Proceedings of KFIS Spring Conference 2005 Volume 15, Number 1

SPC-10기반 소프트웨어 PLC 컴파일러 개발

Development of an Software Programmable Logic Control Compiler based on SPC-10

조영임
수원대학교 정보공학대학 컴퓨터학과

Young Im Cho
Dept. of Computer Science, College of Information Technology
The University of Suwon
E-mail : ycho@suwon.ac.kr

요 약

본 논문에서는 국내에서 상용화되고 있는 삼성전자의 SPC-10의 IEC1131-3 표준 언어(LD, SFC, FBD, ST)가 상호 호환성을 갖도록 목적 언어인 IL로 변환하여 컴파일될 수 있는 IL 컴파일러를 개발하고자 한다. 개발하려는 IL 컴파일러는 IL언어의 활용성을 매우 높여주며, 실제 네트워크 디바이스에 다운로드 하여 사용할 수 있는 IL 언어를 생성하고 실행시키는 점이 특징이다. 이 시스템은 SPC-10에서 많이 사용되는 언어를 패턴인식에 의해 클러스터링하여 자동적으로 IL언어로 변환이 되며, 컴파일에 의해 PLC 프로그램의 동작이 가능하다. 이 시스템에서는 또한 사용자가 발생할 수 있는 문법오류는 물론 논리오류를 지능적 에이전트에 의해 검색하여 수정함으로써 최적화된 환경에서 PC기반 제어가 가능하도록 해준다.

1. 서론

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 추진되었는데 이 표준규격이 IEC1131-3이다[1]. 이 표준규격은 산업용 컨트롤 프로그램의 표준화를 목적으로 개발된 유일한 국제 표준의 산업용 컨트롤 프로그래밍 언어이다. 1992년에 승인되고 1993년에 문서로서 발행된 이 표준규격은 IEC의 TC65/SC65B/WG7/TF3에서 약 10년에 걸쳐 검토한 결과이다.

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. 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[2,3,4,5].

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.

따라서 본 논문은 국제 PLC 표준언어로 제정된 5가지 언어 중 대미 수출은 물론 국내에서 90%이상을 사용하고 있는 PLC 언어인 LD언어에 대한 표준규격을 연구하고, 이것을 중간코드

인 IL(Instruction List)언어로 변환하기 위한 알고리즘과 IL 컴파일러를 개발하고자 한다. 본 논문에서는 이것을 위해 국내의 점유율 1위를 차지하고 있는 삼성전자에서 개발한 SPC-10을 기반으로 IL변환 시스템을 개발하고자 한다.

본 논문의 구성은 다음과 같다. 2장에서는 국내외 PC-based control의 현황과 삼성전자의 SPC-10을 분석하고 3장에서는 LD의 IL 변환알고리즘과 시스템을 설명하고 마지막으로 4장에서 결론 및 향후과제에 관해 설명하고자 한다.

2. SPC-10

삼성전자에서 개발한 SPC-10은 소형 단위기계 제어에 적합한 일체형의 초소형 PLC이며, 릴레이보다 경제적이며 기본 14점, 확장 56점까지 가능하다(그림 1 참조). SPC-10은 산업용 PLC에서 가장 국내 점유율이 높기 때문에 Il로의 변환 알고리즘 적용시 활용도가 매우 높다.

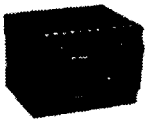


그림 1 SPC-10

또한 고속카운터 및 간이 펄스출력 기능까지 보유하고 있으며, 프로그래밍(소프트웨어 및 명령어)은 N70plus/ N700plus PLC와 호환성이 있다.

SPC-10의 프로그램용량은 2K 스탭까지 가능하다. 또한 다양한 특수기능 보유하고 있는데, 입력노이즈 및 채터링 제거용으로 외부입력 시정수 설정기능이 있고, 고속 카운터 기능이 있어서 2CH(24비트), 엔코더 접속이 가능하다. 또한 펄스출력 기능이 있는데 20Hz - 5KHz 펄스출력이 가능하고, 가감속 제어가 가능하다. 통신 네트워크 기능이 있는데, 일대일 통신이 가능하여 컴퓨터와 직접 접속(아답터 이용)이 가능하고, RS485/232 변환이 자유롭다. 또한 일대다 통신이 가능하여 Twisted Pair Cable로 최대 32대까지 PLC를 네트워크 접속이 가능(9,600bps, 1.2Km)하다. 프로그래밍 Tool 접속이 다양한데, Handy Loader (PGM-10, PGM-500) 사용하며, 사용자 S/W는 GPC (DOS용)와 WinGPC(Window용)가 있다.

3. LD에서 IL로의 변환 알고리즘

본 논문에서는 SPC-10 기반으로 LD에서 IL로의 변환 알고리즘을 제안하여 IL 컴파일이 되도록 하고자 한다.

다음 그림 2는 LD 심벌들의 패턴 분석을 통하여 나타낸 알고리즘 순서도이다. 그림 2의 변환 알고리즘 순서도를 설명하면 다음과 같다.

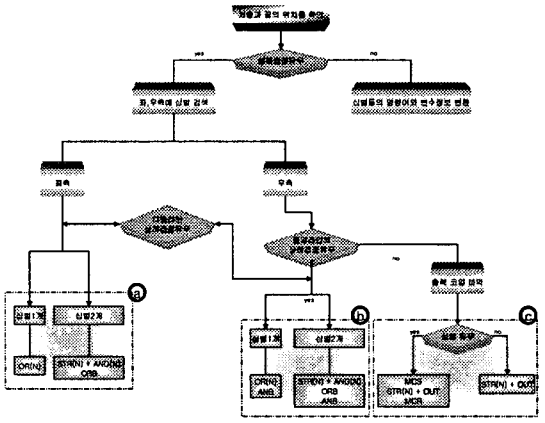


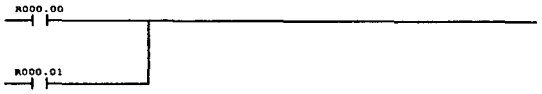
그림 2 본 논문에서 제안하는 LD에서 IL로의 변환 알고리즘

① 처음과 끝의 심벌들에 대한 위치좌표를 파악
② 상하접점이 있는지 파악
③ 상하 접점을 기준으로 좌·우측에 심벌을 검색(x좌표의 값)
④ 좌측에 있는 경우: 다음 라인과 연결된 상하접점이 있는지 검색(y좌표의 값)
④-1 상하접점이 없는 경우 : 심벌의 개수를 파악 -≫ **ⓐ 블록화**
- 심벌개수 1개 : STR(N) 심벌을 OR(N)으로 임시저장소에 변환 저장
- 심벌개수 2개 이상: OR(N)이 아닌 STR(N)과 심벌명령어를 임시저장소에 변환 저장
④-2 상하접점이 있는 경우: 상하접점의 좌·우측에 심벌을 검색 -> ④-1의 ⓐ 블록화 과정을 거쳐 임시 저장소에 변환 저장.
⑤ 우측에 있는 경우: 동일 라인의 상하접점파악.
⑤-1 동일 라인의 상하접점이 없는 경우: 출력코일 파악-> 상하 접점과 출력 코일들 사이에 심벌 유무를 파악. -≫ **ⓒ 블록화**
- 심벌이 있을 경우: MCS + [STR(N) + 심벌 명령어 + OUT] + MCR
- 심벌이 없을 경우: STR(N) + 심벌 명령어 + OUT
⑤-2 동일 라인의 상하접점이 있는 경우: 상하접점과 동일 라인의 상하 접점사이에 다음 라인으로 연결되는 상하접점이 있는지 파악. -≫ **ⓑ 블록화**
- 상하접점이 없는 경우: 상하접점과 동일 라인의 상하접점사이의 심벌을 검색
-> 심벌개수 1개: OR(N) + ANB로 임시저장소에 변환 저장
-> 심벌개수 2개 이상: OR(N)이 아닌 STR(N) + AND(N) + 심벌 명령어 ORB +ANB로 임시 저장소에 변환 저장
- 상하접점이 있는 경우: 동일 라인의 상하접점사이의 심벌을 검색
-> ⓑ 블록화 과정을 거쳐 임시 저장소에 변환 저장

ⓐ 블록화의 패턴 분석

① 좌측에 심벌이 존재할 경우 : ORB를 포함하는 경우

①-1 좌측에 심벌이 1개가 존재할 경우: STR(N)+OR(N)

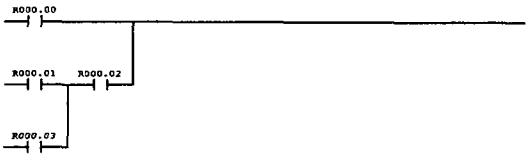


STR R000.00 OR R000.01

①-2 좌측에 심벌이 2개가 존재할 경우 : STR(N)+[STR(N) + AND] +ORB

STRR000.00 STR R000.01 AND R000.02 ORB

①-3 상하접점이 또 하나 존재할 경우 : STR(N)+[STR(N)+AND]+ORB+[A]+ORB



STRR000.00 STR R000.01 OR R000.03 AND R000.02 ORB

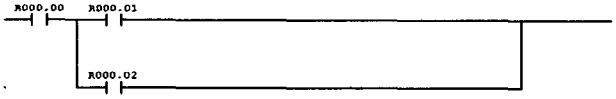
STRR000.00 STR R000.01 AND R000.02 ORB
STRR000.03 AND R000.04 ORB

ⓑ 블록화의 패턴 분석

② 우측에 심벌이 존재할 경우 : ORB와 ANB를 포함하는 경우

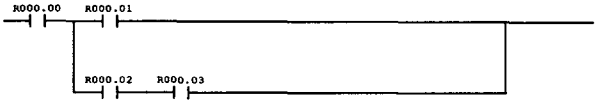
- 그 라인에서 다음의 상하접점의 위치까지를 확인 그 라인에 심벌의 개수를 파악.

②-1 심벌이 1개가 존재할 경우:OR => ANB



STRR000.00 STR R000.01 OR R000.02 ANB

②-2 심벌의 개수가 2개 이상 존재할 경우 : STR(N)+AND(N) => ORB+ANB



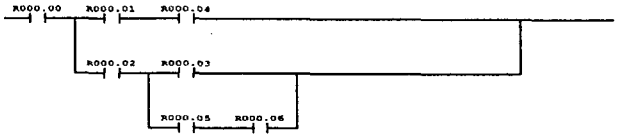
STRR000.00 STR R000.01 STR R000.02 AND R000.03 ORB ANB

②-3 또 다른 상하접점이 존재할 경우 :

- 다음 라인에서 상하접점의 위치까지를 확인하고 그 라인에 심벌의 개수를 파악.

- 다음 라인의 심벌의 개수가 1개: OR => ANB

- 다음 라인의 심벌의 개수가 2개 이상 : STR(N)+AND(N)=>ORB+ANB+ORB+ANB

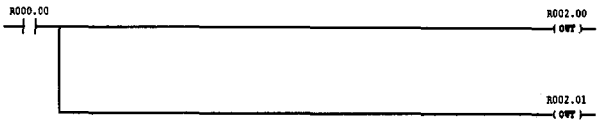


STRR000.00 STR R000.01 AND R000.04
STRR000.02 STR R000.03 STR R000.05
AND R000.06 ORB ANB ORB ANB

ⓒ 블록화의 패턴 분석

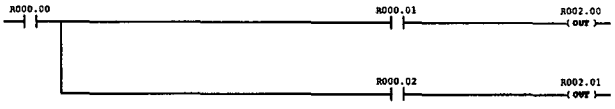
③ 해당라인에 상하접점이 없고 출력 코일 심벌이 2개 이상 존재할 경우: MCS와 MCR을 포함하는 경우.

③-1 상하접점과 끝 접점사이에 어떠한 심벌도 없는 경우 :



STRR000.00 OUT R002.00 OUT R002.01

③-2. 상하접점과 끝 접점사이에 심벌의 개수가 1개 이상 존재하는 경우.



STRR000.00 MCS STRR000.01 OUT R002.00
STRR000.02 OUT R002.01 MCR

4. 에이전트에 의한 LD/IL변환시스템

본 논문에서 에이전트[6,7,8]을 이용하여 제안하는 LD/IL 변환시스템은 다음 그림 3과 같다.

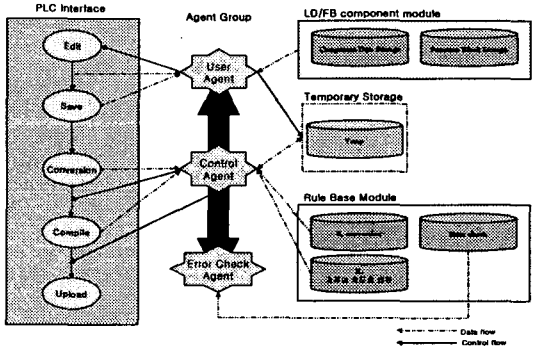


그림 3 LD/IL 변환 시스템

그림 3은 본 논문에서 제안하는 변환시스템으로서, 사용자가 PLC의 에디터화면을 통해서 LD/FB 언어로 편집작업을 하고 저장해서 이를 중간코드 형태인 IL언어로 변환한 후 DeviceNet에 업로드하기 위해 이를 컴파일하여 기계어 코

드로 변환한 후 I/O Driver인 DeviceNet에 업로드하는 컴파일링 과정을 나타낸 것이다. 그림 3의 각 구성을 보면 LD/FB component module과 Rule base module, Agent group로 구분한다.

(1) LD/FB component module은 component data storage와 function block storage로 구성되어 지고 component data storage의 내부 구성은 LD의 심볼들로 구성되어 있고 function block storage의 내부 구성은 FB symbol들의 요소들로 구성되어 있다.

(2) Rule base module은 크게 3가지의 구성요소로 이루어져있는데 ㉠ 중간 코드인 IL 언어로 변환할 때의 Rule ㉡ Error를 체크하는데 사용되어지는 rule ㉢ I/O Driver인 DeviceNet에 업로드하기 위한 기계어코드로 변환되어지는 rule로 구성되어 있다.

(3) Agent group에는 UA(User Agent)와 CA(Control Agent), 그리고 EA(Error check Agent)로 구성되어 있다. User Agent는 사용자와 컨트롤 에이전트와의 상호 작용을 위한 인터페이스로서의 역할을 하는데 사용자가 인터페이스에서 편집하게 되면 LD/FB component module과 Rule base module을 참조하여 수행되는 에이전트이다.

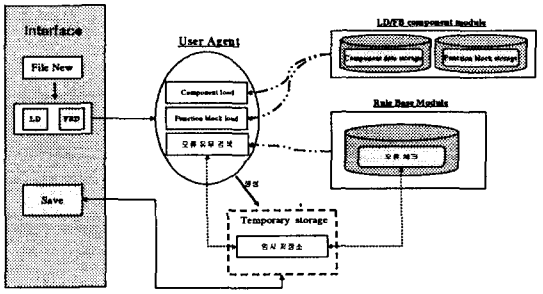


그림 4 User Agent

User Agent를 나타내는 그림으로 user가 편집하기 위해서 LD/FB component module의 각각의 심볼들을 로드하고 로드된 심볼들을 사용자가 편집하고 심볼들의 변수 범위나 위치들이 오류가 있는지의 여부를 체크하는 일련의 과정을 담당하고 사용자가 편집하는 모든 작업은 Temporary storage에 임시 저장되어 지고 사용자가 저장을 하게 되면 Temporary storage안의 심볼들에 대한 정보는 변환하기 위한 Data가 된다. Control Agent는 사용자 에이전트와 상호 작용을 통해서 중간 코드 형태인 IL언어로 변환하고 변환된 IL 언어를 기계어 코드로 변환하여 I/O Driver인 DeviceNet에 업로드하기 위해 Rule base module 참조하여 수행되는 에이전트이다.

사용자가 편집후 저장하면 Temporary storage에 있는 정보들을 Control Agent가 읽어들이고 이를 scanning 하여 Rule base module의 IL Conversion의 rule과 오류체크 rule을 통해서 IL로 변환하게 된다.

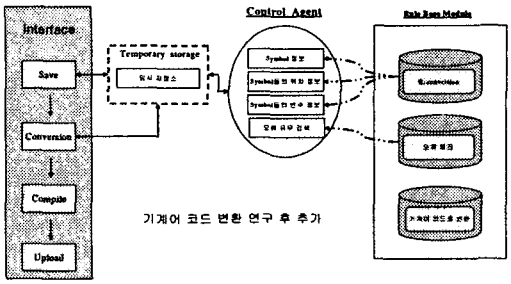


그림 5 Control Agent

Error check Agent는 User Agent 와 Control Agent 메시지 통신을 통한 상호 작용으로 사용자가 editing, conversion, compile하는 것이 올바르게 수행될 수 있도록 Rule base module을 참조하여 수행되는 에이전트이다.

현재 이러한 기능을 갖는 시스템을 구현 중에 있다.

5. 결론

본 논문에서는 IEC1131-3에서 제시하는 5개의 표준 언어들 중, 삼성전자의 SPC-10 기반으로 90%이상을 점유하고 있는 LD(Ladder Diagram)를 IL로 변환하고 컴파일할 수 있는 알고리즘과 시스템을 개발하였다. 본 논문에서는 LD를 IL코드로 변환시키기 위해 LD/FB component module과 Rule base module, Agent group로 구성된 시스템을 개발하였다. 이 시스템은 지능적 에이전트에 의한 논리오류를 체크할 수 있는 장점이 있다. 앞으로 이 시스템을 개발하여 실제 사례에 적용해야 할 것이다.

6. 참고문헌

[1] Norme Internationale International Standard, CEI IEC 1131-3, Premiere edition, First edition, 1993

[2] 원태현외 6인, PLC 제어기술, 제 2판, 복두출판사, 2001

[3] 박양수외 2인, FA를 위한 PLC 실습, 복두출판사, 1998

[4] 김종부외 3인, PLC 이론 및 실습, 복두 출판사, 2002

[5] PLC 이론과 실습, 삼성전자 사내교육 자료

[6] Russell and Norvig, Artificial Intelligence a Modern Approach 2/E Chap 2, Prentice Hall International Co., 1994

[7] John Graham, Real-Time Scheduling in Distributed Multi Agent Systems, Ph.D. Dissertation, University of Delaware, January, 2001

[8] John Graham, "Real-Time Scheduling for Distributed Agents", *AAAI-Spring Symposium on Real-Time Autonomous Systems*, Palo Alto, CA, March, 2000