Research on Industrial Robot Monitoring Technology Based on PC SDK

Xie Kunpeng
Tianjin University of Technology and Education
Institute of robotics and intelligent equipment
Tianjin, China

Liu Chaohua
Tianjin University of Technology and Education
Institute of robotics and intelligent equipment
Tianjin, China

Qi Yuming
Tianjin University of Technology and Education
Institute of robotics and intelligent equipment
Tianjin, China

Zhou Hongchao
Tianjin University of Technology and Education
Institute of robotics and intelligent equipment
Tianjin, China

Abstract— **Around the ABB Industrial robot monitoring technology problem, study elaborated the composition and the application function of PC SDK development Interface Class Library based on Net platform, compared and analyzed the advantages and disadvantages of two development environment** (virtual environment and real environment) **of the robot Monitoring system under PC SDK. Besides, programmed a ABB robot local multi-machine monitoring system using C # language in the virtual development environment. Finally, connected the virtual controller, and verified the usability of the monitoring system. Some functions of the PC SDK were also realized.**

Keywords— Industrial Robot, PC SDK, C#, Monitoring System

I. INTRODUCTION

With the rapid development of modern industry, industrial robots are utilized widely in all walks of life. Currently, many companies are still reading the relevant information from the teaching device to monitor the robot state. Asthis kind of monitoring method needs technical personnel to record, a lot of technical personnel is wasted. Thus, development of remote monitoring system robot is imperative to solve this problem; and how to acquire the robot operation information locally is the key technology. At present, the monitoring system development technology mainly adopt PLC[1,2,3],communication protocol[4,5,6,7,8] (establish the communication between the Controller and PC), or the industrial Internet of Things technology [9] through the external sensor to obtain the robot state information. Despite the easy application，strong interface and good anti-jamming ability, PLC still have its limitations such as less information can be acquired (4-8)Using communication protocols to achieve the acquisition of robot state information always needs its own design,which increased the difficulty of monitoring system development. XIE, et al first adopted the industrial Internet of things technology to get robot state information, however, this method need to use external sensors to assist, which lead to a higher design cost. Compared with the above 3 methods, the ABB brand of industrial robots might be the better candidates to realize remote monitor due to its advantages of low development difficulty without designing communication protocols, high development efficiency and cost savings of system development. At present, foreign scholars have used the PC SDK to verify the high-frequency communication algorithm[10], reducing the controller delay[11].However, the previous study did not introduce the PC SDK clearly, so this paper gives a detailed description about the function of PC SDK in the first section. Furthermore, this paper puts forward a new set of ABB robot local multi-machine monitoring system combining with the PC SDK. Lastly, the paper defines the feasibility of this new monitoring system.

II. PC SDK

The PC SDK is a two-time development interface (API) reserved by ABB Industrial robots and is available. NET language call dynamic link library, through the API can establish the robot controller and PC-side application of Ethernet communication between the PC-side robot information reading and automatic mode of data modification, program control and other functions.

A. Development environment and development mode

The PC SDK has two kinds of development methods under virtual environment and real environment, in the virtual environment, it is necessary to connect the virtual controller established by Robotstudio simulation software to develop and debug the application, and the real environment needs to connect the true controller. Compared with the two development methods, the development method in the virtual environment is more convenient, the PC SDK can realize the communication between the virtual controller and the PC application, and the real environment needs the Ethernet communication with the real controller with the PC interface

This paper is supported by
the key technologies R&D program of Tianjin(18YFZCSF00600).

Communication module installed. Regardless of the development method, the PC SDK version used in PC applications must be compatible with the Robotware version used by the controller. The two development environment configurations are shown in Figure 1:

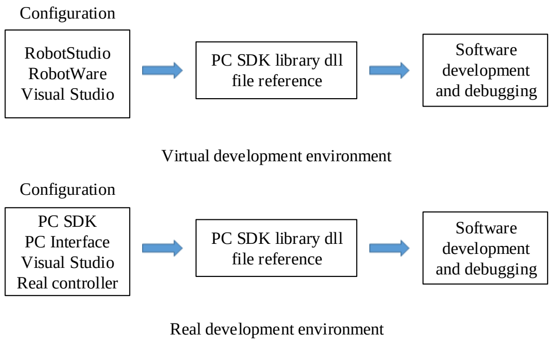

Configuration
RobotStudio
RobotWare
Visual Studio
PC SDK library dll file reference
Software development and debugging
Virtual development environment
Configuration
PC SDK
PC Interface
Visual Studio
Real controller
PC SDK library dll file reference
Software development and debugging
Real development environment

Fig. 1. Two development methods

B. *PC SDK class library composition and implementation functions*

The PC SDK class libraries are organized in the following domains:

1) Controllers

The classes used to access robot controller functionality together make up the Controller API (CAPI).

2) ConfigurationDomain

The Configuration domain namespace enables access to the configuration database of the controller. Using this domain you can read or write the value of a configuration parameter to the configuration database of the controller.

3) Discovery

To create a connection to the controller from a PC SDK application it has to make use of the Netscan functionality of the Discovery namespace. A NetworkScanner object must be created and a scan call must be performed.

4) EventLogDomain

Event log messages may contain information about controller status, RAPID execution, the running processes of the controller, and so on.

5) FileSystemDomain

The File System Domain namespace enables to create, save, load, rename, and delete files in the controller file system. It is also possible to create and delete directories.

6) Hosting

This is the main entrypoint for a host into the CAPI internal functions. It can be used to extend and alter the inner workings of the API.

7) IOSystemDomain

IO System Domain is a robot system uses input and output signals to control processes. Signals can be of digital, analog, or group signal type. Such I/O signals are accessible using the SDK.

8) Messaging

The Messaging domain of the PC SDK can be used to send and receive data between a PC SDK application and a RAPID task.

9) MotionDomain

The MotionDomain namespace lets you access the mechanical units of the robot system.

10) RapidDomain

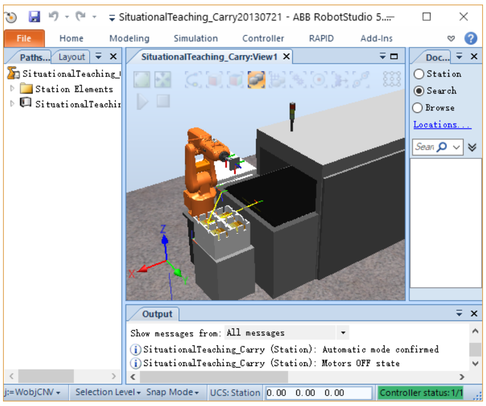
The Rapid Domain namespace enables access to RAPID data in the robot system. There are numerous PC SDK classes representing the different RAPID data types. There is also a UserDefined class used for referring to the RECORD structures in RAPID.

11) UserAuthorizationManagement

In the robot controller there is a system controlling user access: the User Authorization System(UAS). If this feature is used each user needs a user name and a password to log on to a robot controller via RobotStudio. If the controller connection for any reason is lost, you need to log on again.

III. ESTABLISH COMMUNICATION

Using the development mode of the virtual environment to develop the monitoring application, the virtual workstation of IRB120 robot is built by using Robotstudio 5.15 simulation software to simulate the real robot working scene, and the virtual controller and PC communication can be realized. The IRB120 robot virtual workstation is shown in Figure 2: The workstation is able to control the robot with the I/O signal to carry out the conveyor.



```
Show messages from: All messages
SituationalTeaching_Carry (Station): Automatic mode confirmed
SituationalTeaching_Carry (Station): Motors OFF state
0.00  0.00  0.00
```

Fig. 2. IRB120 Robot Virtual workstation

Because the virtual controller is Robotware 5.15, create a new application under the X86 platform. NET Framework2.0 Framework, add a DLL file that references PC SDK 5.15, add a namespace, and complete the communication between the virtual controller and the PC. and using the Networkscanner class to monitor the controller on the communication network to test whether the virtual controller and PC-side communication is established successfully. Figure 3:pc SDK 5.15 dll file, Figure 4:pc SDK 5.15 version of the PC development platform diagram.

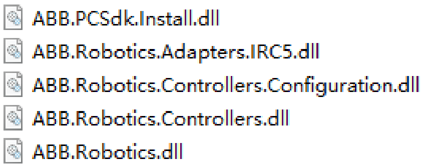


ABB.PCSdk.Install.dll
ABB.Robotics.Adapters.IRC5.dll
ABB.Robotics.Controllers.Configuration.dll
ABB.Robotics.Controllers.dll
ABB.Robotics.dll

Fig. 3. DLL file

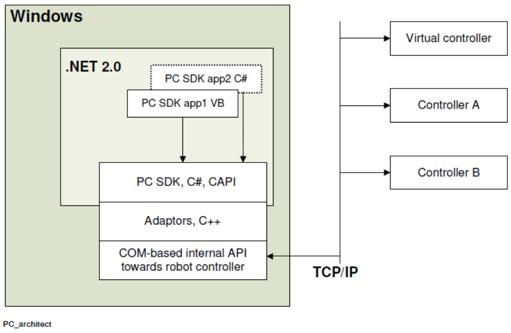


Fig. 4. PC development Platform

IV. MULTI-ROBOT MONITORING SYSTEM

Based on the PC SDK in the virtual development environment, this paper develops a local multi-machine monitoring system for ABB IRB120 models, and the initial interface of the system is shown in Figure 5:

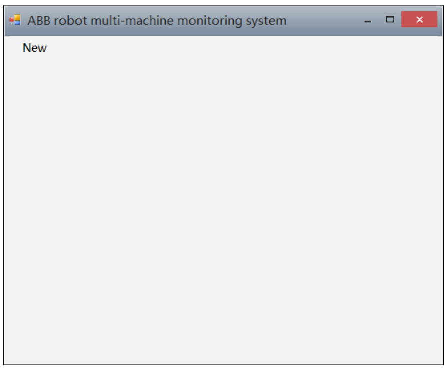
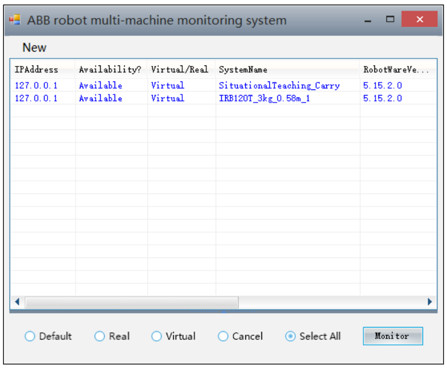


Fig. 5. Initial interface



IPAddress	Availability?	Virtual/Real	SystemName	RobotWareVe...
127.0.0.1	Available	Virtual	SituationalTeaching_Carry	5.15.2.0
127.0.0.1	Available	Virtual	IRB120T_3kg_0.58m_1	5.15.2.0

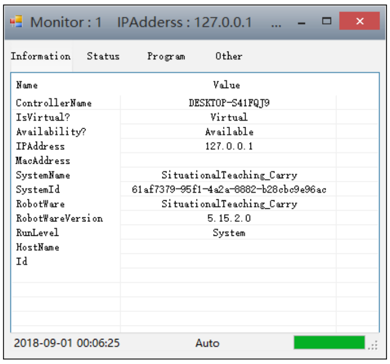
○ Default ○ Real ○ Virtual ○ Cancel ● Select All Monitor

Fig. 6. Network scan Results

The multi-machine monitoring system includes the following functions:

1) Scan all robot controller information on the same network. As shown in Figure 6:

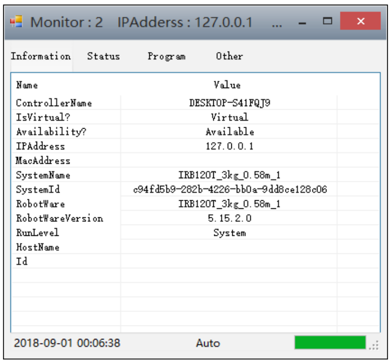
2) Select the robot to be monitored according to the controller information in the scan result and create a monitoring window.



Name	Value
ControllerName	DESKTOP-S41FQJ9
IsVirtual?	Virtual
Availability?	Available
IPAddress	127.0.0.1
MacAddress	
SystemName	SituationalTeaching_Carry
SystemId	61af7379-95f1-4a2a-8882-b28cbc9e96ac
RobotWare	SituationalTeaching_Carry
RobotWareVersion	5.15.2.0
RunLevel	System
HostName	
Id	

2018-09-01 00:06:25 Auto

Fig. 7. Monitoring window 1



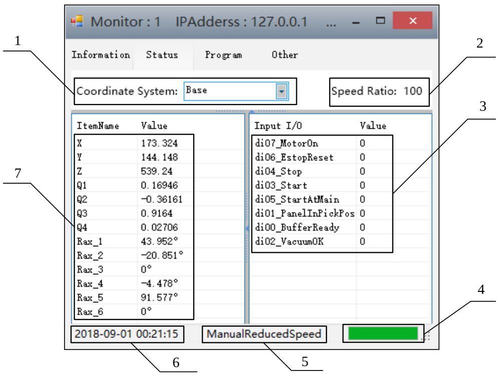
Name	Value
ControllerName	DESKTOP-S41FQJ9
IsVirtual?	Virtual
Availability?	Available
IPAddress	127.0.0.1
MacAddress	
SystemName	IRB120T_3kg_0.58m_1
SystemId	c94fd5b9-282b-4226-bb0a-9dd8ce128c06
RobotWare	IRB120T_3kg_0.58m_1
RobotWareVersion	5.15.2.0
RunLevel	System
HostName	
Id	

2018-09-01 00:06:38 Auto

Fig. 8. Monitoring window 2

3) Obtain the monitored robot information, such as IPAddress, System name, robotware version number, etc., as shown in Figure 7, 8.

4) Always get the working state of the robot, such as the coordinate value of the TCP point, the rotation angle of the six axes, the I/O input status.As shown in Figure 9:



Coordinate System: Base

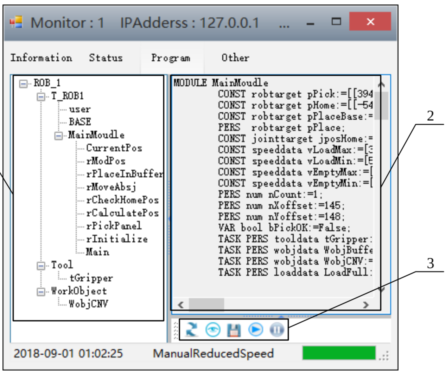
Speed Ratio: 100

ItemName	Value
X	173.324
Y	144.148
Z	539.24
Q1	0.16946
Q2	-0.36161
Q3	0.9164
Q4	0.02706
Rax_1	43.952°
Rax_2	-20.851°
Rax_3	0°
Rax_4	-4.478°
Rax_5	91.577°
Rax_6	0°

Input I/O	Value
di07_MotorOn	0
di06_EstopReset	0
di04_Stop	0
di03_Start	0
di05_StartAtMain	0
di01_PanelInPickPos	0
di00_BufferReady	0
di02_VacuumOK	0

2018-09-01 00:21:15 ManualReducedSpeed

Fig. 9. Running status information



```
ROB_1
  T_ROB1
    user
    BASE
    MainMoudle
      CurrentPos
      rModPos
      rPlaceInBuffer
      rMoveAbsj
      rCheckHomePos
      rCalculatePos
      rPickPanel
      rInitialize
      Main
  Tool
    tGripper
  WorkObject
    WobjCNV
```

```
MODULE MainMoudle
        CONST robtarget pPick:=[[394
        CONST robtarget pHome:=[[-54
        CONST robtarget pPlaceBase:=
        PERS  robtarget pPlace;
        CONST jointtarget jposHome:=
        CONST speeddata vLoadMax:=[3
        CONST speeddata vLoadMin:=[5
        CONST speeddata vEmptyMax:=[
        CONST speeddata vEmptyMin:=[
        PERS num nCount:=1;
        PERS num nXoffset:=145;
        PERS num nYoffset:=148;
        VAR bool bPickOK:=False;
        TASK PERS tooldata tGripper:
        TASK PERS wobjdata WobjBuffe
        TASK PERS wobjdata WobjCNV:=
        TASK PERS loaddata LoadFull:
```

2018-09-01 01:02:25 ManualReducedSpeed

Fig. 10. Program page

The function in Figure 9 is described as:

1: Switching between the world coordinate system, the base calibration system, the workpiece coordinate system, the tool coordinate system, and the custom coordinate system.

2: The running speed ratio of the robot.

3: Input signal status value of the robot workstation.

4: Connection status of the robot controller.

5: The operating mode of the robot.

6: Robot system time.

7: X, Y, Z are the coordinate values of TCP in different coordinate systems, Q1, Q2, Q3, Q4 are the direction quantities of the robot in different coordinate systems, and Rax_1, Rax_2, Rax_3, Rax_4, Rax_5, Rax_6 are robots 1 to 6-axis joint rotation angle value.

5) always acquire the mechanical unit, work task and routine of the current operating system of the robot, and view the text content of the routine program in the form of text, and can control the start and stop of the routine program. As shown in Figure 10:

The features in Figure 10 are:

1: Display all mechanical units, Task, Module, Routine, Tool, workobject information in the controller.

2: Show module text content.

3: Some operations on the routine. among them:

⟳ :Update the information in 1.

◉ :View routine text content.

💾 :Save the module file from the controller to the local computer.

▶ :In automatic mode, start the routine program that is selected in 1.

⏸ :Stop the 1 current routine program in automatic mode.

V. MONITORING SYSTEM VERIFICATION

The experimental verification of the multi-machine monitoring system was carried out using the established IRB120 robot virtual workstation (shown in Figure 5) in Section 3. The current pose of the robot in the virtual workstation is shown in Figure 11:

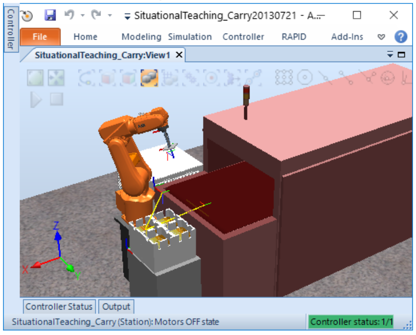
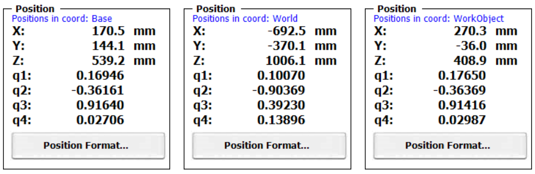

SituationalTeaching_Carry20130721 - A...
File
Home
Modeling
Simulation
Controller
RAPID
Add-Ins
Controller
SituationalTeaching_Carry:View1
Controller Status
Output
SituationalTeaching_Carry (Station): Motors OFF state
Controller status: 1/1

Fig. 11. Pose

The current TCP coordinate value and the direction vector value result of the robot, as shown in Figure 12 and 13:



Position
Positions in coord: Base

X:	170.5	mm
Y:	144.1	mm
Z:	539.2	mm
q1:	0.16946	
q2:	-0.36161	
q3:	0.91640	
q4:	0.02706	

Position Format...

coord:Base

Position
Positions in coord: World

X:	-692.5	mm
Y:	-370.1	mm
Z:	1006.1	mm
q1:	0.10070	
q2:	-0.90369	
q3:	0.39230	
q4:	0.13896	

Position Format...

coord:Word

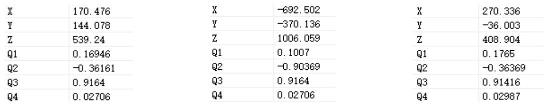
Position
Positions in coord: WorkObject

X:	270.3	mm
Y:	-36.0	mm
Z:	408.9	mm
q1:	0.17650	
q2:	-0.36369	
q3:	0.91416	
q4:	0.02987	

Position Format...

coord:WorkObject

Fig. 12. TCP point coordinates and direction measures in the controller



X	170.476
Y	144.078
Z	539.24
Q1	0.16946
Q2	−0.36161
Q3	0.9164
Q4	0.02706

coord:Base

X	−692.502
Y	−370.136
Z	1006.059
Q1	0.1007
Q2	−0.90369
Q3	0.9164
Q4	0.02706

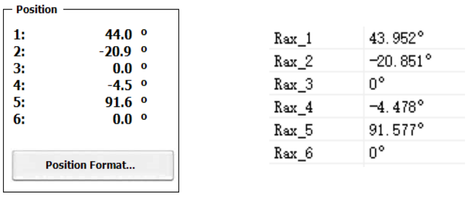
coord:Word

X	270.336
Y	−36.003
Z	408.904
Q1	0.1765
Q2	−0.36369
Q3	0.91416
Q4	0.02987

coord:WorkObject

Fig. 13. TCP point coordinates and direction magnitude in the monitoring system

The current rotation angle value of the robot's 6 joints, as shown in Figure 14:



Position

1:	44.0	o
2:	-20.9	o
3:	0.0	o
4:	-4.5	o
5:	91.6	o
6:	0.0	o

Position Format...

Controller

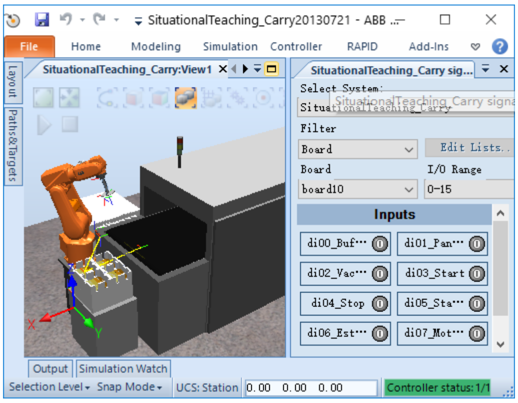
Rax_1	43.952°
Rax_2	−20.851°
Rax_3	0°
Rax_4	−4.478°
Rax_5	91.577°
Rax_6	0°

Monitoring System

Fig. 14. Figure 14 1 to 6 axis joint rotation angle value

Query the current I/O system input signal value of the robot in the RoubotStudio simulation I/O module, as shown in Figure 15:

The multi-machine monitoring system monitors the current I/O input signal in the robot status information as shown in Figure 16:



Select System: SituationalTeaching_Carry

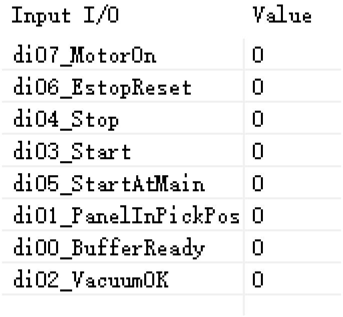
Filter: Board

Board: board10 I/O Range: 0-15

di00_Buf… 0	di01_Pan… 0
di02_Vac… 0	di03_Start 0
di04_Stop 0	di05_Sta… 0
di06_Est… 0	di07_Mot… 0

UCS: Station 0.00 0.00 0.00 Controller status: 1/1

Fig. 15. Control system signal

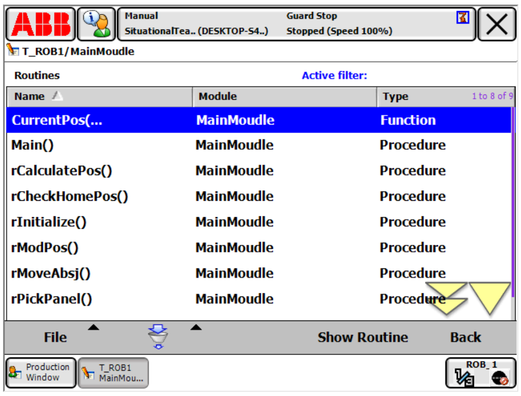


Input I/O	Value
di07_MotorOn	0
di06_EstopReset	0
di04_Stop	0
di03_Start	0
di05_StartAtMain	0
di01_PanelInPickPos	0
di00_BufferReady	0
di02_VacuumOK	0

Fig. 16. Monitoring system signal

The robot routine is queried in the controller, and the result is shown in Figure 17:

Multi-machine Monitoring System Monitor the current robot routine information as shown in Figure 18:



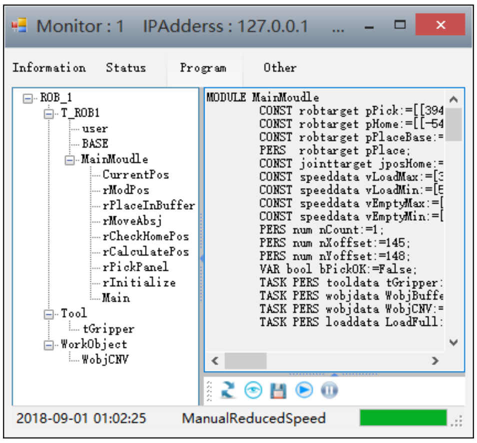
Manual SituationalTea.. (DESKTOP-S4..) Guard Stop Stopped (Speed 100%)

T_ROB1/MainMoudle

Routines Active filter:

Name	Module	Type 1 to 8 of 9
CurrentPos(...	MainMoudle	Function
Main()	MainMoudle	Procedure
rCalculatePos()	MainMoudle	Procedure
rCheckHomePos()	MainMoudle	Procedure
rInitialize()	MainMoudle	Procedure
rModPos()	MainMoudle	Procedure
rMoveAbsj()	MainMoudle	Procedure
rPickPanel()	MainMoudle	Procedure

Fig. 17. Routine in the controller



```
ROB_1
  T_ROB1
    user
    BASE
    MainMoudle
      CurrentPos
      rModPos
      rPlaceInBuffer
      rMoveAbsj
      rCheckHomePos
      rCalculatePos
      rPickPanel
      rInitialize
      Main
  Tool
    tGripper
  WorkObject
    WobjCNV
```

```
MODULE MainMoudle
        CONST robtarget pPick:=[[394
        CONST robtarget pHome:=[[-54
        CONST robtarget pPlaceBase:=
        PERS  robtarget pPlace;
        CONST jointtarget jposHome:=
        CONST speeddata vLoadMax:=[3
        CONST speeddata vLoadMin:=[5
        CONST speeddata vEmptyMax:=[
        CONST speeddata vEmptyMin:=[
        PERS num nCount:=1;
        PERS num nXoffset:=145;
        PERS num nYoffset:=148;
        VAR bool bPickOK:=False;
        TASK PERS tooldata tGripper:
        TASK PERS wobjdata WobjBuffe
        TASK PERS wobjdata WobjCNV:=
        TASK PERS loaddata LoadFull:
```

2018-09-01 01:02:25 ManualReducedSpeed

Fig. 18. Program monitoring results

VI. CONCLUSIONS

The practicability of the PC SDK secondary development interface reserved by control system of ABB IRB120 robot based on the .Net platform was confirmed in this study. And a set of local multi-machine monitoring system for IRB120 industrial robot is designed.This system realized the local monitoring of IRB120 robot, and lays a practical foundation for the development of ABB Industrial robot remote Monitoring system in the future.

REFERENCES

[1] Wang Shengfa, Pan Min. Application of Communication between PLC and PC under VB in Industrial Robot Monitoring System[J]. Mechanical & Electrical Engineering, 2001(06): 35-37.

[2] Wang Shengfa, Zhang Hongbing. Application of PLC Communication under VB in Industrial Robot Monitoring System[J]. Gansu Science and Technology, 2002(Z1): 55-56.

[3] Wang Yudong. Design of OPC-based industrial robot monitoring system and its arm fatigue life prediction [D]. Shanghai University of Engineering and Technology, 2016.

[4] Liu Lei. Design and development of industrial robot remote monitoring and diagnosis service system [D]. Dalian University of Technology, 2014.

[5] Chen Jie. Based on HRT-5 industrial robot network monitoring software development [D]. Wuhan Textile University, 2016.

[6] Pan Xiangfei. Research on industrial robot monitoring system based on Socket communication [D]. Zhejiang University of Technology, 2017.

[7] Lü Hantai. Research on Web3D Monitoring System for Industrial Robots [D]. Zhejiang University of Technology, 2017.

[8] Deng Sihao. Research and implementation of real-time robot monitoring and communication system [D]. Wuhan University of Technology, 2003.

[9] Wen Xiao. Research on robot testing system and remote monitoring platform based on industrial internet of things [D]. Anhui University of Engineering, 2016.

[10] Dalvand M M, Nahavandi S. Teleoperation of ABB industrial robots[J]. Industrial Robot-an International Journal, 2014, 41(3): 286-295.

[11] Dalvand M M, Nahavandi S. Improvements in teleoperation of industrial robots without low-level access[C]. systems, man and cybernetics, 2014: 2170-2175.