ABB robot data collection based on dynamic link library

Zhiyong Wu1, Hongping Wang*,1, Yiwen Zhang1, Minglu Ai1
1Mechanical Engineering, Changchun University of Science and Technology, Changchun, Jilin, 130022, China
* Corresponding author: 1wanghongping@cust.edu.cn

ABSTRACT: At present, with the development of the times and technology, in order to meet the needs of industrial production, robots have begun to be widely used in factories. Although the production efficiency of the robot is very high and relatively cheap compared to the labor cost, when an industrial robot fails, it will have a serious impact on the production capacity of the entire production line. Therefore, it is necessary to monitor various data of the robot. If some abnormal data can be found, it can perform effective predictive maintenance on the robot. This article mainly proposes a method for various data acquisition of ABB robots. The data acquisition interface is developed using C# and the dynamic link library is called. The data of ABB robots can be read in real time. It can realize real-time monitoring of each data of the robot.

1. Introduction
At present, the world has entered the era of Industry 4.0. In order to improve production efficiency, industry and manufacturing have begun to use robots to complete repetitive and even dangerous production operations in colleges and universities instead of manual labor.[1] Although robots can complete many tasks that replace humans, if the robot fails, it will not only lead to production stagnation, affect corporate efficiency, and even threaten the recognized life safety. Therefore, robot maintenance, life prediction, and fault diagnosis become more and more important. The ABB robot has been developed in C# and completed the data collected.[2]

At present, C# and C++ mixed programming is a relatively common method mixed programming. C# is a programming language customized by Microsoft for the Visual Studio.net framework. C# has the same powerful functions as C/C++, and the syntax of C# is simpler than C/C++, making C# one of the most popular development languages. But C# also has some shortcomings.[3] It can only be used for the bottom layer by calling Win32 dynamic link library (DLL, dynamic link library) or dynamic link library written in C++, operate. Moreover, it is widely believed that the confidentiality of C# is weaker than that of C/C++, which can easily cause its source code to be decompiled and cracked. Therefore, it is currently popular to adopt mixed programming to increase the confidentiality of C#. We can encrypt the DLL file with a certain algorithm or compress it and add it to the C# program.[4]

2. Principle and characteristics of DLL
The dynamic link library is a modular program based on Windows. The DLL file not only contains executable programs, data, and resources required for various execution programs. In the interface development program design, the DLL file can be used to integrate the entire system. The program is decomposed into multiple main programs and DLLs, thereby further reducing workload during development. Due to the reduction of program reuse modules, the access speed has been greatly
improved, and if it is necessary to modify a certain part of the underlying program, only the program in the corresponding DLL file can be modified, and the upper-level program does not need to be modified.

In the development process of the data acquisition system, the dynamic link library technology is applied to the functional modules of the system, and each functional module of the system module is made into DLL files, which simplifies the software development and management, and can save more memory space.

3. Dynamic link library

3.1. Import dynamic link library
First, you need to download and install Robot Studio and PCSDK on the ABB official website. ABB’s software development kit PCSDK is a PC software tool that enables programmers to develop customized operator interfaces for the robot’s controller and use the network provided by the SDK. Communication, establish Socket communication (TCP/IP protocol) with Robot Studio. After the installation is complete, there will be three dynamic link libraries referenced by Visual Studio with the suffix dll in the path.

It is relatively simple to use C# to call the managed dynamic link library. You only need to right-click the "reference" in the "solution" and select "add reference", select the listed dynamic link library DLL file, or browse to add the dynamic link library.[5]

![Figure 1. Add dynamic link library](image1)

![Figure 2. Import namespace](image2)

After adding the dynamic link library, use using to import the namespace you want to use, and you can call the functions in the managed dynamic link library just like using your own class.

3.2 The content of the dynamic link library
The classes used to access the functions of the robot controller constitute the controller API.

ABB.Robotics.Controllers.Discover: Scan the domains required by the controller method.

ABB.Robotics.Controllers.RapidDomain: The domain required to access and modify the Rapid program.

ABB.Robotics.Controllers.MotionDomain: The domain required to obtain the robot pose.

The ConfigurationDomain control contains classes for handling the configuration of the robot controller. This namespace allows access to the configuration database of the controller. Using this namespace, the value of the configuration parameter can be read or written to the designated database of the controller.[6] When logging in to the control It can be read-only or write-accessed. Read-only is the default access permission. In order to obtain write access, the client needs to request the master control of the controller resource it wants to operate.
ConfigurationDatabase is to access the configuration database, allowing access to the configuration database to read and write the configuration type and properties of the robot controller.

Reading and writing configuration parameters can be read from the configuration database of the controller. The figure shows how to read the unitmap value of signal TestDI, and how to read the cal offset value of rob11 (moc.cfg), and set signal TestDI The unitmap is set to 17.

```
using ABB.Robotics;
using ABB.Robotics Controllers.ConfigurationDomain;
...
Try
{
    Controller controller = new Controller();
    Mastership master;
    ConfigurationDatabase cfg = controller.Configuration;
    using (master = Mastership.Request(controller.Configuration))
    {
        string[] path = { "EIO", "EIO_SIGNAL", "TestDI", "UnitMap" }; 
        cfg.Write("17", path);
    }
}
```

Figure 3. Read and write configuration parameters

Use DomainCollection to get a list of all domains available in the configuration database. The ConfiguratioDomain.Attribute object contains the description of the configuration domain attribute and cannot be used to read or write the value of the attribute. To read and write the value of the attribute, you should use the GetAttribute() and SetAttribute() methods respectively.

The ConfigurationDomain.Instance object defines an instance of a specific type. This object is a set of disconnected data sets from the configuration database, as shown in the figure below, to create an instance object, use the SetAttribute() and GetAttribute() methods. Create an instance, and set the attribute entry to static, use GetAttribute() to get the same attribute.

When a network scan is executed, a collection of ControllerInfo objects will be returned. This class contains "simple" information about the controller and does not need to be connected to it. Note that when the controller state changes, the ControllerInfo object will not be updated. If you need to know whether the controller is available again, you need to execute a new Scan the network or use an existing controller object and check the status directly.

4. Acquisition system development of Data
First click on the toolbox on the design interface, find the Button-button in the common controls of the toolbox, drag the button into the design interface, and adjust the size to a reasonable size, click on the Button control, press F4 to call up the properties of the Button, and change the text to "Scan the controller", click the event, select and double-click the "Click" option, you can write the program to scan the controller. Select the public control TextBox and drag it into the Form design interface, which is mainly used to receive controller scan information. To edit the Click event of the Button control, you need to first create a network scan object, use the Scan namespace to scan the robot, load all controllers into the memory, and read the end of the scan The information in the returned ControllerInfo object collection is displayed in TextBox1, as shown in Figure.
Also add a button control to the Form design interface, and set the text of button to "connect to the controller". When the controller is scanned, the controller will be loaded into the memory. Click the button to connect to the controller. Drag the textbox again to record the connection, and add a Click event to the button. When writing this event, you must first determine whether the controller is successful. If the controller is loaded, first log off the current user, release or reset the application-defined tasks related to unmanaged resources, so as to achieve the effect of releasing the previous resources. After the resource is released, the controller instance object needs to be created first, and the controller.Logon() method is used to specify the user identity to log in. After the login is successful, the login information will be displayed in the Text Box. The event program code is shown in the figure.

```csharp
private void Button1_Click(object sender, EventArgs e)
{
    this.Scanner = new NetworkScanner(); //Create a new network scanner instance.
    this.Scanner.Scan(); //Start a scan to load all controllers into memory.
    ControllerInfoCollection controllers = this.Scanner.Controllers; //Get the collection of found controllers
    foreach (ControllerInfo info in controllers)
    {
        this.TextBox1.Text += info.ControllerName + "\n"
        this.TextBox1.Text += info.IPAddress + "\n"
        this.TextBox1.Text += info.Version + "\n"
        Controller_Info = info;
    }
}
```

Figure 4. Scan Controller Button

After the start command code is written, add the Timer control. The control can execute the code at regular intervals. According to the actual situation, when setting the Timer control, in the Timer control Tick event, first determine whether the controller is connected. When the controller is connected, "Robot is connected" is displayed, the control mode of the robot is obtained through the GetOperation_mode() method, and the control state is obtained through the GetController_states() method, and displayed. Extract the robot motion data and add 6 TextBoxes, which are mainly used to obtain the current posture of the robot. First, you need to create MotionSystem and JointTarget objects, and call MidpointRounding.AwayFromZero to display the rotation angle of each axis.

After completing the page design and programming, you can run the data acquisition system and read the data of the robot in real time, as shown in the figure for the interface of the robot acquisition system.
Complete the path programming of the ABB robot through the teach pendant of the control cabinet. As shown in the figure, program the robot teach pendant. Set the written program with the Test name.

Connect the PLC of the robot control cabinet and the main box through a network cable, as shown in the figure. Then open the data acquisition system developed in C#, as shown in the figure, you can collect robot data.
Figure 9. Connect to the host to collect robot data

5. Conclusion
The system can connect to the robot in real time, read the relevant data of the robot, monitor various data of the robot, and achieve the effect of predicting the failure of the robot for maintenance through the monitoring, so as to avoid unnecessary economic losses and personal injuries.

References
[1]. Nathan Wallace with, learning to use Visual C++6.0 development of Active X Template Library, University of Electronic Science and Technology Press, 2000.
[2]. Stephen D. Gilbert was, and I learn Visual C++6.0, Machinery Industry Press, 1999.
[3]. Tang Tongyi, Electrical Measurement & Instrumentation Technology in Retrospect and Prospect, Electrical Measurement and Instrumentation, VOLUME 37, No. 1 2000.
[4]. Wang Xiaofeng Li Yunfei Yupeng edited, Visual C++ practice and improve, China Railway Press, 2001.
[5]. Cheng Zhiguang, "Electromagnetic Field Analysis and Verification of Power Transformers", Ph.D. Thesis, Tsinghua University, 1994.
[6]. Zhou Jianming, finite element integrated simulation method of electromagnetic field and research on leakage magnetic field of large transformer, PhD thesis of Huazhong University of Science and Technology, 1990.