdou
find string b:
7
11
find string c:
nil
nil
>Exit code: 0
```

7. string.split(str,delimiter):

Separate the target string str by the delimiter separator and return a table.

```
--string.split

function string.split(str,delimiter)
    if str==nil or str=='' or delimiter==nil then
        return nil
    end
    local result = {}
    for match in (str..delimiter):gmatch("(.-)"..delimiter) do
        table.insert(result,match)
    end
    return result
end

a = "5,6,7,8"
table1 = string.split(a,",")
for k,v in ipairs(table1) do
    print(k.." - "..v)
end
```

```
>lua -e "io.stdc
1 - 5
2 - 6
3 - 7
4 - 8
>Exit code: 0
```

1.8 Lua Language Experiment Operation

1.8.1 Install SciTE. in Windows

Omit

1.8.2 Write a Lua Program that Prints "hello world".

Omit

1.8.3 Custom Function: Returns the Maximum and Minimum Values of 3 Numbers.

```
 2  --test_03
 3
 4  function max_min(a,b,c)
 5      if( a > b ) then
 6          max = a
 7          min = b
 8      else
 9          max = b
10          min = a
11      end
12      if( c > max ) then
13          max = c
14      elseif( c < min ) then
15          min = c
16      end
17
18      return max,min
19  end
20
21  a,b = max_min(3,2,1)
22  print("max = "..max)
23  print("min = "..min)
24
```

```
>lua -e "io.stdout:setvbuf 'no'" "test03.lua"
max = 3
min = 1
>Exit code: 0
```

1.8.4 Create a table

Create a table of 20 elements with elements from 1 to 20. Then print out the 20 elements in the table.

```
--test_04

--创建table
table1 = {}
for i=1,20,1 do
    table1[i] = i
end

--打印table
for j=1,#table1,1 do
    print(table1[j])
end
```

```
>lua -e "io.stdout:setvbuf '
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
>Exit code: 0
```

1.8.5 Printing a Fibonacci sequence within 100

Fibonacci sequence law: 1,1,2,3,5,8,13...

```
--test_05

a = 1
b = 1

print( a )
c = b

while ( c < 100 ) do
    print( c )
    c = a + b
    a = b
    b = c
end
```

```
>lua -e "io.stdou
1
1
2
3
5
8
13
21
34
55
89
>Exit code: 0
```

1.8.6 Convert a byte hexadecimal number to a binary number and print it out.

E.g 0xff -> 11111111,0x07 -> 00000111。


```lua
--test_06

function hex2bin(hex)
    i = 0
    binstr = ''
    while i<=7 do
        if hex >= 0 then
            quotient = hex / 2  --求商
            remainder = hex % 2  --求余数
            hex = math.floor(quotient) --商取整
            binstr = binstr..tostring(remainder)
        else
            binstr = binstr..'0'
        end
        i = i + 1
    end
    return string.reverse(binstr)   --字符串取反
end

print(hex2bin(0x0f))

```

```
>lua -e "io.stdout:setvbuf '
00001111
>Exit code: 0
```

1.8.7 Print the following graphic.

```
>lua -e "io.std
      *
     ***
    *****
   *******
  *********
***********
  *********
   *******
    *****
     ***
      *
>Exit code: 0
```

```lua
--test_07

--2n-1行  2n-1列的菱形
function draw(n)
    tmp = ""
    for i = 1,2*n-1,1 do
        --菱形上一半
        if( i <= n ) then
            for j = 1,2*n-1,1 do
                if( j<n-i+1 ) or ( j>n+i-1 ) then
                    tmp = tmp.." "
                else
                    tmp = tmp.."*"
                end
            end
        --菱形下一半
        else
            for j = 1,2*n-1,1 do
                if( j<i-n+1 ) or ( j>3*n-i-1 ) then
                    tmp = tmp.." "
                else
                    tmp = tmp.."*"
                end
            end
        end

        print(tmp)
        tmp = ""
    end
end

draw(6)

```

1.8.8 Generate a string.

There are three numbers a, b, c, such as a=100, b=1, c=0.8, and generate a fixed format string '\[a,b,c]\', ie '\[100,1,0.8] \'.

```lua
--test_08

function format(a,b,c)
    return "\\["..tostring(a)..","..tostring(b)..","
        ..tostring(c).."]\\"
end

print(format(100,1,0.8))

```

```
>lua -e "io.stdout:setvbuf 'no'" "
\[100,1,0.8]\
>Exit code: 0
```

1.8.9 Splitting the string for processing

Split the string '[a,b,c]' into three numbers a, b, and c. For example, '[100, 1, 0.8]' is splited into 100, 1, 0.8.


```lua
--test_09

function string.split(str,delimiter)
    if str==nil or str=='' or delimiter==nil then
        return nil
    end
    local result = {}
    for match in (str..delimiter):gmatch("(.-)"..delimiter) do
        table.insert(result,match)
    end
    return result
end

a = "[100,1,0.8]"
--去除中括号[ ]
b = string.sub(a,2,-2)
--按照, 分割
table1 = string.split(b,",")
--转换成数字
table1[1] = tonumber(table1[1])
table1[2] = tonumber(table1[2])
table1[3] = tonumber(table1[3])
--打印
for i=1,#table1,1 do
    print(table1[i])
end
```

```
>lua -e "io.stdout:set
100
1
0.8
>Exit code: 0
```

1.8.10 Query string.

If found, the start and end addresses of one or more target strings are printed. If it is not found, print "Not found!". For example: look for 'Lua' in ‘Lua1 Lua2 Lua3’ and return 1 3; 6 8; 11 13 . When found:

```lua
--test_10

function findString(str,substr)
    i = 1
    while ( i < #str ) do
        m_start,m_end = string.find(str,substr,i)
        if( m_start ~= nil ) then
            print(m_start.." ~ "..m_end)
            i = m_end + 1
        else
            if( i == 1 ) then
                print("未找到！")
            end
            break
        end
    end
end

findString("lua1 lua2 lua3","lua")
```

```
>lua -e "io.stdout:se
1 ~ 3
6 ~ 8
11 ~ 13
16 ~ 18
>Exit code: 0
```

When not found:

```lua
--test_10

function findString(str,substr)
    i = 1
    while ( i < #str ) do
        m_start,m_end = string.find(str,substr,i)
        if( m_start ~= nil ) then
            print(m_start.." ~ "..m_end)
            i = m_end + 1
        else
            if( i == 1 ) then
                print("未找到！")
            end
            break
        end
    end
end

findString("lua1 lua2 lua3","lua4")

```

```
>lua -e "io.stdout:setvbu
未找到！
>Exit code: 0
```


2 AUBO Script Programming

2.1 Introduction to AUBO-Script Programming

The AUBO robot can be controlled by the graphical interface of the teach pendant or by script.

AUBO-Script is a scripting language developed on the basis of Lua, supporting Lua language syntax, such as variable types, control flow statements and function definitions.

AUBO-Script has built-in functions for detecting and controlling the I/O and motion of the robot.

2.2 Environment Construction

2.2.1 Scripting Method

Write and execute directly on the teach pendant.

2.2.2 Teach Pendant Script Programming Method

1. [Online Programming]->[Script]->[New]
2. Write the program

```
init_global_move_profile()
set_joint_maxacc({1,1,1,1,1,1})
set_joint_maxvelc({1,1,1,1,1,1})

i=0
while(i<5)do
    move_joint({0,0,0,0,0,0},true)
    sleep(1)
    move_joint({-0.000003, -0.127267, -1.321122, 0.376934, -1.570796, -0.000008},true)
    sleep(1)
    i=i+1
    print(i)
end
```

3. Save the script, for example named <move_joint>
4. [Project]->[New]->[Conditions]->[Advanced Conditions]->Script
5. Select the script file "move_joint" to confirm

6. Save the project.
7. Click [Start] to execute the project.

3 AUBO Script Precautions

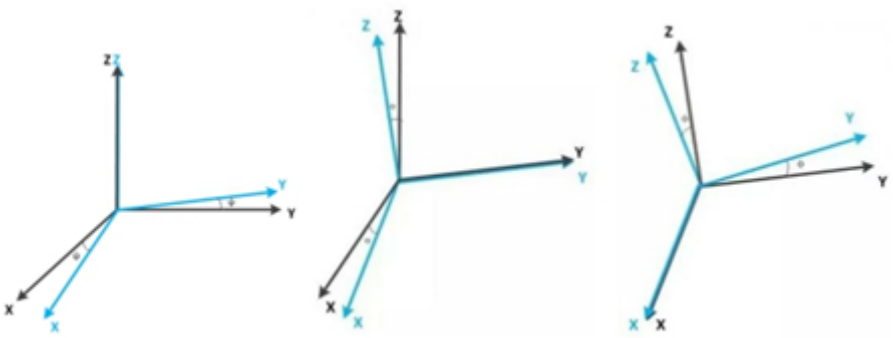
3.1 Unit

Physical Quantity	Unit
Joint angle	rad
Distance	m
Time	s
Joint acceleration	rad/s^2
Joint speed	rad/s
End acceleration	m/s^2
End speed	m/s
Attitude	Quaternion wxyz
Weight	kg

3.2 Attitude Parameters: Quaternion and Euler Angles.

Euler angle: Describes the configuration of the fixed-point rotating rigid body. Three independent coordinate variables are needed to decompose the fixed-point rotation into three independent fixed-axis rotations. The corresponding three independent angles are Euler angles.

The Euler angle of the AUBO robot is rotated in the order of ZYX. For example, the reference coordinate system first rotates RZ around its own Z axis, then rotates RY around the Y axis of the coordinate system after the rotation, and finally rotates RX around the X axis of the coordinate system after the rotation, and the tool coordinate system The direction is the same.



Quaternion: Multiple rotations around the coordinate axis can be equivalent to rotating a certain angle around a certain axis of rotation. Assuming the equivalent rotation axis direction vector →= [, ,], the equivalent rotation angle is θ, then the quaternion q = (, , ,), where

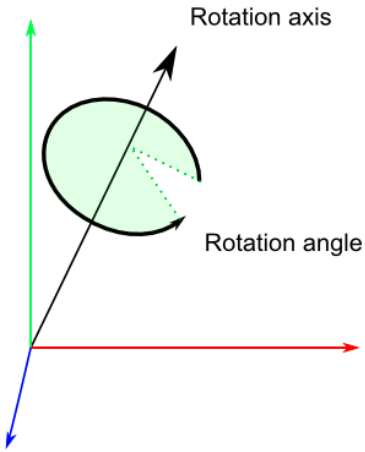
$$x = k_x \cdot \sin\frac{\theta}{2}$$

$$y = k_y \cdot \sin\frac{\theta}{2}$$

$$z = k_z \cdot \sin\frac{\theta}{2}$$

$$w = \cos\frac{\theta}{2}$$

And $x^2+y^2+z^2+w^2=1$



Characteristics of Euler angles and quaternions

	Euler Angle	Quaternion
Advantage	Easy to understand, intuitive	Can avoid the phenomenon of universal joint lock higher efficiency
Disadvantage	Inefficient than quaternion Will cause the	Understanding is more difficult, not intuitive

Euler angles and quaternions transform each other.

AUBO Script provides conversion interface functions: rpy2quaternion and quaternion2rpy. The mathematical calculation of rpy2quaternion (rx, ry, rz unit is rad):

$$w = \cos\frac{rx}{2}\cdot\cos\frac{ry}{2}\cdot\cos\frac{rz}{2} + \sin\frac{rx}{2}\cdot\sin\frac{ry}{2}\cdot\sin\frac{rz}{2}$$

$$x = \sin\frac{rx}{2}\cdot\cos\frac{ry}{2}\cdot\cos\frac{rz}{2} - \cos\frac{rx}{2}\cdot\sin\frac{ry}{2}\cdot\sin\frac{rz}{2}$$

$$y = \cos\frac{rx}{2}\cdot\sin\frac{ry}{2}\cdot\cos\frac{rz}{2} + \sin\frac{rx}{2}\cdot\cos\frac{ry}{2}\cdot\sin\frac{rz}{2}$$

$$z = \cos\frac{rx}{2}\cdot\cos\frac{ry}{2}\cdot\sin\frac{rz}{2} - \sin\frac{rx}{2}\cdot\sin\frac{ry}{2}\cdot\cos\frac{rz}{2}$$

Mathematical calculation of quaternion2rpy (rx, ry, rz units are rad):

$$rx = \tan^{-1}\frac{2(wx + yz)}{1 - 2(x^2 + y^2)}$$

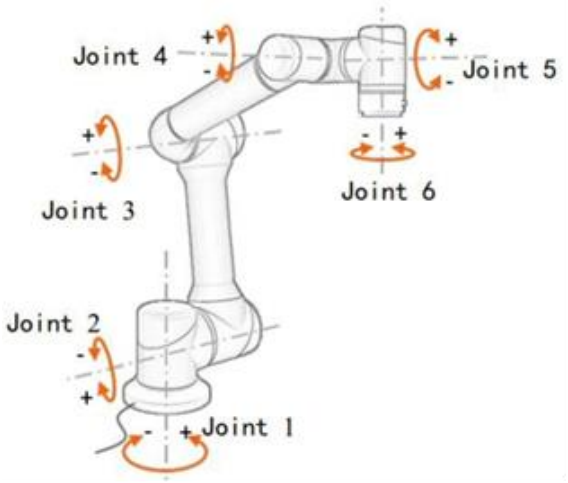
$$ry = \sin^{-1}(2(wy - zx))$$

$$rx = \tan^{-1}\frac{2(wz + xy)}{1 - 2(y^2 + z^2)}$$

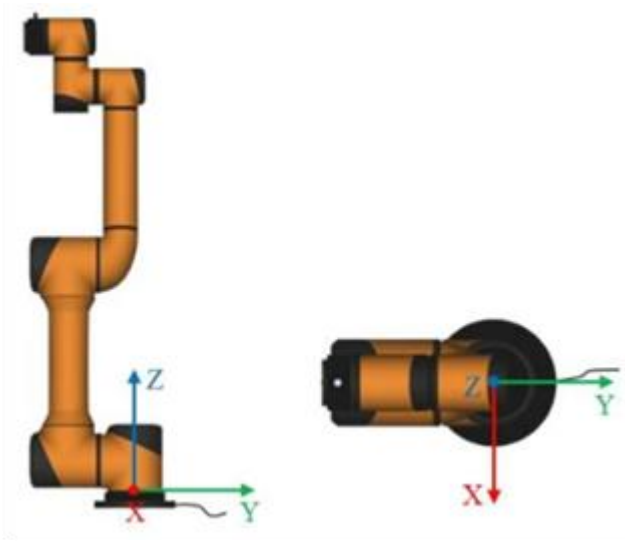
3.3 Introduction to the Coordinate System

AUBO robot has five coordinate systems: the joint coordinate system, the base coordinate system, the End coordinate system, the tool coordinate system, and the user coordinate system.

Joint coordinate system. Each joint rotates from -175° to +175°.



The Base coordinate system is fixed at the center of the robot mounting flange. The user coordinate system is set based on the Base coordinate system.

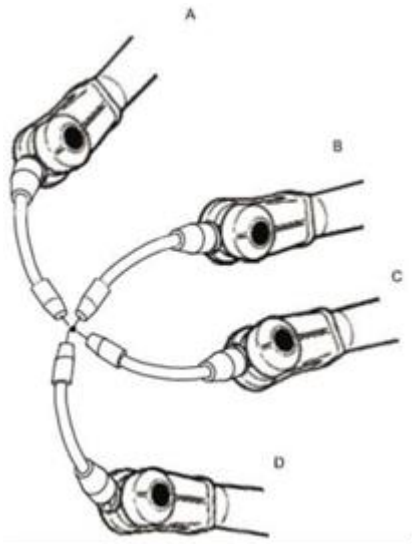


The End coordinate system (flange_center) is fixed at the center of the flange of the end of the arm and will change as the arm moves.

Tool coordinate system. Used to define the position and tool pose of the tool center point (TCP). The tool coordinate system needs to be established before use. The default tool coordinate system of the AUBO robot is the End coordinate system.

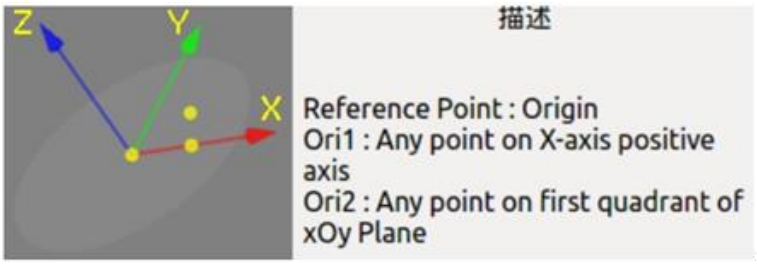
The AUBO robot supports multiple tool coordinate systems and can be switched in real time in the program.

Tool coordinate system TCP point position calibration: 4+ point calibration method, position unchanged, change 4 different postures.



Tool coordinate system attitude calibration: 2 points need to be taught again, there are two types of calibration methods.

The first category: including xOxy, yOyz, and zOzx, requires a reference point, that is, one of the 4+ points of the position calibration point is selected as the coordinate system origin. Taking xOxy as an example, the ray formed by the first teaching point of the reference point and the attitude calibration is the X positive half axis of the tool coordinate system. The ray formed by the reference point and the second teaching point of the attitude calibration is in the xOy plane.



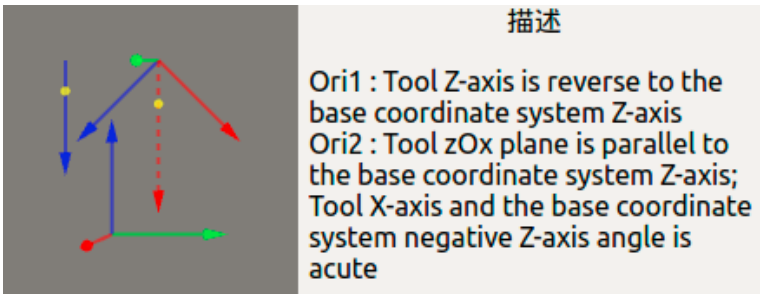
The second category: contains TxRBz_TxyPBzAndTyABnz, TyRBz_TyzPBzAndTzABnz, TzRBz_TzxPBzAndTxABnz three kinds, no reference point. First define the abbreviation:

F_t: tool coordinate system

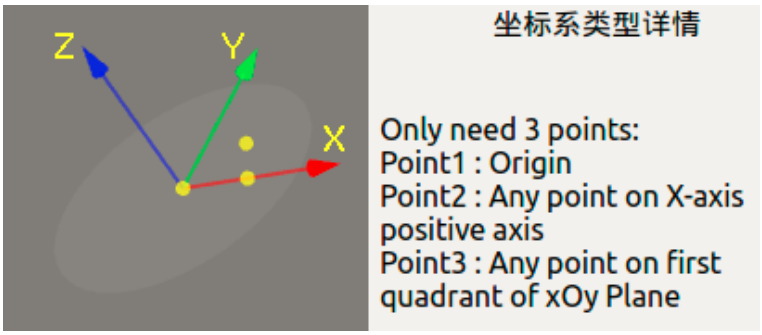
Z_b: Base coordinate system Z axis

Z_t: Tool coordinate system Z axis

Taking TzRBz_TzxPBzAndTxABnz as an example, the first point of the attitude calibration satisfies Z_t and Z_b parallel but opposite directions; the second teaching point of the attitude calibration satisfies the zOx plane of F_t parallel to Z_b, and the Z_b negative direction vector is projected on the zOx plane of F_t The angle with Z_t is an acute angle.



The user coordinate system can also be called the workpiece coordinate system. Contains 9 calibration methods: xOy, yOz, zOx, xOxy, xOxz, yOyz, yOyx, zOzx, zOzy. Taking xOxy as an example, the first point to be calibrated is the origin of the coordinate system, the second point is at any point on the positive half of the X-axis, and the third point is formed at any point in the first quadrant of the xOy plane. The angle is an acute angle.



3.4 Parameter Terminology

Must pass parameter	Must be passed, even if it's not used.
Select pass parameter	According to the parameter description of the function and the business logic to choose whether to pass
Bundled parameter	Bundle with a parameter, select the delivery method according to the parameter description of the function
Must not pass parameter	Must not be passed, generally used in conjunction with the bundled parameter, the parameter description of the
Tool parameter	Composed of tool kinematics parameter and tool dynamics parameter
Tool kinematics parameter	It consists of the tool end position parameter and the tool end attitude parameter, which can also describe
Tool dynamics parameter	Consisting of tool load and center of gravity parameters
When the tool is the center of the flange, the kinematic position parameter is {0,0,0} and the attitude parameter is {1,0,0,0}. Load 0, center of gravity {0,0,0}.	

Reference coordinate system: There are 3 types, Base coordinate system, tool coordinate system and user coordinate system.

Base Coordinate	
Tool Coordination	There are position parameters and attitude parameters. When the tool coordinate system is the End coordinate system, the position parameter is {0,0,0}, and the attitude parameter is {1,0,0,0}
User coordinate system parameter	By the user coordinate system calibration method, the center of the flange used for calibration is based on three waypoints of the base coordinate system, and the end position parameters of the tool used for the calibration are composed.

4 Enumerated Types

4.1 User Coordinate System Calibration Method

```
enum CoordCalibrateMethod{
    xOy,
    yOz,
    zOx,
    xOxy,
    xOxz,
    yOyz,
    yOyx,
    zOzx,
    zOzy
}
```

4.2 Coordinate System Type

```
enum CoordType{
    BaseCoordinate,
    EndCoordinate,
    UserCoordinate
}
```

4.3 Track Motion Type

```
enum MoveTrackType{
    Arc,
    Cir,
    ArcWithOriRot,
    CirWithOriRot,
    CARTESIAN_MOVEP,
    JOINT_GNUBSPLINEINTP,
    CARTESIAN_GNUBSPLINEINTP
}
```


4.4 Onboard IO Type

```
enum RobotIOType{
    RobotBoardControllerDI,
    RobotBoardControllerDO,
    RobotBoardControllerAI,
    RobotBoardControllerAO,
    RobotBoardUserDI,
    RobotBoardUserDO,
    RobotBoardUserAI,
    RobotBoardUserAO,
    RobotToolDI,
    RobotToolDO,
    RobotToolAI,
    RobotToolAO
}
```

4.5 Tool IO Power Supply Voltage

```
enum ToolPowerType{
    OUT_0V
    OUT_12V
    OUT_24V
}
```

4.6 Safey IO (16 DI, 16 DO)

U_DI_00	U_DI_10	U_DO_00	U_DO_10
U_DI_01	U_DI_11	U_DO_01	U_DO_11
U_DI_02	U_DI_12	U_DO_02	U_DO_12
U_DI_03	U_DI_13	U_DO_03	U_DO_13
U_DI_04	U_DI_14	U_DO_04	U_DO_14
U_DI_05	U_DI_15	U_DO_05	U_DO_15
U_DI_06	U_DI_16	U_DO_06	U_DO_16
U_DI_07	U_DI_17	U_DO_07	U_DO_17

4.7 USER IO (16 DI, 16 DO)

U_DI_00	U_DI_10	U_DO_00	U_DO_10
U_DI_01	U_DI_11	U_DO_01	U_DO_11
U_DI_02	U_DI_12	U_DO_02	U_DO_12
U_DI_03	U_DI_13	U_DO_03	U_DO_13
U_DI_04	U_DI_14	U_DO_04	U_DO_14
U_DI_05	U_DI_15	U_DO_05	U_DO_15
U_DI_06	U_DI_16	U_DO_06	U_DO_16
U_DI_07	U_DI_17	U_DO_07	U_DO_17

4.8 4 Analog Voltage Input

U_VI_00
U_VI_01
U_VI_02
U_VI_03

4.9 Analog Output (2 Voltage Outputs, 2 Current Outputs)

U_VO_00
U_VO_01
U_CO_00
U_CO_01

4.10 4 Configurable DI/O (Tool IO)

T_DI/O_00
T_DI/O_01
T_DI/O_02
T_DI/O_03

4.11 2 Analog Voltage Inputs (Tool IO)

T_AI_00
T_AI_01

5 Commonly Used Mathematical Functions

double cos(double f)	Returns the cosine of the f radians
double sin(double f)	Returns the sine of the f radians
double tan(double f)	Returns the tangent of the angle of f radians
double sqrt(double f)	Returns the square root of f. If f<0, an error will be reported.
double r2d(double r)	Return radians r to angle values
double d2r(double d)	Return angle d is converted to radians

rpy2quaternion({oriRX,oriRY,oriRZ})	Return quaternion {oriW,oriX,oriY,oriZ}
quaternion2rpy({oriW,oriX,oriY,oriZ})	Return to Euler {oriRX,oriRY,oriRZ}

Example:

```
a=sqrt(64)
print("a="..a)
b=log(10,1000)
print("b="..b)
c=pow(10,3)
print("c="..c)
d=ceil(2.1)
print("d="..d)
e=floor(3.9)
print("e="..e)
f=r2d(3.14159265)
print("f="..f)
g=d2r(180)
print("g="..g)
h=cos(d2r(30))
print("h="..h)
i=sin(d2r(30))
print("i="..i)
j=tan(d2r(30))
print("j="..j)
```

```
- a=8
- b=3
- c=1000
- d=3
- e=3
- f=180
- g=3.14159
- h=0.866025
- i=0.5
- j=0.577351
```

6 Common Motion Modules

6.1 init_global_move_profile

void init_global_move_profile (void)	
Function	Initialize global motion properties
Description	The default is: 0: maximum speed of articulated motion 25°/s, maximum acceleration 25°/s2 1: Maximum speed of end type motion is 0.03m/s, maximum acceleration is 0.03m/s2 2: Circle number of circles 0 3: Offset: no offset 4: tool parameter attribute: no tool, ie flange coordinate system 5: Reference coordinate system: base marking system

6.2 set_joint_maxacc

void set_joint_maxacc({double joint1MaxAcc, double joint2MaxAcc, double joint3MaxAcc,double joint4MaxAcc, double joint5MaxAcc, double joint6 MaxAcc})	
Function	Set the maximum acceleration of 6 joints in rad/s2
Description	Set before articulationJ1~J3: 0 ~ 17.3 J4~J6: 0 ~ 20.7
Example	set_joint_maxacc({1,1,1,1,1,1})

6.3 set_joint_maxvelc

void set_joint_maxvelc ({double joint1MaxVelc, double joint2MaxVelc, double joint3MaxVelc,double joint4MaxVelc, double joint5MaxVelc, double joint6MaxVelc})	
Function	Set the maximum speed of 6 joints, in rad/s
Description	Set before articulation J1~J3: 0 ~ 2.596 J4~J6: 0 ~ 3.11
Example	set_joint_maxvelc({1,1,1,1,1,1})

6.4 set_end_maxacc

void set_end_maxacc(double endMaxAcc)

Function	Set the maximum acceleration at the end, in m/s2
Description	Set before end-type motion, such as linear motion and trajectory motion 0 ~ 2
Example	set_end_maxacc(1)

6.5 set_end_maxvelc

void set_end_maxvelc(double endMaxVelc)	
Function	Set the maximum speed at the end, in m/s
Description	Set before end-type motion, such as linear motion and trajectory motion 0 ~ 2
Example	set_end_maxvelc(1)

6.6 move_joint

void move_joint({double joint1Angle, double joint2Angle, double joint3 Angle,double joint4Angle, double joint5Angle, double joint6Angle}, bool isBlock)	
Function	Joint Move
Description	Move to a waypoint, the waypoint is a table containing 6 joint angles, the unit rad Can be set to block, true means blocking, fasle means non-blocking In blocking mode, the data will be returned when the arm is in motion; in non-blocking mode, The data is returned when the arm begins to move. In general, it is recommended to use blocking mode.
Example	move_joint({1,1,1,1,1,1},true)

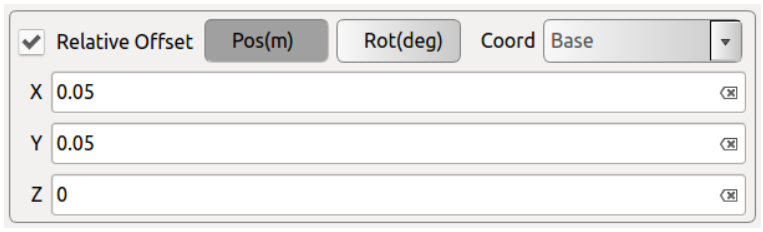
6.7 move_line

void move_line({double joint1Angle, double joint2Angle, double joint3Angle,double joint4Angle, double joint5Angle, double joint6Angle}, bool isBlock)	
Function	Linear motion
Description	Move to a waypoint, the waypoint is a table containing 6 joint angles, the unit rad Can be set to block, true means blocking, fasle means non-blocking In blocking mode, the data will be returned when the arm is in motion; in non-blocking mode, The data is returned when the arm begins to move. In general, it is recommended to use blocking mode.

Example	move_line({1,1,1,1,1,1},true)

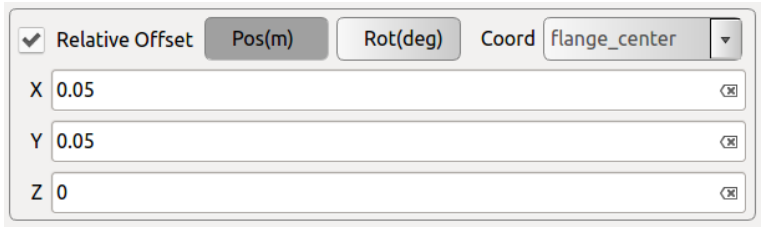
6.8 set_relative_offset

void set_relative_offset({double posOffsetX, double posOffsetY, double posOffsetZ}, {double oriOffsetW,double oriOffsetX, double oriOffsetY, double oriOff setZ}, {double toolEndPosX, toolEndPosY, toolEndPosZ}, {double toolEndOriW, toolEndOriX, toolEndOriY, toolEndOriZ}, CoordCalibrateMethod coordCalibrateMethod, {double point1Joint1, double point1Joint2, double point1Joint3, double point1Joint4, double point1Joint5, double point1Joint6}, {double point2Joint1, double point2Joint2, double point2Joint3, double point2Joint4, double point2Joint5, double point2Joint6}, {double point3Joint1, double point3Joint2, double point3Joint3, double point3Joint4, double point3Joint5, double point3Joint6}, {double toolEndPosXForCalibCoord, toolEndPosYForCalibCoord, toolEndPosZForCalibCoord})	
Function	Set relative offset properties
Description	The red part is the must pass parameter，The value passed in represents: the position offset based on the reference coordinate system. If no position offset is required, pass in {0,0,0} The yellow part is the select pass parameter，The value passed in represents: the attitude offset based on the reference coordinate system. If no pose offset is required, you can either pass or pass {1,0,0,0} The blue part is the select pass parameter. When the reference coordinate system is the base coordinate system or the user coordinate system, the ratio is not transmitted; when the reference coordinate system is the tool coordinate system, the must pass parameter, {double toolEndPosX, toolEndPosY, toolEndPosZ} is the position parameter of the tool coordinate system, {double toolEndOriW, toolEndOriX, toolEndOriY, toolEndOriZ} are the pose parameters of the tool coordinate system. The green part is the select pass parameter，This includes the coordinate system calibration method, the joint angle corresponding to the 3 teach points, and the tool position parameters used to calibrate the user coordinate system (if the flange center is used, {0, 0, 0} or no pass). When based on the base system or the tool coordinate system, this parameter must not be transmitted; when based on the user coordinate system, it must be transmitted.
Example	**Relative offset under the base coordinate.**



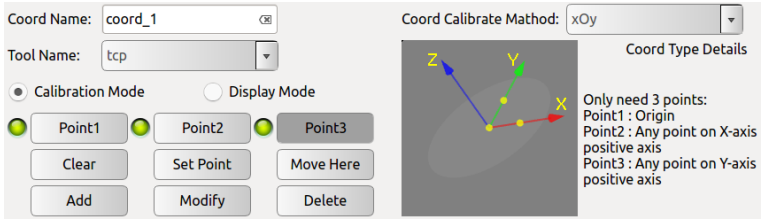
set_relative_offset({0.05,0.05,0})

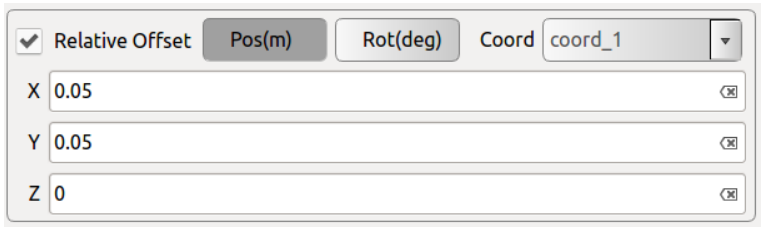
Relative offset in the tool coordinate system



set_relative_offset({0.05, 0.05, 0}, {0.000000, 0.000000, 0.000000}, {1.000000, 0.000000, 0.000000, 0.000000})

Relative offset in the user coordinate system





set_relative_offset({0.05, 0.05, 0}, CoordCalibrateMethod.xOy, {-0.000003, -0.127267, -1.321122, 0.376934, -1.570796, - 0.000008}, {-0.000004, -0.347513, -1.480267, 0.438042, - 1.570796, -0.000009}, {-0.363607, -0.169896, -1.356376, 0.384316, -1.570793, -0.363612}, {0.000000, 0.000000, 0.100000})

6.9 get_current_waypoint

A waypoint is a nested table containing position, pose, and joint angle.

```
{
"pos" = {
        "x" = posX
        "y" = posY
        "z" = posZ
}
```


"ori" = {
"w" = oriW
"x" = oriX
"y" = oriY
"z" = oriZ
}
"joint" = {
"j1" = joint1Angle
"j2" = joint2Angle
"j3" = joint3Angle
"j4" = joint4Angle
"j5" = joint5Angle
"j6" = joint6Angle
}
}

get_current_waypoint(void)

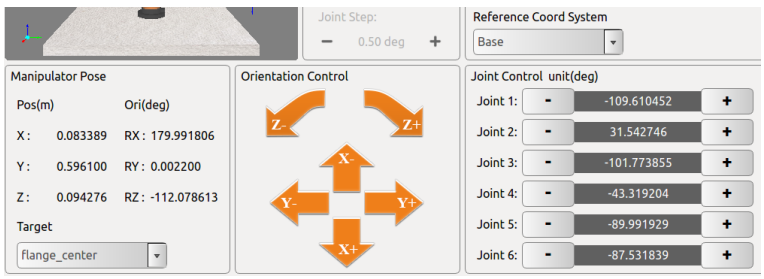
Function	Returns the real-time waypoint of the current arm
Example	

```
pose=get_current_waypoint()

print(string.format("posX=%6.6f",pose.pos.x))
print(string.format("posY=%6.6f",pose.pos.y))
print(string.format("posZ=%6.6f",pose.pos.z))

print(string.format("oriW=%6.6f",r2d(pose.ori.w)))
print(string.format("oriX=%6.6f",r2d(pose.ori.x)))
print(string.format("oriY=%6.6f",r2d(pose.ori.y)))
print(string.format("oriZ=%6.6f",r2d(pose.ori.z)))

print(string.format("Joint1=%6.6f",r2d(pose.joint.j1)))
print(string.format("Joint2=%6.6f",r2d(pose.joint.j2)))
print(string.format("Joint3=%6.6f",r2d(pose.joint.j3)))
print(string.format("Joint4=%6.6f",r2d(pose.joint.j4)))
print(string.format("Joint5=%6.6f",r2d(pose.joint.j5)))
print(string.format("Joint6=%6.6f",r2d(pose.joint.j6)))
```

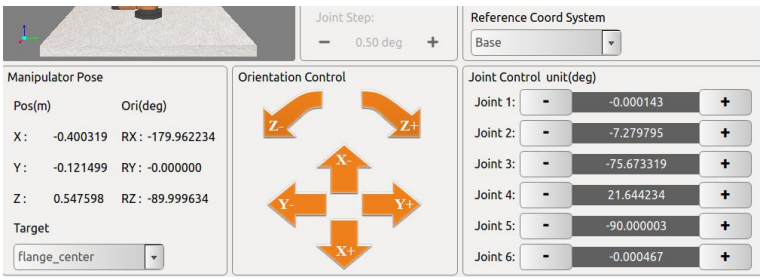


	22/08/2019 17:42:49 [INFORMATION] - **posX=0.083389** 22/08/2019 17:42:49 [INFORMATION] - **posY=0.596100** 22/08/2019 17:42:49 [INFORMATION] - **posZ=0.094276** 22/08/2019 17:42:49 [INFORMATION] - **oriW=0.001349** 22/08/2019 17:42:49 [INFORMATION] - **oriX=32.006800** 22/08/2019 17:42:49 [INFORMATION] - **oriY=-47.522300** 22/08/2019 17:42:49 [INFORMATION] - **oriZ=-0.004020** 22/08/2019 17:42:49 [INFORMATION] - **Joint1=-109.610000** 22/08/2019 17:42:49 [INFORMATION] - **Joint2=31.542700** 22/08/2019 17:42:49 [INFORMATION] - **Joint3=-101.774000** 22/08/2019 17:42:49 [INFORMATION] - **Joint4=-43.319200** 22/08/2019 17:42:49 [INFORMATION] - **Joint5=-89.991900** 22/08/2019 17:42:49 [INFORMATION] - **Joint6=-87.531800**

6.10 get_target_pose

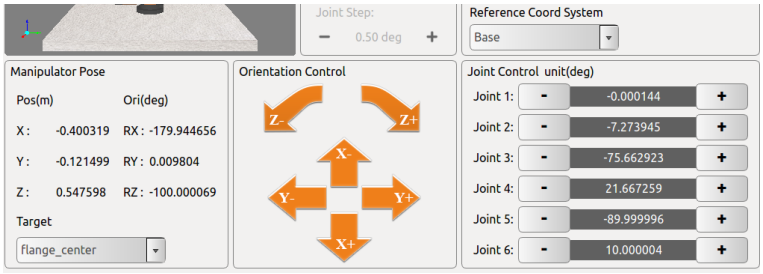
get_target_pose({ikRefPointJoint1Angle, ikRefPointJoint2Angle, ikRefPointJoint3Angle, ikRefPointJoint4Angle, ikRefPointJoint5Angle, ikRefPointJoint6Angle}, {doubletoolEndPosXOnUserCoord, toolEndPosYOnUserCoord, toolEndPosZOnUserCoord}, {double toolEndOriWOnUserCoord, toolEndOriXOnUserCoord, toolEndOriYOnUserCoord,toolEndOriZOnUserCoord}, bool enableEndRotate, double endRotateAngle, {double toolEndPosX, toolEndPosY, toolEndPosZ}, {double toolEndOriW, double toolEndOriX, toolEndOriY, toolEndOriZ}, CoordCalibrateMethod coordCalibrateMethod, {double point1Joint1, double point1Joint2, double point1Joint3, double point1Joint4, double point1Joint5, double point1Joint6}, {double point2Joint1, double point2Joint2, double point2Joint3, double point2Joint4, double point2Joint5, double point2Joint6}, {double point3Joint1, double point3Joint2, double point3Joint3, double point3Joint4, double point3Joint5, double point3Joint6}, {double toolEndPositionForCalibCoordX, toolEndPositionForCalibCoordY, toolEndPositionForCalibCoordZ})	
Function	Given the position and attitude of the tool coordinate system in the reference coordinate system, the end rotation angle parameter, return the inverse solution Post joint angle table
Description	{ikRefPointJoint1Angle, ikRefPointJoint2Angle, ikRefPointJoint3Angle,ikRefPointJoint4Angle, ikRefPointJoint5Angle, ikRefPointJoint6Angle} is the select pass parameter, this point is used as the inverse

	solution reference point; when the parameter is not passed, the current robot arm real-time waypoint is used as the inverse solution reference point. {double toolEndPosXOnUserCoord, toolEndPosYOnUserCoord, toolEndPosZOnUserCoord}must pass parameter，Indicates the position of the tool coordinate system under the reference coordinate system. {double toolEndOriWOnUserCoord, toolEndOriXOnUserCoord, toolEndOriYOnUserCoord, toolEndOriZOnUserCoord} is the select pass parameter indicates the posture of the tool coordinate system in the reference coordinate system. If the parameter is not passed, the current real-time waypoint posture is maintained by default.bool enableEndRotate is the must pass parameter，Can be set to true or false Double endRotateAngle is the select pass parameter. When the previous parameter is true, this parameter is the must pass parameter, and finally J6 will move to the parameter; when the previous parameter is false, this parameter is the must not pass parameter. {double toolEndPosX, toolEndPosY, toolEndPosZ}, {double toolEndOriW, double toolEndOriX, toolEndOriY, toolEndOriZ} is a select pass parameter indicating the kinematic parameters of the tool coordinate system. When based on the center of the flange, this parameter may not pass or pass {0,0,0}, {1,0,0,0}CoordCalibrateMethod coordCalibrateMethod, {double point1Joint1, double point1Joint2, double point1Joint3,d ouble point1Joint4, double point1Joint5, double point1Joint6}, {double point2Joint1, double point2Joint2, double point2Joint3,d ouble point2Joint4, double point2Joint5, double point2Joint6}, {double point3Joint1, double point3Joint2, double point3Joint3,d ouble point3Joint4, double point3Joint5, double point3Joint6}, {double toolEndPositionForCalibCoordX, toolEndPositionForCalibCo ordY,toolEndPositionForCalibCoordZ} Is a reference pass parameter, indicating the reference coordinate system. When the reference coordinate system is the base mark system, the parameter is must not pass parameter; when the reference coordinate system is the user coordinate system, it is a silent pass parameter.
Example	1、Keep the current attitude moving to the waypoint under the base marking system {-0.400319, -0.121499, 0.547598}, the tool coordinate system is the flange center joint=get_target_pose({-0.400319,-0.121499,0.547598}, false,{0,0,0},{1,0,0,0}) init_global_move_profile() set_joint_maxacc({1,1,1,1,1,1}) set_joint_maxvelc({1,1,1,1,1,1}) move_joint(joint,true)



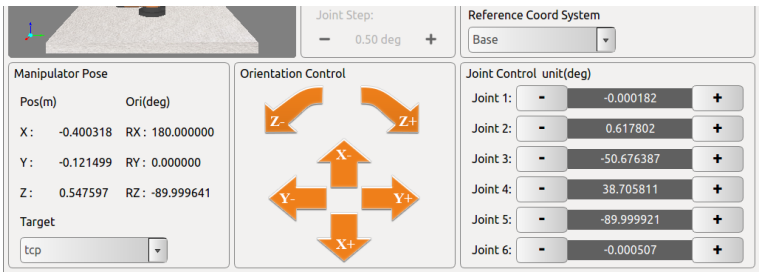
2、Keep the current attitude to the waypoint under the base marking system {-0.400319, -0.121499, 0.547598}, the tool coordinate system is the flange center, and the J6 finally moves to 10°.

```
joint = get_target_pose({-0.400319,-
0.121499,0.547598},true,d2r(10),{0,0,0},{1,0,0,0})
init_global_move_profile()
set_joint_maxacc({1,1,1,1,1,1})
set_joint_maxvelc({1,1,1,1,1,1})
move_joint(joint,true)
```

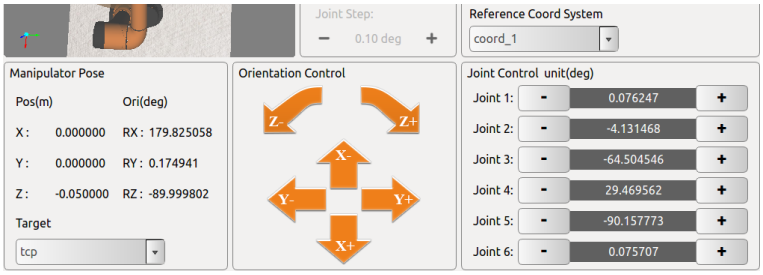


3、Keep the current posture motion to the waypoint {-0.400319, -0.121499, 0.547598}, which is {0,0,0. 1}, {1,0,0,0} tool coordinate system (tcp) under the base mark position.

```
joint = get_target_pose({-0.400319, -
0.121499,0.547598},false,{0,0,0.1},{1,0,0,0})
init_global_move_profile()
set_joint_maxacc({1,1,1,1,1,1})
set_joint_maxvelc({1,1,1,1,1,1})
move_joint(joint,true)
```



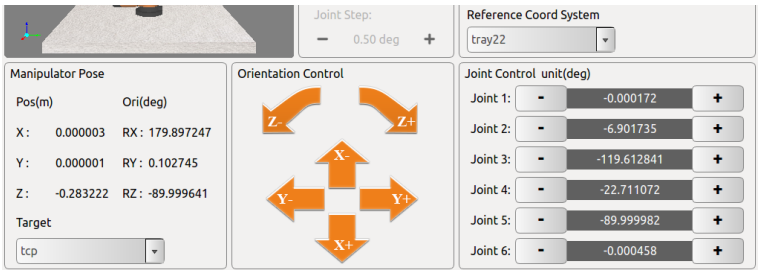
4、Keep the current pose motion to the waypoint {0,0,0.05}, which is {0,0,0.1}, {1,0,0,0} tool coordinate system (tcp) in CoordCalibrateMethod.xOy, {-0.000003, -0.127267 , -1.321122, 0.376934, -1.570796, -0.000008}, {-0.000004, -0.347513, -1. 480267, 0.438042, -1.570796, -0.000009}, {-0.363607, -0.169896, -1.356376, 0.384316, -

	1.570793, - 0.363612}, {0.000000, 0.000000, 0.100000}) The position under the reference coordinate system (coord_1). joint = get_target_pose({0,0,0.05}, false, {0,0,0.1},{1,0,0,0}, CoordCalibrateMethod.xOy, {-0.000003, -0.127267, -1.321122, 0.376934, -1.570796, -0.000008}, {-0.000004, -0.347513, -1.480267, 0.438042, -1.570796, -0.000009}, {-0.363607, -0.169896, -1.356376, 0.384316, -1.570793, -0.363612}, {0.000000, 0.000000, 0.100000}) init_global_move_profile() set_joint_maxacc({1,1,1,1,1,1}) set_joint_maxvelc({1,1,1,1,1,1}) move_joint(joint,true) 

6.11 base_to_user

The return value is position and pose {posX, posY, posZ}, {oriW, oriX, oriY, oriZ} base_to_user({double joint1Angle, double joint2Angle, double joint3Angle, double joint4Angle, double joint5Angle, double joint6Angle}, {double toolEndPosX, toolEndPosY, toolEndPosZ}, {double toolEndOriW, double toolEndOriX, toolEndOriY, toolEndOriZ}, CoordCalibrateMethod coordCalibrateMethod, {double point1Joint1, double point1Joint2, double point1Joint3, double point1Joint4, double point1Joint5, double point1Joint6}, {double point2Joint1, double point2Joint2, double point2Joint3, double point2Joint4, double point2Joint5, double point2Joint6}, {double point3Joint1, double point3Joint2, double point3Joint3, double point3Joint4, double point3Joint5, double point3Joint6}, {double toolEndPositionForCalibCoordX, toolEndPositionForCalibCoordY, toolEndPositionForCalibCoordZ})	
Function	Convert the position and pose parameters of the flange_center under Base to tool under userCoord Position and attitude parameters
Description	{double joint1Angle, double joint2Angle, double joint3Angle, double

	joint4Angle, double joint5Angle, double joint6Angle}, is a must pass parameter, six joint arcs (determining a unique flange_center pose).{double toolEndPosX, toolEndPosY, toolEndPosZ}, {double toolEndOriW, double toolEndOriX, toolEndOriY, toolEndOriZ},select pass parameter，Indicates the position and attitude parameters of the tool coordinate system. When the tool coordinate system is flange_center, you can pass no arguments or pass in {0,0,0},{1,0,0,0} CoordCalibrateMethod coordCalibrateMethod, {double point1Joint1, double point1Joint2, double point1Joint3,d ouble point1Joint4, double point1Joint5, double point1Joint6}, {double point2Joint1, double point2Joint2, double point2Joint3,d ouble point2Joint4, double point2Joint5, double point2Joint6}, {double point3Joint1, double point3Joint2, double point3Joint3,d ouble point3Joint4, double point3Joint5, double point3Joint6}, {double toolEndPositionForCalibCoordX, toolEndPositionForCalibCo ordY,toolEndPositionForCalibCoordZ} Is a reference pass parameter, indicating the reference coordinate system. When the reference coordinate system is the base mark system, the parameter is must not pass parameter; when the reference coordinate system is the user coordinate system, it is a silent pass parameter.

	1. The pose parameter of flange_center under base is converted to the pose parameter of flange_center under userCoor d(tray22).
	`pose = get_current_waypoint()`
	`joint = {pose.joint.j1,pose.joint.j2,pose.joint.j3,pose.joint.j4,pose.joint.j5,pose.joint.j6}`
	`pos_user,ori_user = base_to_user(joint,{0,0,0},{1,0,0,0},get_user_coord_param("tray22"))`
	`print(string.format("X_user = %6.6f",pos_user[1]))`
	`print(string.format("Y_user = %6.6f",pos_user[2]))`
	`print(string.format("Z_user = %6.6f",pos_user[3]))`
	`RPY_user = quaternion2rpy({ori_user[1],ori_user[2],ori_user[3],ori_user[4]})`
	`print(string.format("RX_user = %6.6f",r2d(RPY_user[1])))`
	`print(string.format("RY_user = %6.6f",r2d(RPY_user[2])))`
	`print(string.format("RZ_user = %6.6f",r2d(RPY_user[3])))`
	 
	21/08/2019 16:17:55 [INFORMATION] - **Start program**
	21/08/2019 16:17:55 [INFORMATION] - **X_user=0.000003**
	21/08/2019 16:17:55 [INFORMATION] - **Y_user=0.000001**
	21/08/2019 16:17:55 [INFORMATION] - **Z_user=-0.283223**
	21/08/2019 16:17:55 [INFORMATION] - **RX_user=179.897000**
	21/08/2019 16:17:55 [INFORMATION] - **RY_user=0.102745**
	21/08/2019 16:17:55 [INFORMATION] - **RZ_user=-89.999600**
	21/08/2019 16:17:55 [INFORMATION] - **ScriptController::enqueueCmd cmdType : 47**
	21/08/2019 16:17:55 [INFORMATION] - **commandInfoQueue size : 1**
	21/08/2019 16:17:55 [INFORMATION] - **ScriptStateMachine State : 0, cmdType : 47**
	2. The pose parameter of the flange_center under Base is converted to the

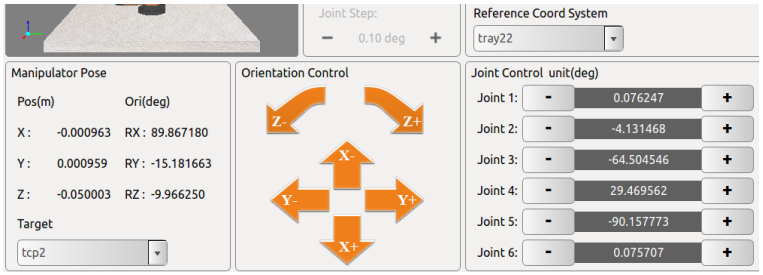
pose parameter of the tool coordinate system (tcp2) under userCoord(tray22).


```
pose=get_current_waypoint()

tool_pos={0,0,0.45}
tool_ori=rpy2quaternion({1.570796,-0.261799,-0.174533})

joint={pose.joint.j1,pose.joint.j2,pose.joint.j3,pose.joint.j4,pose.joint.j5,pose.joint.j6}
pos_user,ori_user=base_to_user(joint,tool_pos,tool_ori,get_user_coord_param("tray22"))
print(string.format("X_user=%6.6f",pos_user[1]))
print(string.format("Y_user=%6.6f",pos_user[2]))
print(string.format("Z_user=%6.6f",pos_user[3]))
RPY_user=quaternion2rpy({ori_user[1],ori_user[2],ori_user[3],ori_user[4]})
print(string.format("RX_user=%6.6f",r2d(RPY_user[1])))
print(string.format("RY_user=%6.6f",r2d(RPY_user[2])))
print(string.format("RZ_user=%6.6f",r2d(RPY_user[3])))
```

```
22/08/2019 15:36:22 [INFORMATION] - Start program
22/08/2019 15:36:22 [INFORMATION] - X_user=-0.000963
22/08/2019 15:36:22 [INFORMATION] - Y_user=0.000959
22/08/2019 15:36:22 [INFORMATION] - Z_user=-0.050003
22/08/2019 15:36:22 [INFORMATION] - RX_user=89.867300
22/08/2019 15:36:22 [INFORMATION] - RY_user=-15.181700
22/08/2019 15:36:22 [INFORMATION] - RZ_user=-9.966200
```



6.12 get_user_coord_param

CoordCalibrateMethod coordCalibrateMethod, {double point1Joint1, double point1Joint2, double point1Joint3, double point1Joint4, double point1Joint5, double point1Joint6}, {double point2Joint1, double point2Joint2, double point2Joint3, double point2Joint4, double point2Joint5, double point2Joint6}, {double point3Joint1, double point3Joint2, double point3Joint3, double point3Joint4, double point3Joint5, double point3Joint6}, {double toolEndPosXForCalibCoord, toolEndPosYForCalibCoord, toolEndPosZForCalibCoord} get_user_coord_param(string userCoordName)	
Function	Get user coordinate system parameters
Parameter	userCoordName - User coordinate system name

	Need to be consistent with the user coordinate system name in the teaching device user coordinate system calibration interface
Return	Returns the user coordinate system parameters (including: Method enumeration for calibrating the user coordinate system; The center of the flange used to calibrate the user coordinate system is based on the three point joint angles of the base coordinate system; Calibrate the tool end position parameters used by the user coordinate system.)
Instructions	get_user_coord_param("userCoordName ")

6.13 move_track

void move_track(MoveTrackType trackType, bool isBlock)	
Function	Trajectory movement, according to the global waypoint list (added by the add_waypoint function)
Parameter	trackType The track type. Reference enumeration type chapters include arcs, circles, moveP, B-splines, and so on.
	isBlock Motion blocking flag. When true, the interface blocks until it moves to the target waypoint; if false, the interface returns immediately.
Return	in vain
Description	Use with add_waypoint, for example, the track has 3 points, first execute add_waypoint to join 3 points, then execute Currently the track motion type supports arc/circle, moveP, see enumeration type
Example	**--MoveP** init_global_move_profile() set_joint_maxvelc({1.298089,1.298089,1.298089,1.555088,1.555088,1.555088}) set_joint_maxacc({8.654390,8.654390,8.654390,10.368128,10.368128,10.368128}) move_joint({1.060116, -0.615040, -1.874408, 0.336996, -1.588076, 1.064821}, true) init_global_move_profile() set_end_maxvelc(0.500000) set_end_maxacc(0.500000) add_waypoint({1.060116, -0.615040, -1.874408, 0.336996, -1.588076, 1.064821})

	add_waypoint({0.203591, -0.683445, -1.901615, 0.356287, -1.601432, 0.208205}) add_waypoint({-0.256614, -0.431207, -1.803535, 0.188129, -1.599854, -0.252338}) add_waypoint({-0.984889, -0.696295, -1.914051, 0.326062, -1.585615, -0.980493}) set_blend_radius(0.05) move_track(MoveTrackType.CARTESIAN_MOVEP, true) **--MoveArc** init_global_move_profile() set_joint_maxvelc({1.298089,1.298089,1.298089,1.555088,1.555088,1.555088}) set_joint_maxacc({8.654390,8.654390,8.654390,10.368128,10.368128,10.368128}) move_joint({-0.189573, -0.351721, -1.482747, 0.439770, -1.570795, -0.189578}, true) init_global_move_profile() set_end_maxvelc(1.000000) set_end_maxacc(1.000000) add_waypoint({-0.189573, -0.351721, -1.482747, 0.439770, -1.570795, -0.189578}) add_waypoint({-0.565416, -0.540795, -1.571577, 0.540014, -1.570792, -0.565421}) add_waypoint({-0.652200, -0.370080, -1.493317, 0.447559, -1.570792, -0.652205}) move_track(MoveTrackType.Arc, true) **--MoveCir** init_global_move_profile() set_joint_maxvelc({1.298089,1.298089,1.298089,1.555088,1.555088,1.555088}) set_joint_maxacc({8.654390,8.654390,8.654390,10.368128,10.368128,10.368128}) move_joint({-0.189573, -0.351721, -1.482747, 0.439770, -1.570795, -0.189578}, true) init_global_move_profile() set_end_maxvelc(1.000000) set_end_maxacc(1.000000) add_waypoint({-0.189573, -0.351721, -1.482747, 0.439770, -1.570795, -0.189578}) add_waypoint({-0.565416, -0.540795, -1.571577, 0.540014, -1.570792, -0.565421}) add_waypoint({-0.652200, -0.370080, -1.493317, 0.447559, -1.570792, -0.652205}) set_circular_loop_times(1)

	move_track(MoveTrackType.Cir, true)

6.14 clear_global_waypoint

void clear_global_waypoint_list(void)	
Function	Clear global waypoint list
Description	When using move_track multiple times, you need to clear the last track waypoint
Example	clear_global_waypoint_list()

6.15 add_waypoint

void add_waypoint({double joint1Angle, double joint2Angle, double joint3Angle, double joint4Angle, double joint5Angle, double joint6Angle})	
Function	Add a waypoint to the global waypoint list for the move_track function
Parameter	type table
Description	in vain
Example	add_waypoint({-0.000003, -0.127267, -1.321122, 0.376934, -1.570796, -0.000008})

6.16 set_circular_loop_times

void set_circular_loop_times(int times)	
Parameter	Times The number of laps in a circular motion, used with move_track. When times=0, it is a circular motion. When times>0, it is a circular motion.
Return	in vain
Example	set_circular_loop_times(0)

6.17 set_blend_radius

void set_blend_radius(double blendRadius)	
Function	Set the blending radius

Parameter	blendRadius is blending radius, unit: meter
Return	in vain
Example	set_blend_radius(0.01)

6.18 set_arrival_ahead_distance_mode

void set_arrival_ahead_distance_mode(double distance)	
Funtion	set advance arrival distance mode
Parameter	distance Advance distance
Return	in vain
Example	set_arrival_ahead_distance_mode (0.01)

6.19 set_arrival_ahead_time_mode

void set_arrival_ahead_time_mode(double time)	
Function	set advance time mode
Parameter	time advance time
Return	in vain
Example	set_arrival_ahead_time_mode(0.01)

6.20 set_arrival_ahead_blend_mode

void set_arrival_ahead_blend_mode(double blendRadius)	
Function	set the advance in-position blending radius mode
Parameter	blendRadius: blending radius
Return	in vain
Example	set_arrival_ahead_blend_mode (0.01)

6.21 set_robot_collision_class(6)

void set_robot_collision_class(int collisionClass)	
Funtion	Set the collision level
Parameter	collisionClass 0~10

Return	in vain
Example	set_robot_collision_class(6)

6.22 set_tool_kinematics_param

void set_tool_kinematics_param({double posX, double posY, double posZ}, {double oriW = 1, double oriX = 0, double oriX = 0, double oriX = 0})	
Function	Set tool kinematic parameters, the gesture can not be passed
Parameter	Two tables, the first table is the position must pass, the second table is the gesture, you can not pass.
Return	in vain
Example	set_tool_kinematics_param({0.1,0.2,0.3})

6.23 get_tool_kinematics_param

{double toolEndPosX, toolEndPosY, toolEndPosZ}, {double toolEndOriW, toolEndOriX, toolEndOriY, toolEndOriZ} get_tool_kinematics_param(string toolName)	
Function	Get tool kinematic parameters
Parameter	toolName : the name of the tool, which needs to be the same as the name of the tool in the calibration tool of the Teach Pendant tool.
Return	Return tool end position parameter, tool end attitude parameter
Example	get_tool_kinematics_param("toolName")

6.24 set_tool_dynamics_param

void set_tool_dynamics_param(double payload, {double gravityCenterX, double gravityCenterY, double gravityCenterZ}, {double inertiaXX = 0, double inertiaXY = 0, double inertiaXZ = 0, double inertiaYY = 0, double inertiaYZ = 0, double inertiaZZ = 0})	
Function	Set the tool dynamics parameters, the load and center of gravity xyz must pass, the inertia can not pass

Parameter	Payload unit: kg Center of gravity: table Inertia: table
Return	in vain
Example	set_tool_dynamics_param(3,{0.1,0.2,0.3})

6.25 robot_pause

void robot_pause(void)	
Function	The arm is suspended. It can be called if and only if the arm is in motion.
Parameter	in vain
Return	in vain
Example	robot_pause()

6.26 robot_continue

void robot_continue(void)	
Function	The robotic arm resumes movement. Can be called if and only if the arm is in the pause state.
Parameter	in vain
Return	in vain
Example	robot_continue()

6.27 robot_slow_stop()

void robot_slow_stop(void)	
Function	The arm is slowly stopped. It can be called if and only if the arm is in motion.
Parameter	in vain
Return	in vain
Example	robot_slow_stop()

6.28 robot_fast_stop

void robot_fast_stop(void)	
Function	The arm is stopped. It can be called if and only if the arm is in motion.
Parameter	in vain
Return	in vain
Example	robot_fast_stop()

7 Internal Modules

7.1 sleep

void sleep(double second)	
Function	Delay
Description	unit s
Example	sleep(0.1)

7.2 set_robot_io_status

void set_robot_io_status(RobotIOType ioType, string name, double valu	
Function	Set arm IO status
Description	
Example	Set U_DO_00 to 1 set_robot_io_status(RobotIOType.RobotBoardUserDO,"U_DO_00",1)

7.3 get_robot_io_status

double **get_robot_io_status**(RobotIOType ioType, string name)	
Function	Get arm IO status
Description	Returns the value of the IO state, which is double
Example	Get the value of U_DI_00 a=get_robot_io_status(RobotIOType.RobotBoardUserDI,"U_DI_00") print("a="..a)

7.4 get_modbus_io_status

double get_modbus_io_status(string ioName)	
Function	Get modbus IO status
Description	ioName: modbus IO name (name in the IO configuration in teach pendant modbus)
Example	get_modbus_io_status(“M_DO_00”)

7.5 set_modbus_io_status

void set_modbus_io_status(string ioName,double ioValue)	
Function	Set modbus IO status
Description	ioName: modbus IO name (name in the IO configuration in teach pendant modbus)
Example	set_modbus_io_status(“M_DO_00”,1)

7.6 get_global_variable

variant get_global_variable(string varName)	
Function	Get the teach pendant global variable value
Description	
Example	Get the value of V_I_b in the teach pendant [variable configuration] by script a=get_global_variable ("V_I_b") print("a="..a)

7.7 set_global_variable

void set_global_variable(string varName, variant varValue)	
Function	Set the teach pendant global variable value
Description	Support for int, double, and bool
Example	Set V_B_a to true in the teach pendant [Variable Configuration] by script set_global_variable("V_B_a",true) Set the V_I_b in the teach pendant [Variable Configuration] to 5 by script. set_global_variable("V_I_b",5) Set the teach pendant [variable configuration] V_D_c to 8.8 by script set_global_variable("V_D_c",8.8)

7.8 init_global_variables

void init_global_variables(string varNameList)	
Function	Initialize the teach pendant global variable value (variable value in the variable configuration interface)
Parameter	varNameList variable name list string, separated by a comma, for example: " varName1, varName2 ". If the parameter is empty, initialize all teacher variable values

Description	in vain
Example	init_global_variables("varName1, varName2")

7.9 set_tool_power_voltage

void set_tool_power_voltage(ToolPowerType toolPowerType)	
Function	Set tool supply voltage
Parameter	toolPowerType tool power voltage enumeration value, refer to the enumeration type chapter.
Description	Corresponding to the state value of IO, double type
Example	set_tool_power_voltage(ToolPowerType.OUT_12V)

7.10 get_plc_io_status

double get_plc_io_status(string ioName)	
Function	Get PLC IO status
Parameter	ioName PLC IO name (defined by teach pendant Extensions→Peripheral → Plc → IO Config)
Description	PLC IO status
Example	get_plc_io_status ("P_DO_0")

7.11 set_plc_io_status

void set_plc_io_status(string ioName, double ioValue)	
Function	set PLC IO status
Parameter Description	ioName PLC IO name (defined by teach pendant Extensions→Peripheral → Plc → IO Config)
	ioValue PLC IO status
Example	in vain
Function	set_plc_io_status ("P_DO_0", 1)

8 TCP Communication

8.1 TCP Net Assistant

Used for debugging of TCP communication.

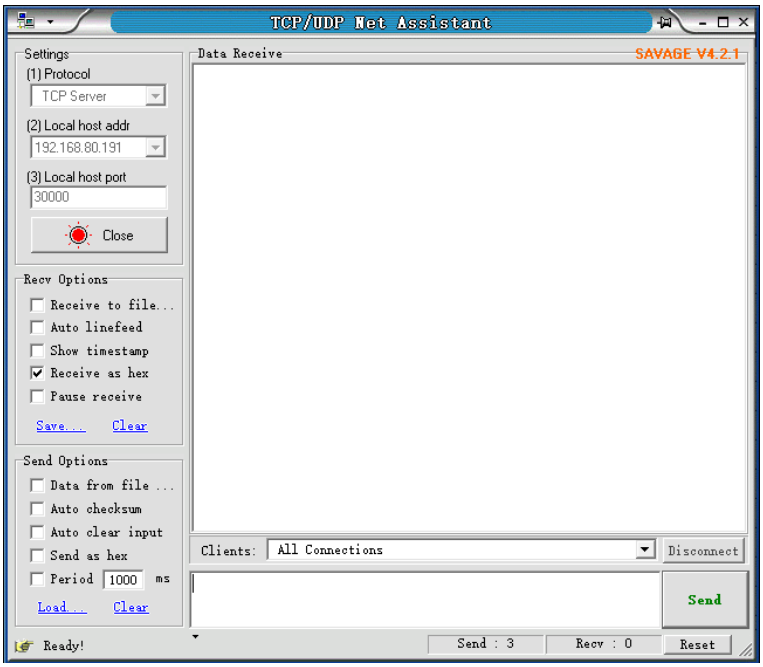
8.1.1 TCP Net Assistant as a server.

Communication mode: TCP Server

Local port: custom settings

Click to start monitoring, when there is a client connection, the indicator light is on.

Click Stop Listening to shut down the server.



8.1.2 TCP Net Assistant as a Client

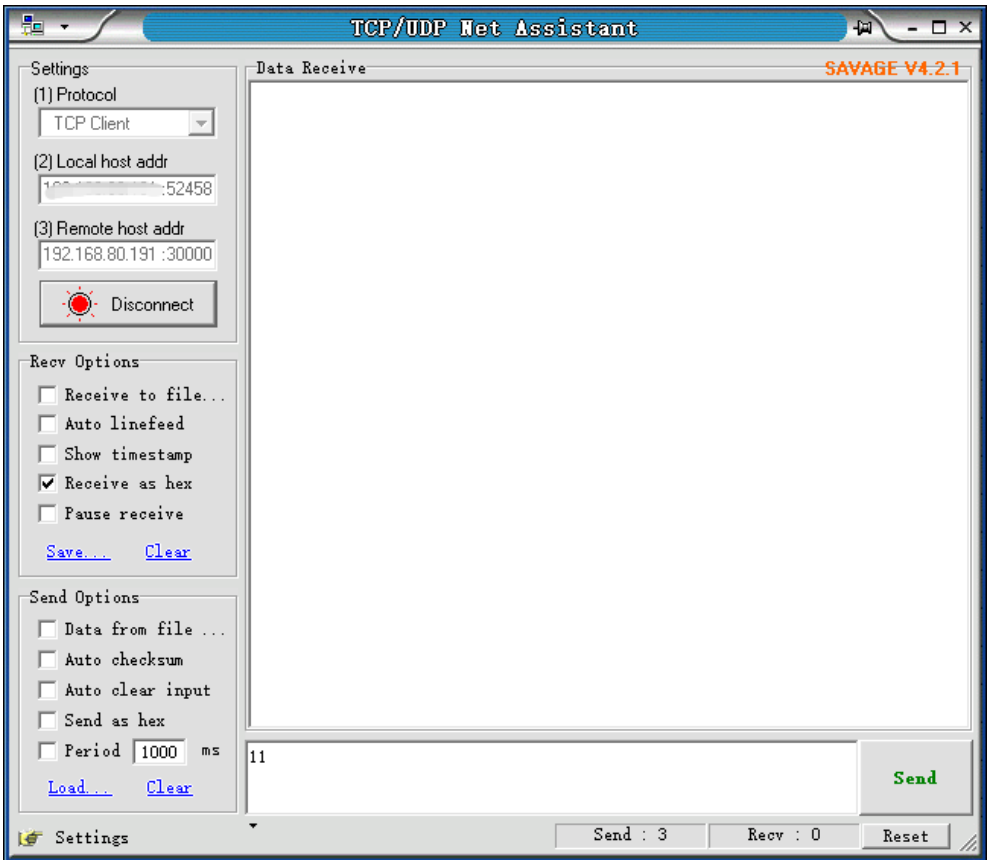
Communication mode: TCP Client

Remote host: Ip address of the connected server

Remote port: the port number of the connected server

Click to connect to the network, when the connection is successfully connected to the server, the indicator light is on.

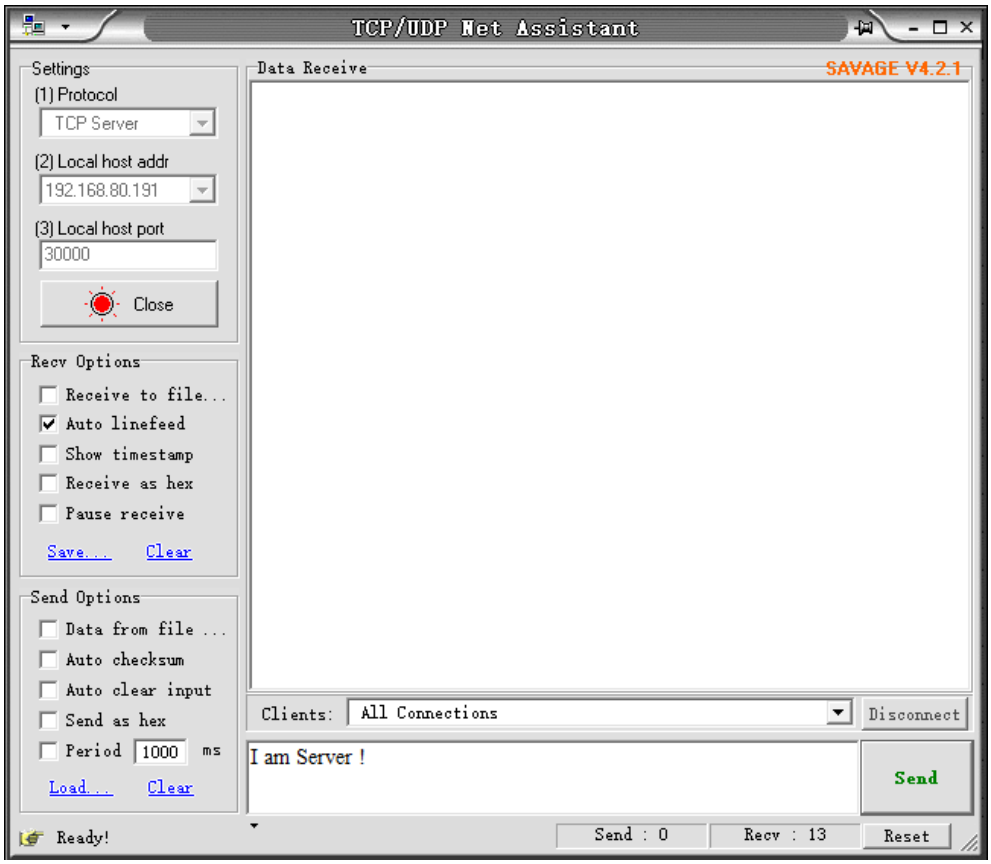
Click Disconnect Network to disconnect from the server.



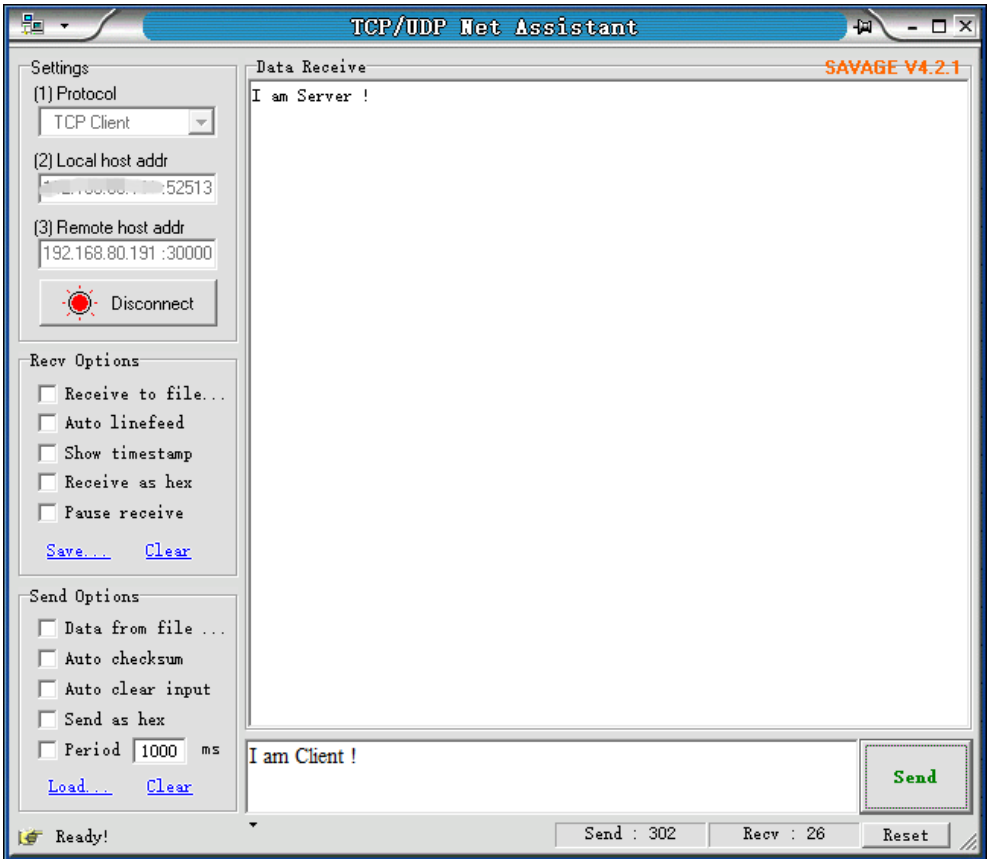
8.1.3 Server Sends Data to the Client

Ensure that the server and client connect successfully.

Enter the desired content in the sending area, such as ‘I am Server!’ and click [send].



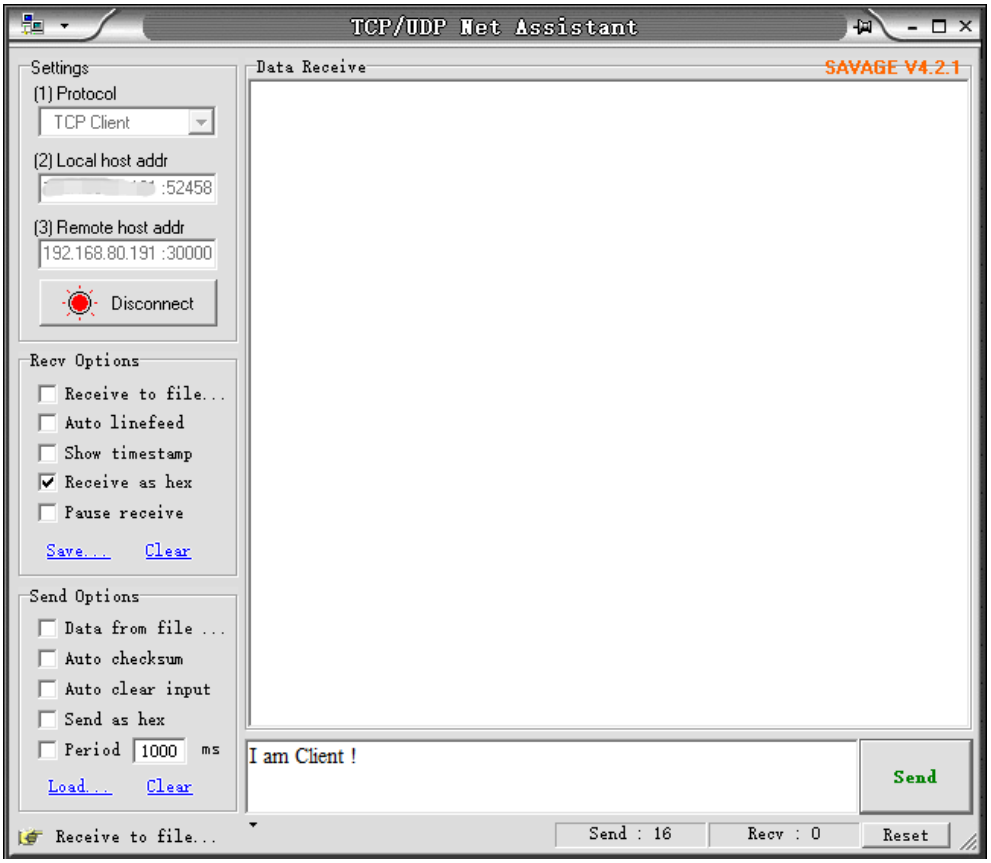
In the receiving area of the client, you can go to the data 'I am Server!'



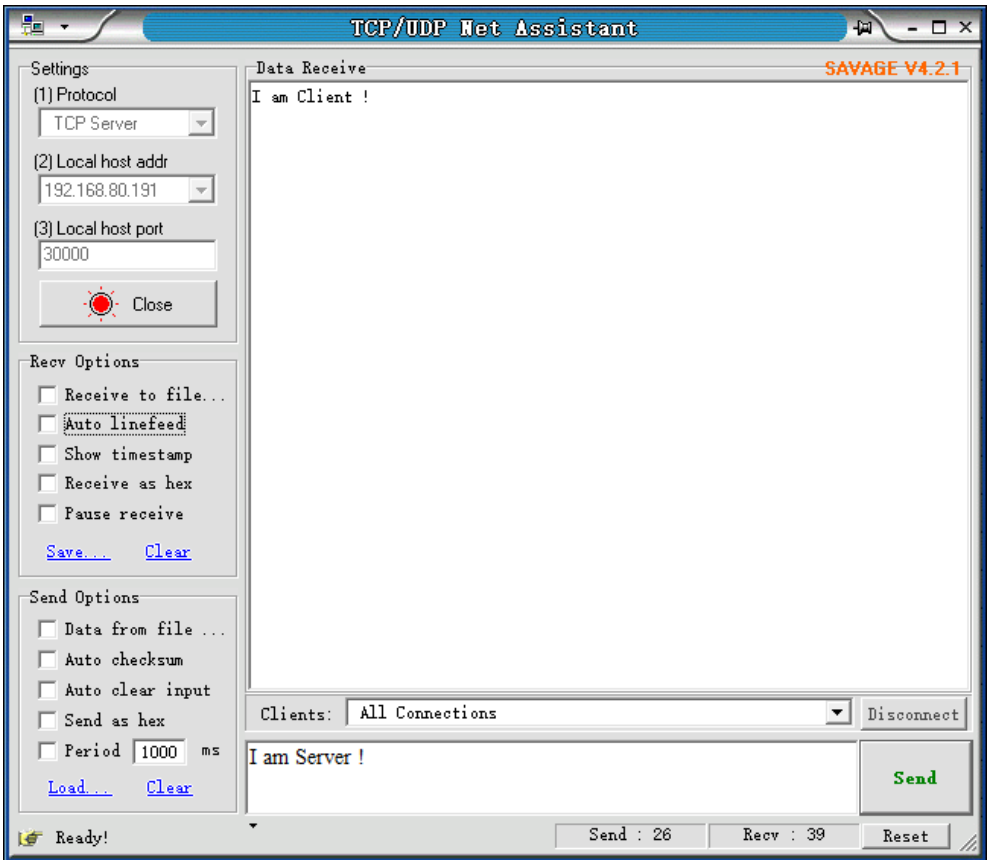
8.1.4 Client sends Data to the Server

Ensure that the client and server are connected successfully.

Enter the desired content in the client send area, such as ‘I am Client!’ click [send]



In the server receiving area you can go to the same content ‘I am Client!’

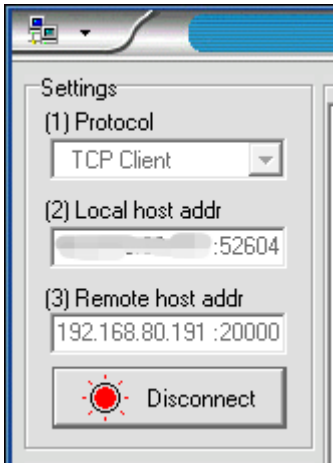


8.2 AUBO as a TCP Server

8.2.1 Listening Port

void listen(int port)	
Function	TCP server listening port port
Description	port： Port number
Example	port1 = 20000 tcp.server.listen (port1)

void initTCPServer(int port) This function is deprecated and only backward compatible	
Function	Initialize the TCP server
Description	port： Port number
Example	port1 = 20000 initTCPServer(port1)

	 note: An AUBO can only initialize one server. After the server is initialized, the server will be shut down only after the software is restarted.

8.2.2 Judging Whether to Connect

bool is_connected(string IP)	
Function	Determine the IP address of the client whether to establish a connection to the local server
Parameter	IP：IP address
Return	Returns true if the IP in the parameter is already connected to the TCP server; otherwise returns false.
Example	while (tcp.server.is_connected(ip) ~= true) do sleep(1) end print("connection succeeded")

bool isClientConnected(string IP) **This function is deprecated and only backward compatible**	
Function	Determine if the IP is connected to the server
Parameter	IP：IP address
Return	Return true if the IP in the parameter is already connected to the TCP server; false otherwise
Example	ret=isClientConnected(8888)

8.2.3 Receive client data as a string

string recv_str_data(string IP)	
Function	The local server receives the data from the client with the IP address as a string.
Parameter	IP: client's ip address

Example	while(true)do
	ip_client="192.168.2.1"
	port=8890
	--TCP server listening port
	tcp.server.listen(port)
	--waiting for an external client connection
	while (tcp.server.is_connected(ip_client) ~= true) do
	  sleep(1)
	end
	print("connection succeeded")
	-- The local server receives the data from the client with IP address as a string
	recv=""
	while(recv~="quit")do
	  recv=tcp.server.recv_str_data(ip_client)
	  while(recv~="")do
	  print("recv="..recv)
	  recv=""
	  sleep(1)
	  end
	  sleep(1)
	end
	end

std::string serverRecvData(std::string IP) **This function is deprecated and only backward compatible**	
Function	Receive data from IP
Parameter	IP: client's ip address
Example	recv = serverRecvData(“192.168.1.100”)

8.2.4 Send a message to the client as a string

void send_str_data(string IP, string msg)

Function	The local server sends a message to the client with the address IP as a string.
Parameter	IP: client's ip address; msg: string

Example	while(true)do
	ip_client="192.168.2.1"
	port=8890
	--TCP server listening port
	tcp.server.listen(port)
	--waiting for an external client connection
	while (tcp.server.is_connected(ip_client) ~= true) do
	  sleep(1)
	end
	print("connection succeeded")
	--the local server sends a message to the client with the address IP as a string.msg
	sleep(3)
	tcp.server.send_str_data(ip_client, "connect succ")
	sleep(0.5)
	end

void serverSendData(std::string IP, std::string msg) **This function is deprecated and only backward compatible**	
Function	Send the string msg to the IP address
Parameter	IP: client's ip address; msg: string

Example	port1 = 20000
	ip1 = "192.168.2.129"
	
	initTCPServer(port1)
	ret = false
	while (ret ~= true) do
	  ret = isClientConnected(ip1)
	  sleep(0.1)
	end
	while (1) do
	  recv = serverRecvData(ip1)
	  serverSendData(ip1,recv)
	  sleep(0.01)
	end

8.2.5 Receive client data in ASCII format

table recv_asc_data(string IP)	
Function	The local server receives data from the client with IP address in ASCII format
Parameter	IP：IP address
Return	Returns the received data, the return value is in the form of a one-dimensional table, and the key takes the default value (starting at 1)

Example	ip_client="192.168.2.1"
	port=88900
	--TCP server listening port
	tcp.server.listen(port)
	-- Waiting for an external client connection
	while (tcp.server.is_connected(ip_client) ~= true) do
	  sleep(1)
	end
	print("connection succeeded")
	-- The local server receives data from the client with IP address in ASCII format
	sleep(1)
	while(true)do
	recv2=tcp.server.recv_asc_data(ip_client)
	ab=#recv2
	while(ab>0)do
	  print(recv2[1])
	  print(recv2[2])
	  print(recv2[3])
	  print(recv2[4])
	  print(recv2[5])
	  sleep(0.5)
	  ab=0
	end
	sleep(0.5)
	end

8.2.6 Send a message to the client in ASCII format

void send_asc_data(string IP, table msg)	
Function	The local server sends the message msg to the client with IP address in ASCII format.
Parameter	IP：IP address
Return	msg：The message sent in the format of a one-dimensional table, the key

	takes the default value (starting from 1)。
Example	in vain
Function	ip_client="192.168.2.1" port=8890 --TCP server listening port tcp.server.listen(port) -- Waiting for an external client connection while (tcp.server.is_connected(ip_client) ~= true) do sleep(1) end print("connection succeeded") -- The local server sends the message msg to the client with IP address in ASCII format. hello={string.byte("hello",1),string.byte("hello",2),string.byte("hello",3),string.byte("hello",4),string.byte("hello",5)} world={string.byte("world",1),string.byte("world",2),string.byte("world",3),string.byte("world",4),string.byte("world",5)} tcp.server.send_asc_data(ip_client, hello) tcp.server.send_asc_data(ip_client, world)

8.2.7 Stop listening and disconnect

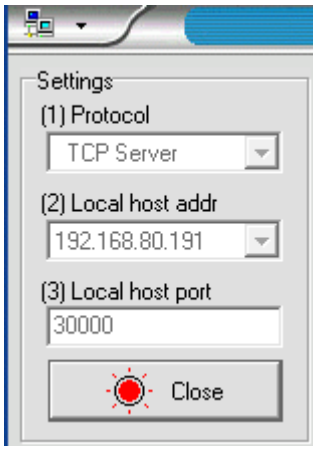
void close(void)	
Function	The local server stops listening and disconnects all established connections
Parameter	in vain
Return	in vain
Example	tcp.server.close()

8.3 AUBO as a TCP client

8.3.1 Connecting to the Server

void connect(string IP, int port)	
Function	Connect to the TCP server of the specified IP and port
Description	IP: IP address; port: port number
Example	ip_server="192.168.80.191"

	port_server=10000 tcp.client.connect (ip_server,port_server)

void connectTCPServer(std::string IP, int port) **This function is deprecated and only backward compatible**	
Function	Connect to the server of the specified IP and port
Description	IP: IP address; port: port number
Example	port1 = 30000 ip1 = "192.168.80.191" connectTCPServer(ip1,port1)  Note: Can connect to multiple servers

8.3.2 Receive server data as a string

string recv_str_data(string IP, string port)	
Function	Receive data sent from the TCP server of the specified IP and port as a string
Description	IP: IP address; port: port number
Example	ip_server="192.168.80.191" port_server=10000 tcp.client.connect (ip_server,port_server) --connect to TCP server recv=tcp.client.recv_str_data(ip_server,port_server) print(recv)

string clientRecvData(string IP, string port) **This function is deprecated and only backward compatible**	
Function	Receive data sent from the TCP server of the specified IP and port
Description	IP: IP address; port: port number
Return	Return data sent from the server

Example	recv=clientRecvData("127.0.0.1", "7777")

8.3.3 Send data to the server as a string

void send_str_data(string IP, string port, string msg)	
Function	Sends data to the TCP server of the specified IP and port as a string msg
Description	IP: IP address port: port number msg: sent data
Example	ip_server="192.168.80.191" port_server=10000 tcp.client.connect (ip_server,port_server) --connect to TCP server tcp.client.send_str_data(ip_server, port_server, "Hello server")

void clientSendData(string IP, string port, string msg) **This function is deprecated and only backward compatible**	
Function	Send data to the TCP server of the specified IP and port msg
Description	IP: IP address port: port number msg: sent data
Example	clientSendData("127.0.0.1", "7777", "OK")

8.3.4 Receive server data in ASCII

table recv_asc_data(string IP, string port)	
Function	Receive data sent from the TCP server of the specified IP and port in **ASCII format**
Description	IP: IP address port: port number
Return	Returns the received data in a one-dimensional table with the default value of the key (starting at 1)
Example	ip_server="192.168.80.191" port_server=10000 --connect to TCP server

	tcp.client.connect (ip_server,port_server)
	sleep(1)
	while(true)do
	recv2=tcp.client.recv_asc_data(ip_server, port_server)
	ab=#recv2
	while(ab>0)do
	print(recv2[1])
	print(recv2[2])
	print(recv2[3])
	print(recv2[4])
	print(recv2[5])
	sleep(0.5)
	ab=0
	end
	sleep(0.5)
	end

8.3.5 Send data to the server in ASCII

void send_asc_data(string IP, string port, table msg)	
Function	Send data to the server in **ASCII**
Description Return	IP: IP address Port: port number Msg: The data to be sent, in the format of a one-dimensional table, the key takes the default value (starting at 1).
Example	ip_server="192.168.80.191" port_server=10000 --connect to TCP server tcp.client.connect (ip_server,port_server) sleep(1) tcp.client.send_str_data(ip_server, port_server, "Hello") world = {string.byte("world",1), string.byte("world",2), string.byte("world",3), string.byte("world",4),string.byte("world",5)} tcp.client.send_asc_data(ip_server, port_server, world)

8.3.6 Disconnect the TCP server

void disconnect(string IP, int port)	
Function	Disconnect the TCP server of the specified IP and port

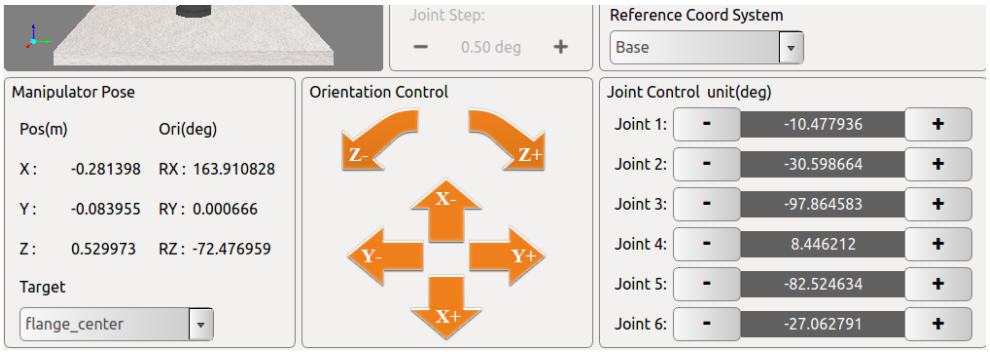
Description	in vain
Return	in vain
Return	tcp.client.disconnect("127.0.0.1", 7777)

void disconnectTCPServer()**This function is deprecated and only backward compatible**	
Function	Disconnect all clients from the TCP server
Description	in vain
Return	in vain
Return	disconnectTCPServer()

9 Scripting Exercises

9.1 Obtain the Coordinates of the Center of the Arm Flange Under Base

Obtain the position and attitude parameters of the center of the flange of the robot arm under Base, and the six joint angles, and print them out as follows:




```
21/08/2019 16:40:36 [INFORMATION] - Start program
21/08/2019 16:40:36 [INFORMATION] - X_base=-0.281398
21/08/2019 16:40:36 [INFORMATION] - Y_base=-0.083955
21/08/2019 16:40:36 [INFORMATION] - Z_base=0.529973
21/08/2019 16:40:36 [INFORMATION] - RX_base=163.911000
21/08/2019 16:40:36 [INFORMATION] - RY_base=0.000660
21/08/2019 16:40:36 [INFORMATION] - RZ_base=-72.476900
21/08/2019 16:40:36 [INFORMATION] - Joint1=-10.477900
21/08/2019 16:40:36 [INFORMATION] - Joint2=-30.598700
21/08/2019 16:40:36 [INFORMATION] - Joint3=-97.864600
21/08/2019 16:40:36 [INFORMATION] - Joint4=8.446210
21/08/2019 16:40:36 [INFORMATION] - Joint5=-82.524600
21/08/2019 16:40:36 [INFORMATION] - Joint6=-27.062800
```

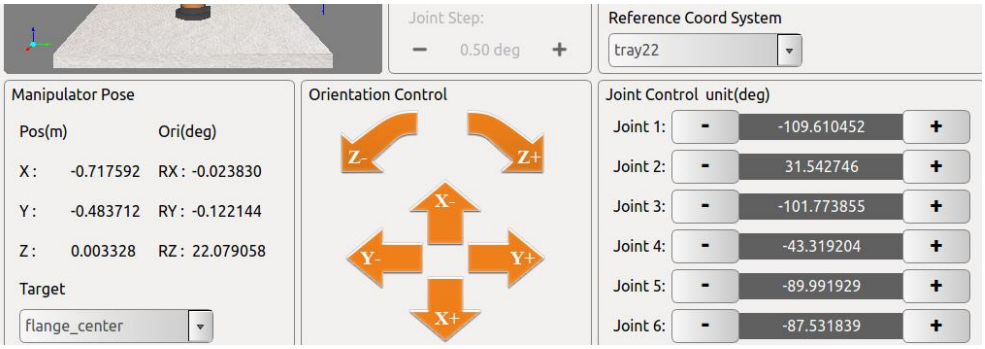
Reference procedure:


```
pose = get_current_waypoint()

print(string.format("X_base = %6.6f",pose.pos.x))
print(string.format("Y_base = %6.6f",pose.pos.y))
print(string.format("Z_base = %6.6f",pose.pos.z))
RPY = quaternion2rpy({pose.ori.w,pose.ori.x,pose.ori.y,pose.ori.z})
print(string.format("RX_base = %6.6f",r2d(RPY[1])))
print(string.format("RY_base = %6.6f",r2d(RPY[2])))
print(string.format("RZ_base = %6.6f",r2d(RPY[3])))
print(string.format("Joint1 = %6.6f",r2d(pose.joint.j1)))
print(string.format("Joint2 = %6.6f",r2d(pose.joint.j2)))
print(string.format("Joint3 = %6.6f",r2d(pose.joint.j3)))
print(string.format("Joint4 = %6.6f",r2d(pose.joint.j4)))
print(string.format("Joint5 = %6.6f",r2d(pose.joint.j5)))
print(string.format("Joint6 = %6.6f",r2d(pose.joint.j6)))
```

9.2 Obtain the Coordinates of the Center of the Arm Flange in the User Coordinate System

Obtain the position and attitude parameters of the center of the arm flange in the user coordinate system (tray22) and print it out:




```
22/08/2019 17:46:39 [INFORMATION] - Start program
22/08/2019 17:46:39 [INFORMATION] - X_user=-0.717592
22/08/2019 17:46:39 [INFORMATION] - Y_user=-0.483712
22/08/2019 17:46:39 [INFORMATION] - Z_user=0.003328
22/08/2019 17:46:39 [INFORMATION] - RX_user=-0.023830
22/08/2019 17:46:39 [INFORMATION] - RY_user=-0.122144
22/08/2019 17:46:39 [INFORMATION] - RZ_user=22.079000
```

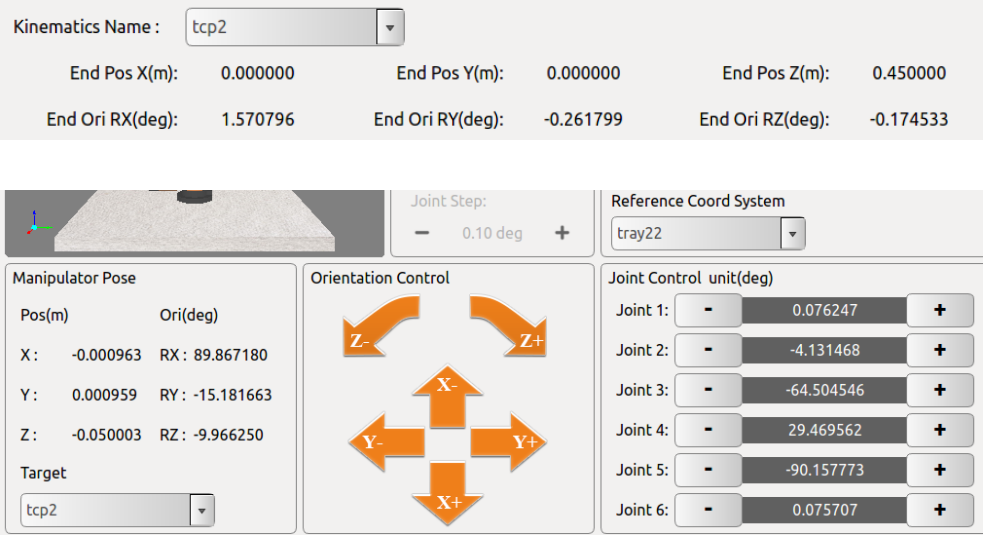
Reference procedure:


```
pose = get_current_waypoint()

joint = {pose.joint.j1,pose.joint.j2,pose.joint.j3,pose.joint.j4,pose.joint.j5,pose.joint.j6}
pos_user,ori_user = base_to_user(joint,{0,0,0},{1,0,0,0},get_user_coord_param("tray22"))
print(string.format("X_user = %6.6f",pos_user[1]))
print(string.format("Y_user = %6.6f",pos_user[2]))
print(string.format("Z_user = %6.6f",pos_user[3]))
RPY_user = quaternion2rpy({ori_user[1],ori_user[2],ori_user[3],ori_user[4]})
print(string.format("RX_user = %6.6f",r2d(RPY_user[1])))
print(string.format("RY_user = %6.6f",r2d(RPY_user[2])))
print(string.format("RZ_user = %6.6f",r2d(RPY_user[3])))
```

9.3 Obtaining the Coordinates of the Robot Arm TCP in the User Coordinate System

Obtain the position and attitude parameters of the robot arm tcp(tcp2) in the user coordinate system (tray22) and print it out:




```
22/08/2019 15:36:22 [INFORMATION] - Start program
22/08/2019 15:36:22 [INFORMATION] - X_user=-0.000963
22/08/2019 15:36:22 [INFORMATION] - Y_user=0.000959
22/08/2019 15:36:22 [INFORMATION] - Z_user=-0.050003
22/08/2019 15:36:22 [INFORMATION] - RX_user=89.867300
22/08/2019 15:36:22 [INFORMATION] - RY_user=-15.181700
22/08/2019 15:36:22 [INFORMATION] - RZ_user=-9.966200
```

Reference procedure:

```
pose=get_current_waypoint()

tool_pos={0,0,0.45}
tool_ori=rpy2quaternion({1.570796,-0.261799,-0.174533})

joint={pose.joint.j1,pose.joint.j2,pose.joint.j3,pose.joint.j4,pose.joint.j5,pose.joint.j6}
pos_user,ori_user=base_to_user(joint,tool_pos,tool_ori,get_user_coord_param("tray22"))
print(string.format("X_user=%6.6f",pos_user[1]))
print(string.format("Y_user=%6.6f",pos_user[2]))
print(string.format("Z_user=%6.6f",pos_user[3]))
RPY_user=quaternion2rpy({ori_user[1],ori_user[2],ori_user[3],ori_user[4]})
print(string.format("RX_user=%6.6f",r2d(RPY_user[1])))
print(string.format("RY_user=%6.6f",r2d(RPY_user[2])))
print(string.format("RZ_user=%6.6f",r2d(RPY_user[3])))
```

9.4 Joint Movement

The robot arm reciprocates 3 times between wp1 and wp2 points.

wp1 = {-16.418907,11.715555,-133.748860,-55.472799,-89.999982,-14.340675}

wp2 = {-16.898397,-10.512184,-88.484384,12.019414,-89.999914,-14.820165}

The reference procedure is as follows:

```
wp1 = {-16.418907,11.715555,-133.748860,-55.472799,-89.999982,-14.340675}
wp2 = {-16.898397,-10.512184,-88.484384,12.019414,-89.999914,-14.820165}

--Angle turning
function deg_to_rad(table1)
    table_ret={}
    for i=1,#table1,1 do
        table_ret[i]=d2r(table1[i])
    end
    return table_ret
end

wp1_rad=deg_to_rad(wp1)
wp2_rad=deg_to_rad(wp2)

i=1

while(i<=3)do
    init_global_move_profile()
    set_joint_maxacc({1,1,1,1,1,1})
    set_joint_maxvelc({1,1,1,1,1,1})


    move_joint(wp1_rad,true)
    move_joint(wp2_rad,true)
    i=i+1
end
```

9.5 Linear Motion

The robot arm reciprocates 3 times between wp1 and wp2 points.

wp1 = {-16.418907,11.715555,-133.748860,-55.472799,-89.999982,-14.340675}

wp2 = {-16.898397,-10.512184,-88.484384,12.019414,-89.999914,-14.820165}

The reference procedure is as follows:

```
wp1 = {-16.418907,11.715555,-133.748860,-55.472799,-89.999982,-14.340675}
wp2 = {-16.898397,-10.512184,-88.484384,12.019414,-89.999914,-14.820165}

function deg_to_rad(table1)
    table_ret={}
    for i=1,#table1,1 do
        table_ret[i]=d2r(table1[i])
    end
    return table_ret
end

wp1_rad=deg_to_rad(wp1)
wp2_rad=deg_to_rad(wp2)

i=1
while(i<=3)do
    init_global_move_profile()
    set_end_maxacc(0.3)
    set_end_maxvelc(0.3)

    move_line(wp1_rad,true)
    move_line(wp2_rad,true)
    i=i+1
end
```

9.6 Circular Motion

3 waypoints that make up the circle:

wp1={0.208890, -0.044775, -1.246891, 0.368688, -1.570800, 0.208869}

wp2={-0.237646, -0.169014, -1.355669, 0.384145, -1.570793, -0.237655}

wp3={-0.000009, 0.087939, -1.110852, 0.372015, -1.570793, -0.000007}

```
wp1={0.208890, -0.044775, -1.246891, 0.368688, -1.570800, 0.208869}
wp2={-0.237646, -0.169014, -1.355669, 0.384145, -1.570793, -0.237655}
wp3={-0.000009, 0.087939, -1.110852, 0.372015, -1.570793, -0.000007}
--Move to ready point
init_global_move_profile()
set_joint_maxvelc({1,1,1,1,1,1})
set_joint_maxacc({1,1,1,1,1,1})
move_joint(wp1, true)
 --Move to the first track point
 init_global_move_profile()
 set_end_maxvelc(1)
 set_end_maxacc(1)
 --Move cir
 init_global_move_profile()
 set_end_maxvelc(1)
 set_end_maxacc(1)
 add_waypoint(wp1)
 add_waypoint(wp2)
 add_waypoint(wp3)
 set_circular_loop_times(2)
 move_track(MoveTrackType.ARC_CIR, true)
```


9.7 User IO Read and Set

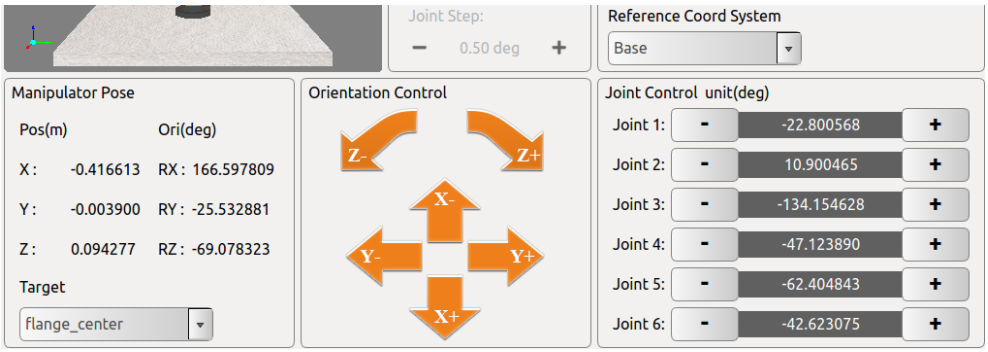
The state of U_DO_00 is read by a script and printed. Control U_DO_00 to turn on and off at 1Hz.

```
while(true) do
    print(get_robot_io_status(RobotIOType.RobotBoardUserDO, "U_DO_00"))
    set_robot_io_status(RobotIOType.RobotBoardUserDO, "U_DO_00", 1)
    sleep(0.5)
    print(get_robot_io_status(RobotIOType.RobotBoardUserDO, "U_DO_00"))
    set_robot_io_status(RobotIOType.RobotBoardUserDO, "U_DO_00", 0)
    sleep(0.5)
end
```

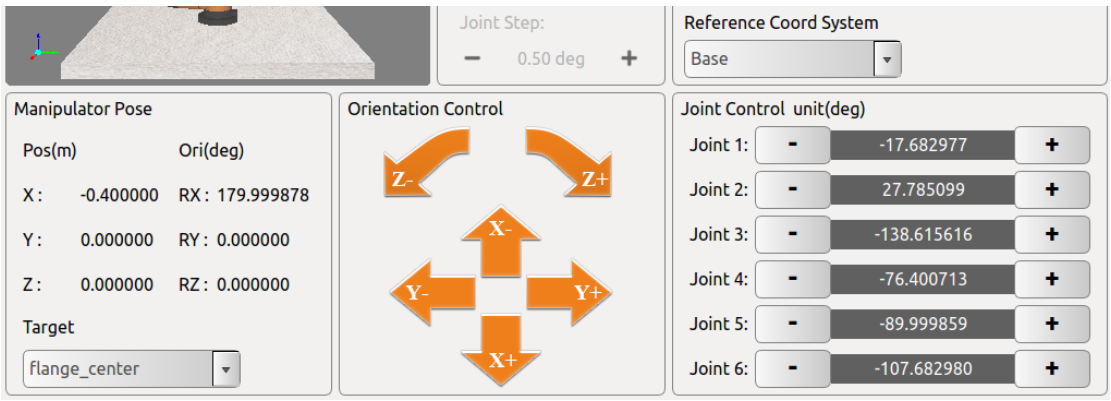
9.8 Robot Arm Moves to the Target Position

Robot arm current position

wp1={-22.800568,10.900465,-134.154628,-47.123890,-62.404843,-42.623075}



Move the arm to the target position XYZ={-0.4,0,0}, RPY={180,0,0}. as follows



Reference procedure


```
wp1={-22.800568,10.900465,-134.154628,-47.123890,-62.404843,-42.623075}

function deg_to_rad(table1)
    table_ret={}
    for i=1,#table1,1 do
        table_ret[i]=d2r(table1[i])
    end
    return table_ret
end

wp1_rad=deg_to_rad(wp1)

init_global_move_profile()
set_joint_maxacc({1,1,1,1,1,1})
set_joint_maxvelc({1,1,1,1,1,1})
move_joint(wp1_rad,true)

pos_target={-0.4,0,0}
ori_target=rpy2quaternion({d2r(180),d2r(0),d2r(0)})

joint=get_target_pose(pos_target,ori_target,false,{0,0,0},{1,0,0,0})

```

```
init_global_move_profile()
set_joint_maxacc({1,1,1,1,1,1})a
set_joint_maxvelc({1,1,1,1,1,1})
move_joint(joint,true)
```

9.9 TCP Server Communication 1

The robot acts as a TCP server and waits for an external client connection. When a client connection is successful, the string "connect succ" is sent to the client.

```
ip_client="192.168.2.1"
port=8890
tcp.server.listen(port)
while (tcp.server.is_connected(ip_client) ~= true) do
    sleep(1)
end
tcp.server.send_str_data(ip_client, "connect succ")
```

9.10 TCP Server Communication 2

As a TCP Server, the robot receives the data (number string) sent by the client and assigns it to the global variable V_D_recv. When the string "quit" is received, the program is exited.

```
ip_client="192.168.2.1"

port=8890
tcp.server.listen(port)

while (tcp.server.is_connected(ip_client) ~= true) do
  sleep(1)
end

recv=""
while(recv~="quit")do
    recv=tcp.server.recv_str_data(ip_client)
    if(recv~="")then
        set_global_variable("V_D_recv",tonumber(recv))
        end
        sleep(0.01)
end
```


9.11 TCP Client Communication 1

As the TCP Client, the robot sends the coordinates xyz under the current base system to the server. as follows:

Reference procedure:

```
ip_server="192.168.80.191"
port_server=10000

--connect to TCP server
tcp.client.connect (ip_server,port_server)
sleep(1)

wp=get_current_waypoint()
str="x="..tostring(wp.pos.x).." "
    .."y="..tostring(wp.pos.y).." "
   .. "z="..tostring(wp.pos.z).." "

tcp.client.send_str_data(ip_server, port_server, str)
```

9.12 TCP Client Communication 2

As a TCP Client, the robot receives the data sent by the server. The data format is { x_offset, y_offset, t_offset } (such as {100,200,300}), x_offset is the x offset (in mm), and y_offset is the y offset (in mm). , t_offset is the RZ offset (in deg).

The robot reference point wp1={-16.418907, 11.715555, -133.748860, -55.472799, -89.999982, -14.340675}. The robot compensates for the offset and moves to the target position. The robot exits the program after receiving "quit".

Reference procedure:


```lua
--String split function
function string.split(str, delimiter)
    if str==nil or str=='' or delimiter==nil then
        return nil
    end
    local result = {}
    for match in (str..delimiter):gmatch("(.-)"..delimiter) do
        table.insert(result, match)
    end
    return result
end
--Angle turning
function deg_to_rad(table1)
    table_ret={}
    for i=1,#table1,1 do
        table_ret[i]=d2r(table1[i])
    end
    return table_ret
end
--server ip and port
ip_server="192.168.80.191"
port_server=40000

--connect tcp server
tcp.client.connect (ip_server,port_server)
sleep(1)

--Move to ready point
wp1  =  {-16.418907,11.715555,-133.748860,-55.472799,-89.999982,-14.340675}
wp1_rad=deg_to_rad(wp1)
init_global_move_profile()
set_joint_maxvelc({1,1,1,1,1,1})
set_joint_maxacc({1,1,1,1,1,1})
move_joint(wp1_rad, true)

wp=get_current_waypoint()
pos_current={wp.pos.x,wp.pos.y,wp.pos.z}
ori_current={wp.ori.w,wp.ori.x,wp.ori.y,wp.ori.z}
rpy_current=quaternion2rpy(ori_current)

print("OK")
--Received quit exit program
recv=""
while(recv~="quit")do
    if(recv~="")then

        --delete []
        recv=string.sub(recv,2,-2)

        --Use", "split string
        table1=string.split(recv,",")

        -- Get the offset and convert it to a number for easy calculation
        x_offset=tonumber(table1[1])/1000
        y_offset=tonumber(table1[2])/1000
        t_offset=d2r(tonumber(table1[3]))

        --XY plus position offset
        pos_target={}
        pos_target[1]=pos_current[1]+x_offset
        pos_target[2]=pos_current[2]+y_offset
        pos_target[3]=pos_current[3]

        --RZ plus angle offset
        rpy_target={0,0,0}
        rpy_target[1]=rpy_current[1]
        rpy_target[2]=rpy_current[2]
        rpy_target[3]=rpy_current[3]+t_offset
        ori_target=rpy2quaternion(rpy_target)
```



```
70
71          -- Find the inverse of the target position
72          joint=get_target_pose(pos_target,ori_target,false,{0,0,0},{1,0,0,0})
73
74          -- joint movement to the target position
75          move_joint(joint,true)
76      end
77      -- Receive data from the server
78      --recv=clientRecvData(ip_server,port_server)
79      recv=tcp.client.recv_str_data(ip_server,port_server)
80      print(recv)
81      sleep(0.01)
82  end
83
```


遨博智能
AUBO

AUBO (Beijing)Robotics Technology Co.,Ltd.

Add: The 3rd Floor, Sunshine Buliding,China 102300（Headquarters）
Tel : +86 010-88595859 / 60864660

Email：info@aubo-robotics.cn
Web：www.aubo-robotics.cn

AUBO (Jiangsu)Robotics Co.,Ltd.

Add：The 3rd Floor, B Blotk Of Zhongke Business Center, Changzhou Science and Education Town, Changzhou, Jiangsu, China
Tel：+86 0519-86339960