Journal of the Korea Academia-Industrial cooperation Society
Vol. 14, No. 10 pp. 5134-5139, 2013

http://dx.doi.org/10.5762/KAIS.2013.14.10.5134

SFC언어에서 인터럽트 프로그램 시간개선에 관한 연구

유정봉[1*]
[1]공주대학교 전기전자제어공학부

Study on the Time Improvement of Interrupt Program by SFC

Jeong-Bong You[1*]
[1]**Division of Electrical & Electronics & Control Eng. Kongju National University**

요 약 복잡한 현대의 제어시스템 설계에 PLC를 사용하면 프로그램은 LD언어나 SFC언어를 사용한다. 대부분은 LD 언어를 사용하지만 최근에는 SFC 언어의 사용빈도수가 높아졌다. SFC 언어는 제어의 흐름을 이해하기가 쉽지만, 조합논리를 표현하는데는 단점을 가지고 있다. SFC언어에서 인터럽트를 처리할 때 인터럽트 요인이 발생하게 되면 메인프로그램을 중지하고 인터럽트 프로그램을 실행하여 프로그램이 종료된 후 메인프로그램으로 복귀하게 된다. 그러면 인터럽트 프로그램이 복잡할수록 메인프로그램 정지시간은 그만큼 길어지게 된다. 본 논문에서는 SFC언어에서 메인프로그램의 휴지시간이 없는 인터럽트 처리방법을 제안하고, 시뮬레이션을 통해 그의 타당성을 확인하였다.

Abstract Ladder Diagram(LD) or Sequential Function Chart(SFC) is used for the design of complex modern control system with Programmable logic controller(PLC). LD is the most widely utilized among PLC standard language. But recently, SFC is used frequently. SFC is very easy to grasp the sequential flow of control logic but is difficult for describing combinational logic. When the interrupt factor is occurred, the main program is stopped. And after the interrupt program is completed, the main program is restart. Therefore the more complex the interrupt program, the main program is interrupted downtime will be that much longer. In this paper, we propose the method for interrupt implementation without the dwell time of the main program by SFC language and confirm his feasibility through the simulation.

Key Words : PLC, LD, SFC, Control system, Interrupt

1. 서론

요즘 같은 복잡한 현대사회에서는 다양한 IT 매체가 발달하고 있다. IT 매체를 활용할 기기는 소형화되고 고속의 처리속도를 갖는 고기능, 고속의 기기가 경쟁력이 있게 된다. 이러한 기기를 생산하는 기업체에서는 생산설비의 첨단화 및 고속화를 이루어야 제품에 대한 경쟁력을 갖추게 된다. 생산설비의 첨단화 및 고속화를 위한 제어기기로는 대표적으로 컴퓨터 제어기기 및 PLC (Programmable Logic Controller)가 가장 대표적으로 사용된다.

현대의 PLC는 초기에 개발된 이후로 마이크로프로세서의 발전에 의해 고도의 기능을 갖는 PLC가 출현하여 대규모의 입출력을 처리할 수 있고, 공장자동화(Factory Automation, FA)의 요구에 맞춰 고도의 기능을 갖게 되었고, 고속의 처리능력을 갖추게 되었다[1-3].

PLC를 제어하기 위해서는 제어 프로그램이 필요하다. 제어프로그램은 IEC-1131-3의 국제 규격에 제시된 표준 언어를 사용하게 되며, IL(Instruction List), ST(Structured Text)의 텍스트 기반언어와 LD (Ladder Diagram), FBD(Function Block Diagram), SFC(Sequential Function Chart)의 그래픽 기반 언어를 사용하게 된다[4,5].

SFC언어는 프랑스에서 공장자동화를 위해 이해하기 쉽고 구현하기 편리하도록 1977년에 개발된 GRAFCET

본 논문은 공주대학교 자체학술연구비로 수행되었음.
*Corresponding Author : Jeong-Bong You(Kongju Nation Univ.)
Tel: +82-41-521-9156 email: jbyou@kongju.ac.kr
Received August 23, 2013 Revised (1st September 23, 2013, 2nd September 30, 2013) Accepted October 10, 2013

에 근거한 국제 표준언어이다. SFC언어는 순차 제어 논리의 기술에 적합한 그래픽언어이므로 제어의 흐름을 이해하기 쉬우며, 유지보수가 용이하고, 프로그램의 기술성이 뛰어나고 기계장애의 진단성이 우수하다는 장점이 있다.

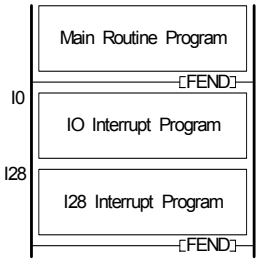
SFC언어를 사용하여 프로그램을 하다보면 조건 처리 및 인터록(Interlock)을 구현하기가 까다롭다는 단점이 있다. 또한 입출력의 타스크(Task)를 처리하기 위해서는 인터럽트(Interrupt)로 처리해야 하는데 인터럽트로 처리하게 되면 인터럽트 루틴으로 들어갔을 때는 CPU는 처리하던 작업을 정지 하게 되고 인터럽트 처리를 하게 된다. 그러면 CPU 작업 효율이 떨어지고 전체 작업 시간이 늦어지게 된다. 본 논문에서는 이러한 단점을 극복하고자 매크로 블록을 사용하였고, 매크로 블록을 사용하게 되면 CPU가 정지하여 인터럽트를 처리하지 않고 CPU는 CPU대로 프로그램실행을 하고, 인터럽트는 CPU와 상관없이 실행되어 인터럽트 처리 시간을 개선할 수 있는 방법을 제시하였다. 또한 제안된 방법을 캐링 로봇에 적용하여 인터럽트 처리되는 시간을 개선하고 있음을 확인하였다.

2. 인터럽트 프로그램 처리

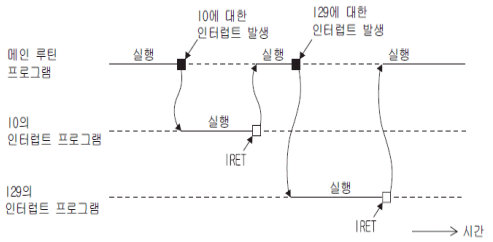
2.1 래더 다이어그램으로 인터럽트 처리

래더 다이어그램으로 프로그램할 때 인터럽트 프로그램은 시퀀스 프로그램이 동작되는 상태에서 인터럽트 요인이 발생하게 되면 시퀀스 프로그램을 중지하고 인터럽트 프로그램을 실행하게 된다. 즉, 인터럽트가 발생할 때에는 CPU는 다른 작업을 하지 못하고 인터럽트 프로그램을 실행하게 된다.

Fig. 1에서 메인 프로그램이 동작을 하다가 인터럽트 요인이 발생을 하게 되면, 메인 프로그램을 중지하고 인터럽트 포인터 번호에 해당하는 인터럽트 프로그램이 실행된다. 이러한 인터럽트 프로그램에 대한 실행 타이밍은 Fig. 2와 같다.



[Fig. 1] Interrupt Implementation by Ladder diagram

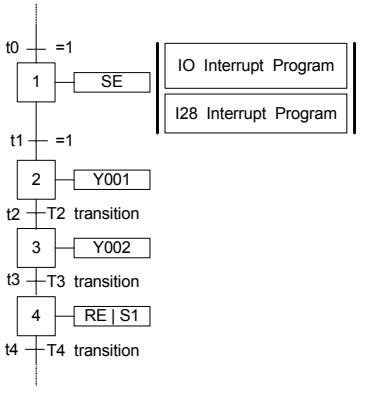


[Fig. 2] The Operation Timing of Interrupt Program

Fig. 2에서 알 수 있듯이 인터럽트 요인이 발생하여 인터럽트 프로그램이 실행될 때는 메인 프로그램이 중지하기 때문에 인터럽트 프로그램이 길면 길수록 메인 프로그램은 그만큼 시간에 대한 손실이 커지게 된다.

2.2 SFC언어를 사용한 기존의 방법

SFC언어를 사용하여 인터럽트 프로그램을 처리하기 위해서는 기존에 Fig. 3과 같은 방법을 사용하게 된다.



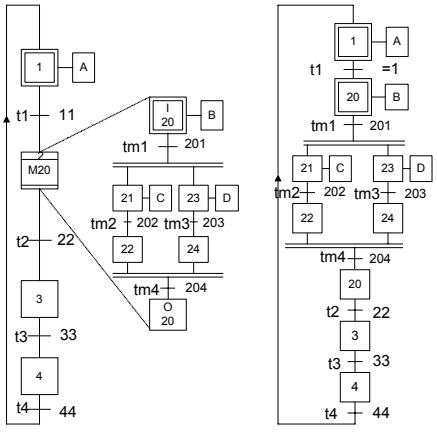
[Fig. 3] SFC Program using Set Instruction to Process Example

Fig. 3에서 인터럽트로 처리하고자하는 스텝은 1번 스텝으로 되어있다. 1번 스텝 내부 전체에 대해 SET 명령(Fig. 3에서 1번 스텝의 SE)을 사용하는 방법이 있다. 1번 스텝을 활성화 시킨 상태에서 t1 천이조건이 만족되어 2번 스텝이 활성화되면 1번 스텝을 비활성 상태로 되어야 한다. 이것은 SFC 언어의 규칙에 해당한다. 그러나 Fig. 3에서 1번 스텝을 [SE] 액션 제한자를 사용하였기 때문에 t1 천이조건을 만족하여 S2 스텝으로 천이된다 하더라도 1번 스텝은 그대로 활성상태로 남아있어 항상 인터럽트 처리를 기다리고 있게 된다. 그러나 이와 같은 방법은 인터럽트를 처리하지 않게 되거나, 프로그램이 종료하게 되면 Fig. 3의 4번 스텝과 같이 [RE] 액션 제한자를 사용하여 1번 스텝의 SET 상태를 해제해주어야 한다. 그

러면 불필요한 스텝이 첨가되어 프로그램메모리의 용량이 늘어나 비효율적이 된다.

3. 제안된 인터럽트 처리

본 논문에서 제안하는 인터럽트 처리방법은 인터럽트를 처리할 스텝을 매크로 스텝으로 처리하는 방법이다. 매크로 스텝은 제어하고자 하는 시스템이 복잡할 때 사용하게 되는데, 하나의 스텝을 또다른 SFC 언어로 기술하여 세부적으로 SFC 언어의 그래픽 표현을 명확히 하게 된다. Fig. 4에 매크로 스텝의 예를 나타낸다[6].



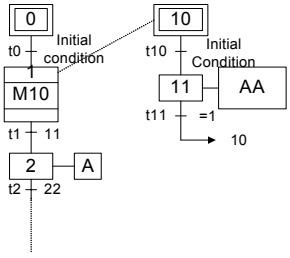
[Fig. 4] Macrostep (a) SFC with Macrostep (b) Macrostep Expansion (c) Equivalent SFC without Macrostep

Fig. 4 (a)는 메인 SFC 프로그램을 나타내고 있다. 총 4개의 스텝으로 구성되어 있는데, 이중 2번 스텝 M20 이 Fig. 4 (b)와 같은 매크로 스텝으로 구성되어 있다. M20 매크로스텝은 6개의 스텝으로 구성되게 된다. Fig. 4 (b)의 매크로스텝을 메인 SFC 프로그램에 삽입하여 단일 시퀀스로 구성하게 되면 Fig. 4 (c)와 같게 된다. 매크로 스텝은 하나의 입력스텝과 출력스텝을 갖게 되며, 이전 천이조건의 모든 점화는 그 확장의 입력 스텝을 활성화 시킨다. 또한 매크로 스텝의 확장의 출력 스텝은 그 매크로 스텝을 포함하는 SFC의 구조에 따라 그 다음 천이조건을 가능하게 한다.

매크로 스텝은 2가지로 나누어진다. 하나는 매크로 스텝의 확장이 END를 확인한 후 메인 SFC의 다음 스텝으로 진행하는 매크로 스텝과 END를 확인하지 않고 메인 SFC의 다음 스텝으로 진행하는 매크로 스텝이다.

첫 번째 매크로 스텝의 확장이 END를 확인한 후에 메인 SFC 프로그램의 다음 스텝으로 진행하게 되면 매크로 스텝이 종료될 때까지 메인 프로그램은 대기상태가 되어 시간적인 손실이 발생하게 되고, 2.2절에서 기술된 SET 명령을 사용하는 방법과 동일하게 된다. 따라서 이러한 방법은 아무 의미가 없게 된다.

두 번째 매크로 스텝의 확장이 END를 확인하지 않고 메인 SFC 프로그램의 다음 스텝으로 진행하게 되면 매크로스텝의 확장과 메인 프로그램이 동시에 진행되는 결과가 되어 시간적인 손실이 전혀 발생하지 않게 된다. 이와 같은 매크로 스텝은 Fig. 5와 같다.



[Fig. 5] Macrostep that doesn't check END

SFC 언어의 규칙중에서 현스텝에서 천이조건을 만족하여 다음스텝으로 천이되면 이전 스텝은 비활성 상태로 된다는 규칙이 있다. Fig. 5에서 1번 스텝이 활성화되어 매크로 블록 M10이 활성화되어 M10이 동작되고 t1 천이조건을 만족하게 되면 2번 스텝으로 천이한다. 그러나 1번 스텝의 M10은 비활성상태로 되는 것이 아니고, M10 매크로 블록을 강제로 무한 루프로 만들어 계속해서 1번 스텝이 활성화 상태로 남아있게 된다.

본 논문에서는 Fig. 5와 같은 END를 확인하지 않는 매크로 블록을 사용해서 인터럽트를 처리하여 기존의 인터럽트 루프에 비해 시간적으로 훨씬 효율성있게 처리하였다.

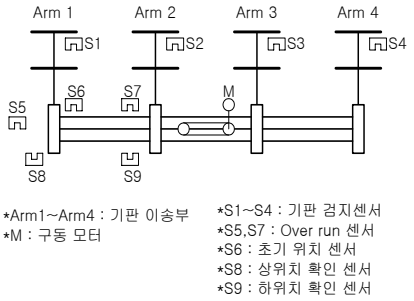
4. 시뮬레이션

4.1 시스템의 개요

본 연구에서 사용된 시스템은 제품을 이송하는 시스템이다. 간단하게 4개의 arm으로 구성되어 있다.

캐링 로봇은 유리 제조공정에서 유리를 이송시키거나, LCD 제조공정에서 LCD를 다음공정으로 이송시키기 위해 사용된다. 본 캐링 로봇은 LCD 제조공정에서 4개의 공정으로 LCD 기판을 각각의 공정으로 이송시키기 위해 사용되며, 4개의 Arm으로 구성되어 각각의 Arm에서 4개

의 공정으로 배분된다.

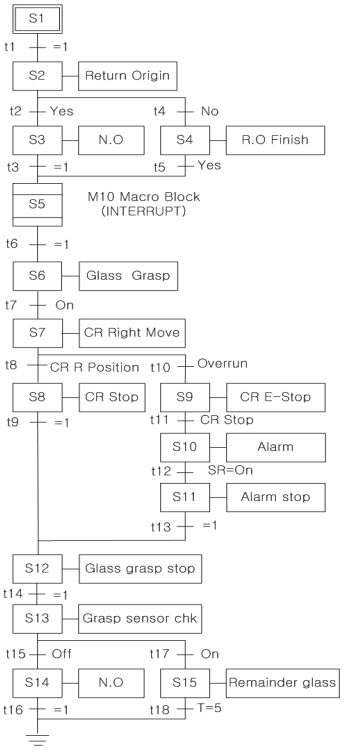


[Fig. 6] Carrying Robot

4.2 알고리즘 설계

본 연구에서 사용된 시스템에 대한 제어 사양은 다음과 같다.

(1) 제품을 잡고 우측으로 한번 이동한다.
(2) 좌측 동작이나 또다른 우측 동작도 동일한 내용으로 구성된다.
(3) 알고리즘은 선택시퀀스로 구성된다.
(4) 인터럽트 루틴은 매크로 블록으로 구성된다.



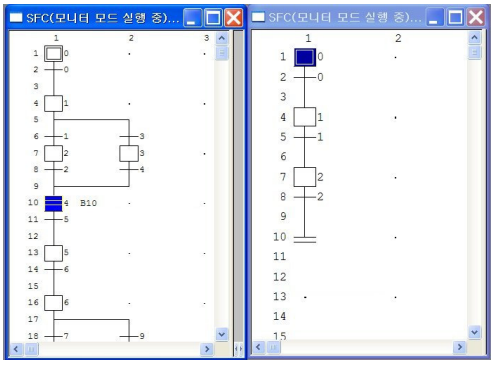
[Fig. 7] Algorithm of Carrying Robot

Fig. 7의 캐링 로봇 알고리즘에서 스텝 S5가 매크로 블록으로 설정되어 인터럽트 프로그램이 삽입되어 있다. 이 END를 체크하지 않는 매크로 블록은 S5 스텝이 활성화된 후, t6 조건을 만족하여 다음 스텝 S6스텝으로 천이되어도 S5 스텝은 비활성화되지 않고 활성화된 상태로 되어 메인 프로그램이 동작되면서 인터럽트 프로그램도 동시에 동작되게 된다.

4.3 시뮬레이션 결과

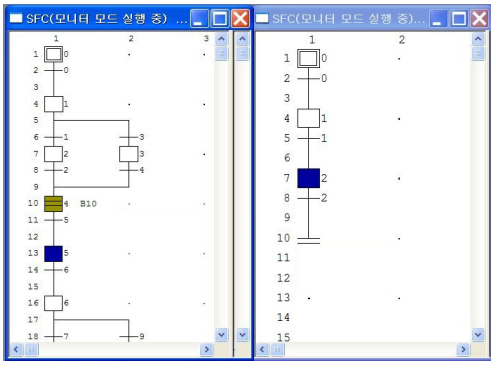
본 연구에서 사용된 PLC는 미쓰비시 Q03UD PLC이다. 또한 에디팅 프로그램은 GX Developer 8.0을 사용하였다[6]. 캐링 로봇은 4개의 Arm에 대해 전체의 동작에 대한 인터럽트 동작은 공통으로 구성된다. 따라서 Fig. 7의 4번 스텝인 S4가 인터럽트 스텝으로 구성되어 전체 프로그램이 구동되어도 S4 스텝은 항상 활성화 상태로 되어 있어야 한다.

Fig. 8은 인터럽트 스텝인 S4 매크로 블록이 활성화되어 있음을 보여준다. 또한 이 매크로 블록은 또다시 내부에 SFC 언어로 구성을 할 수가 있으며, 이 내부 SFC의 0번 스텝이 활성화 되어 있음을 알 수 있다. 즉, 인터럽트가 동작되고 있음을 나타내고 있다.



[Fig. 8] Results applying Interrupt Block

Fig. 8의 S4 매크로 블록이 활성화 되어 있는 상태에서 트랜지션 t5가 조건을 만족하게 되면 다음 스텝 S5로 천이하게 된다. SFC의 규칙에서 현재의 스텝이 활성화되어있을 때 천이조건을 만족하여 다음스텝으로 천이하게 되면 그 이전스텝은 비활성화 된다. 그러나 본 연구에서는 END를 검사하지 않는 매크로 블록을 사용하여 인터럽트 블록으로 처리하였기 때문에 t5 천이조건을 만족하여도 다음 스텝 S5가 활성화될 때 S4 매크로 블록은 비활성화가 아닌 활성화상태로 되어 인터럽트 루틴을 계속 수행하게 된다. 이것은 Fig. 9에서 나타난다.

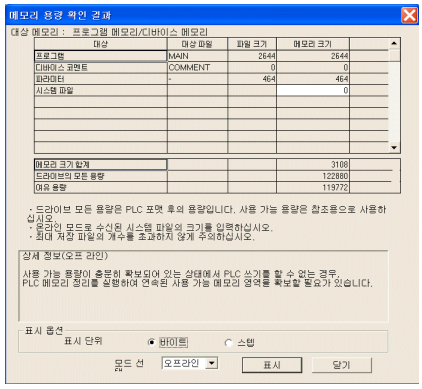


[Fig. 9] Results applying Next Step

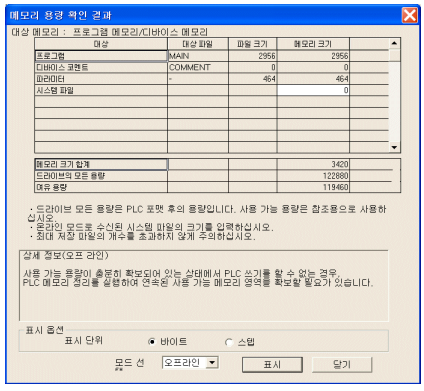
Fig. 9에서 알 수 있듯이 S4 매크로 블록이 활성화 된 상태에서 t5 천이조건을 만족하여 S5 스텝으로 천이되어 S5 스텝이 활성화 되어 있다. 그래도 이전스텝인 S4 스텝의 내부 SFC에서 S2 스텝이 활성화되어 있어 인터럽트가 지속적으로 동작하고 있음을 알 수 있다. 또한, SFC언어에서 기존에 사용하는 방법인 Fig. 3의 방법을 사용했을 때와 각각의 트랜지션에 모든 인터럽트 접점을 추가했을 때의 메모리 크기를 Fig. 10에 나타내었다. Fig. 10에서 나타난 메모리 용량을 비교한 것이 Table 1이다.

Table 1에서 기존의 각 트랜지션에 모든 인터럽트 접점을 추가했을 때 프로그램의 메모리가 가장 크고, SE 제한자를 사용했을 때 메모리 크기가 좀 더 줄어든다. 그리고 본 논문에서 제시한 매크로블록을 사용했을 때가 프로그램 메모리 크기가 가장 적다. 세가지의 방법에서는 메모리의 크기가 많이 차이가 나지 않지만, 전체 프로그램의 크기가 커지고 인터럽트 프로그램 용량이 커질수록 본 논문에서 제시한 방법이 메모리에 대한 효율도 좋아짐을 확인할 수 있다.

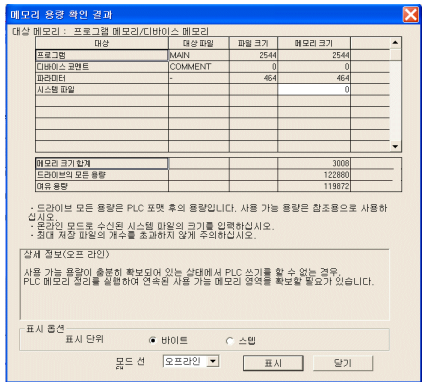
기존의 방법언어에서는 인터럽트 처리할 때 메인 프로그램에서 인터럽트가 발생하게 되면, 메인 프로그램을 중지하고 인터럽트 프로그램으로 분기하여인터럽트 프로그램을 실행하고 인터럽트 프로그램이 종료된 후에 다시 메인 프로그램으로 복귀하기 때문에 메인 프로그램의 처리 시간이 그만큼 길어지게 된다. 그러나 본 연구에서 제안한 방법은 메인 프로그램과 인터럽트 프로그램이 동시에 실행되기 때문에 메인 프로그램은 기존의 방법에 비해 시간적 효율성이 매우 높음을 알 수 있으며, 메모리의 효율도 매우 높음을 알 수 있다. 또한 인터럽트 프로그램의 길이가 길수록 본 연구에서 제안한 방법이 더 효과적이라는 것이 매우 고무적인 일이다.



(a)



(b)



(c)

[Fig. 10] Memory size
(a) SE Qualifier (b) Add In Transition
(c) Macro Block

[Table 1] Comparison of Memory size

[Unit: Byte]

	SE Qualifier	Add in Transition	Macro Block
Memory size	3108	3420	3008

5. 결론

공장자동화 알고리즘을 설계할 때 PLC를 사용하면 대부분 LD 언어를 사용하였지만 LD 언어는 조합논리로 처리되기 때문에 인터록을 기술하거나 인터럽트를 처리할 때 기술하기는 용이하다. 그러나 순차논리에는 부적합하기 때문에 최근에 SFC 언어를 사용하여 프로그램하는 빈도수가 높아지고 있다. SFC 언어에서는 인터록을 처리하거나 인터럽트를 처리할 때는 단점을 안고 있었다. 본 논문에서 제시된 END를 검사하지 않는 매크로 블록을 사용한 인터럽트 처리를 하게 되면 기존의 방법에 비해 처리 시간을 훨씬 단축할 수 있으며, 메모리의 효율도 높일 수 있게 된다. 따라서 좀 더 효율적인 프로그램 처리가 가능하게 되었다.

References

[1] Jeong-Bong You, “Improvement Implementation of Interlock Using Management Step Described by SFC”, The Korean Institute of Illuminating and Electrical Installation Engineers, Vol. 19, No.3, May 2005.

[2] M. Zhou and E Twiss, "Design of Industrial automated systems via relay ladder logic programming and Petrinets", IEEE Trans on Systems, Man and Cybernetics -part C ; Applications and Reviews, Vol 28, No 1, pp137- 150, 1998.
DOI: http://dx.doi.org/10.1109/5326.661096

[3] Giuseppe Casalino, Giorgio Cannata, Giorgio Panin, Adrea Caffaz "On a Two level Hierarchical Structure for the Dynamic Control of Multifingered Manipulation", Proceedings of the 2001 IEEE, International Conference on Robotics & Automation Seoul Korea, 2001.
DOI: http://dx.doi.org/10.1109/ROBOT.2001.932533

[4] Bong-Suk Kang and Kwang-Hjun Cho, "Discrete Event Model Conversion Algorithm for Systematic Analysis of Ladder Diagrams in PLCs" Journal of Control, Automation and systems Engineering, Vol 8. No5, p401- 406, May, 2002.
DOI: http://dx.doi.org/10.5302/J.ICROS.2002.8.5.401

[5] G.Frey and L.Litz, Formal methods in PLC Programming, Proceedings for the IEEE Conference on Systems Man and Cybernetics SMC 2000, Nachville, Oct. 8-11, 2000.
DOI: http://dx.doi.org/10.1109/ICSMC.2000.884356

[6] "Operating Manual (SFC)", Mitsubishi GX Developer Version 8, 2006

유 정 봉(Jeong-Bong You) **[종신회원]**

- 1988년 2월 : 단국대학교 전자공학과 (공학사)
- 1990년 8월 : 단국대학교 전자공학과 (공학석사)
- 1998년 8월 : 단국대학교 전자공학과 (공학박사)
- 1990년 7월 ~ 1993년 9월 : (주) 신도리코
- 1999년 3월 ~ 2000년 2월 : 생산기술연구원 비상근 연구원
- 1999년 8월 ~ 현재 : 공주대학교 전기전자제어공학부 교수

<관심분야>
PLC제어, 마이크로프로세서 제어, BLDC 모터제어, 공장자동화 알고리즘 설계