지능적인 멀티에이전트 기반 소프트웨어 PLC

Intelligent Multiagent Based Software Programmable Logic Control

조 영 임*
(Young Im Cho)

Abstract : In this paper, I developed an intelligent multiagents based softPLC(IMPLC). In IMPLC, the standard IEC 1131-3 PLC languages(LD, SFC, FBD, ST) programmed by a user are converted to IL, which is one of intermediate codes, in order to make them interactions. And then the IL is converted to the standard C code regarding some extension and transplanting, which can be used in a commercial editor such as visual C++. In IMPLC, the logical errors and syntax errors occurred by users are detected, so that the optimal PC control based softPLC can be possible. IMPLC provide easy programming platform to such beginner as well as professionals. The study of code conversion is firstly tried in the world as well as KOREA.
I applied IMPLC to 3 steps conveyer belt system. The simulation results say that the debugging steps by IMPLC using multiagents are decreased than the conventional softPLC's.

Keywords : software PLC, multiagents, IEC1131-3, intelligent multiagenst based PLC

I. 서론

PLC 프로그램이란 PLC가 일을 할 수 있도록 PLC에게 주어지는 명령의 일종으로 컴퓨터적인 고도의 지식보다 릴레이 등의 시퀀스 제어에 숙달되면 충분히 구현할 수 있도록 다각화되어 개발되고 있으며, 제어의 내용이나 적용분야에 따라 프로그래밍 언어의 특징에 적합한 것을 선정하여 사용할 수 있도록 구성되어 있다[1].

그러나 PLC 프로그래밍 언어는 나라마다 컴퓨터 기종마다 서로 상이하고 표준화가 되어있지 않기 때문에 오픈화와 분산화 시대에 맞지 않다는 문제점이 있다. 따라서 1993년 유럽을 중심으로 PLC 프로그램 언어의 표준활동이 추진되었다. 이 표준규격이 1992년에 승인되고 1993년에 문서로서 발행된 IEC의 TC65/SC65B/WG7/TF3에서 약 10년에 걸쳐 검토한 IEC 1131-3이다[2].

IEC1131-3에서는 다음과 같은 5개의 언어를 규정하여 처리하고 있다. 즉, 텍스트계의 언어로 IL(Instruction List), ST(Structured Text)가 있고, 그래픽계의 언어로는 LD(Ladder Diagram)과 FBD(Function Block Diagram)이 정의되고 있으며, 중요한 공통요소로 SFC(Sequential Function Chart)가 있다[3,4]. IL은 독일 비롯한 유럽에서 많이 쓰이며, FBD는 프로세스 제어의 신호 플로를 수반하는 애플리케이션용 회로도와 비슷한 형태의 언어이다. ST는 리얼타임 애플리케이션으로 개발된 프로그래밍과 유사하여 복잡한 펑션 블록의 정의에 효과적으로 사용된다. LD는 미국의 사다리형에서 기원한 언어인데, 일본이나 캐나다 등지에서 사용되고 있으며 입출력을 조합하여 프로그래밍을 한다. 우리나라에서도 90% 이상의 기업들에서 LD를 사용하고 있다. 이렇게 정의된 5개의 언어들은 어느 나라나 지역에 편중되지 않고 전통적인 PLC용 언어를 지원하므로 프로그래머의 능력에 따라 자유롭게 실제의 애플리케이션으로 개발할 수 있다.

* 책임저자(Corresponding Author)
논문접수 : 2004. 9. 1., 채택확정 : 2004. 11. 13.
조영임 : 수원대학교 컴퓨터학과(ycho@suwon.ac.kr)

이와 같이 1970년대초 릴레이를 대체시키고자 개발되었던 PLC는 이후 20-30년간 눈부신 성장을 거듭해 왔지만 80년대 초반 PC의 등장과 함께 PC-based control(soft logic 또는 software PLC라고 함)이 등장하게 되었다[1-5]. 즉, 지금까지 산업현장에서의 제어는 PLC만을 사용하여 왔으나 1990년대 초 미국 GM사가 처음으로 PC기반으로 자동차 라인에 적용한 사례를 시점으로 현재는 미국을 비롯한 유럽 각지까지 PLC에서 PC기반 제어로의 전이가 일어나고 있다.

그러나 국내에서는 이 기술에 대한 인지도 및 적용률이 미진하여 세계적인 변화에 편승하지 못하고 있는 실정이다. 우리나라는 세계에서 장비출하 댓수가 1위인 나라임에도 불구하고 장비에 들어가는 제어기의 90% 이상을 일본에서 수입한 PLC 제어에 의존하고 있는 형편이다. 그러나 앞으로는 PC 중심의 반도체 장비 중심의 제어가 주류를 이루고 있으므로 향후 PC 중심의 자동 제어방법이 중요하게 대두될 것이다. 이렇게 되면 반도체 장비에 하나의 패키지 형태도 제공함으로써 PC기반에서도 제어가 가능하게 될 것이다.

개방화, 오픈화 플랫폼 경향에 따라 기존 PLC 하드웨어 공급자에게 의존하는 방식은 점차 PC-based control을 이용하여 사용자 자신의 프로세스에 맞는 자원을 선택할 수 있도록 윈도우즈 환경 하에서 보다 쉽고 빠른 개발시간과 안정된 관리 및 유연한 확장성을 필요로 하게 되었다. 그동안 산업현장에서의 PC는 데이터 관리나 단지 HMI(Human Machine Interface)를 위한 용도로 인지되어 왔었으나 이제 산업현장에서도 PC의 빠른 발전속도와 뛰어난 성능을 접목시켜 직접 제어하는 시대가 도래하고 있으므로 이 연구는 이러한 시점에서 매우 중요한 의미를 갖는다. 즉, PC-based control은 하드웨어적 PLC의 여러 단점들을 보완하고 뛰어난 성능과 유연성으로 PLC를 대체할 차세대의 솔루션으로 인정받고 있다.

그러나 IEC1131-3이라는 표준언어 제정에도 불구하고 여전히 PLC 프로그래밍 언어는 일반인이 사용하기에는 매우

어려운 언어이며 웹 환경에서 범용성을 갖기 어려운 언어이다. 또한 전문가라 해도 프로그램시 논리오류를 찾기는 매우 어렵다는 문제점을 갖는다. 이 문제를 해결하기위한 선행 연구 결과인 ISPLC(Intelligent Agent System based Software Programmable Logic Controller)[6]는 국제 PLC 표준 언어로 제정된 5가지 언어[8] 중 국내에서 90% 이상 사용하고 있는 LD(Ladder Diagram)언어에 대한 표준규격을 연구하고, 이것을 중간코드인 IL(Instruction List)언어로 변환하고 기존 상용화된 편집기(Visual C++)에서 활용 가능한 표준 C 코드로 변환하는 기술이었다. 그러나 ISPLC는 LD언어에 대한 변환만을 시도한 것이므로 다른 언어들에 대한 변환이 이루어지지 않았으며 변환되는 언어들의 형태도 매우 제한적이고 일반적이지 못한 단점을 갖는다.

따라서 본 논문에서는 IEC1131-3 표준언어 중 LD언어 뿐만 아니라 FBD(Function Block Diagram), SFC(Sequential Function Chart), ST(Structured Text), IL(Instruction List)언어들이 중간코드인 IL언어로 변환하고 이를 상용화된 편집기에서 활용 가능한 표준 C코드로 변환함으로써 고급언어에 익숙한 일반인이 사용할 수 있도록 효율성을 높이기 위한 지능적 멀티에이전트 기반의 통합 시스템 IMPLC(Intelligent Multi Agent System based Software Programmable Logic Controller)를 연구 개발하고자 한다. IL 중간코드로 변환하는 이유는 모든 PLC 언어들의 단일 형태화 시킴으로써 코드간 상호관련성을 높이기 위한 것이며, C언어로 변환하는 이유는 C코드의 재활용성을 높이기 위해서이다.

IMPLC에서는 4가지 언어(LD, FBD, ST, SFC)에서 IL로 변환되는 과정에서 1차 오류 검색을 하고 C로 변환하는 과정에서 2차 오류를 검색하여 발생 가능한 오류의 수를 현저히 줄이고 C에서 논리오류 검출기능을 할 수 있어 매우 효율적이다. GUI기반 인터페이스를 제공하고 멀티 에이전트에 의해서 사용자가 수행하는 모든 행동양식들을 에이전트들을 통해서 핸들링 함으로써 초보자는 물론 PLC에 익숙한 사용자들에게도 효율적인 프로그래밍을 할 수 있도록 하는 플랫폼을 제공하고자 한다. IMPLC를 실제 산업현장에서 사용되는 3단 컨베이어 벨트에 적용하여 효율적인 제어가 되도록 사용자 GUI기반 웹 환경을 구축하고, 프로그래밍 오류 검색 뿐 아니라 프로그래밍 시간을 매우 단축시켜줄 수 있도록 하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 국내의 softPLC현황을 분석하여 IMPLC의 필요성을 제시하고 3장에서는 IMPLC를 제안하고 4장에서는 실제로 시스템에 적용한 사례를 분석하고 5장에서는 결론을 내리고자 한다.

II. 국내외 softPLC 현황 분석

국외 산업현장의 발전을 위한 기존에 개발된 PC-based control 제품들 중[7,8] 상업적 시스템 툴로써 이미 개발된 KW system이 있는데 사용자가 툴에 대한 친숙도를 고려하여 비주얼한 인터페이스를 제공해주는 시스템이다. 또한 embedded super PLCs[9,10]가 있는데, 이 시스템은 LD 프로그래밍을 기반으로 Basic 언어와 통합하여 단일 비트와 디지털 I/O를 핸들링하는 데이터를 프로세싱하는 PC 기반의 시스템으로 윈도우, 도스, 브라우저 등의 버전에 따라 달리 프로그래밍 한다.

국내에는 산업용으로 아직 개발된 사례가 없다. 그 이유는 국내 시장이 외국에 비해 비교적 작고 중소기업에서 투자하여 연구하기에는 인력 및 기술이 부족하고 수지타산이 맞지 않기 때문인 것으로 분석된다.

그러나 (주)리얼게인에서 교육용 PLC 언어 학습 패키지(realPLC)를 개발하여 보급하고 있는데, PLC소프트웨어만으로 다양한 언어를 학습할 수 있고, 다양한 모니터링 및 애니메이션을 할 수 있어 적은 비용으로 최상의 PLC 학습을 할 수 있는 특징을 갖는다. 특히 여러 종류의 애니메이션 시스템을 제공하여 대상 시스템 없이도 PC내에서 프로그램을 검증할 수 있고, LD 뿐만 아니라, SFC, IL, FBD의 모든 국제 표준 PLC 언어에 대한 학습을 할 수 있도록 구성되어 있어 PLC 언어에 대한 체계적인 학습을 할 수 있다. 그러나 이 제품은 교육용만으로 국한되어 있어 산업용으로는 활용하기 어렵다[11]. 리얼게인 시스템은 KW system과 매우 유사하나, 주된 차이점은 KW system과 달리 표준언어들 중 SFC언어를 기반으로 프로그래밍을 하며, 저 가격으로 사용할 수 있는 시스템이다. 이는 애니메이션과 사용자가 쉽게 실험할 수 있는 테스트베드를 갖고 있다.

국내에서는 LG 산전과 삼성에서 개발 사용되고 있는 PLC 시스템들이 있다. 삼성에서 윈도우용 WinGPC, LG산전에서는 master-K를 개발하였으나 자체 PLC용 장비에 대한 인터페이스만을 제공하고 있으므로 범용성이 부족하다.

다음 표 1은 국외의 대표적인 softPLC인 KW system과 국내의 리얼게인(real gain) 시스템의 인터페이스 및 장단점을 비교 분석하였다. 두 시스템 모두 IEC1131-3에서 정한 표준언어들의 인터페이스는 제공하나 표준 언어들 간의 상호 호환성이 부족하고 고급 언어로의 코드 변환은 제공하고 있지 않고 있다.

지금까지 살펴본 국내외 개발 시스템은 몇 가지 문제점이 있다. 첫째, PLC 기반 언어로 프로그래밍을 하는 동안의 오류(특히 논리오류) 분석, 해석, 처리기능이 부족하다. 둘째 오픈 소스가 아닌 블랙 보드로서의 결과만이 확인 가능하다. 셋째, IEC1131-3에서 제공하는 표준 언어들 간의 호환성이 부족하다. 넷째, 제한적인 시뮬레이션 기능을 가지고 있

표 1. KW system과 realgain 비교.

Table 1. KW system vs. realgain.

Compare Advantage with Disadvantage between KW and Real Gain System			
KW System		Real Gain System	
[Point of Tool Characterisic]			
Advantage	Disadvantage	Advantage	Disadvantage
Support 5L	Optional	Learning 4L	LD based
Combination	Hard Convert	Monitoring	Only Simulation
MultiProg Fox	WinPLC ProConOS	Animation	Restricted Demo
Window based		Easy Experiment	Entire Module
Evaluation DL		Low Cost	
MultiTasking		Download	
[Point of User and Customer]			
Advantage		Disadvantage	
User Friendly (OS)		Manualized Option	
Drag and Drop		Black Board	
Monitoring Simulation		Hard to Error search	
Select each language		User Handling (Expert)	
		High efficient PC based	
		Limited Simulation	

으므로 일반 사용자들의 기대를 만족시키기에는 부족한 점이 많다. 다섯째, 자체개발한 PLC용 언어만을 지원한다는 것이다.

III. 지능적 에이전트기반 softPLC 설계 및 구현

1. IMPLC의 개요

본 논문에서는 에이전트[12,13] 기반의 PLC 소프트웨어 시스템은 PLC 프로그래밍이 가능한 툴 내에서 사용자를 대신하여 지적 대리인으로서 사용자가 원하고, 사용자에게 적합한 컴포넌트를 제공해 줌으로써, 그 결과를 사용자가 원하는 형태로 필터링하여 최적의 코드를 생성해 주고, 요구되는 사항들을 고려하여 자동적이고 자율적이며 전문적인 통합형 PC 기반의 지능형 PLC 소프트웨어 에이전트 시스템인 IMPLC(Intelligent Soft Multiagent based Programmable Logic Controller)을 개발하고자 한다.

이 시스템에서는 IEC1131-3에서 제공하는 표준 4개의 언어를 나머지 한 언어인 IL로 변환하여 언어간 호환성을 갖게 하여 표준 C 컴파일러에서 컴파일 함으로써 사용자가 발생하는 논리오류를 수정하여 최적화된 환경에서 PC기반으로 제어하도록 한다.

본 논문에서 제안하는 PC기반의 지능형 softPLC 편집기인 IMPLC 시스템의 구성요소는 크게 에이전트 그룹(agent group), 컴포넌트 데이터 스토리지(component data storage), 규칙 베이스 구조(rule base architecture) 3가지로 구성되어 있다.

에이전트 그룹에는 4개의 에이전트가 있는데 ① 사용자 에이전트(User Agent : UA), ② 스캐닝 에이전트(Scanning Agent : SA), ③ 컨트롤 에이전트(Control Agent : CA), ④ 오류 체크 에이전트(Error Check Agent : EA)가 있다. 컴포넌트 데이터 스토리지는 사용자가 많이 사용하는 프로그래밍 언어의 컴포넌트를 추천하고 관리하는 기능을 한다. 규칙 베이스 구조에는 5가지의 모듈이 있는데 ① IL언어로 변환하는 모듈 ② IL언어가 C언어로 변환하는 모듈 ③ 오류 체크 모듈 ④ 메모리 매핑 모듈 ⑤ 사용자 학습파일 모듈로 구성되어 있다.

IMPLC의 에디터 화면은 다음 그림 1과 같다. 구성한 IMPLC 에디터 화면의 전체 개념은 IEC1131-3의 표준 언어(LD, SFC, FBD, ST)들이 중간 코드 형태인 IL언어로 변환하고 이 IL언어를 고급(C)언어로 변환해서 컴파일 하는 것으로 윈도우 환경에서 실행 가능하도록 구성한다. 에디터는 GUI기반으로 되어 있으며 에이전트들에 의해 지능적으로 오류 수정 및 코드 변환이 수행되는 점이 특징이다.

2. IMPLC의 세부모듈

IMPLC의 전체적인 구조도는 다음 그림 2와 같다.

본 논문에서 IMPLC 시스템의 각 모듈의 역할 및 특징은 다음과 같다.

2.1 에이전트 그룹(agent group) : IMPLC 시스템 내에서 활동하는 agent들의 집합체를 말한다. 본 논문의 IMPLC에 있는 에이전트 그룹에는 4가지의 에이전트로 구성되어 있다. 각 에이전트의 역할을 설명하면 다음과 같다.

- 스캐닝 에이전트(scanning agent) : 사용자와 컨트롤

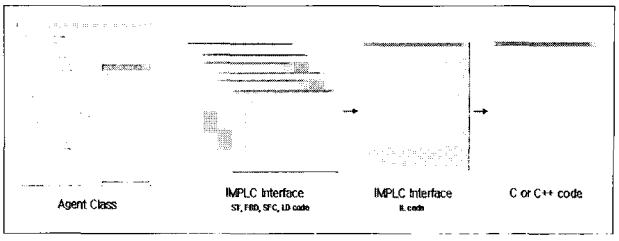


그림 1. IMPLC의 지능적 에디터.

Fig. 1. The intelligent editor of IMPLC.

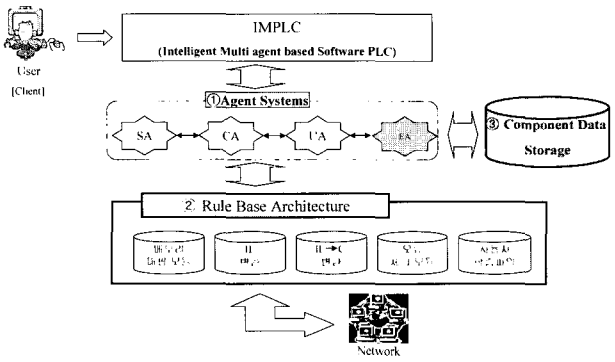


그림 2. IMPLC 전체 시스템 구조.

Fig. 2. The overview of IMPLC.

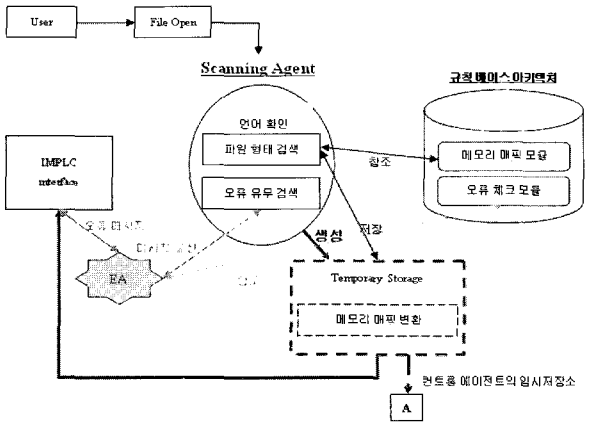


그림 3. 스캐닝 에이전트.

Fig. 3. Scanning agent.

에이전트와의 상호 작용을 위한 인터페이스로서의 역할을 하고 사용자가 파일을 불러올 때 수행되는 에이전트이다. 스캐닝 에이전트의 수행 및 행동양식에 대한 그림은 위의 그림 3과 같다.

스캐닝 에이전트의 행동 절차를 보면 에디터에서 해당 파일의 유무를 파악한 후, 파일의 확장자를 읽어 들여 작성되어 있는 언어(예, LD, FBD 등)를 확인한다. 그런 다음 규칙베이스 구조의 메모리 매핑 모듈을 참조하여 명령어(operator), 변수(operand)에 맞게 변환해서 임시저장소에 저장하게 되고 EA(오류체크 에이전트)와 메시지 통신을 통해 논리나 문법오류 유무를 검색한 후, 오류 발생시 오류 메시지를 사용자에게 알려주는 기능을 담당한다. 만약 오류가

없으면 다음 프로그램을 진행한다.

• 사용자 에이전트(user agent) : 스캐닝 에이전트와 같이 사용자와 컨트롤 에이전트와의 상호 작용을 위한 인터페이스로서의 역할을 하는데 사용자가 IMPLC에서 편집하게 되면 컴포넌트 데이터 스토리지를 참조하여 수행되는 에이전트이다. 사용자 에이전트의 수행 및 행동양식에 대한 그림은 다음 그림 4와 같다.

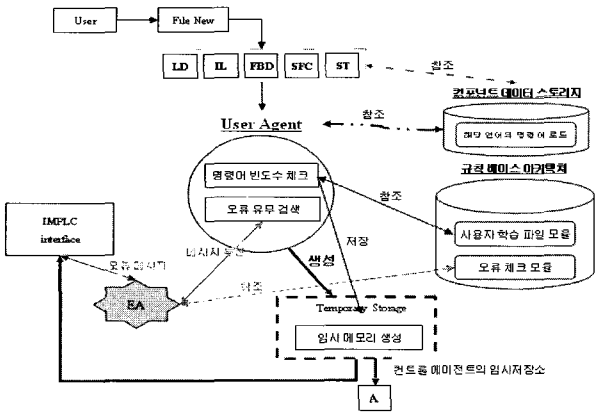


그림 4. 사용자 에이전트.

Fig. 4. User agent.

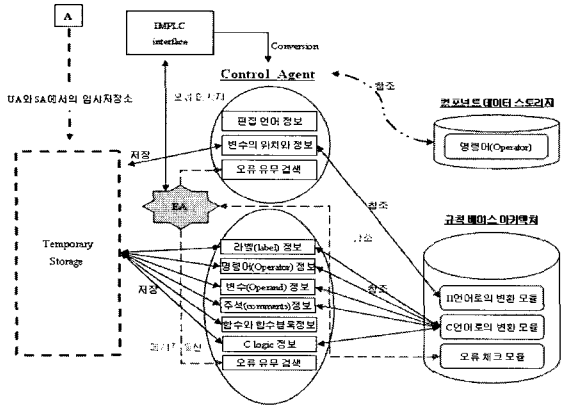


그림 5. 컨트롤 에이전트.

Fig. 5. Control agent.

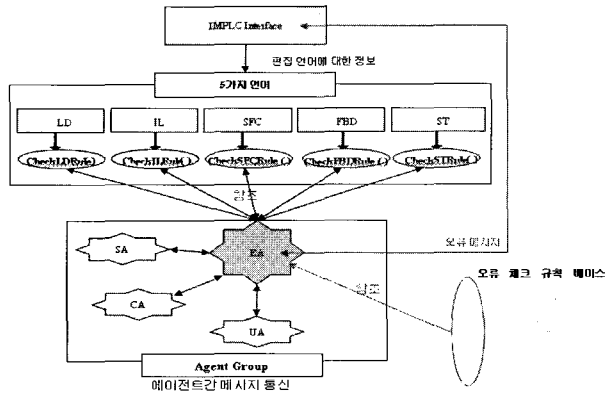


그림 6. 오류 체크 에이전트.

Fig. 6. Error check agent.

사용자 에이전트의 행동 절차를 보면, 사용자가 IMPLC에서 언어를 선택 시 컴포넌트 데이터 스토리지를 참조하여 명령어를 로드하게 되고, 사용자가 편집 하게 되면 실시간적으로 임시 저장소에 저장되어진다. 사용자 에이전트는 규칙베이스 구조의 사용자 학습 파일 모듈을 참조하여 사용자 성향에 알맞은 컴포넌트를 추천하며 EA와 메시지 통신을 통해서 오류 유무를 검색하여 오류메시지를 사용자에게 알려주는 기능을 담당한다.

• 컨트롤 에이전트(control agent) : 스캐닝 에이전트와 사용자 에이전트와 상호 작용을 통해서 IEC1131-3의 표준 언어 규정에 맞게 각각의 언어를 중간 코드 형태인 IL언어로 변환하고, 변환된 IL언어를 C언어로 변환할 때 규칙베이스 구조를 참조하여 수행되는 에이전트이다. 컨트롤 에이전트의 수행 및 행동양식에 대한 그림은 다음 그림 5와 같다.

컨트롤 에이전트의 행동 절차를 보면, 사용자가 편집하는 언어에 대한 정보와 해당언어의 명령어와 변수의 위치 정보를 임시 저장소에 저장하게 되고, 저장되어 있는 정보를 가지고 규칙 베이스 구조의 IL언어로의 변환 모듈을 참조하여 IL로 변환하는 것에 1차적으로 관여한다. IL언어를 C언어로 변환하기 위하여 IL정보(라벨 정보, 명령어 정보, 변수 정보, 주석 정보, 함수와 함수 블록 정보)를 검색한 후 규칙베이스 구조의 C언어로의 변환 모듈을 참조하여 C언어로 변환하게 되는 과정을 2차적으로 관여하는 것을 담당한다.

• 오류 체크 에이전트(error check agent) : 오류 체크 에이전트의 수행 및 행동양식에 대한 그림은 다음 그림 6과 같다.

사용자가 편집하거나 파일을 불러온 경우, 각 언어들이 IEC1131-3의 표준 언어 규정에 맞는지 규칙 베이스 구조 오류 체크 모듈을 참조하여 체크하게 되는데 내부적으로 5가지 언어들의 표준 언어 규정들을 클래스화하여 정의하였다. 오류 체크 에이전트의 가장 큰 특징은 스캐닝 에이전트와 사용자 에이전트 그리고 컨트롤 에이전트와 항상 메시지 통신을 해서 오류 유무에 대한 것을 사용자에게 알려주는 기능을 담당한다.

2.2 규칙 베이스 구조(rule base architecture) : IMPLC의 주요 언어변환기능을 담당한다.

• 메모리 매핑 모듈(memory mapping module) : IEC1131-3의 표준에 근거하여 외부에서 작업을 한 파일을 IMPLC에서 읽어 들여 저장된 모든 파일들을 IEC1131-3에 근거하여 자동 형 변환 해주는 메모리 매핑 규칙을 모듈화 한 것이다. 이것이 필요한 이유는 IMPLC에서 편집한 파일은 물론, 외부 다른 편집기에서 편집한 PLC 파일도 사용할 수 있도록 하기위해서이다. 메모리 매핑을 하기 위해서는 자동으로 변환해주는 규칙을 갖는 모듈이 필요한데 이것을 본 논문에서는 메모리 자동변환이라 정의하였다. 메모리 매핑에서는 그래픽 (LD, FBD, SFC)언어는 각 회사마다 그래픽 저장방식이 상이하기 때문에 변환이 불가능해서 제외하였고 텍스트 기반 언어인 (IL, ST)언어만을 변환하도록 하였다.

• IL언어로의 변환 모듈(IL conversion module) : 각각의 언어를 IL언어로 변환하는 과정에 관여하는 모듈로서 다음 그림 7과 같이 스캐닝 변환 알고리즘을 통해서 IL로 변환하게 된다.

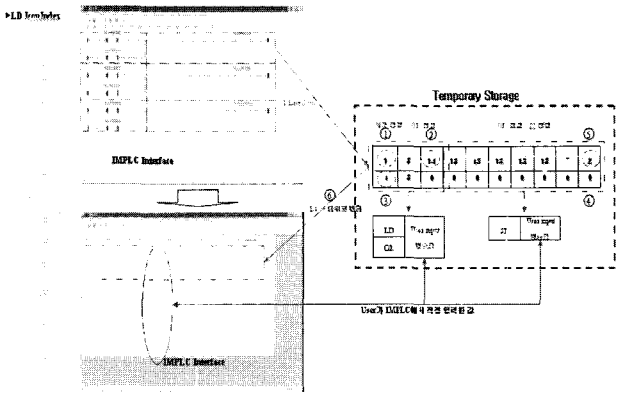


그림 7. LD언어의 IL로의 변환 과정.
Fig. 7. The conversion process of LD into IL.

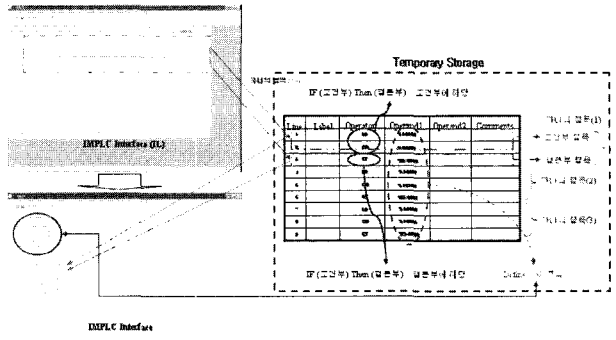


그림 8. IL언어의 C로의 변환 과정.
Fig. 8. The conversion process of IL into C.

위의 그림 7은 LD언어의 예이다. 접점의 위치 정보를 알기 위해 접점을 시작접점과 OR접점, 끝 접점이 세 가지로 구분하게 되고 변수의 위치 및 정보, 각 언어들의 컴포넌트 정보를 얻기 위해 컴포넌트 데이터 스토리지를 참조하게 된다. 변환 스캐닝 알고리즘을 보면, 첫 번째 라인의 첫 번째 시작접점을 검색 → 첫 번째 라인의 시작 접점부터 스캐닝을 시작하여 OR접점이나 끝접점을 검색 → OR접점을 찾게 되면 다음 라인으로 이동하여 시작 접점을 검색 → 두 번째 라인의 시작 접점부터 스캐닝을 시작하여 OR접점이나 끝 접점을 검색한다. 이때 OR접점을 찾게 되면 다음 라인으로 이동해서 시작 접점을 검색하게 되고 접점을 발견하지 못할 시에는 전(첫 번째)라인으로 이동 → 첫 번째 라인의 OR접점 이후부터 또 다른 OR접점이나 끝 접점을 검색하게 된다. 위의 과정들을 1Line Scan(하나의 스텝)이라 하여 이를 블록 단위의 IL로 변환 하게 되고 사용자가 IMPLC상에서 변수의 정보를 입력하면 IL언어의 operand 부분에 저장되게 된다. 이러한 일련의 과정을 통해서 IL언어로 변환이 가능하게 되는 것이다. 본 논문에서는 프로그래밍 언어별로 접점으로 나누어서 각각 인덱스를 부여하여 컴포넌트 데이터 스토리지에 저장되어 있으며, 언어변환시 사용한다.

• C언어로의 변환 모듈(C conversion module) : IL언어를 C언어로 변환하는 과정에 관여하는 모듈로서 다음 그림 8과 같이 스캐닝 변환 알고리즘을 통해 C로 변환하게 된다.

IL언어로 변환된 것을 C언어로 변환하기 위한 모듈로서, C로 변환하기 위해서는 IL언어의 라벨(label)정보, 명령어(operator)정보, 변수(operator)정보, 주석(comments)정보가 필요하다. 이 정보를 가지고 C언어의 스캐닝 변환 알고리즘을 통해서 C언어로 변환하게 되는데 라벨, 명령어, 변수, 주석에 대한 정보들은 IEC1131-3의 정의 되어있는 문법들을 클래스화하여 정의하였다. 변환 스캐닝 알고리즘을 보면, IMPLC에서 IL언어로 변환된 에디터를 스캐닝하여 라벨, 명령어, 변수, 주석의 정보를 스트링형의 배열로 임시저장소에 저장하게 되고 명령어 부분과 operand부분으로 나누어서 명령어 부분은 처음라인에서 LD라는 명령어를 검색 → 다음 라인의 명령어를 검색 : OR(N) 혹은 AND(N)까지는 IF-Then의 조건부에 해당된다는 정보를 컨트롤 에이전트가 기억 → 다음 라인부터 ST(N)라는 명령어를 검색 : IF-Then의 결론부에 해당된다는 정보를 컨트롤 에이전트가 기억한다. 그런 다음 C언어로의 변환 규칙을 참조하여 명령어 부분에 해당되는 것들을 C 언어의 연산자 부분과 패턴 매칭하여 C 로직에 맞게 변환하고, operand 부분은 C 코드의 "#define"문으로 정의한다. 그런 다음 C언어 문법에 맞는 조합과정을 통해서 IL언어가 C언어로 변환하게 된다.

• 오류 체크 모듈 (syntax & logic error check module) : 오류 체크 에이전트가 상주하여 사용자가 작성하는 모든 편집작업을 항상 스캐닝한다. 만약 오류가 발생되면 EA(오류 체크 에이전트)가 오류 규칙 베이스를 참조하여 오류 메시지를 CA, UA, SA와 메시지 통신을 통해서 사용자에게 제안해주는 모듈이다. 오류 규칙 베이스는 IEC1131-3의 표준 규정에 대한 것을 클래스화하여 정의되어 있는데, 이 클래스를 오류 체크 에이전트가 주기적으로 스캐닝하게 되는 것이다.

• 사용자 학습파일 모듈(user profile module) : 사용자의 프로그램 편집과정을 패턴화하여 개인화 코드로 컴포넌트 또는 클래스 파일 형태를 빈도수의 증가로 저장한다. 이 모듈은 사용자가 인터페이스 상에서 이전 프로그램의 편집과정에 대한 컴포넌트 및 클래스 패턴의 히스토리를 가지고서 제안을 해주는 모듈이다.

2.3 컴포넌트 데이터 스토리지 (component data storage) : IMPLC시스템에서 각각의 5가지 표준언어들의 명령어와 오퍼랜드등 문법 구성요소를 갖고 있는 모듈로서 UA에게 사용자가 원하는 데이터 즉, 컴포넌트들을 제공하여 사용자로 하여금 인터페이스에서 편집하는데 도움을 주는 모듈이다. IMPLC 시스템은 사용자가 인터페이스를 통해 편집하거나 파일을 불러오는 경우에 사용자 에이전트와 스캐닝에이전트가 사용자의 질의에 맞게 규칙베이스와 컴포넌트 데이터 스토리지를 참조하여 사용자에게 원하는 데이터를 전달하고 각각의 언어를 IL로 변환하고 다시 C언어로 변환하는데 컨트롤 에이전트가 관여함으로써 사용자가 프로그래밍을 쉽게 할 수 있도록 도와주는 것이고 최종 C로 변환된 코드를 가지고 재사용이 가능하게 되는 것이다.

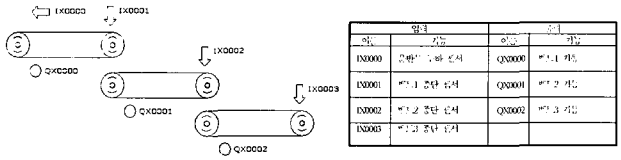
IV. IMPLC 시스템의 시뮬레이션 및 성능평가

1. 3단 컨베이어 벨트 시뮬레이션

본 논문에서는 IMPLC의 적용 시스템으로써 산업현장에

서 사용되고 있는 3단 컨베이어 벨트 제어 시스템에 적용하여 시뮬레이션 하고자 한다. 이 시스템은 운반물들의 이동 및 적재시 사용되는 것으로 입출력 리스트를 보면 다음 그림 9와 같다.

본 논문에서 개발한 3단 컨베이어 벨트 시스템의 모델링 시뮬레이션 구성도는 다음 그림 10과 같다.



(a) 3 steps conveyer belt (b) I/O list

그림 9. 3단 컨베이어 벨트.

Fig. 9. 3 steps conveyer belt.

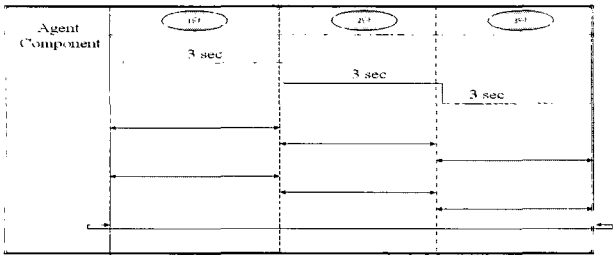


그림 10. 3단 컨베이어 벨트 시스템 모델링.

Fig. 10. The system modelling of 3 steps conveyer belt.

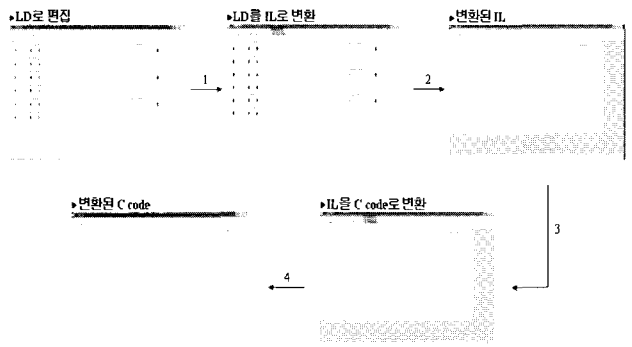


그림 11. IMPLC상에서의 3단 컨베이어 벨트 처리과정.

Fig. 11. The process of 3 steps conveyer belt on IMPLC.

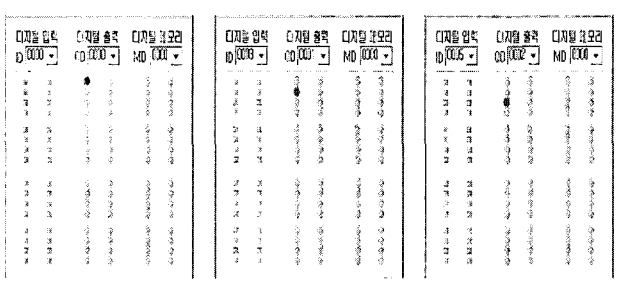


그림 12. 3단 컨베이어 벨트 모니터링.

Fig. 12. The monitoring of 3 steps conveyer belt.

그림 10에 적색 3가지는 3단 컨베이어 벨트의 센서를 나타내고 있다. 각각 1단, 2단, 3단의 타이머 설정시간은 3초로 하였다. 운반물 투하센서인 %IX0000이 운반물의 투하를 확인하면 1단 벨트인 %QX0000은 운반물이 3초 후에 종단센서인 %IX0001을 만나기 전까지 기동(ON)한다. 1단 벨트 종단센서를 만나면 %QX000은 정지한다. 운반물이 2단 벨트로 내려가 %QX0001이 기동(ON)하고 3초 후에 2단 벨트 종단세서인 %IX0002를 만나면 정지한다. 다음 운반물이 3단 벨트로 내려가 %QX0002이 기동(ON)하고 3초 후에 3단 벨트 종단센서인 %IX0003을 만나기 전까지 운반물이 기동된다[13].

실제적으로 IMPLC상에서 작성해서 실행되는 동작과정은 그림 11과 같다.

위 그림 11은 사용자가 LD로 작성하여 IL로 변환하고 다시 IL을 C code로 변환하는 과정을 3단 컨베이어 벨트를 예제로 실행하는 과정을 단계적으로 나타낸 그림이다. 이 실행과정의 결과를 모니터링하게 되면 그림 12와 같다.

2. IMPLC 성능평가

IMPLC의 기존 소프트웨어 PLC의 오류처리 알고리즘을 설명하면 IMPLC에서는 PLC 프로그래밍 언어(LD, FBD, ST, SFC) 접점에 따른 전문가 작성 프로그램의 패턴을 분류하여 사용자 에이전트가 규칙베이스를 구축하여 IL 코드로 변환한다. 사용자 에이전트는 사용자 데이터를 관리하고 사용자의 사용패턴을 학습한다. 컴포넌트 라이브러리 등을 제공하고 사용자에게 올바른 C코드를 추천해 준다. 이것을 위해 IL->C 로의 규칙적인 학습패턴을 유지하여 규칙베이스에 저장함으로써 가능하게 된다. 따라서 IMPLC는 사용자가 많이 사용하면 할수록 C코드로의 변환이 더 효율적으로 된다. 컨트롤 에이전트는 실제로 변환된 C 코드로부터 컴파일을 하고 오류나 패턴을 필터링하여 사용자에게 피이드백 함으로써 PLC 프로그램을 재작성할 수 있도록 상호 작용한다(그림 13).

IMPLC와 기존 소프트웨어 PLC의 논리오류 디버깅에 소요되는 시간을 측정하기 위해 3단 컨베이어 벨트에서 LD 프로그래밍 언어의 명령어 패턴수를 증가시키면서 실험을 해보았다. 실험결과, IMPLC는 논리오류 만큼의 프로그램내

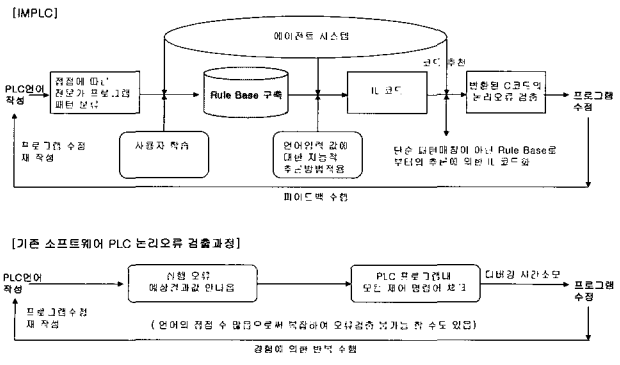
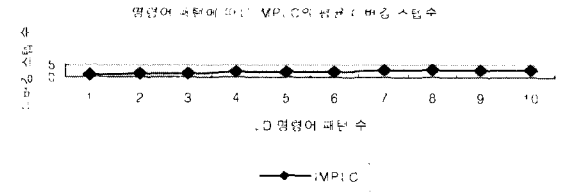
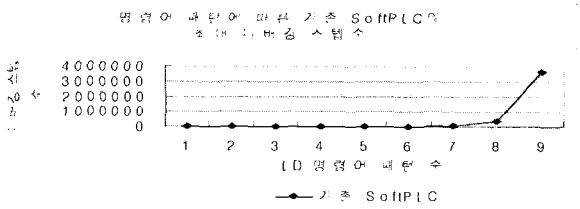


그림 13. IMPLC와 기존 소프트웨어 PLC 과정 비교.

Fig. 13. The process comparison between IMPLC and the conventional PLC.



(a) The debugging steps of IMPLC



(b) The debugging steps of the conventional PLC

그림 14. IMPLC와 기존 소프트웨어 PLC의 디버깅 스텝수 비교.
Fig. 14. The comparison of debugging steps between IMPLC and the conventional PLC.

의 특정 부분의 제어관련 명령어들을 확인하고 메시지 흐름의 순서에 따라 원인을 순서대로 규명해야 되므로 프로그램이 복잡해지더라도 평균적으로 $\sqrt{n}$(n: LD명령어패턴의갯수) 정도의 디버깅 스텝수를 갖는 경향이 있음을 알 수 있었다. 그러나 기존 소프트웨어 PLC는 LD 프로그램 작성 후 실행이 안되었을 경우 디버깅하기 위해서는 프로그램내의 모든 제어관련 컴포넌트들을 확인하고 제어흐름의 순서에 따라 모든 명령어들의 상관관계를 분석하면서 오류 원인을 차례로 규명해야 하므로 $nPn = n!$의 디버깅 스텝수를 가짐을 알 수 있었다. 따라서 LD 접점수가 많으면 많을수록 오류 검출 가능성은 줄어들게 된다. 이것을 그래프로 나타내면 지수함수 비율로 디버깅 스텝수가 증가함을 알 수 있었다 (그림 14). 따라서 IMPLC를 사용할 경우 기존 소프트웨어 PLC보다 논리오류를 현저하게 줄여줌을 알 수 있었다.

V. 결론

본 논문에서는 지능형 softPLC인 IMPLC를 구현하였다. 또한 IMPLC를 각각의 언어, C 또는 Visual C++ 매핑과 더불어 에이전트를 기반으로 한 3단 컨베이어벨트 시스템에 적용함으로써 IMPLC에서 각각의 언어가 IL로 변환되고 다시 C 코드로 변환시킴으로써 표준 C컴파일러 상에서 지능적 에이전트에 의한 논리 오류를 체크할 수 있는 장점이 있다. 이러한 장점은 각각의 언어 상에서 논리오류를 찾아내는 것 보다는 C 컴파일러 상에서 에이전트에 의한 논리 오류의 수정기능을 갖게 되므로 매우 효율적이다.

그러나 IMPLC는 IEC1131-3 표준언어가 IL에서, IL에서 다시 C로의 코드 변환을 통합된 환경에서 제공하나 역관계로 코드변환이 이루어지지 않으므로, 만약 논리 오류 발견 시 실제 사용자는 변환된 코드로부터 다시 표준언어 프로그램을 수정해야 하므로 개발 툴의 활용능력이 우수해야 한다는 단점을 갖는다. 즉, IMPLC는 완전 자동화가 아닌 반 자동화 형식으로 사용자에게 코드변환 관계를 설명한다. 또한 IMPLC에서 사용되는 에이전트의 기능이 다소 제약적이므로 보다 효율적인 에이전트 알고리즘이 개발되어야 한다.

따라서 ISPLC의 제약 사항들을 해결하고 보다 지능적인 시스템이 되게 하기 위해서는 숙련된 개발 툴과 알고리즘을 적용하여 보다 효율적이고 구조적인 환경과 사용자의 수행능력, 전문가 시스템을 위한 노력이 많이 필요하다. 또한 실제 산업용에서는 IL을 중간코드로 한 표준 언어들과 IL, IL과 표준 언어들간의 양방향 변환 모듈의 개발이 필요하다. 향후 PLC 실행 엔진과 네트워크 통신을 통해 실제적으로 적용하게 되면 프로그램상에 문제가 없는 논리 오류들의 검출기능이 더욱더 강화 될 것으로 생각되고 IMPLC에 추가적으로 네트워크 통신 기능을 첨부하여 실제적으로 산업현장에서 사용할 수 있는 통합 시스템 개발이 필요하다.

참고문헌

[1] PLC 이론과 실습, 삼성전자 사내교육 자료.
[2] Norme Internationale International Standard, CEI IEC 1131-3, Premiere edition, First edition, 1993.
[3] 김정렬외 3인, PLC활용과 모니터링(KGL-WIN용), 테크미디어, 2002.
[4] 임윤식외 2인, 시퀀스 및 PLC제어, 북두 출판사, 2003.
[5] 원태현외 6인, PLC 제어기술, 제2판, 북두 출판사, 2001.
[6] 조영임, 심재홍, "ISPLC: 지능적인 에이전트 기반 소프트웨어 PLC", 한국 멀티미디어학회 추계 논문집, 제6권 제2호, pp. 557-560, 2003.11.21-22.
[7] www.angelfire.com/in/bsommer/softplc.html
[8] IsaGRAF user's guide, Version 2.1, CJ International, 1994
[9] http://www.intellution.co.kr
[10] http://www.deltaww.com
[11] 리얼게인 연구소, PLC실험 실습, 청문각, 2003.
[12] Russell and Norvig, *Artificial Intelligence a Modern Approach 2/E Chap 2*, Prentice Hall International Co., 1994.
[13] http://www.fipa.org/repository/managementspecs.html

조 영 임

1963년 8월 21일생. 1988년 고려대학교 전산과학과 졸업(학사). 1990년 고려대학교 전산과학과졸업(석사). 1994년 고려대학교 전산과학과졸업(박사). 1995년~1996년 삼성전자 멀티미디어 연구소 선임연구원. 1996년~2005년 2월 평택대학교 컴퓨터과학과 교수. 1999년~2000년 University of Massa- chusetts, Dept. of Computer Science, Post-doc. 2003년~현재 한국퍼지 및 지능시스템학회 이사. 한국여성정보인협회 운영위원. 2004년~현재 한국전자상거래학회 이사(학술위원장). 2005년 3월~현재 수원대학교 컴퓨터학과 교수. 관심분야는 뉴로퍼지시스템, 지능형 에이전트, 제어시스템.