

# ER 系列工业机器人 Python 脚本应 用参考手册

## RCS2 V2.0.2

南京埃斯顿自动化股份有限公司  
南京埃斯顿机器人工程有限公司

## 文档修订记录

序号	版本号	修订日期	修订概述	修订人
1	V2.0.1	2021.11.17	新建文档	应用软件部
2	V2.0.2	2022.04.15	修改 2.1 节的描述	应用软件部

# 目录

前言 .....	1
读者对象 .....	1
注意事项 .....	1
安全说明 .....	1
<b>第 1 章 Python 脚本配置篇 .....</b>	<b>2</b>
1.1 本节概要 .....	2
1.2 操作步骤 .....	2
<b>第 2 章 ModbusTcp 篇 .....</b>	<b>7</b>
2.1 本节概要 .....	7
2.2 配置启动脚本的 ModbusTCP .....	7
<b>第 3 章 脚本机器人 3D 接口篇 .....</b>	<b>11</b>
3.1 本节概要 .....	11
3.2 脚本 3d 接口配置 .....	11
<b>第 4 章 欧姆龙 Fins 协议篇 .....</b>	<b>16</b>
4.1 本节概要 .....	16
4.2 Fins 通信模板程序 .....	16
4.3 标准通信程序 .....	18
<b>第 5 章 三菱 Mc 协议篇 .....</b>	<b>21</b>
5.1 本节概要 .....	21
5.2 Mc 通信模板程序 .....	21
5.3 Mc 标准通信程序 .....	23
<b>第 6 章 机器人接口篇 .....</b>	<b>26</b>
6.1 本节概要 .....	26
6.2 机器人接口 demo 例程 .....	26
6.3 Java 下 Python 多线程编程示例 .....	28

# 前言

本手册适用于控制系统 RCS2 系列，介绍埃斯顿 ER 系列机器人相关通信使用和调试方法。




## 读者对象

本手册仅供埃斯顿机器人相关技术支持人员使用。

## 注意事项

- 在安装和调试这些组件时，操作人员必须严格遵循本文档的说明和解释。
- 相关负责人员必须确保所述产品的应用或使用满足所有安全要求，包括相关法律、法规、准则和标准。
- 尽管本文档经过精心编制，但由于其中所描述的产品仍处于不断更新换代中，我们可能不会在每次更新后都检查文档中所描述的产品性能数据、标准或其它特性总是与实际产品相一致。
- 本文档中难免会出现一些技术或者编辑错误，我们保留随时对文档信息做出修改之权力，恕不另行通知。对于已经变更的产品，如果本文档中的数据、图表以及文字描述没有修改，我们将不再特别加以声明。
- 任何人不得对软、硬件配置进行文本档中规定之外的修改，ESTUN 公司对因此而造成的一切后果不承担任何责任。
- 本文档中出现图示单位在没有特别标注说明时，默认单位为毫米 mm。

## 安全说明

 <b>警告</b>	<b>受伤的危险</b> 不遵守本标志相关的安全说明将危及个人生命和健康安全。
 <b>注意</b>	<b>对环境和设备有危险</b> 不遵守本标志相关安全说明可能明显危害环境和设备安全。
 <b>说明</b>	<b>说明或提示</b> 该标志表示这些信息能够帮助您更好的理解安全说明。

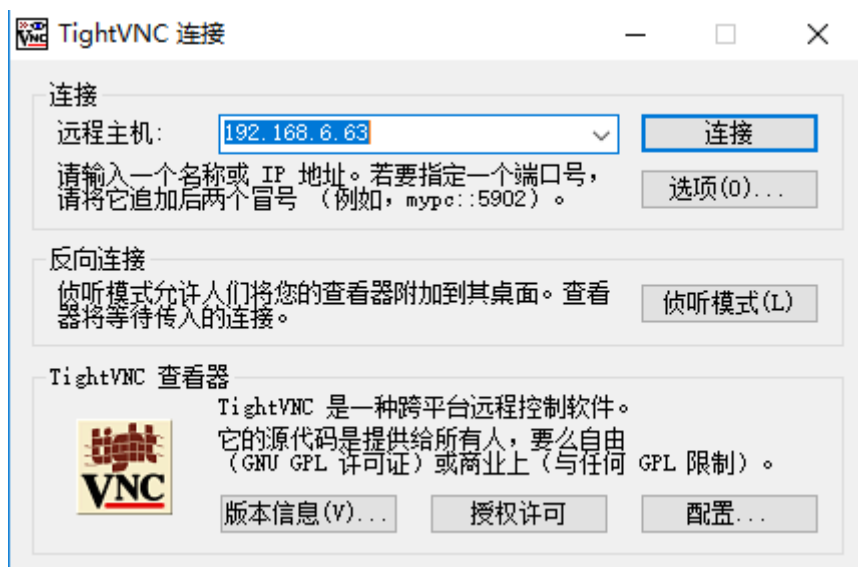
# 第 1 章 Python 脚本配置篇

## 1.1 本节概要

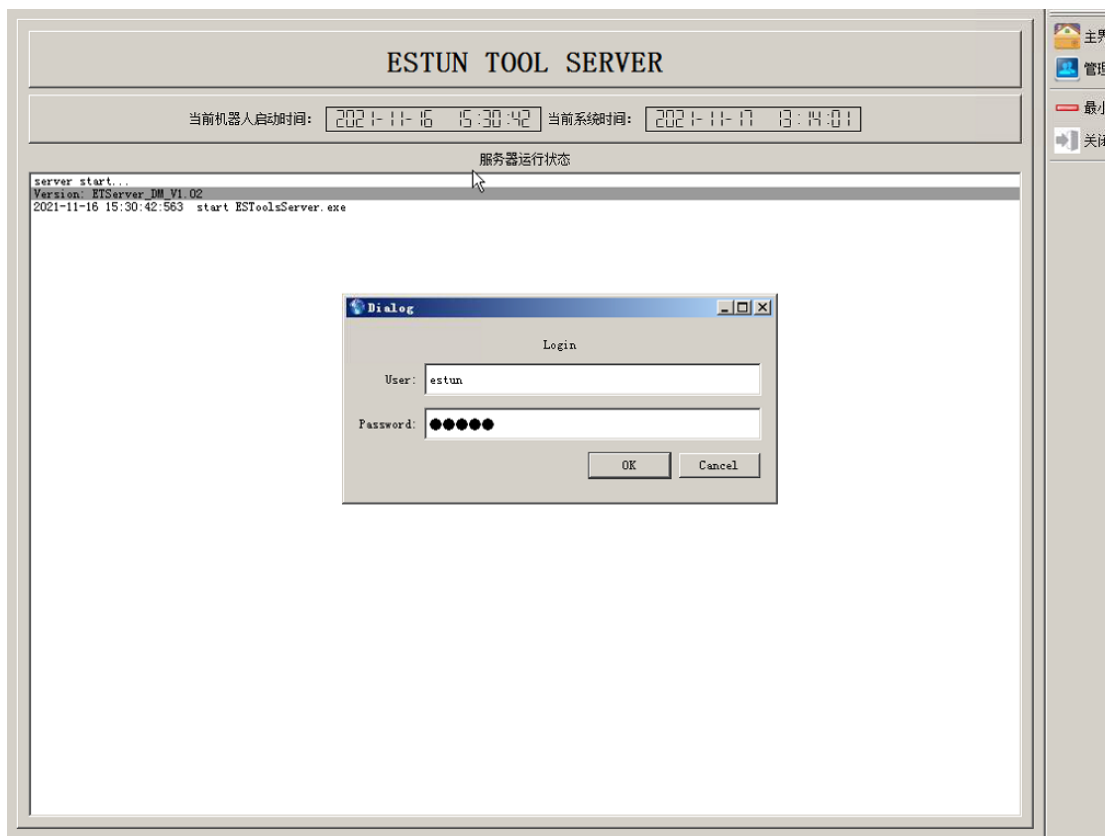
脚本的最新配置说明，在老版本的 runtime 下如何配置脚本的环境。在发布的 runtime 下如何升级脚本文件。

## 1.2 操作步骤

1. 用vnc连接进入控制器，在如下界面中输入控制器IP地址(默认IP地址是192.168.6.63)和密码：ERC654321



VNC连接控制器

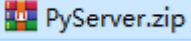


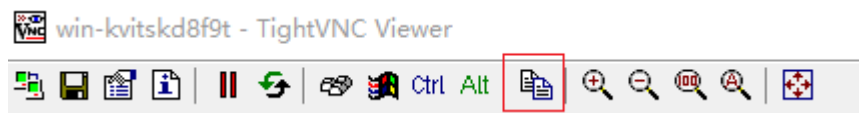
输入EstunTool密码,点击最小化进入系统

2. 检查控制器C:\runtime\python目录下是否存在PyServer文件夹:



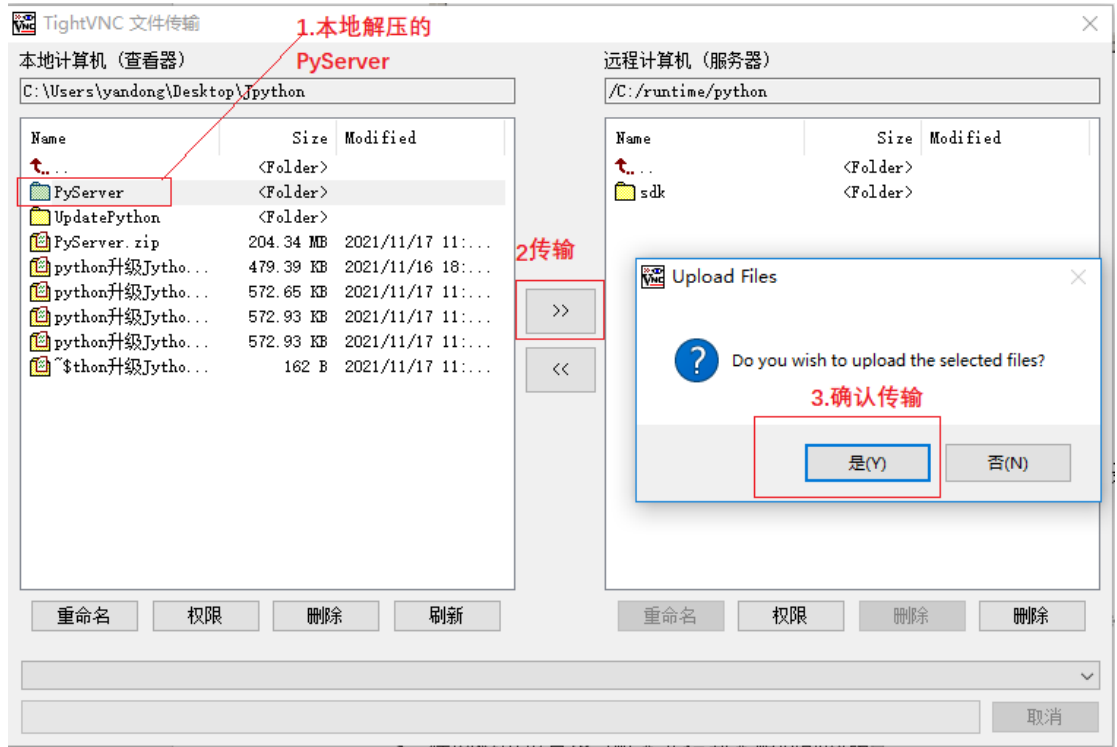
PyServer文件夹目录

如果不存在PyServer文件夹,把  PyServer.zip 文件夹解压到本地目录,点击vnc中的传输按钮,如下图

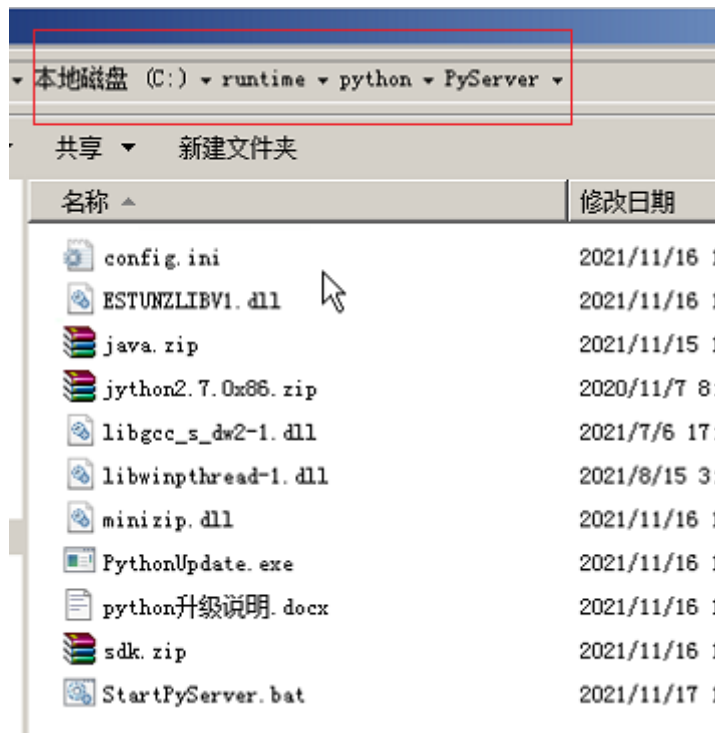


Vnc传输菜单

选中本地解压的PyServer文件夹,传输到控制器中,如下图所示

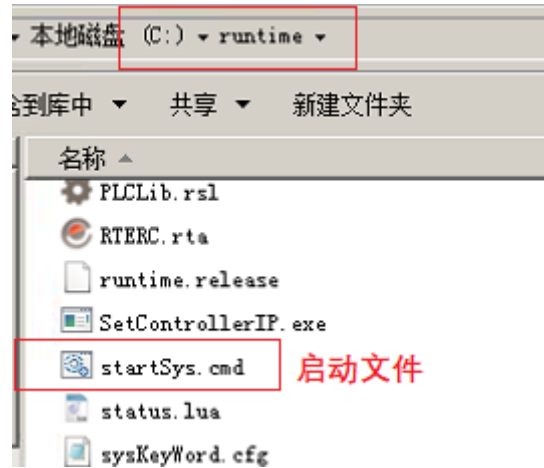


PyServer传输



传输后控制器的PyServer目录

- 在runtime包里的启动文件中配置PyServer的启动项，添加  
`cd /d C:\runtime\python\PyServer`  
`start C:\runtime\python\PyServer\StartPyServer.bat`



Runtime包里面的启动文件

```

startSys.cmd - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
cd /d c:\runtime
ping 127.0.0.1 -n 1 >nul
start C:\runtime\ERC_HMI\updateESToolsServer.cmd
ping 127.0.0.1 -n 3 >nul
start C:\runtime\SetControllerIP.exe
ping 127.0.0.1 -n 3 >nul
nodemgr "stop NodeA"
ping 127.0.0.1 -n 3 >nul
nodemgr "start NodeA"
ping 127.0.0.1 -n 8 >nul
ldrta.exe C:\runtime\rtk\prt.img
ping 127.0.0.1 -n 10 >nul
ldrta.exe "C:\Program Files\Intime\bin\usb3.rta"
ping 127.0.0.1 -n 5 >nul
ldrta.exe C:\runtime\RTERC.rta
ping 127.0.0.1 -n 120 >nul
start C:\runtime\rtk\EipAdapter\Adapter.img
cd /d C:\runtime\python\PyServer
start C:\runtime\python\PyServer\StartPyServer.bat
  
```

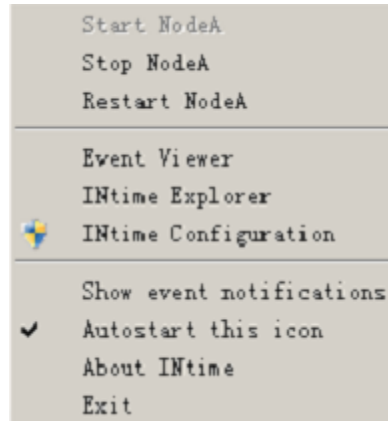
PyServer的启动配置

启动文件中添加的内容

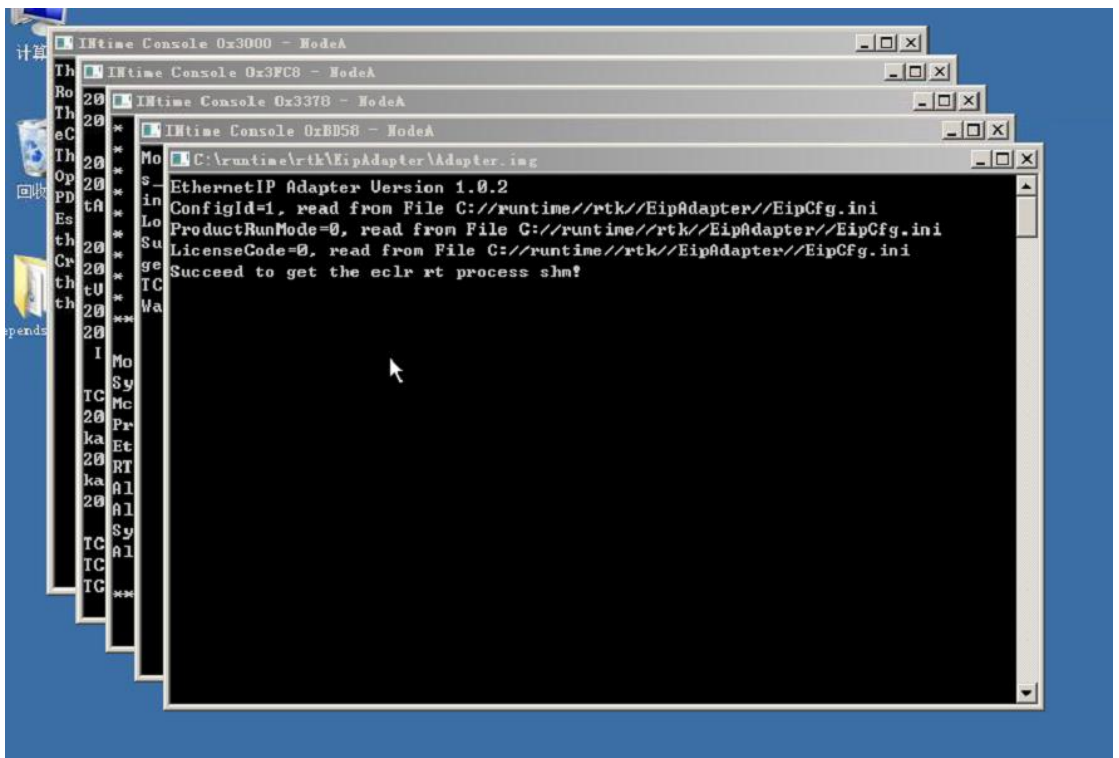
4. 保存，关闭intime，以及现有窗口，再启动项中启动statSys.bat文件
5. 关闭intime操作：首先，右击电脑右下角intime图标，在左击出现的Stop NoteA，随后进行所有控制台窗口的关闭。



红色图标为intime图标



选择Stop NoteA



控制台窗口

6. 配置完成，立即生效

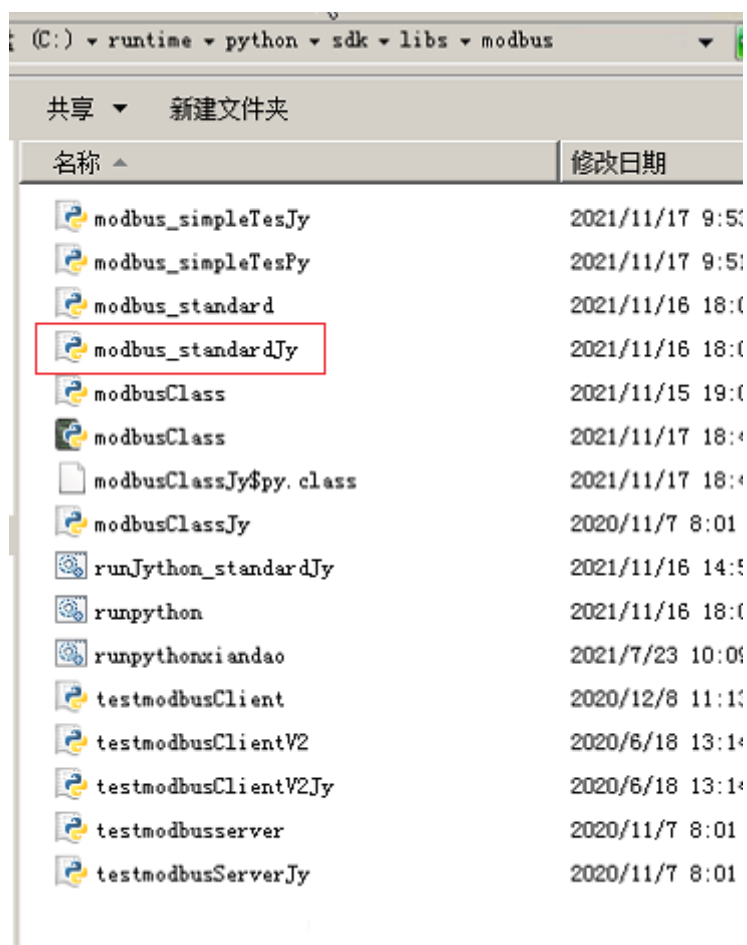
## 第 2 章 ModbusTcp 篇

### 2.1 本节概要

介绍如何配置modbus服务端的端口以及启动Modbus服务端。默认的modbusTCP的点表同<<ER系列工业机器人ModBusTCP接口调试手册\_RCS2 V1.5>>,除地址40005~40014地址对应的显示当前加载工程名功能、40054~40063地址对应的设置加载工程名功能、40100地址对应的读写标志位请求功能的功能不再支持，其余的和文档一致。

### 2.2 配置启动脚本的 ModbusTCP

1. 同样使用vnc进入控制器，打开C:\runtime\python\sdk\libs\modbus目录，



Modbus库目录

启动modbusTCP需要先配置一下modbusTCP的端口号，打开该目录下的

modbus\_standardJy.py文件，如下图修改IP地址为控制器IP地址，端口号为modbusTCP的服务器的端口号

```

modbus_standardJy.py - C:\runtime\python\sdk\libs\modbus\modbus_standardJy.py
File Edit Format Run Options Windows Help

import math
from struct import *

import threading
import sys
sys.path.append("C:\\runtime\\python\\sdk\\bin")
sys.path.append("C:\\runtime\\python\\sdk\\libs\\modbus")
sys.path.append("C:\\runtime\\python\\sdk\\libs\\RobotFace")
sys.path.append("C:\\runtime\\python\\sdk\\bin\\loadLibraryForJy.jar")

from com.Estun.App import loadLibraryForJy
from java.lang import String

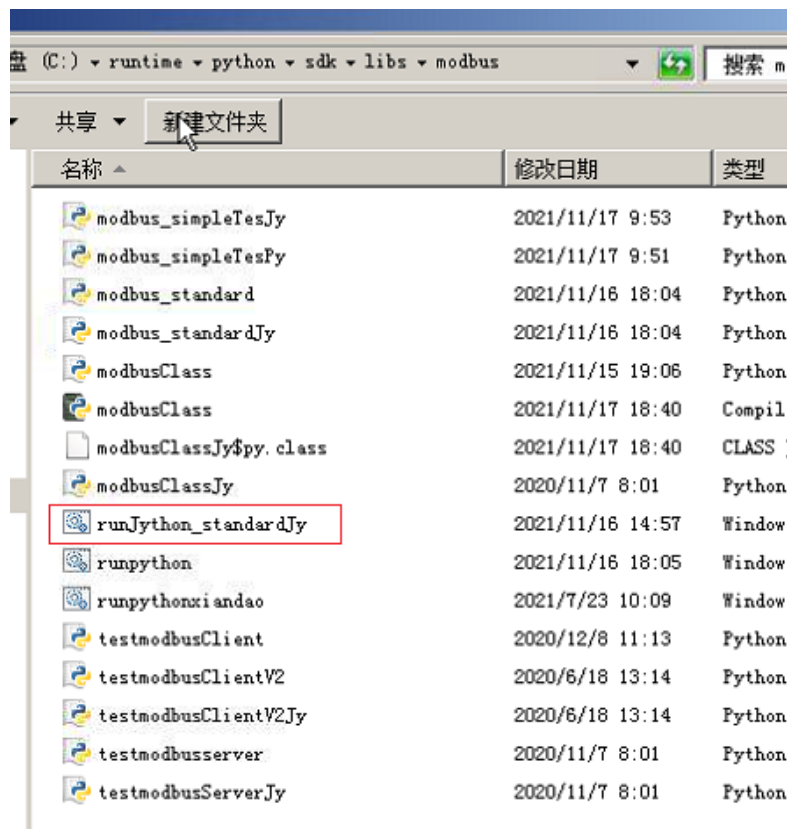
g_loadlibRobot = loadLibraryForJy()
g_loadlibRobot.loadlibrary("RobotFaceForJava")
g_loadlibRobot.loadlibrary("ModbusForJava")

from modbusClassJy import ModbusServerV3
from RobotFaceClassJy import RobotFaceJy

g_Robot = RobotFaceJy()
g_ModbusServer = ModbusServerV3("192.168.6.63", 503, 2000)
    
```

ModbusTCP服务器中的IP地址和端口号

2. 双击运行runJython\_standarJy.bat启动文件，启动ModbusTCP



modbusTCP启动文件

```

tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
disconnect with plc!!
disconnect with plc!!
disconnect with plc!!
disconnect with plc!!
disconnect with plc!!
disconnect with plc!!
disconnect with plc!!
('main : time gap1:', 0.01699
('gap time:', 0.0160000324249
('simdo2 = ', 0.0)
('isMoving = ', 0)
('main : time gap1:', 0.03200
('gap time:', 0.0320000648498
('simdo2 = ', 0.0)
('isMoving = ', 0)
('main : time gap1:', 0.01699
('gap time:', 0.0160000324249
('simdo2 = ', 0.0)
('isMoving = ', 0)
('main : time gap1:', 0.01699
('gap time:', 0.0320000648498
('simdo2 = ', 0.0)
('isMoving = ', 0)
('main : time gap1:', 0.01600
('gap time:', 0.0319998264312
('simdo2 = ', 0.0)
('gap time:', 0.0160000324249
('isMoving = ', 0)
('simdo2 = ', 0.0)

```

Modbus服务器脚本正常运行的打印信息

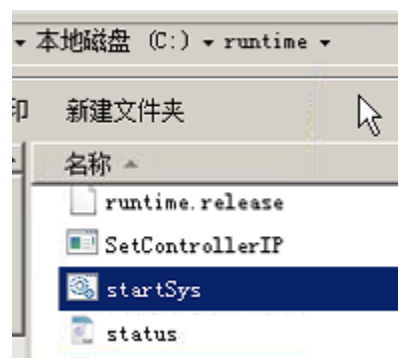
- 脚本正常运行后，需要把runJython\_standardJy.bat加到

C:\runtime\startSys.bat启动文件中

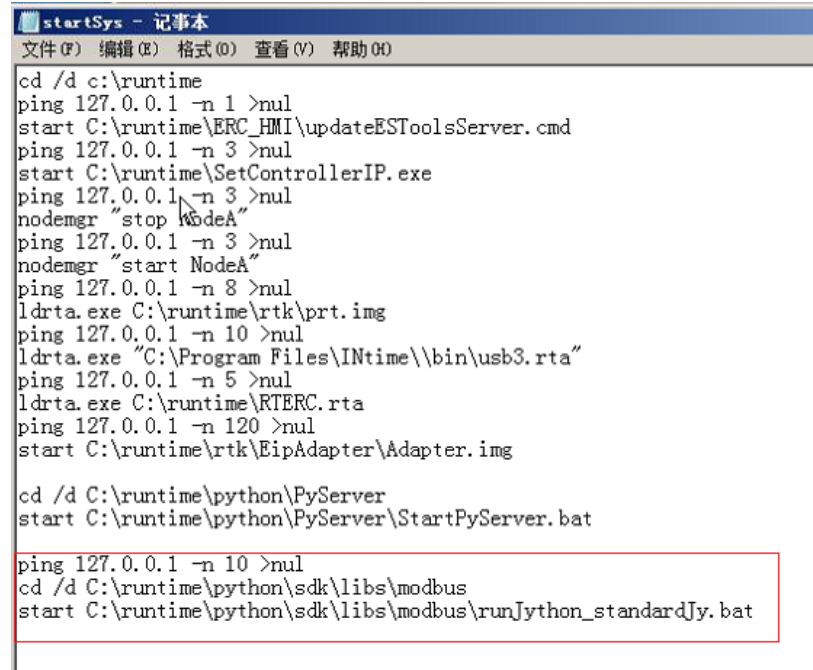
```
ping 127.0.0.1 -n 10 >nul
```

```
cd /d C:\runtime\python\sdk\libs\modbus
```

```
start C:\runtime\python\sdk\libs\modbus\runJython_standardJy.bat
```



Runtime包启动文件



```
startSys - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

cd /d c:\runtime
ping 127.0.0.1 -n 1 >nul
start C:\runtime\ERC_HMI\updateESToolsServer.cmd
ping 127.0.0.1 -n 3 >nul
start C:\runtime\SetControllerIP.exe
ping 127.0.0.1 -n 3 >nul
nodemgr "stop NodeA"
ping 127.0.0.1 -n 3 >nul
nodemgr "start NodeA"
ping 127.0.0.1 -n 8 >nul
ldrta.exe C:\runtime\rtk\prt.img
ping 127.0.0.1 -n 10 >nul
ldrta.exe "C:\Program Files\INtime\bin\usb3.rta"
ping 127.0.0.1 -n 5 >nul
ldrta.exe C:\runtime\RTERC.rta
ping 127.0.0.1 -n 120 >nul
start C:\runtime\rtk\EipAdapter\Adapter.img

cd /d C:\runtime\python\PyServer
start C:\runtime\python\PyServer\StartPyServer.bat

ping 127.0.0.1 -n 10 >nul
cd /d C:\runtime\python\sdk\libs\modbus
start C:\runtime\python\sdk\libs\modbus\runJython_standardJy.bat
```

#### ModbusTCP启动项添加

4. 手动关闭系统，重启双击运行一下startSys.bat.一切正常，即可关机重启。

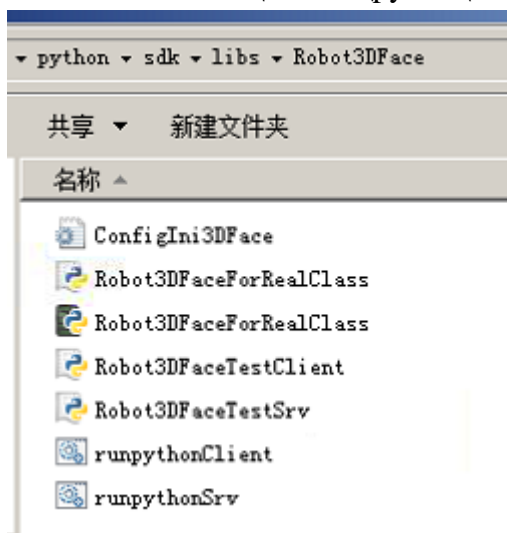
## 第 3 章 脚本机器人 3D 接口篇

### 3.1 本节概要

如何配置和启动3d接口的服务端和客户端。3d接口启动后，协议符合文档《ER系列工业机器人三维视觉调试手册(python下Robot3DFace库)\_RCS2 V1.5.1》

### 3.2 脚本 3d 接口配置

1. 同样使用 vnc 进入控制器，打开 C:\runtime\python\sdk\libs\Robot3DFace



3d 接口目录

打开 Robot3DFaceTestClient.py 文件打开脚本，这是 3d 接口的客户端，需要设置其中的连接的目标的 IP 地址和端口号

```
Robot3DFaceTestClient.py - C:\runtime\python\sdk\libs\Robot3DFace\Robot3DFaceTe
File Edit Format Run Options Windows Help
|
import sys
sys.path.append("C:\\runtime\\python\\sdk\\bin")
sys.path.append("C:\\runtime\\python\\sdk\\libs\\Robot3DFace")
import time

from Robot3DFaceForRealClass import Robot3DFaceForReal

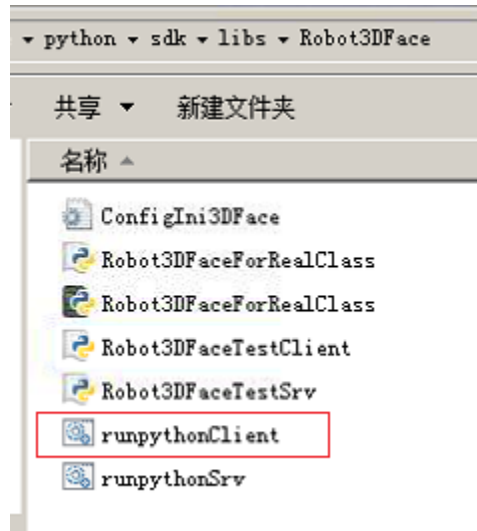
n3DFace = Robot3DFaceForReal()

#print("Face:",n3DFace.Init_Cam2d3d_Server(7080)) 目标IP地址 目标端口号
print("Client Face:",n3DFace.Init_Cam2d3d_Client("127.0.0.1",8523))

while True:
    time.sleep(10)
```

## 3d 接口客户端 IP 地址和端口号设置

双击运行 runpythonClient.bat，启动 3d 接口客户端的脚本，客户端有自动重连功能



3d 接口启动文件

```
tran id 1
tran id 2
(<'Client Face:', True>)
user count: 0
user count: 0
由于目标计算机积极拒绝，无法连接。
ThreadProc_SendRev_client quit
ThreadProc_SendRev_client quit
user count: 1
user count: 1
user count: 1
user count: 1
反复重连目标服务器
TcpClient_ptr delete
TcpClient_ptr delete
由于目标计算机积极拒绝，无法连接。
ThreadProc_SendRev_client quit
由于目标计算机积极拒绝，无法连接。
ThreadProc_SendRev_client quit
user count: 1
user count: 1
user count: 1
user count: 1
TcpClient_ptr delete
TcpClient_ptr delete
```

运行打印窗口



### 测试接口

2. 脚本正常运行后，需要把 `runpythonClient.bat` 加到 `C:\runtime\startSys.bat` 文件中

```
ping 127.0.0.1 -n 10 >nul
```

```
cd /d C:\runtime\python\sdk\libs\Robot3DFace
```

```
start C:\runtime\python\sdk\libs\ Robot3DFace \runpythonClient.bat
```

```
startSys - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

cd /d c:\runtime
ping 127.0.0.1 -n 1 >nul
start C:\runtime\ERC_HMI\updateESToolsServer.cmd
ping 127.0.0.1 -n 3 >nul
start C:\runtime\SetControllerIP.exe
ping 127.0.0.1 -n 3 >nul
nodemgr "stop NodeA"
ping 127.0.0.1 -n 3 >nul
nodemgr "start NodeA"
ping 127.0.0.1 -n 8 >nul
ldrta.exe C:\runtime\rtk\prt.img
ping 127.0.0.1 -n 10 >nul
ldrta.exe "C:\Program Files\INtime\bin\usb3.rta"
ping 127.0.0.1 -n 5 >nul
ldrta.exe C:\runtime\RTERC.rta
ping 127.0.0.1 -n 120 >nul
start C:\runtime\rtk\EipAdapter\Adapter.img

cd /d C:\runtime\python\PyServer
start C:\runtime\python\PyServer\StartPyServer.bat

ping 127.0.0.1 -n 10 >nul
cd /d C:\runtime\python\sdk\libs\Robot3DFace
start C:\runtime\python\sdk\libs\Robot3DFace\runpythonClient.bat
```

### 启动配置文件修改

3. 服务端的配置，需要打开 `C:\runtime\python\sdk\libs\Robot3DFace` 目录下的 `Robot3DFaceTestSrv.py`

```
Robot3DFaceTestSrv.py - C:\runtime\python\sdk\libs\Robot3DFace\Robot3DFaceTest
File Edit Format Run Options Windows Help
|
import sys
sys.path.append("C:\\runtime\\python\\sdk\\bin")
sys.path.append("C:\\runtime\\python\\sdk\\libs\\Robot3DFace")
import time

from Robot3DFaceForRealClass import Robot3DFaceForReal

n3DFace = Robot3DFaceForReal()

print("Face:",n3DFace.Init_Cam2d3d_Server(7080))
#print("Client Face:",n3DFace.Init_Cam2d3d_Client("127.0.0.1",8523))

while True:
    time.sleep(10)
```

### 3d 接口服务器脚本

修改脚本的端口号。启动 runpythonsrv.bat 文件。运行脚本

```
C:\Windows\system32\cmd.exe
tran id 2
tran id 3
tran id 1
get shm success
get shm success
get shm success
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
get shm success
get shm success
get shm success
tran id 3
tran id 1
tran id 2
Cam2d3d init!!!Cam2d3d init!!!<'Face:', True>
```

### 3d 接口服务端脚本



### 测试 3d 接口服务端

- 脚本正常运行后，需要把 runpythonSrv.bat 加到启动项中  
 ping 127.0.0.1 -n 10 >nul  
 cd /d C:\runtime\python\sdk\libs\Robot3DFace  
 start C:\runtime\python\sdk\libs\ Robot3DFace \runpythonSrv.bat

```

startSys - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

cd /d c:\runtime
ping 127.0.0.1 -n 1 >nul
start C:\runtime\ERC_HMI\updateESToolsServer.cmd
ping 127.0.0.1 -n 3 >nul
start C:\runtime\SetControllerIP.exe
ping 127.0.0.1 -n 3 >nul
nodemgr "stop NodeA"
ping 127.0.0.1 -n 3 >nul
nodemgr "start NodeA"
ping 127.0.0.1 -n 8 >nul
ldrta.exe C:\runtime\rtk\prt.img
ping 127.0.0.1 -n 10 >nul
ldrta.exe "C:\Program Files\INtime\bin\usb3.rta"
ping 127.0.0.1 -n 5 >nul
ldrta.exe C:\runtime\RTERC.rta
ping 127.0.0.1 -n 120 >nul
start C:\runtime\rtk\EipAdapter\Adapter.img

cd /d C:\runtime\python\PyServer
start C:\runtime\python\PyServer\StartPyServer.bat

ping 127.0.0.1 -n 10 >nul
cd /d C:\runtime\python\sdk\libs\Robot3DFace
start C:\runtime\python\sdk\libs\Robot3DFace\runpythonSrv.bat
  
```

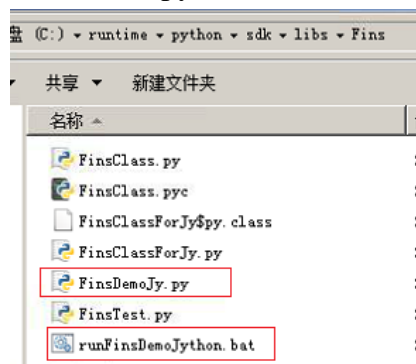
## 第 4 章 欧姆龙 Fins 协议篇

### 4.1 本节概要

介绍 Fins 通信模板程序；标准通信程序的配置、启动，标准通信程序是用了 modbusTCP 的点表，只不过地址是 Fins 的地址不是 modbus 地址，这样直接启动，安装点表可以与 PLC 程序通信。

### 4.2 Fins 通信模板程序

1. 同样进入 vnc，打开 C:\runtime\python\sdk\libs\Fins 目录



脚本 Fins 目录

打开 FinsDemoJy.py 文件，在自定义逻辑区就用 Fins 接口类写自己的逻辑了。

```
FinsTestJy.py - C:\runtime\python\sdk\libs\Fins\FinsTestJy.py
File Edit Format Run Options Windows Help

import sys
sys.path.append("C:\\runtime\\python\\sdk\\libs\\CommPyHead")

import time
import CommHeadJy
from FinsClassForJy import FinsTcpV2

g_finsclient = FinsTcpV2("192.168.6.100", 9600, 63)

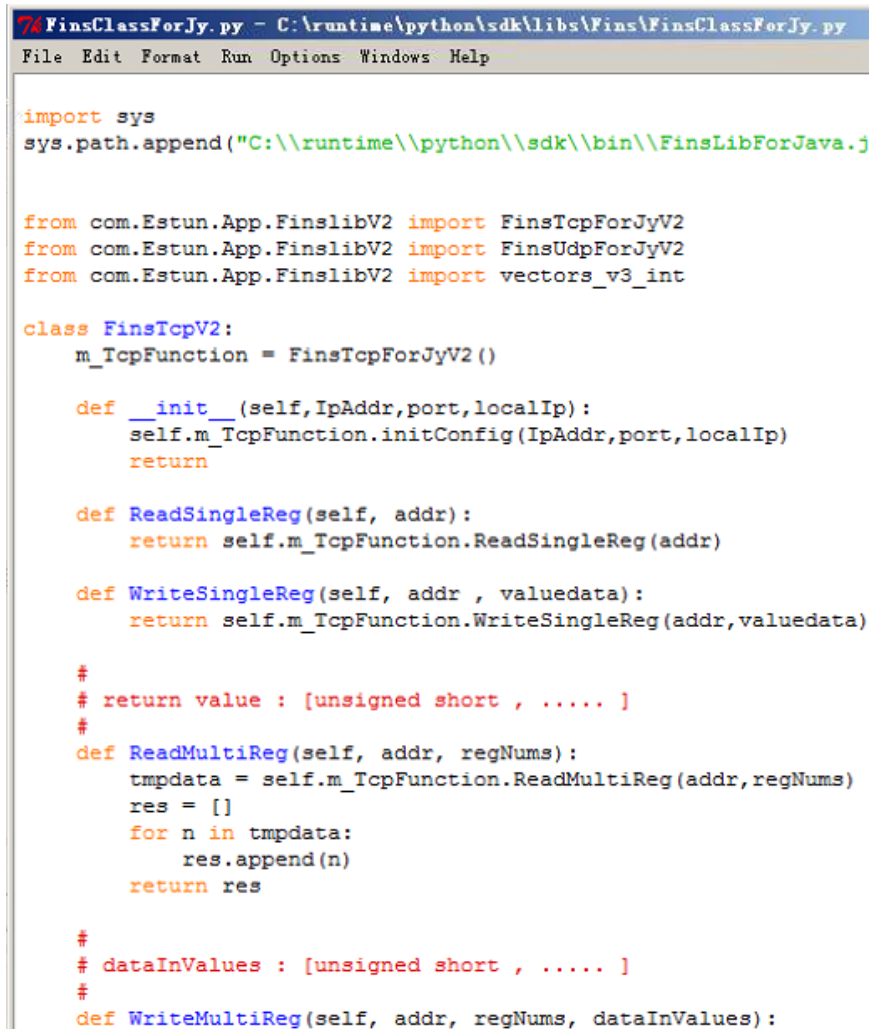
while True:
    cmd = raw_input("cmd:")
    if int(cmd) == 1:
        t1 = time.time()
        print("WriteMultiReg:", g_finsclient.WriteMultiReg(302))
        print("time gap:", time.time() - t1)
    elif int(cmd) == 2:
        t1 = time.time()
        vv = g_finsclient.ReadMultiReg(3020, 10)
        print("time gap:", time.time() - t1)
        for n in vv:
            print(n)
    elif int(cmd) == 3:
        break
    else:
        continue
```

FinsDemo 模板

在通信对象创建的时候需要指定 PLC 的 IP 地址和端口号，以及本机 IP 地址的最后一位（例如 192.168.6.63 是控制器 IP 地址，这里就是本机 63）

```
g_finsclient = FinsTcpV2("192.168.6.100",9600,63)
```

### Fins 的 IP 地址和端口号以及本机 IP



```
FinsClassForJy.py - C:\runtime\python\sdk\libs\Fins\FinsClassForJy.py
File Edit Format Run Options Windows Help

import sys
sys.path.append("C:\\runtime\\python\\sdk\\bin\\FinsLibForJava.j

from com.Estun.App.FinslibV2 import FinsTcpForJyV2
from com.Estun.App.FinslibV2 import FinsUdpForJyV2
from com.Estun.App.FinslibV2 import vectors_v3_int

class FinsTcpV2:
    m_TcpFunction = FinsTcpForJyV2()

    def __init__(self, IpAddr, port, localIp):
        self.m_TcpFunction.initConfig(IpAddr, port, localIp)
        return

    def ReadSingleReg(self, addr):
        return self.m_TcpFunction.ReadSingleReg(addr)

    def WriteSingleReg(self, addr, valuedata):
        return self.m_TcpFunction.WriteSingleReg(addr, valuedata)

    #
    # return value : [unsigned short, .....]
    #
    def ReadMultiReg(self, addr, regNums):
        tmpdata = self.m_TcpFunction.ReadMultiReg(addr, regNums)
        res = []
        for n in tmpdata:
            res.append(n)
        return res

    #
    # dataInValues : [unsigned short, .....]
    #
    def WriteMultiReg(self, addr, regNums, dataInValues):
```

### Fins 通信的接口类

接口类在文件 FinsClassForJy.py 中。

2. 运行 runFinsDemoPython.bat, 就可以启动 demo 程序的脚本

```

C:\Windows\system32\cmd.exe
C:\runtime\python\sdk\libs\Fins>cd /d C:\runtime\python\sdk\libs\Fins
C:\runtime\python\sdk\libs\Fins>jython.exe FinsTestJy.py
user count: 0
connect Plc ok
cmd:1head packet ok !!send head ok!!!cmd:
('WriteMultiReg:', 0)
('time gap:', 0.015000104904174805)
cmd:2
('time gap:', 0.016000032424926758)
1
2
3
4
5
6
7
8
9
10
cmd:1
('WriteMultiReg:', 0)
('time gap:', 0.03099989891052246)
cmd:

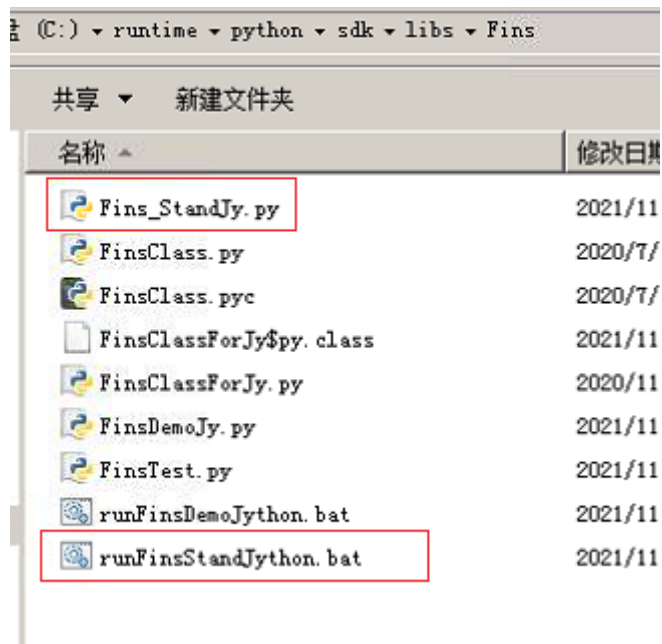
```

FinsDemo 测试程序运行窗口

3. 脚本正常运行后,需要把runFinsDemoJython.bat 加到 c:\runtime\startSys.bat 中和上一章节都一样。这里不再重复

### 4.3 标准通信程序

6. 1. 同样使用vnc进入控制器, 打开C:\runtime\python\sdk\libs\Fins目录,



Fins库目录

启动Fins标准通信需要先配置一下PLC的IP地址和端口号, 打开该目录下的Fins\_StandJy.py文件, 如下图修改IP地址为PLC的IP地址, 端口号为PLC的端口号

```

74 Fins_StandJy.py - C:\runtime\python\sdk\libs\Fins\Fins_StandJy.py
File Edit Format Run Options Windows Help
sys.path.append(C:\runtime\python\sdk\bin)
sys.path.append("C:\\runtime\\python\\sdk\\libs\\Fins")
sys.path.append("C:\\runtime\\python\\sdk\\libs\\RobotFace")
sys.path.append("C:\\runtime\\python\\sdk\\bin\\loadLibraryForJy.java")
sys.path.append("../bin")
from com.Estun.App import loadLibraryForJy
from java.lang import String

g_loadlibRobot = loadLibraryForJy()
g_loadlibRobot.loadlibrary("RobotFaceForJava")
#g_loadlibRobot.loadlibrary("ModbusForJava")
g_loadlibRobot.loadlibrary("FinslibForJavaV2")
from FinsClassForJy import FinsTcpV2
from RobotFaceClassJy import RobotFaceJy

g_Robot = RobotFaceJy()

#g_ModbusServer = ModbusServerV3("192.168.6.12", 5033, 2000)
g_FinsTcpServer = FinsTcpV2("172.31.6.100", 9601, 63)

```

PLC的IP地址和端口号

本机IP最后一位数

PLC的IP地址和端口号以及本机IP

7. 双击运行runFinsStandJy.bat启动文件，启动Fins标准通信脚本

```

C:\Windows\system32\cmd.exe
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
user count: 0
connect Plc fail!!
connect Plc fail!!
disconnect with plc!!
disconnect with plc!!

C:\Windows\system32\cmd.exe
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
user count: 0
connect Plc ok
send head packet ok !!send

```

Fins标准通信脚本正常运行的打印信息

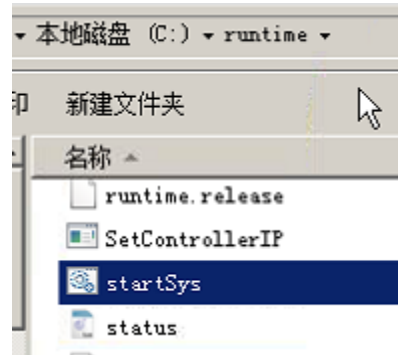
8. 脚本正常运行后，需要把runFinsStandJy.bat加到C:\runtime\startSys.bat启

动文件中

```
ping 127.0.0.1 -n 8 >nul
```

```
cd /d C:\runtime\python\sdk\libs\Fins
```

```
start C:\runtime\python\sdk\libs\Fins\runFinsStandJython.bat
```



Runtime包启动文件

```
startSys.cmd - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
cd /d c:\runtime
ping 127.0.0.1 -n 1 >nul
start C:\runtime\ERC_HMI\updateESToolsServer.cmd
ping 127.0.0.1 -n 3 >nul
start C:\runtime\SetControllerIP.exe
ping 127.0.0.1 -n 3 >nul
nodemgr "stop NodeA"
ping 127.0.0.1 -n 3 >nul
nodemgr "start NodeA"
ping 127.0.0.1 -n 8 >nul
ldrta.exe C:\runtime\rtk\prt.img
ping 127.0.0.1 -n 10 >nul
ldrta.exe "C:\Program Files\INtime\bin\usb3.rta"
ping 127.0.0.1 -n 5 >nul
ldrta.exe C:\runtime\RTERC.rta
ping 127.0.0.1 -n 120 >nul
start C:\runtime\rtk\EipAdapter\Adapter.img

cd /d C:\runtime\python\PyServer
start C:\runtime\python\PyServer\StartPyServer.bat

ping 127.0.0.1 -n 10 >nul
cd /d C:\runtime\python\sdk\libs\Fins
start C:\runtime\python\sdk\libs\Fins\runFinsStandJython.bat
```

Fins标准通信脚本启动项添加

手动关闭系统，重启双机运行一下 startSys.bat.一切正常，即可关机重启。

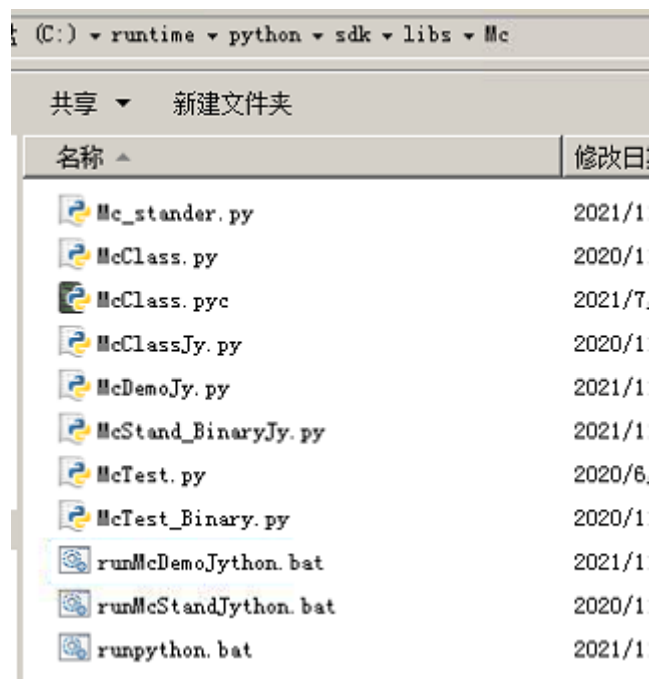
## 第 5 章 三菱 Mc 协议篇

### 5.1 本节概要

介绍 Mc 通信模板程序；标准通信程序的配置、启动，标准通信程序是用了 modbusTCP 的点表，只不过地址是 Mc 的地址不是 modbus 地址，这样直接启动，安装点表可以与 PLC 程序通信。

### 5.2 Mc 通信模板程序

1. 同样进入 vnc，打开 C:\runtime\python\sdk\libs\Mc 目录



脚本 Mc 目录

打开 McDemoJy.py 文件，在自定义逻辑区用 Mc 接口类写自己的逻辑了

```

7% McDemoJy.py - C:\runtime\python\sdk\libs\Mc\McDemoJy.py
File Edit Format Run Options Windows Help

import sys
sys.path.append("C:\\runtime\\python\\sdk\\libs\\CommPyHead")

import time
import CommHeadJy  头文件和类对象导入区

from McClassJy import McClientTcpJy

g_ff = McClientTcpJy("192.168.6.200",5300)

while True:
    cmd = raw_input("cmd:")  自定义逻辑区

    if (int(cmd) == 1):
        print("WriteMultiReg_D",g_ff.WriteDBy3EBinary(80,5,[88,
    elif (int(cmd) == 5):
        gg1 = g_ff.ReadDBy3EBinary(80,5)
        for nn in gg1:
            print("nn:",nn)
    elif (int(cmd) == 6):
        print("WriteMultiReg_M",g_ff.WriteMBy3EBinary(10,5,[1,
    elif (int(cmd) == 7):
        gg2 = g_ff.ReadMBy3EBinary(10,5)
        for nn in gg2:
            print("nn:",nn)

```

### 三菱McDemo程序

在创建通信对象的时候，需要指定PLC的IP地址和端口号。

```
g_ff = McClientTcpJy("192.168.6.200",5300)
```

### 通信IP地址和端口号

2. 运行runMcDemoJython.bat,启动Mcdemo的脚本，运行效果如下

```

C:\Windows\system32\cmd.exe
0\bin;C:\runtime\Python\sdk\Java\bin
C:\runtime\python\sdk\libs\Mc>cd /d C:\runtime\python\
C:\runtime\python\sdk\libs\Mc>jython.exe McDemoJy.py
WSAIoctl res : 0 -----
connect Plc ok
cmd:1
(<'WriteMultiReg_D', True)
cmd:2
cmd:5
(<'nn:', 88)
(<'nn:', 99)
(<'nn:', 111)
(<'nn:', 222)
(<'nn:', 333)
cmd:6
(<'WriteMultiReg_M', True)
cmd:7
(<'nn:', 1)
(<'nn:', 1)
(<'nn:', 1)
(<'nn:', 1)
(<'nn:', 1)
cmd:

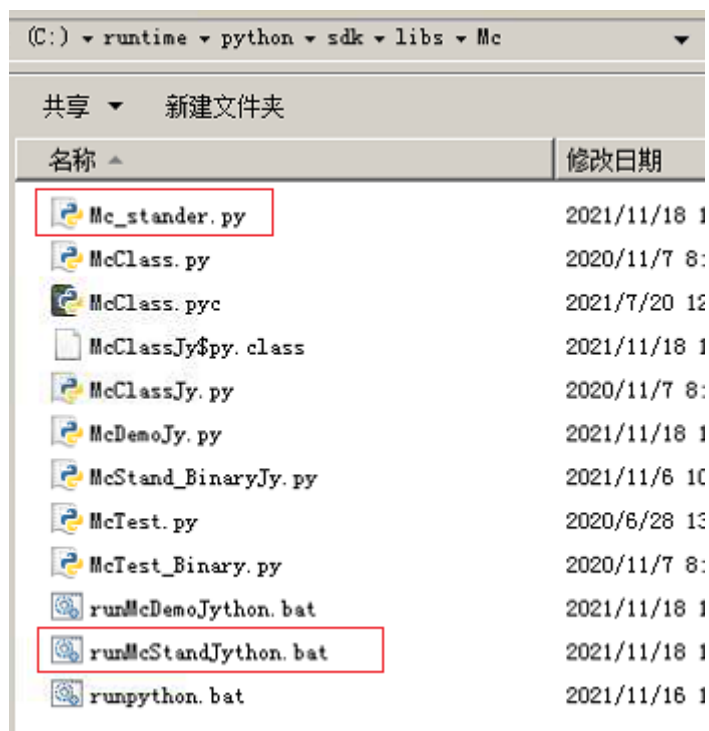
```

### McDemo测试程序脚本运行

- 脚本正常运行后，需要把runMcDemoJython.bat加到c:\runtime\startSys.bat中，同上面章节一致，这里不再重复

## 5.3 Mc 标准通信程序

- 同样使用vnc进入控制器，打开C:\runtime\python\sdk\libs\Mc目录，



Mc库目录

启动Mc标准通信程序需要先配置一下PLC的Ip地址和端口号，打开该目录下的Mc\_stander.py文件，如下图修改IP地址为PLC的IP地址，端口号为PLC的服务器的端口号

```

78 Mc_stander.py - C:\runtime\python\sdk\libs\mc\Mc_stander.py
File Edit Format Run Options Windows Help

import time
import threading
import sys
from struct import *

import sys
sys.path.append("C:\\runtime\\python\\sdk\\bin")
sys.path.append("C:\\runtime\\python\\sdk\\libs\\Mc")
sys.path.append("C:\\runtime\\python\\sdk\\libs\\RobotFace")
sys.path.append("C:\\runtime\\python\\sdk\\bin\\loadLibraryForJy")

from com.Estun.App import loadLibraryForJy

g_loadlib = loadLibraryForJy()
g_loadlib.loadlibrary("JavaMcLib")
g_loadlib.loadlibrary("RobotFaceForJava")

from McClassJy import McClientTcpJy
from RobotFaceClassJy import RobotFaceJy

g_mc = McClientTcpJy("192.168.6.200", 5300)
g_Robot = RobotFaceJy()

```

PLC的IP地址和端口号

Mc的PLC的IP地址和端口号

9. 双击运行runMcStandJython.bat启动文件，启动Mc标准通信脚本程序

```

C:\Windows\system32\cmd.exe
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
WSAIocctl res : 0 -----
connect Plc fail!!
CICPSocket::Close_1
disconnect with plc!!
disconnect with plc!!
disconnect with plc!!

C:\Windows\system32\cmd.exe
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
WSAIocctl res : 0 -----
connect Plc ok

```

Mc标准通信脚本正常运行的打印信息

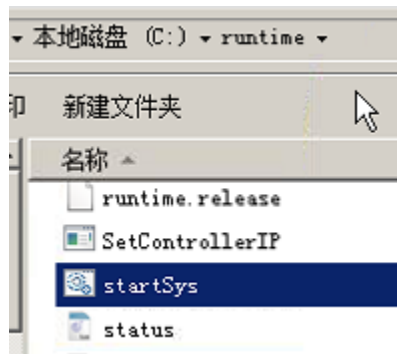
10. 脚本正常运行后，需要把runMcStandJython.bat加到

C:\runtime\startSys.bat启动文件中

```
ping 127.0.0.1 -n 10 >nul
```

```
cd /d C:\runtime\python\sdk\libs\Mc
```

```
start C:\runtime\python\sdk\libs\Mc\runMcStandJython.bat
```



Runtime包启动文件

```
startSys.cmd - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
cd /d c:\runtime
ping 127.0.0.1 -n 1 >nul
start C:\runtime\ERC_HMI\updateESToolsServer.cmd
ping 127.0.0.1 -n 3 >nul
start C:\runtime\SetControllerIP.exe
ping 127.0.0.1 -n 3 >nul
nodemgr "stop NodeA"
ping 127.0.0.1 -n 3 >nul
nodemgr "start NodeA"
ping 127.0.0.1 -n 8 >nul
ldrta.exe C:\runtime\rtk\prt.img
ping 127.0.0.1 -n 10 >nul
ldrta.exe "C:\Program Files\INtime\bin\usb3.rta"
ping 127.0.0.1 -n 5 >nul
ldrta.exe C:\runtime\RTERC.rta
ping 127.0.0.1 -n 120 >nul
start C:\runtime\rtk\EipAdapter\Adapter.img

cd /d C:\runtime\python\PyServer
start C:\runtime\python\PyServer\StartPyServer.bat

ping 127.0.0.1 -n 10 >nul
cd /d C:\runtime\python\sdk\libs\Mc
start C:\runtime\python\sdk\libs\Mc\runMcStandJython.bat
```

Mc标准启动项添加

手动关闭系统，重启双机运行一下 startSys.bat.一切正常，即可关机重启。。

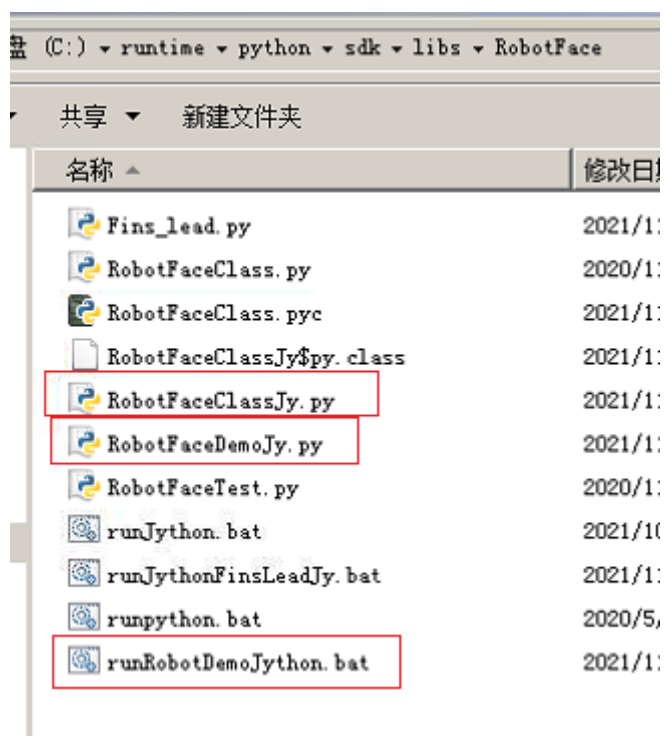
## 第 6 章 机器人接口篇

### 6.1 本节概要

介绍机器人接口的使用，和程序模板。

### 6.2 机器人接口 demo 例程

1. 同样进入vnc，打开C:\runtime\python\sdk\libs\RobotFace目录



机器人接口目录

打开RobotFaceDemoJy.Py文件，在自定义逻辑区用Robot接口写程序

```

RobotFaceDemoJy.py - C:\runtime\python\sdk\libs\RobotFace\RobotFaceDemoJy.py
File Edit Format Run Options Windows Help

import sys
sys.path.append("C:\\runtime\\python\\sdk\\libs\\CommPyHead")

import time
import CommHeadJy  头文件 和接口类导入

from RobotFaceClassJy import RobotFaceJy

g_robot = RobotFaceJy()

while True:

    cmd = raw_input("cmd:")  自定义逻辑区

    if int(cmd) == 1:
        print(g_ff.GetPoint_SerializationIFace("P0",1))
        print(g_ff.GetPoint_SerializationIFace("P3",1))
    elif int(cmd) == 2:
        print(g_ff.ReadVar_SerializationIFace("CameraU1",0,1))
        print(g_ff.ReadVar_SerializationIFace("CameraID",0,1))
    elif int(cmd) == 3:
        print(g_ff.SetVar_SerializationIFace("CameraU1",2133.22,0,1))
        print(g_ff.SetVar_SerializationIFace("CameraID",5445,0,1))
        print(g_ff.SetVar_s_SerializationIFace("CameraU1",2133.22,0,1))
        print(g_ff.SetVar_s_SerializationIFace("CameraID",5445,0,1))
    elif int(cmd) == 4:
        # list:[mod,x,y,z,a,b,c],[a1,a2,a3,a4,a5,a6]
        print(g_ff.SetPointPos_SerializationIFace("P3",[0,1.1,2.2,3.3,4.4,5.5,6.6],0,1))
        print(g_ff.SetPointPos_SerializationIFace("P0",[21.1,22.2,23.3,24.4,25.5,26.6],0,1))
        print(g_ff.SetPointPos_s_SerializationIFace("P3",[0,1.1,2.2,3.3,4.4,5.5,6.6],0,1))
        print(g_ff.SetPointPos_s_SerializationIFace("P0",[21.1,22.2,23.3,24.4,25.5,26.6],0,1))
    else:
        continue

```

### RobotFace的demo程序

2. 机器人的接口类比较多在文件RobotFaceClassJy.py文件中，跟以前的接口是一致的

```

RobotFaceClassJy.py - C:\runtime\python\sdk\libs\RobotFace\RobotFaceClassJy.py
File Edit Format Run Options Windows Help

import sys
sys.path.append("C:\\runtime\\python\\sdk\\bin\\RealCtrlJava.jar")

from com.RealRobotFace import CRobotFaceForJava
from com.RealRobotFace import vectors
from com.RealRobotFace import vectors_double
from com.RealRobotFace import vectors_int
from com.RealRobotFace import vectors_short
from com.RealRobotFace import vectors_uchar
from com.RealRobotFace import vectors_uint
from com.RealRobotFace import vectors_ushort

print("in RobotFaceClassJy")

class RobotFaceJy:
    m_RobotFace = CRobotFaceForJava()

    def __init__(self):
        return

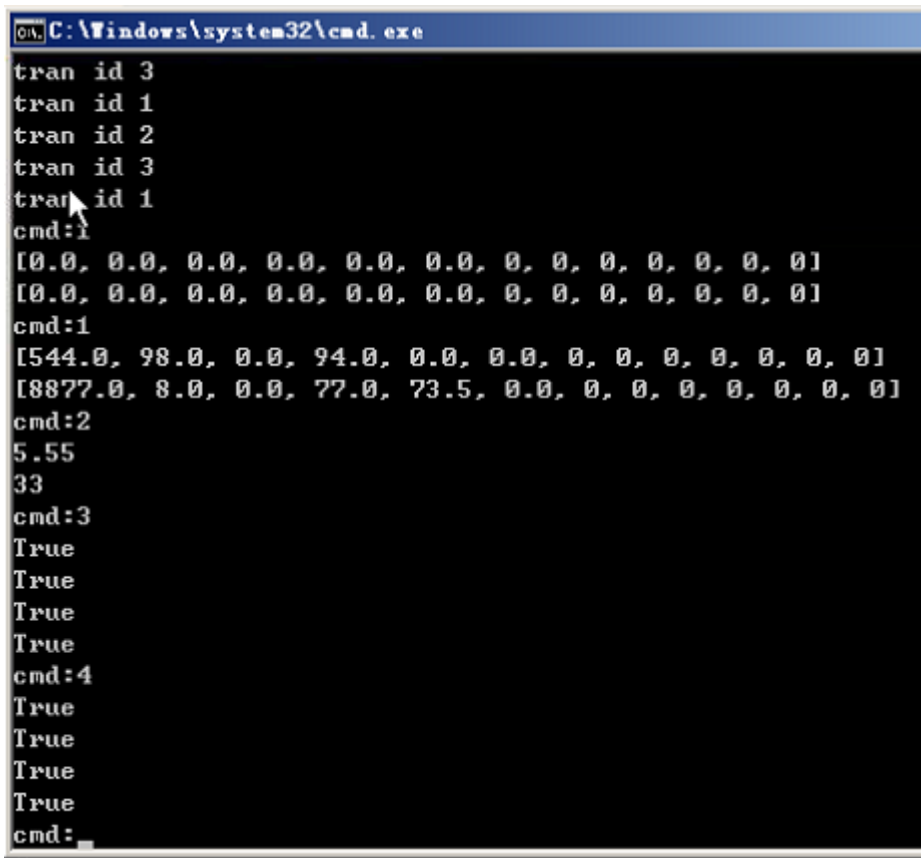
    def GetCurJPos_SerializationIFace(self):
        tt1 = self.m_RobotFace.GetCurJPos_SerializationIFace()
        res1 = []
        for n in tt1:
            res1.append(n)
        return res1

    """
    功能: 获取当前世界坐???
    参数: 无
    返回值: list:[x,y,z,a,b,c]
    """

```

## 机器人接口类

3. 运行runRobotDemoJython.bat，启动测试程序。



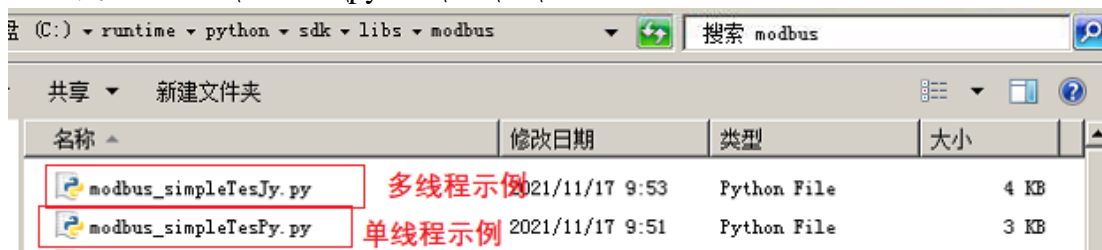
```

C:\Windows\system32\cmd.exe
tran id 3
tran id 1
tran id 2
tran id 3
tran id 1
cmd:1
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0, 0, 0, 0, 0, 0, 0]
[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0, 0, 0, 0, 0, 0, 0]
cmd:1
[544.0, 98.0, 0.0, 94.0, 0.0, 0.0, 0, 0, 0, 0, 0, 0, 0]
[8877.0, 8.0, 0.0, 77.0, 73.5, 0.0, 0, 0, 0, 0, 0, 0, 0]
cmd:2
5.55
33
cmd:3
True
True
True
True
True
cmd:4
True
True
True
True
True
cmd:
  
```

4. 脚本正常运行后，需要把runRobotDemoJython.bat加到  
c:\runtime\startSys.bat中，操作同上面章节一致，这样每次开机脚本都会  
启动

## 6.3 Java 下 Python 多线程编程示例

示例代码是在 c:\runtime\python\sdk\lib\modbus 目录下



Jython 多线程 demo

下面的代码是有两个操作

第一操作，读取机器人的速度、伺服状态、运行状态，把读取的值设置到 PLC 寄存器上。

第二操作，读取 PLC 寄存器的值，取上升沿，调用机器人的启动、停止、复位命令。

由于操作是顺序执行的，执行机器人接口的时候，PLC 的通信接口就必须等待；反之，执行 PLC 通信接口时，机器人接口就无法调用。

```
while True:
    t1 = time.time()

    #global readWritePlcError
    if g_ModbusServer.isConnected() == True:

        t1 = time.time()
        # -----过程1-----
        # 设置PLC寄存器10号、11号、12号地址为速度、伺服状态、运行状态
        # 获得速度
        CurSpeed = g_Robot.GetGlobalSpeed_SerializationIFace()
        # 获得伺服状态
        servoStr = g_Robot.GetServoSts_SerializationIFace()
        # 获得运行状态
        sysState = g_Robot.CurSysMode_SerializationIFace()
        # 设置寄存器值
        g_ModbusServer.setMultiRegs(10,3,3,[int(CurSpeed), \
                                           int(servoStr),int(sysState)])

        # -----过程2-----
        # 读PLC寄存器20号、21号、22号地址值取上升沿，分别启动、停止、复位错误
        # 读PLC寄存器
        Reg20_23 = g_ModbusServer.getMultiRegs(20, 3, 3)
        startProgramRegister = Reg20_23[0]
        stopProgramRegister = Reg20_23[1]
        resetRobotStateRegister = Reg20_23[2]

        # 取上升沿，发送启动命令
        if startProgram == 0 and startProgramRegister == 1:
            g_Robot.StartRun_SerializationIFace()
            startProgram = startProgramRegister

        # 取上升沿，发送停止命令
        if stopProgram == 0 and stopProgramRegister == 1:
            g_Robot.StopRun_SerializationIFace()
            stopProgram = stopProgramRegister

        # # 取上升沿，发送复位命令
        if resetRobotState == 0 and resetRobotStateRegister == 1:
            g_Robot.rstError_SerializationIFace()
            resetRobotState = resetRobotStateRegister

        print("gap time:",time.time()-t1,"-----")

    time.sleep(0.002)
```

### 使用 Jython 多线程修改该代码示例

第一步：把 2 个过程分解中的通信部分单独写一个线程，交互的数据使用全局变量保存，当其它线程需要 PLC 数据时从全局变量里取数据。当其它线程需要向 PLC 中设置数据时把数据放到全局变量里面，通信线程会取全局数据设置到 PLC 数据的。

```

setPLCData = [0] * 100
getPLCData = [0] * 100

# -----过程0-----
# PLC的modbus通信,全局变量setPLCData和getPLCData是交互的值
def ReadWritePLC():
    global setPLCData
    global getPLCData

    while True:
        if g_ModbusServer.isConnected() == True:
            # 一次读10个寄存器值
            g_ModbusServer.setMultiRegs(10, 10, 3, setPLCData)
            # 一次写10个寄存器值
            getPLCData = g_ModbusServer.getMultiRegs(20, 10, 3)
            time.sleep(0.002)

thread0 = threading.Thread(target=ReadWritePLC)
thread0.setDaemon(True)
thread0.start()

```

第二步，把过程 1 写成一个线程，机器人的状态会在该线程中刷新，刷新后数据放到 PLC 的写全局变量里面，由通信线程把数据设置到 PLC 上

```

# -----过程1-----
# 设置PLC寄存器10号、11号、12号地址为速度、伺服状态、运行状态
# 赋值到写PLC上去
def ReadRobotStatus():
    global CurSpeed
    global servoStr
    global sysState
    global setPLCData

    while True:
        # 获得速度
        CurSpeed = g_Robot.GetGlobalSpeed_SerializationIFace()
        # 获得伺服状态
        servoStr = g_Robot.GetServoSts_SerializationIFace()
        # 获得运行状态
        sysState = g_Robot.CurSysMode_SerializationIFace()

        setPLCData[10] = CurSpeed
        setPLCData[11] = servoStr
        setPLCData[12] = sysState
        time.sleep(0.002)

thread1 = threading.Thread(target=ReadRobotStatus)
thread1.setDaemon(True)
thread1.start()

```

第三步，把过程 2 写成一个线程，不断检测 PLC 数据的全局变量，全局变量的数据值，由通信线程刷新。该线程检测到全局变量有上述沿，会执行机器人控制命令。

```
# -----过程2-----
# 读PLC寄存器20号、21号、22号地址值取上升沿，分别启动、停止、复位错误
# 读PLC寄存器
def SetRobotCmd():
    global getPLCData
    startProgram = 0;
    stopProgram = 0;
    resetRobotState = 0;

    while True:
        startProgramRegister = getPLCData[0]
        stopProgramRegister = getPLCData[1]
        resetRobotStateRegister = getPLCData[2]

        # 取上升沿，发送启动命令
        if startProgram == 0 and startProgramRegister == 1:
            g_Robot.StartRun_SerializationIFace()
            startProgram = startProgramRegister

        # 取上升沿，发送停止命令
        if stopProgram == 0 and stopProgramRegister == 1:
            g_Robot.StopRun_SerializationIFace()
            stopProgram = stopProgramRegister

        # # 取上升沿，发送复位命令
        if resetRobotState == 0 and resetRobotStateRegister == 1:
            g_Robot.rstError_SerializationIFace()
            resetRobotState = resetRobotStateRegister
            time.sleep(0.002)

thread2 = threading.Thread(target=SetRobotCmd)
thread2.setDaemon(True)
thread2.start()
```

说明：原来的1个线程变成3个线程，因为Jython是正真的多线程并行运行，所以效率会有很大提升。综合现场反馈，效率比原来提升1倍。