

Software development environment design in Robot device operation based on IEC61131-3

Soon-Bum kwon*, Hee-Yeon Cho**, Dong-Ryeol Shin***

School of Information and Communication Engineering Sungkyunkwan University, Suwon, Korea

{period99, perfume772}@skku.edu, drshin@ece.skku.ac.kr

Abstract — The PLC language which IEC organization standardized as IEC 61131-3 standard is used in various fields of industry such as robot control system and industry automation system. However, it can't provide suitable development environments for developers who don't have enough knowledge about robot devices. Besides, it is difficult to support branch code, real-time operation and to interpret without extensive device knowledge.

For these reasons, this paper provides useful development architecture for easy development and well-defined interfaces. Especially, we more focus on block diagram model in order for application developers to easily construct their applications. Our architecture model support development environment based on block diagram model. It can be possible to provide robot developer with reliable development environments.

Keywords— PLC, Robot, IEC61131-3

I. INTRODUCTION

A Programmable logic controller (PLC) [1] is a digital computer which is used to automate electromechanical processes, such as control of machinery on factory assembly lines, airline, railway automation systems and power plant real-time robot operating systems. PLCs are used in many industries and machines. Currently, PLC programs are commonly written in the personal computer and it is downloaded to special machine using a direct connection cable or over a network to the PLC.

The initial model of PLC programming languages was incompatible. PLC programming model usually depend on special purpose, country's policy, mechanics type or domain features. Because of above mentioned reasons, it makes software developer confusing. Therefore, IEC 61131-3(PLC open) [2] is standardized by International Electrotechnical Commission(IEC).

IEC 61131-3 specifies the syntax and semantics of a unified suite of programming languages for programmable controllers (PCs). These consist of two textual languages, IL (Instruction List) and ST (Structured Text), and two graphical languages, LD (Ladder Diagram) and FBD (Function Block Diagram). Sequential Function Chart (SFC) elements are defined for structuring the internal organization of programmable controller programs and function blocks. Also, configuration elements are defined which support the installation of programmable controller programs into programmable controller systems.

Most widely used language is Ladder Diagram (LD). LD which was invented to replace hardwired relay control systems is a mainstay in the U.S, Korea, Japan today, used in probably 95 percent of all applications. The LD is familiar with the electric circuit format, even a non-programmer with an electrical background can follow the program for purposes of troubleshooting a problem. It's also easy to start writing a program in Ladder Diagram. However functions such as PID, trigonometry, data analysis are commonly required in many control applications, but difficult to implement. Another problem is that as program size grows, the ladder can become very difficult to read and interpret, unless it's extensively documented. For such reasons, LD is not suitable for robot systems configured by embedded devices.

The aim of this paper is to propose block diagram based working model in order for application developers to easily use, understand robot programming and to introduce a graphical design tool for PLC software development. The software tool was developed using Java with an additional graphical library.

The paper is constructed as follows: Section 2 will concentrate on graphical PLC programming platform, Section 3 will suggest software architecture for graphical PLC programming and Section 4 gives conclusions on the development way hereafter..

II. RELATED WORK

There are several trials to provide robot programming development environments. In this section, we briefly summarize current robot programming development tools or interfaces. From those robot programming environments, we describe some advantages and disadvantages and our future directions.

A. BECKHOFF TWINCAT

TWINCAT provides visual development tools and various field buses(Profibus, Interbus, CANOpen, DeviceNet, Ethernet, EtherCAT). User directly accesses the connected device through these field buses. This way allows simple input/output operation without device control code and provides user with convenience.

TWINCAT incorporating with Microsoft operating systems allows the user to access PC resource through the well-defined interfaces of the operating system. It consists of run-time systems that execute control programs in real-time and the development environments for programming, diagnostics and

configuration. During the runtime, it is possible to add function in real time.

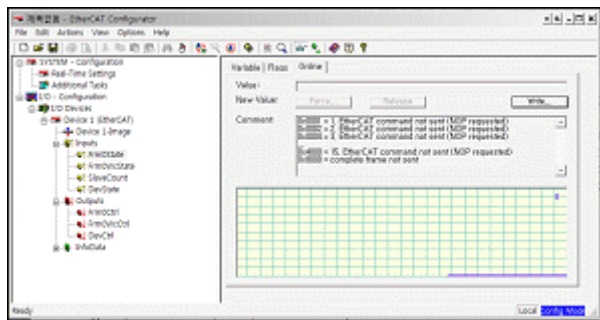


Figure 1. BECKHOFF TWINCAT

However it is difficult to control device without understanding device running model and cannot control device if TWINCAT doesn't support the device driver.

B. Microsoft Robotic studio

Microsoft MSRS(Microsoft Robotic studio) [3] is based on UML modeling programming. It is possible to easily develop robot device program without device knowledge. MSRS actualized this concepts through VPL(Visual Programming Language) and Visual Simulation Environment.

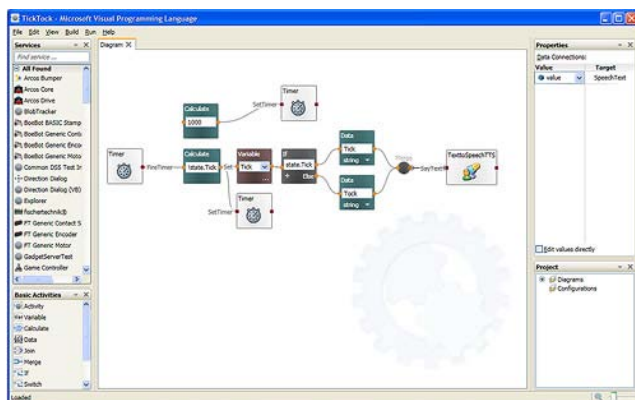


Figure 2. Microsoft Robotics studio

VPL make it possible to construct specific services with simple task. VPL project is configured by block diagrams and connection lines. Each block represents a service, a calculation, a condition, or a nested diagram, and these call activities which control the robot devices. The lines between the blocks represent messages flowing from one service to another. It is possible to implement a simple device control service without writing a single line of code.

MSRS provides several services that connected directly to the simulation engine. The simulation environment supports both indoor and outdoor virtual scenes and comes with a variety of simulated robots. It is much cheaper to destroy a virtual robot with a programming error than a real one.

However, MSRS is difficult that direct apply the control codes to connected device. The connected device must support the Microsoft operating system. It is highly depend on operating system environments.

C. Open source project Beremiz

Beremiz [4] consists of a PLC open editor(visual development tool) and a beckend compiler. Using the PLC open editor, the user may develop programs in IEC61131-3 of the four programming languages and a state machine definition language. The beckend compiler is used to convert these programs into equivalent C programs.

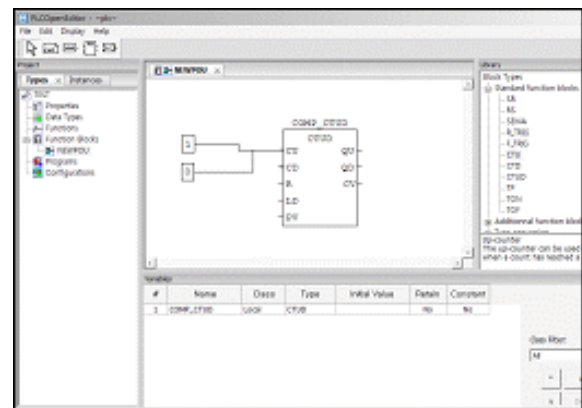


Figure 3. Open source project Beremiz

Beremiz is difficult to develop device control programming without extensive device knowledge and doesn't support direct communication to connected device. Therefore, the user needs to work for device connection and upload control files to target device.

This paper propose a novel development environment model providing graphic development tools based on block diagram and supporting code converting to ST or C language. It easily creates device control codes through graphic development tools and provides direct connection for real time operation.

III. DEVELOPMENT ARCHITECTURE

We have plan on developing architecture model providing robot programming development environment with well-defined interfaces. We have two main goals. First, our architecture model support full graphical interface and automation development environments. Second, we are focus on providing OS independent architecture model. As above mentioned in section 2, current architecture models supporting robot programming development environments don't provide both properties.

For providing both properties, our architecture model has layered architecture. Figure 4 depicts our layered architecture model. Application programmers develop their application programming logic using graphical interfaces. OS Abstraction Layer support operating system independent development processes so that application programmer is not affected by operating system specific constraints. Convertor Layer support

automated conversion from diagram information on visual interface to C language. Communication Layer provide remote development environment. Application developer can develop their programming logic in their personal computer.

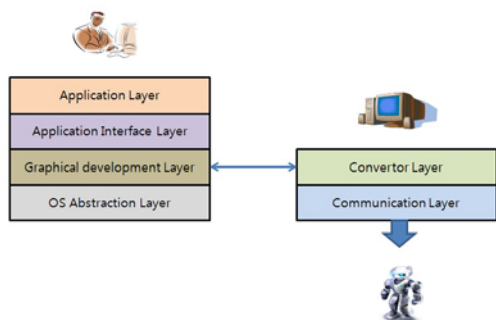


Figure 4. Layered architecture model

Currently, we more focus on developing graphical interfaces. Visual Interface allows the users to insert and delete programming elements easily. The user creates overall control processes through a visual interface. Visual Interface consists of several components, shown in Figure 5.

Each of control processes is mapped with block diagrams which have proper object information [5]. The object information consists of object diagram information and connection diagram information.

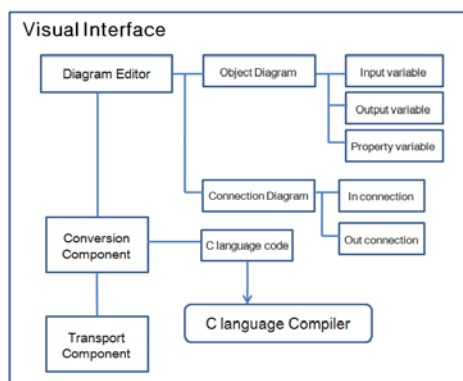


Figure 5. Configuration in Visual Interface

The object diagram information defines proper tasks such as calculation, device connection and device control procedure. Each of object diagrams is connected by connection diagrams. It presents connection relation through the in connection variable and the out connection variable. These variables have object identifier, shown in Figure 6.

Constructing graphical interface may be complex process. For managing software complexity, we maintain object repository. Object repository is not new concept. [6] shows a object repository approach to manage software complexity and provide efficient data processing. Object information is stored in the object repository. In addition, the object repository stores various type of resource such as object identifiers, diagrams drawing position, object variable data.

These resources which represent the relationship of objects are used to maintain information related to configurations. Besides, it is necessary part for converting C language code. The object repository is maintained in the extra files. When programmer makes the any change to the object in run-time, the information related to object is reflected to object repository.

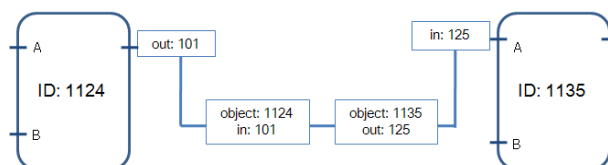


Figure 6. Diagram connection mechanism

Conversion component creates C language code through the resource which is stored in the object repository.

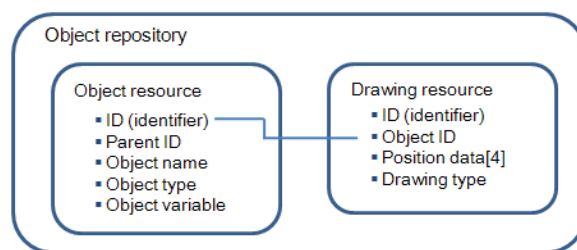


Figure 7. The stored table in object repository

Each of the object resource managed in the one record, shown in Figure 7. A record is identified by ID which connects Object ID in the drawing resource. The drawing resource represents object diagram in Visual Interface. Position data have a drawing position of object diagram and drawing type decides what figure is drawn in Visual Interface. Object type decides what a procedure is executed for device control.

For example, one move object has two input variables and one output variable, it is depicted in figure 8.

C language code created by conversion component, It is compiled through C compiler and to create binary file for a device control. In addition, the transport component sends a binary file to target device. This connection mechanism is peer to peer based on TCP/IP protocol. The user can call a remote procedure for device control through this connection.

IV. CONCLUSIONS

This paper describes a novel architecture model supporting robot programming development environments. We have two main goals. One is to constructing full graphical development environments. Other is to support operating system independent development environments. Currently, we more focus on making well-defined graphical environments. Operating system abstraction is our next step. We have plan on designing and developing this OS abstraction in the next year. We convince that our novel robot development environment

provide application developers with high quality of development environments.

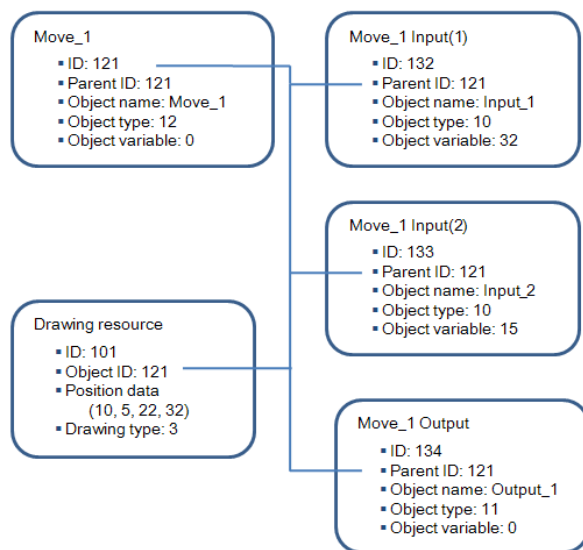


Figure 8. The stored table in object repository

ACKNOWLEDGMENT

This work is financially supported by the Ministry of Education, Science and Technology(MEST), the Ministry of Knowledge Economy(MKE) through the fostering project of HUNIC

REFERENCES

- [1] IEC, International Standard for Programmable Controllers: Programming Languages (Part 1) 2nd edition, January 2003.
- [2] IEC, International Standard for Programmable Controllers: Programming Languages (Part 3) 2nd edition, January 2003.
- [3] Kyle Johns, Trevor Taylor, "Microsoft Robotics Developer Studio", pp 83-87, January 2008.
- [4] Edouard Tisserant, Laurent Bessard, Mario de Sousa An Open Source IEC 61131-3 Integrated Development Environment.
- [5] Michael F Zaeh, Clemens Poernbacher "Model-driven development of PLC software for machine tools", German Academic Society for Production Engineering (WGP) 2008.
- [6] Gregor Kandare, Giovanni Godena, Stanko Strmcnik "A new approach to PLC software design", The Instrumentation, Systems, and Automation Society(ISA) 2003