COMPARISON BETWEEN TWO METHODS OF ROBOT CONTROL IN THE PRODUCTION OF PRISMATIC PARTS

Csokmai Lehel Szabolcs
1
1 University of Oradea, lcsokmai@uoradea.ro

ABSTRACT: In this paper two robot programming methods will be compared. The two methods are the PC SDK which uses ABB .NET libraries to communicate with the IRC5 robot controller, the second method will use the PC Interface option of the IRC5 robot controller.
KEY WORDS: ABB, IRC5, IRB 1600, C#, RAPID, LAN, socket, network, internet, communication.

1. INTRODUCTION

The system contains two IRB 1600 robots and one IRC5 controller from ABB.

The IRC5 is ABB’s fifth generation robot controller. Its motion control technology, TrueMove & QuickMove, is key to the robot’s performance in terms of accuracy, speed, cycle-time, programmability and synchronization with external devices.

RobotStudio, is a powerful PC tool for working with IRC5 data on-line as well as off-line. In off-line mode, RobotStudio provides a perfect digital copy of the robot system together with strong programming and simulation features.

The IRC5 key features are:

• Safety, operator safety is a central quality of the IRC5, fulfilling all relevant regulations with good measure, as certified by third party inspections. Electronic position switches and SafeMove represent a new generation of safety, enabling more flexible cell safety concepts, e.g. involving collaboration between robot and operator;
• RAPID programming language provides the perfect combination of simplicity, flexibility and power. It is a truly unlimited language with support for structured programs, shop floor language and advanced features. It also incorporates powerful support for many process applications;
• Communication, the IRC5 supports the state-of-the-art field busses for I/O and is a well-behaved node in any plant network. Sensor interfaces, remote disk access and socket messaging are examples of the many powerful networking features;
• RobotStudio, is a powerful PC tool for working with IRC5 data on-line as well as off-line. In off-line mode, RobotStudio provides a perfect digital copy of the robot system together with strong programming and simulation features.

The IRB 1600 key features are:

• Reliable - high production up time, low maintenance requirements, and short repair times
• Fast - short cycle times, faster than any other robot in its class.
• Accurate - consistent parts quality, outstanding position repeatability - ± 0.05mm.
• Robust - protected for harsh production environment, IP 67 classified
• Versatile - flexible integration and production, mounting options - wall, floor, inverted or tilted

In this paper two robot programming methods will be compared. The two methods are the PC SDK which uses ABB .NET libraries to communicate with the IRC5 robot controller, the second method will use the PC Interface option of the IRC5 robot controller.
2. PC SDK

ABB’s PC Software Development Kit (PC SDK) is a software tool, which enables programmers to develop customized operator interfaces for the IRC5 robot controller.

The application is divided into two parts. One is written in the C# programming language using the .NET libraries and the second part is a RAPID program which is running on the IRC5 robot controller.

The remote client written in C# do not have all the privileges of a local client. For example, both PC and RAPID applications can reset the program pointer and start the execution, but for a PC SDK application to do this there are certain restrictions. Mastership of the RAPID domain must be requested explicitly by the PC application and the IRC5 controller has to be in automatic operating mode.

An advantage of a remote client, on the other hand, is the possibility to monitor and access several robot controllers from one location. As for large applications the PC platform is also less limited than the IRC5 robot controller as regards memory resources and process power.

A minimum response time for a real controller should be expected to be in the order of 10-100 milliseconds, meaning that hard real time demands cannot be met on any platform.

The PC SDK offer controller functionality through the public application interface called Controller API. The interface can be seen as a specification of the controller services available. The PC SDK class libraries are organized in the following domains:

- Controllers
- ConfigurationDomain
- Discovery
- EventLogDomain
- FileSystemDomain
- Hosting
- IOSystemDomain
- Messaging
- MotionDomain
- RapidDomain
- UserAuthorizationManagement

The classes used to access robot controller functionality together make up the Controller API (CAPI). The following illustration shows a part of the CAPI object model:

For the comparison we made a virtual controller in the RobotStudio application.
The following application has been made in RAPID programming language for the IRC5 robot controller.

```rapid
MODULE prog_01
VAR robtarget p50 := [ [740, -325, 504], [0.707170, 0, 0.707170, 0], [0, 0, 0, 0], [9E9, 9E9, 9E9, 9E9, 9E9, 9E9] ];
VAR bool flag_exec := FALSE;
VAR bool flag_stop := FALSE;
proc main_01()
WHILE flag_stop=FALSE DO
IF flag_exec=TRUE THEN
  MoveJ p50, v200, fine, tPen;
  flag_exec:=FALSE;
ENDIF
ENDWHILE
ENDPROC
ENDMODULE

A default position is defined for the IRB 1600 robot where, after the RAPID program starts the robot will move. After the RAPID program is started the PC application is started, then the application logs in the IRC5 robot controller.

After the login is accepted we can set the target coordinates then the variable flag_exec to TRUE. When the robot reaches the coordinates set by the PC application the flag_exec will be FALSE

This cycle can be executed for any number of coordinates, when we want to stop the RAPID application the flag_stop must be TRUE;

3. PC INTERFACE

PC Interface is used for communication between the controller and a PC connected to an Ethernet network, and is required for some software products from ABB, such as WebWare and some functionality in RobotStudio.

With PC Interface, data can be sent to a PC. This is, for example, used for:
- backup
- production statistics logging
- operator information presented on a PC.

The major advantage of the PC Interface is, the connection can be made from any platform. The client application can be made in any programming language which can use the TCP/IP protocols.

With the help of the PC Interface any RAPID program can utilize the Socket Messaging. The purpose of Socket Messaging is to allow a RAPID programmer to transmit application data between computers, using the TCP/IP network protocol.

A socket represents a general communication channel, independent of the network protocol being used. Socket communication is a standard that has its origin in Berkeley Software Distribution Unix. Besides Unix, it is supported by, for example, Microsoft Windows. With Socket Messaging, a RAPID program on a robot controller can, for example, communicate with a C/C++ program on another computer.

Code example for client, contacting server with IP address 192.168.0.2:

```rapid
VAR socketdev socket1;
VAR string received_string;
PROC main()
SocketCreate socket1;
SocketConnect socket1, "192.168.0.2", 1025;
SocketSend socket1 \\Str:="Hello server";
SocketReceive socket1 \\Str:=received_string;
TPWrite "Server wrote - " + received_string;
received_string := "";
! Continue sending and receiving...
! Shutdown the connection
SocketSend socket1 \\Str:="Shutdown connection";
SocketReceive socket1 \\Str:=received_string;
SocketWrite "Server wrote - " + received_string;
SocketClose socket1;
ENDPROC
```

Figure 7. Illustration of socket communication

Code example for server (with IP address 192.168.0.2):

```rapid
VAR socketdev temp_socket;
VAR socketdev client_socket;
VAR string received_string;
VAR bool keep_listening := TRUE;
PROC main()
SocketCreate temp_socket;
SocketBind temp_socket, "192.168.0.2", 1025;
SocketListen temp_socket;
WHILE keep_listening DO
  SocketAccept temp_socket, client_socket;
  SocketReceive client_socket \\Str:=received_string;
  TPWrite "Client wrote - " + received_string;
  received_string := "";
  SocketSend client_socket \\Str:="Message acknowledged";
  SocketReceive client_socket \\Str:=received_string;
  TPWrite "Client wrote - " + received_string;
  SocketSend client_socket \\Str:="Shutdown acknowledged";
  SocketClose client_socket;
ENDWHILE
SocketClose temp_socket;
ENDPROC
```

The client-server application which connects to the IRC5 controller is made with the help of the RAD Studio IDE.

Code example for client-server PC application:

```
try
  with tcp_client_01 do
    SocketCreate socket1;
    SocketConnect socket1, "192.168.0.2", 1025;
    SocketSend socket1 \\Str:="Hello server";
    SocketReceive socket1 \\Str:=received_string;
    TPWrite "Server wrote - " + received_string;
    received_string := "";
    ! Continue sending and receiving...
    ! Shutdown the connection
    SocketSend socket1 \\Str:="Shutdown connection";
    SocketReceive socket1 \\Str:=received_string;
    SocketWrite "Server wrote - " + received_string;
    SocketClose socket1;
  end;
end;
```
begin
if Connected then Disconnect;
Host := '192.168.125.1';
Port := 65502;
Connect;
IOHandler.WriteLn('R2_coord');
mem0_lines.Add(IOHandler.ReadLn());
IOHandler.WriteLn('847.92,9.45,749.22');
mem0_lines.Add(IOHandler.ReadLn());
IOHandler.WriteLn('0.701172,0.0135639,0.712714,0.014585');
mem0_lines.Add(IOHandler.ReadLn());
IOHandler.WriteLn('0, -1,0,1');
mem0_lines.Add(IOHandler.ReadLn());
disconnect;
end;
except
on E : Exception do
begin
ShowMessage(E.Message);
end;
end;
The coordinates for the IRB 1600 robot has to be sent in three parts because the IRC5 robot controller cannot accept strings with more than 60 characters.

Figure 8. PC application for socket messaging
An example for a socket server written in the LabVIEW programming environment.

Figure 9. Simple Messaging Reference Library (STM) in LabVIEW
Code example for server written in PHP:

```php
$socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
socket_bind($socket, $address, $port);
socket_listen($socket);
echo "Waiting for a connection\n";
$conn = false;
switch(@socket_select($r = array($socket), $w = array($socket), $e = array($socket), 60)) {
case 2:
echo "Connection refused\n";
break;
case 1:
echo "Connection accepted\n";
$conn = @socket_accept($socket);
bson:
break;
case 0:
echo "Connection timed out\n";
bson:
break;
}
if ($conn !== false) {
}
```

4. CONCLUSIONS
We demonstrated above that the use of the PC Interface functions to control the IRB 1600 robots, are more versatile than using the PC SDK. The PC SDK’s most restrictive part is the need to use the .NET libraries. The .NET libraries can be used only on Windows platforms and with only few programming languages while with PC Interface almost any language can communicate with the robots.

5. ACKNOWLEDGEMENTS
This work was partially supported by the strategic grant POSDRU/107/1.5/S/80272, inside POSDRU Romania 2007-2013 co-financed by the European Social Fund - Investing in People.

6. REFERENCES