SYMBOL NAMING RULES FOR PLANT MODEL GENERATION IN THE PLC SIMULATION

Hyeongtae Park, Ginam Wang and Sang C. Park
Department of Inudstrial Engineering
Ajou University
Yeongtong-Gu, Suwon, Republic of Korea
E-mail: {taiji416, gnwang, scpark}@ajou.ac.kr

KEYWORDS
Programmable Logic Controller (PLC), PLC Symbol, Simulation, Virtual Factory

ABSTRACT

Presented in the paper is the PLC symbol naming rules in order to simulate the virtual factory which is operated by the PLC control logic. In the PLC simulation, the corresponding model, normally called as the plant model, to the PLC receives the PLC output signal, operates virtual devices, and sends the changed PLC input signal which is updated by the virtual device movements. Therefore the plant model is necessary to verify PLC code and to simulate the virtual factory. Normally, the plant model construction takes much time and effort, requires in-depth knowledge about the simulation and PLC code, and even it might has modelling errors. To prevent the modelling error and to save the time and effort for the I/O model construction, this paper proposes the naming rules of PLC symbol to extract the necessary information. The simple example cell is applied to test the proposed approach.

INTRODUCTION

To survive and prosper in the modern manufacturing era, a manufacturing company should be capable of adapting reduced life cycle of products in a continuously changing market place. Simulation is a useful tool for manufacturers to adapt this kind of rapidly changing market to design and analyze complex systems that are difficult to model analytically or mathematically. (Choi and Kim 2000) Manufacturers who are using simulation technology can reduce the time to reach the stable condition of automated manufacturing system by utilizing statistics, finding bottlenecks and pointing out scheduling errors etc. The discrete event system simulation makes possible to extract these kinds of simulation results. (Klingstam and Gullander 1999; Al-Ahmari and Ridgway 1999) ARENA and AUTOMOD is widely used for discrete event system simulation software.

Other than the discrete event system simulation, there is another very important concept which has to be considered in today's simulation environment. With the huge improvement of computing power, many user want to create much more realistic simulation model, which can forecast not only the production capability of the system but also the logical verification, physical validity, and the efficiency of co-working machines. The demand results in the concept of virtual factory (VF), which can be described as a model executing manufacturing processes in computers just like the real world. (Onosato and Iwata 1993; Onosato et al. 1995; Lin et al. 1999) Because the delay in construction of the automated production line and the line stop during the automated mass production system cause the large loss of revenue for the manufacturing company, the VF simulation technologies for process validation prior to the real implementation are becoming more important in many manufacturing areas. Especially, not only showing the movement of the automated virtual devices, but also the running with real control logic. Simulation running with the real control logic is getting important to verify the production system. Control program verification and validation is a rising topic among the current control engineers.

After the successful programming of a Programmable Logic Controller (PLC) code, in conventional control logic verification, checking the correctness of the program is a mandatory job in a shop floor control system. The accuracy of the control logic plays a vital role in the automated manufacturing line for the proper functioning of the production process. (Lucas and Timbury 2003) Normally the PLC verification work has been done manually. This process typically starts with writing the PLC program as per the IEC 61131-3 standard. (IEC, 2003) In next step, the program is downloaded to the PLC, then the input value is provided or changed by the user to check the logical flow of the program. Finally, the PLC program can be simulated by giving the Boolean inputs manually. Although it is easy to check the short and small scale of control code, the checking of a large and more complex program is almost impossible as it is a time-consuming and error-prone process. In addition, it is very difficult to decompose complex PLC code manually and to verify all possible dynamic properties. (Younis and Frey 2003)

To solve the aforementioned difficulties of simulation running with PLC code, the counterpart model which is corresponding to the PLC signal is necessary. Generally, it is called the Plant model. The Plant model receives the PLC output signal and operates the virtual devices in the 3D space. According to the movement of virtual

Proceedings 23rd European Conference on Modelling and Simulation ©ECMS Javier Otamendi, Andrzej Bargiela, José Luis Montes, Luis Miguel Doncel Pedrera (Editors)
ISBN: 978-0-9553018-8-9 / ISBN: 978-0-9553018-9-6 (CD)

device, the renewed signals such as the sensor detection of virtual device movement is sent to the PLC as a PLC input.

There have been several approaches for the Plant model. A simple real-time I/O simulator was proposed to test the PLC program, in which the benefits of an object-oriented simulation approach to perform control program testing quickly was introduced. (David 1998) However no concrete methodology and result were presented. A virtual factory consisting of a 3D graphic model and behavior model was combined with an actual factory to verify the ladder program in the PLC. (Hibnio et. Al. 2006) Park et al. proposed the I/O model as a Plant model which can interchange the signals with PLC. (Park et. al. 2006)

In I/O model approach, the virtual devices are represented with the device states. The state changes are caused by the PLC output signal. Once a state is changed by the PLC signal, the corresponding signals derived by the state change is transfered to the PLC as an input. Even though the I/O model is easy to understand and to construct, constructing the I/O model is time consuming work. Especially the field workers are not willing to construct another work (model), namely I/O model, to verify the PLC code. In addition, if the process scale is as large as the line level, there are not only the huge effort to construct the I/O model, but also some possibility of I/O model error in itself. To overcome the mentioned problems, the I/O model is supposed to be generated automatically. The input factors for constructing the I/O model are PLC code and PLC symbols. Originally, the I/O model is constructed by analyzing the PLC code and PLC symbols which is named with no dominant rule depending to the programmer's propensity are used in the code. Consequently, this paper proposes the naming principle of PLC symbols to construct the I/O model automatically. The presented PLC symbol naming principle permits to contain the necessary information to the symbol to construct the I/O model automatically. With the proposed methodoloty, the PLC simulation is ready to do just by importing the symbol list to the simulator.

PLC SIMULATION

The Programmable Logic Controller (PLC) is a dedicated industrial computer used to control automated processes in manufacturing. (Parr 1999) PLC is designed for multiple inputs and outputs arrangements and it detects process state data through the sensing devices such as limit sensors, proximity sensors, or from the signals of the robots. It executes Boolean logic according to updated input memory and triggers the next command to the actuator such as motor, solenoid valve or command signal for the robots etc. PLC code can be represented in different types of languages. IEC published IEC 61131-3 to standardize PLC languages including Ladder diagram, Sequential Function Chart, Structured Text and Function Block Diagram. (Maslar 1996)

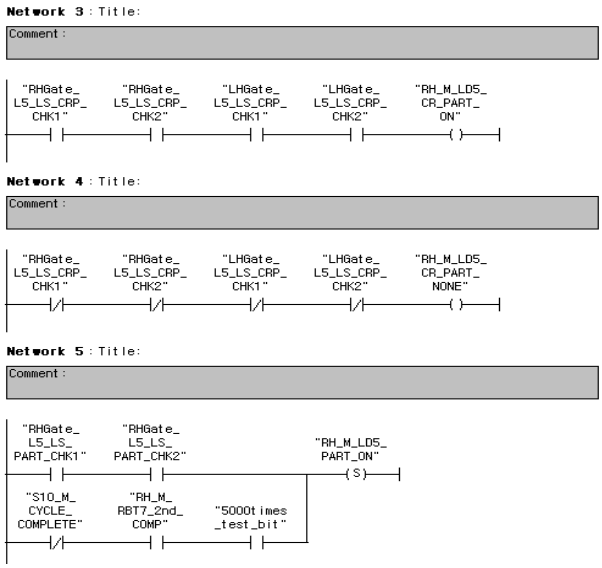


Figure 1: PLC Code Example in the form of Ladder diagram

A PLC can be considered as a system having input and output in essence. To run a PLC, a counterpart system which is called Plant Model is required to interact with the input and output of the PLC. The behaviour of the Plant model should be similar to that of the actual system to achieve correct PLC verification. Thus, the procedures start with the virtual modelling of the actual manufacturing system. Simulation is conducted to check the behaviour of the entire system. If desired behaviours are not observed or undesired behaviours are observed in the simulation result, then the code errors can be corrected such as robot conflict, non detection of emergency condition and over run problem. The verification architecture of the PLC simulation is shown in Figure 2.

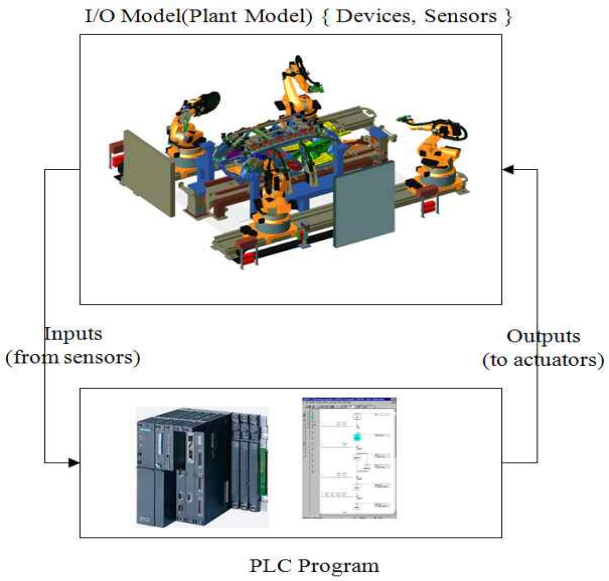


Figure 2: PLC Simulation architecture

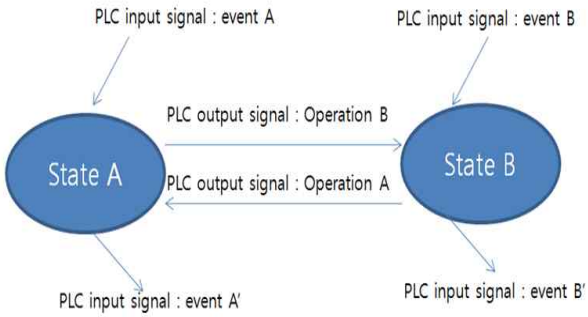
I/O MODEL AS A PLANT MODEL

Manufacturing devices consisting the process line can be divided into three categories. The first are intelligent devices (such as a transporter, handling robot and welding robot), the second are simple actuators (JIG, servo motor) and the third are sensing systems (proximity sensor, light sensor). Because the PLC controls various counterpart devices, virtual process is also needed to model the each resource to correspond to the PLC. As a counterpart modeling (Plant model) technique, this paper employs the I/O model. (Park et. al. 2006) The I/O model is to describe the devices of virtual process line using a state-based object model. The principles of the I/O model involve device modeling with the PLC point of view, which is focused on the control signals and decomposition of the entire system according to their functional role as many as possible. The user can imitate the target system using a finite set of an I/O model, including operations, states and events. And the I/O model receives the PLC output signals and sends the corresponding signals as the PLC inputs. The specified I/O model consists of eight denotations:

$$M = < I, O, S, \delta_{int}, \delta_{ext}, \lambda_{enter}, \lambda_{leave}, \lambda_{disable} >$$

I : *input set*
O: *output set*
S: *state set*
δ_{int} : *internal transition function*
δ_{ext} : *external transition function*
λ_{enter} : *entering output function*
λ_{leave} : *leaving output function*
$\lambda_{disable}$:*disable function*

In the components of an I/O model, a device is partitioned into a set of states, inputs and outputs. The behavior of a device is defined by five function-sets: external/internal transitions, entering/leaving output functions and a disable function. The behavior of the system is controlled by the PLC, which sends signals to I/O model. An external transition function changes its state when it receives external input from the PLC: $\delta_{ext}(A, I \mid \Delta t) = B$ represents "If the current state is A and the model receives external input I, then the state will change to B during Δt". An internal transition function describes an automatic transition during a pre-defined time: $\delta_{\text{int}}(B \mid \Delta t) = C$ represents "If the current state is B, the state will change to C during Δt time. The entering output function generates an outgoing message to the PLC when the model enters a state: $\lambda_{enter}(B) = O_1$ indicates "If the model enters State B, then the function emits a O_1 signal to the external controller (PLC). The leaving output function generates an outgoing message to the PLC when the model leaves from a state: $\lambda_{leave}(B) = O_2$ indicates "If the model leaves State B, then the function emits a O_2 signal to the external controller (PLC). The disable function sets a specific I/O model to disabled mode when the model enters a state: $\lambda_{disable}(B) = M_1$ indicates "If the model enters State B, then the model M_1 is disabled; if the model leaves State B, than the model M_1 is enabled." This method is useful for the modeling of automated process line from the point of view of the controller.
The graphical notation of an I/O model is shown in Figure 3.



Denotation	signal
Entering output function	event A, event B
Leaving output function	event A', event B'
External(internal) transition function	Operation A, Operation B

Figure 3: I/O model as a methodology of Plant model

PLC SYMBOL RULE FOR THE I/O MODEL AUTO GENERATION

Even though the IEC 61131-3 standardized the PLC in many aspects, (Parr 1999) the PLC symbol name structure is not defined. In the real field, there is no rule for the symbol name. It varies from the PLC programmer's naming habit and from the different PLC vendor because of hardware characteristics. As a result, only the PLC programmer can recognize the exact usage of a symbol from the name. Here the necessity of the standardization of symbol name is found to extract the meaningful information from the symbol name.
Generally, the symbol name can be partitioned by under bar ('_'). Despite of this is a general truth; there is no detail rule which indicates the process information. This paper proposes the naming principle to generate the I/O model automatically.
The I/O model contains each device's state and related signals, which can change the state or device movement and can report the device's state. Therefore, symbol name need to have the device information, especially the device hierarchy or the involving higher level process information for the larger line. In addition to the device information, the states are needed to be defined in the symbol name to generate the states for each device. In case of Operation, this signal is the PLC output signal which changes the device states. By putting the next state name to the operation name, we can extract the state and operation name at the same time. From the state change, the event will report the state change to the

PLC. Unlike the operation is the PLC output signal, the event signal is the input for the PLC. Therefore we can give the similar name pattern to the signal just by distinguishing if the one is input or output signal. Proposed signal structure is shown in Figure 4.

SYMBOL structure

Level 1	Level 2	Level 3	Level 4	Level 5

Level	Contents	example
Level 1	Line name	BB(body build)
Level 2	Process number	S301(station 301)
Level 3	Device name	RBT3(robot3)
Level 4	Input or Output	I or O
Level 5	Task	WELD(welding)

Figure 4: Proposed PLC Symbol structure

In the procedure of the I/O model auto construction, the information from level 1 to level 3 is referenced to recognize the device, extract all the related symbols for a selected device, and to create device hierarchy in terms of whole process line.
Once the related symbols for one device are extracted, next step is to create the device states. Among the PLC output symbols whose 4th name field is 'O' is used to create the states. The 5th name field information of operations symbol indicates the terminal state, namely this operation is a triggering symbol to that terminal state.
The remained work is to assign the PLC input, namely event symbol in the I/O model, to the each state. The Event symbols are the symbol which senses the state changes and report to the PLC as a PLC input. The Event symbol is recognized by the 4th name field of symbol. If 4th name is showing the 'I', this means that current symbol is PLC input symbol, which detects the state in 5th level name field and transfer the signal to the PLC. While the operation symbol indicates the next state which will be reached by the this operation symbol, the event symbol show the current detected state. With this method, the symbols will be referenced to generate the I/O model automatically. Detail procedure and example is shown in next chapter.

EXAMPLE CELL AND IMPLEMENTATION

In this chapter, we will observe a small work cell example. Figure 5 shows the small cell example. At first, an entity will be generated. Then this will lay on the AGV machine in P1, then the AGV senses this raw part and moves to the P2 for machining. When the machine detects the part arrival by the AGV in P2 position, the machine starts to operate. We can consider two devices in this example cell for the I/O model, the AGV and the Machine .

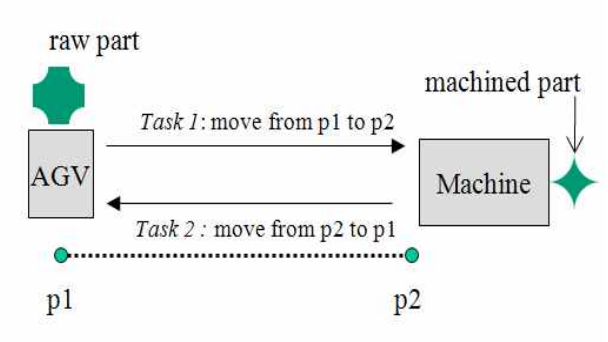


Figure 5: Example Cell

The AGV model can have two states. One state is when the AGV is in P1 position and another is when the AGV is in P2. Therefore, it has two input signals that sense the each position, and two operation signals that command to move. With the proposed symbol naming rule, we can give the signals for inputs as 'EXCELL_MB_AGV_I_P1', 'EXCELL_MB_AGV_I_P2'. Here, it is assumed that process name is MB (Main Buck). Similarly, we can make the operation signals. The first one is 'EXCELL_MB_AGV_O_P2' that command to move to the P2 position from the P1 position. And the second is 'EXCELL_MB_AGV_O_P1' that commands to move to the P1 position from the P2 position. In case of Machine device, the states can be defined into two. The first one is the Idle when the Machine is not operating or waiting for the next raw part. The second is the Run when the machine is running. Likewise the AGV symbols, we can create the symbols of the Machine. The event signals can be'EXCELL_MB_MC_I_IDLE1','EXCELL_MB_MC_I_RUN2', and the operation can be 'EXCELL_MB_MC_O_IDLE1', 'EXCELL_MB_MC__RUN2' respectively. The number in the last digit of the level 5 means the state sequence. The machine is on IDLE state at first and it changes its state to the RUN.

	A	B	C	D	E	F
1	Class	Identifier	MIT-Addr.	IEC-Addr.	Type	Initial
2	VAR_GLOBAL	EXLINE_MB_AGV_I_P1	X0001	%IX0001	BOOL	FALSE
3	VAR_GLOBAL	EXLINE_MB_AGV_I_P2	X0002	%IX0002	BOOL	FALSE
4	VAR_GLOBAL	EXLINE_MB_AGV_O_P2	Y0001	%QX0001	BOOL	FALSE
5	VAR_GLOBAL	EXLINE_MB_AGV_O_P1	Y0002	%QX0002	BOOL	FALSE
6	VAR_GLOBAL	EXLINE_MB_MC_I_IDLE1	X0003	%IX0003	BOOL	FALSE
7	VAR_GLOBAL	EXLINE_MB_MC_I_RUN2	X0004	%IX0004	BOOL	FALSE
8	VAR_GLOBAL	EXLINE_MB_MC_O_IDLE1	Y0003	%QX0003	BOOL	FALSE
9	VAR_GLOBAL	EXLINE_MB_MC_O_RUN2	Y0004	%QX0004	BOOL	FALSE

Figure 6: Symbol list according to the proposed rule

With the example cell, the PLC symbol list is presented in Figure 6. This list is formatted for GX IEC developer version 7.0 produced by Mitsubishi Electric Corporation. The above Excel file format can be acquired by exporting the symbol table from the GX IEC developer, in which the symbols are declared and the PLC code is programmed as well. Once the symbol list is decided, it will be inputted to the simulator which is comprised of the I/O model and the virtual graphic factory. Figure 7 shows the generated I/O models in a simulator after importing the PLC symbols. As we can see, the device I/O models are suspended in 'ExLine' project tree. And two I/O models are generated.

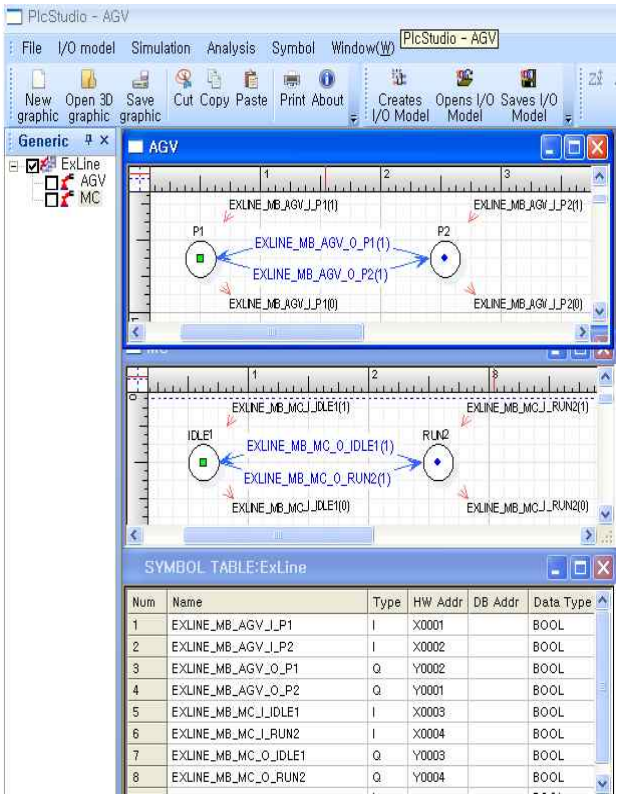


Figure 7: Generated I/O models for example cell

CONCLUSIONS

To simulate the virtual factory running with the PLC logic code, the counterpart model that receives the PLC output signals and sends the corresponding the signals is necessary. Among the previous approaches for counterpart (plant) model, this paper proposes rapid and efficient method to construct the I/O model. Currently, there is no dominant naming principle for the PLC symbol. By publishing the naming principle of PLC symbol in terms of constructing the I/O model automatically, we can extract the necessary information such as device states, event and operation signals. Even though example cell is simple, we can apply to the larger automated line for the I/O model generation if satisfied that the symbol rule is applied with the proposed naming principle.

The generated I/O model will operate the virtual devices and interfaces the signals with the PLC from the result of the device movement in virtual simulator. The Figure 8 shows the overall feature of virtual simulator that is constituted the generated I/O model and the graphic model.

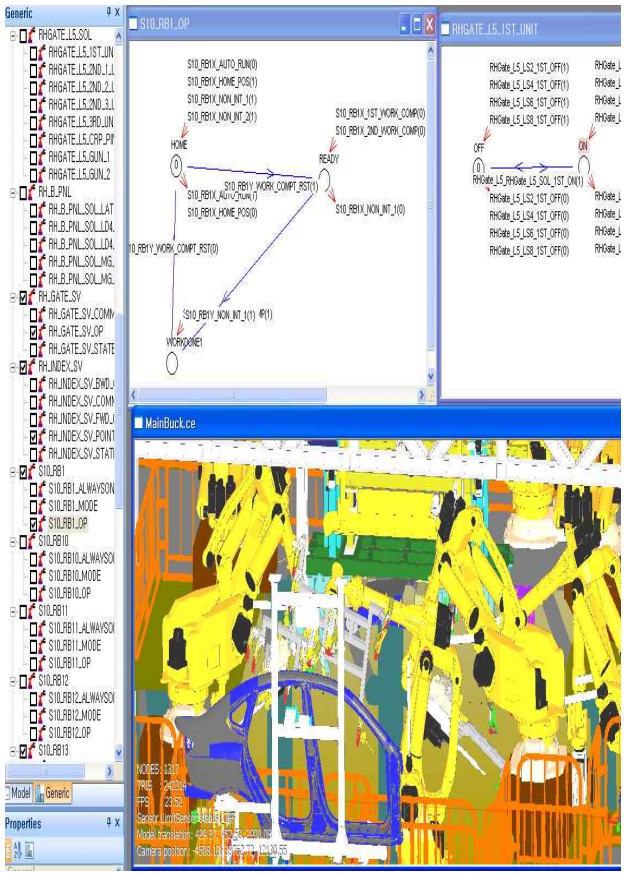


Figure 8: I/O model and the graphic model of larger scale of virtual factory

In a real factory, the some automatic device doesn't operate in decided sequence. Their operation sequence can be varied and can be complex depend on the neighboring devices. In this case, the I/O model might be complicated and the proposed naming principle is not concerned this limitation. To be the more realistic and practical simulator, however, the I/O model and the corresponding PLC naming principle are needed to cover this issue as the future work.

REFERENCES

Al-Ahmari, A. and Ridgway, K., 1999. "An integrated modeling method to support manufacturing system analysis and design.", *Computers In Industry,* Vol. 38, 225-238.

Choi, B. K. and Kim, B. H. 2000. "Trend in CIM: Virtual manufacturing systems for next generation manufacturing. *Current advances in Mechanical Design and Production Seventh Cairo University International MDP Conference,* 425-436.

David, J. D. 1998. “Application and benefits of Real-time I/O Simulation for PLC and PC Control System.”, *ISA Transaction,* Vol. 36, 305-311.

Hibnio, H., Inukai, T. and Fukuda, Y., 2006. “Efficient Manufacturing System Implementation Based on Combination between real and virtual factory.”, *International Journal of Production Research,* Vol. 44(18), 3897-3915.

IEC(International Electrotechnical Commission), 2003, *“*International Standard IEC 61131-3: Programmable Controllers-Part 3 Programming Languages, Second Edition”.

Iwate, K., Onosato, M. and Teramoto, K., 1995. “A modelling and simulation architecture for virtual manufacturing systems”, *CIRP,* Vol. 44, 399-402.

Klingstam, P. and Gullander, P., 1999. “Overview of simulation tools for computer-aided engineering.”, *Computers In Industry,* Vol. 38, 173-186.

Lin, M., Fu, L. and Shin, T., 1999. “Virtual Factory – a novel testbed for an advanced flexible manufacturing system.”, *IEEE International Conference on Robotics & Automation,* Detroit, MI, 2422-2427.

Lucas, M. and Tilbury, D., 2003. “A Study of Current Logic Design Practices in the Automotive Manufacturing Industry.”, *International Journal of Human-Computer Studies,* Vol. 59, 725-753.

Maslar, M. 1996. “PLC Standard programming Language: IEC 61131-3.”, *IEEE Pulp and Paper Industry Technical Conference,*

Onosato, M. and Iwata, K., 1993. “Development of a virtual manufacturing system by integrating product models and factory models.”, *CIRP,* Vol. 42, 475-478.

Park, C. M., Sachin, M., Park, S., C. and Wang G. N., 2006. “Development of Virtual Simulator for Visual Validation of PLC Program.”, *International Conference on Intelligent Agent Web Technologies & Internet Commerce,* Australia.

Parr, E. A. 1999. Programmable Logic Controllers: An engineer’s guide 3rd edition.

Younis, M. *B.* and Frey, G. M., 2003. *“*Formalization of Existing PLC Programs: A Survey.*”, Proceeding of CESA,* Paper No. S2-R-00-0239.

AUTHOR BIOGRAPHIES

HYEONG T. PARK is currently a Ph.D candidate/researcher in the Department of Industrial & information systems at Ajou University, South Korea. His area of research is related to PLC verification and simulation. His other research interest is the auto generation of PLC code from the flow chart and the Digital Manufacturing System. He can be reached at taiji416@ajou.ac.kr

GI N. WANG is a full professor of Ajou University, South Korea. He received his BS in Ajou University, MS in KAIST, and PhD in Texas A&M. After his degree, he worked in Hyundai electronics as a researcher of department of production information control systems, and was a visiting professor in the university of Texas at Austin. His research area is related to industrial control, artificial intelligence, and PLC. He can be reached at gnwang@ajou.ac.kr

SANG C. PARK is an associate professor in the Department of Industrial & Information Systems Engineering at Ajou University. Before joining Ajou, he worked for DaimlerChrysler Corp. and CubickTek Co., developing commercial and in-house CAD/CAM/CAPP/simulation software systems. He received his BS, MS, and PhD degrees from KAIST in 1994, 1996, and 2000, respectively, all in industrial engineering. His research interests include geometric algorithms in CAD/CAM, Digital Manufacturing System, process planning, engineering knowledge management, and discrete event system simulation. He can be reached via email at scpark@ajou.ac.kr