PLC 프로그래밍 언어의 표준화 동향

Standardization Trend of the Programming Language of Programmable Logic Controllers

원 진 명, 신 동 민, 이 진 수
포항공과대학교 전자전기공학과

1. 서론

공장자동화 산업은 제조업 전체의 경쟁력을 좌우하는 핵심 산업이다. 공작기계 산업에 기반을 둔 고전적 의미의 자동화 기기는 물론이고 마이크로 프로세서와 네트워크의 발달에 힘입은 첨단 자동화 기기들에 대한 연구와 투자는 제조업 부문의 경쟁력과 직접적인 상관관계를 갖는다. 일찍이 이러한 노력을 기울인 선진 각국들이 경제대국으로 부상한 사실은 이를 뒷받침 해 주고있다.

각종 자동화 기기들 중에서도 가장 중요한 것이 논리연산 제어장치(Programmable Logic Controllers - PLCs)이다. PLC는 기존의 릴레이, 타이머, 카운터 등의 릴레이 제어 기반 자동화 기기들을 마이크로프로세서를 이용해 통합시킨 장치이다 고성능 마이크로프로세서의 도입을 통해 PLC의 프로그램 환경은 기존의 시퀀스 제어는 물론 각종 산술연산, 논리연산, 함수연산, 조절연산 및 데이터 처리를 가능할 수 있게 한다. 또한 PLC는 계래식 자동화 기기에 비해 제어기능과 신뢰성이 뛰어나고 제어내용을 손쉽게 수정할 수 있으며 여러 가지 복잡한 제어 알고리즘을 수행할 수 있는 등 많은 장점을 지니고 있다 이러한 필요성 때문에 현제 세계 유수의 대기업들은 물론 다수의 국내 기업들도 PLC 관련 기술 개발에 박차를 가하고 있다.

그러나 최근 PLC의 대규모화와 이에 대한 의욕적인 투자에도 불구하고 거대 PLC 공급 업체들의 폐쇄적 기술 운영에 따른 비효율성 때문에 PLC가 담당해 왔던 많은 부분을 다른 기기들이 대체하는 결과를 초래했다. 최근 이런 문제를 해결하기 위하여 PLC는 IEC(International Electrotechnical Commission)와 IEEE(The Institute of Electrical and Electronics Engineers)등의 표준규격에 기초를 둔 개방형 제품으로 발 빠르게 탈바꿈하고 있는 추세이다.

현재 개발되고 있는 대부분의 PLC 들은 IEC1131이라 일컬어지는 표준 규격에 기반을 두고 제작되고 있다. 이에 따라 아직까지 선진 기술에 비해 뒤져있는 현상황을 극복하고 그들에 필적하는 기술을 갖추기 위해서는 국제적으로 추진되고 있는 표준화의 내용을 정확히 이해하고 제품 개발에 적용해야 할 것이다. 본 논고에서는 PLC의 최근 기술 동향에 대해 살펴보고 PLC의 개발자 입장에서 프로그래밍의 표준화 하기위한 요소와 규격에 대해 기술하고 있는 IEC1131-3의 내용과 이를 효율적으로 구현하기 위한 방안을 제시한다.

2. PLC의 기술 동향

2.1 기존 PLC의 문제점

현재 공장자동화 관련 기술은 눈부신 발달을 거듭하고 있다. 예를 들어 공장자동화 관련 소프트웨어의 성능은 5년마다 2배정도 향상되고 있고 마이크로프로세서의 성능 역시 매년 25%의 신장세를 보이고 있다. 이같은 추세에 따라 제어 프로그램의 용량도 점점 커지고 있으며 2000년대에는 제어 프로그램의 평균 용량이 평균 200킬로 바이트를 상회할 것으로 예상된다.

한편으로 공장 자동화 기기의 대형화/복잡화는 높은 수준의 공장 제어 프로그램을 요구하게 되어 프로그램의 난이도를 높이게 되며 필연적으로 프로그래밍 기간의 연장과 신뢰도 저하라는 문제점을 낳게 된다. 만약 기존의 PLC 프로그래밍 언어인 래더(ladder) 다이어그램 만이 계속 사용된다면 한명의 숙련된 기술자가 200 킬로 바이트 정도의 공장 제어 프로그램을 오류 없이 작성하는 데는 수년의 시간이 소요될 것이다

또한 미래의 공장 자동화는 제조기업의 주컴퓨터를 정점으로 생산현장에서부터 최고경영자의 의사

결정에 이르기까지의 모든 정보를 활용, 생산성을 극대화 시킬 수 있도록 유기적으로 결합시키는 광의의 개념으로 이해될 수 있으며, 이에 따라 PLC는 기존의 순차 제어(sequence control) 뿐만이 아닌 인간과 기계 간의 접속(Man-Machine Interface - MMI). 공정의 네트워크화, 품질계통 관리 등의 작업도 수행하게 될 것으로 전망된다. 이 또한 제어 프로그램의 난이도를 높게 만드는 요소로 향상된 프로그램 품질관리기능이 도입되지 않는다면 2000년대의 프로그램 오류 발생은 80년대의 발생율에 비해 백배 이상에 이를 것으로 전문가들은 내다보고 있다.

다양한 제품의 출현 또한 사용자들에게는 부담으로 작용하고 있다. PLC 부문에선 초보 단계로 평가받는 우리나라에서도 매년 다양한 PLC가 개발, 출시되고 있으며 전세계적으로는 50억 달러를 상회하는 시장에서 수백 여 개의 제품들이 경합중이다. 예전에는 각 제조업체의 제품들이 각자의 독특한 프로그래밍 구조와 사용자 인터페이스를 가지고 있었으므로 PLC의 사용자 입장에서는 한 제조업체의 제품을 계속 사용함으로써 엔지니어의 재교육에 필요한 비용을 최소화할 수 밖에 없었다.

2.2 개방형 PLC

타 기종 간의 호환성이 떨어지는 문제를 극복하기 위해 PLC는 IEC나 IEEE 등에서 제정한 표준규격에 기반을 두고 개방형 구조로 개발되어 왔다. 현재 국제적으로 추진되고 있는 개방형 시스템의 주된 이슈는 이식성(portability), 연결성(connectivity) 등의 개념을 포함하는 표준화(standardization)이다.

이식성은 사용자로 하여금 한 기종에서 개발된 제어 프로그램을 다른 기종에서도 사용할 수 있도록 해주는 기능이다. 즉 사용자가 개발한 전체 제어 프로그램 혹은 부분적인 제어 프로그램을 다른 플랫폼에도 사용할 수 있도록 해주는 것이다. 연결성은 각기 다른 제조업체로부터 제조된 PLC 시스템간의 연결이 가능함을 의미한다. 이는 하나의 완성된 시스템간의 연결뿐만 아니라 한 시스템의 특정 모듈도 타 기종의 모듈로 대체할 수 있다는 포괄적인 개념이다.

이식성과 연결성을 보장하는 PLC는 원가를 절감시키고 안정성을 보장하게 된다. 이같은 표준화를 통해 사용자 입장에서는 서로 다른 PLC를 취급하는 엔지니어에 대한 재교육시간을 단축할 수 있게 한다. 또한 다른 시스템으로 교체하기 위한 노력이 줄어들고 더욱 안정화된 시스템과 높은 수준의 프로그램을 지향할 수 있게 된다. 이런 이유 때문에 선진국의 경우에는 PLC의 세계 규격이 완성되는 대로 대학의 관련 학과에서 이를 교육하도록 의무화하고 있다.

2.3 IEC1131의 출현

원래 PLC의 규격화 작업은 국가별로 따로 진행되어 왔으나 79년 IEC내에 전담 부서가 설치되면서 전세계적으로 통합되었다. 그 후, PLC의 국제 규격인 IEC1131이 90년대에 들어서 제정되었다. IEC1131은 다음과 같은 5부분으로 구성되어 있다.

- IEC1131-1: 1992년에 제정되었으며 PLC의 일반 정보를 다루고 있음.
- IEC1131-2: 역시 1992년에 제정되었으며 PLC의 요구기능 및 테스트 조건을 다루고 있음
- IEC1131-3: 1993년에 제정되었으며 PLC 프로그래밍 언어의 규격을 기술함.
- IEC1131-4: 사용자를 위한 지침서.
- IEC1131-5: PLC의 통신 규격.

IEC1131은 세계 각국에서 이미 자국의 PLC 규격으로 채택된 것을 비롯하여 대학, 소프트웨어 업체 및 관련 민간단체까지 확산되고 있다. 이에 따라 현재 IEC1131을 적용한 PLC가 본격적으로 시장을 장악하고 있으며 앞으로는 이식성이 우수하고 다양한 소프트웨어 도구와 라이브러리 개발이 가능한 IEC1131형의 PLC가 향후 시장을 주도하게 될 것으로 분석된다.

2.4 소프트웨어 중심의 PLC

최근에는 PC 또는 워크스테이션용 마이크로프로세서의 성능이 강화됨에 따라 최소한의 하드웨어로 PLC의 기능을 수행하는 것이 가능하게 되었다. 이 같은 추세에 따라 종래의 하드웨어에 종속되었던 제품이 소프트웨어가 중심이 되는 제품으로 변화되고 있는 양상이며 하드웨어 부문에서는 PC나 워크스테이션용 범용 마이크로프로세서가 제어기용 마이크로프로세서 시장으로 도전해 옴에 따라 그 영역싸움이 본격화되고 있다.

이런 여건 속에서도 IEC1131은 여전히 채택되고 있으며, 소프트웨어 중심의 PLC에서는 친숙한 사용자 인터페이스와 프로그래밍 환경, 네트워크를 통한 기능 등이 제품의 수준을 가늠하는 중요한 기준이 되므로, 특히 프로그래밍 언어의 규격을 기술하고 있는 IEC1131-3은 과거에 비해 오히려 그 중요도가 커지고 있다.

3. IEC1131-3

IEC1131-3은 전체 PLC 시스템의 추상화 된 모델, 프로그래밍 언어의 공통 사항 들과 5가지 프로그래밍 언어에 대한 설명으로 구성되어 있다.

소프트웨어 모델과 통신 모델은 전체 PLC의 각 부분의 명칭과 정의, 그리고 그곳에서 실행되는 각

프로그램의 의미와 프로그램 간의 데이터 교환이 어떻게 이루어 질 수 있는가에 대해 기술하고 있다.

프로그래밍 언어의 공통 원소에서는 프로그래밍에 사용되는 식별자(identifiers), 키워드(keywords), 데이터 타입(data types), 변수(variables) 등과 같은 기본적인 사항 이외에도 함수(functions), 함수 블록(function blocks), 프로그램(programs), 리소스(resources), 태스크(tasks), 컨피규레이션(configurations) 등과 같은 PLC의 구성에 직적 연관되는 프로그램 요소와 종류에 관해 설명하고 있다.

끝으로 프로그래밍 언어 부분에서는 함수, 함수 블록, 프로그램을 작성하는데 사용되는 ST(Structured Text), FBD(Function Block Diagram), LD(Ladder Diagram), IL(Instruction List), SFC(Sequential Function Chart) 등의 다섯 가지 PLC 프로그래밍 언어들의 상호 호환성과 사용법에 대해 논하고 있다

4. 컨피규레이션 모델

4.1 소프트웨어 모델

IEC1131-3에 따르면 PLC는 그림 1과 같은 대략적인 소프트웨어 모델을 갖는다.

컨피규레이션은 하나의 PLC 단위를 의미한다. 여기에는 컨피규레이션 전역 변수들(global variables)과 다수의 리소스, 즉 연산보드가 존재할 수 있다. 그리고 각 리소스에는 리소스 전역 변수들과 태스크가 존재한다. 이처럼 제어 프로그램을 제외한 나머지 컨피규레이션 구성요소를 컨피규레이션 정의(configuration definition)라 한다.

여기에서 각 태스크들은 프로그램과 연동되어 실행되게 되는데 프로그램 내부에 존재하는 함수 블록 역시 자신의 태스크를 갖고 독립적으로 실행 가능할 수 있게 한 것이 IEC1131-3의 기본 골격이다. 프로그램, 함수, 함수 블록, 그리고 SFC 언어에서 정의되는 액션(action), 전이(transition)를 통틀어 프로

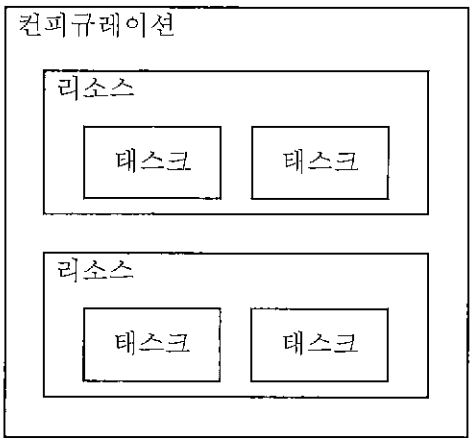

그림 1. 컨피규레이션 정의

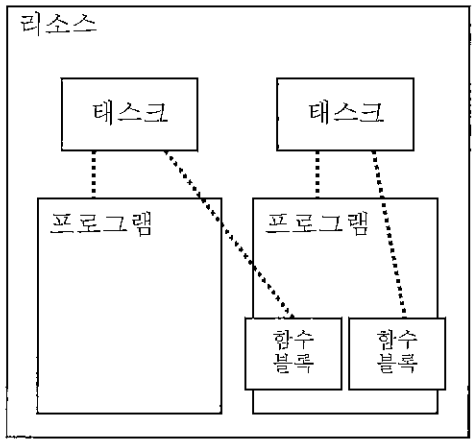

그림 2. 컨피규레이션 구현과 태스크간의 관계

그램 조직 단위(Program Organization Units - POUs)라 부르고 컨피규레이션 상에서는 컨피규레이션 구현(configuration implementation)으로 정의된다.

4.2 통신 모델

PLC의 통신은 컨피규레이션 간의 데이터 교환은 물론, 한 컨피규레이션 내의 데이터 교환까지 포함하는 포괄적인 개념이다.

프로그램 조직 단위 내에서의 데이터 교환은 내부 변수(internal variables)를 통해 이루어지고 프로그램, 함수 및 함수 블록에 데이터를 입력하거나 출력할 때는 입력 변수(input variables), 출력 변수(output variables), 입출력 변수(input/output variables)를 사용한다.

프로그램 조직 단위들이 상위의 리소스나 컨피규레이션 정보를 공유하고 싶을 때는 리소스나 컨피규레이션 전역 변수를 외부 변수(external variables)로 선언해서 사용한다.

컨피규레이션 간의 데이터 전달은 접근 경로(access pathes)로 정의된 통신 객체를 이용한다. IEC1131-5를 기반으로 작성되는 이 접근 경로를 통해 서로 다른 부분의 함수간, 서로 다른 컨피규레이션 내부의 프로그램간, 혹은 프로그램 가능한 컨피규레이션과 프로그램 할 수 없는 컨피규레이션 사이의 정보 교환이 가능하다.

4.3 프로그래밍 모델

프로그래밍 모델에서는 앞으로 정의될 프로그램 언어 공통원소 들간의 전후 생성 관계를 기술하고 있다. IEC1131-3 프로그래밍 모델의 핵심 개념은 '파생 및 재사용'이다. 가령, IEC1131-3에 따르면 기본 데이터 타입으로부터 사용자가 새로운 데이터 타입을 정의해서 사용할 수 있다.

또한, 표준 함수들로부터 새로운 함수 또는 함수 블록을 생성해서 라이브러리화 하여 다른 프로그램

작성에 사용할 수도 있다. 이런 기능을 통해 사용자는 고급 프로그래밍 언어에서 사용되는 프로그래밍 기법들을 PLC 프로그래밍에 사용할 수 있다. 프로그램의 라이브러리화는 동 기종 내에서만 이루어지는 것은 아니다. 한 기종에서 작성된 프로그램이 큰 변경 없이 다른 기종에서 사용될 수 있다면 그 가치는 큰 것이다.

이상의 세가지 컨피규레이션 모델은 PLC 프로그래밍 환경을 제작하는 제조업체가 기본적으로 지원해야 하는 사항들이다.

5. 프로그래밍 언어의 공통 원소

5.1 식별자와 키워드

IEC1131-3에서는 다음과 같은 식별자의 제약 조건을 두고 있다.

"식별자는 글자나 밑줄 친 문자로 시작되는 글자, 아라비아 숫자, 밑줄 친 문자들의 문자열이다. 밑줄은 식별자에서 의미를 가진다. 예를 들어, "A_BCD"와 "AB_CD"는 다른 식별자로 해석된다. 또한, 식별자는 공백 문자(space)를 포함하지 않아야 한다. 키워드는 각 구문 요소들로 이용되는 문자들의 특수한 형태이다. 키워드는 변수 이름 등으로 사용될 수 없다."

이같은 제약은 제조업체에 따라서는 완화될 수도 있는 사항이지만 완화된 조건에 의해 생성된 프로그램 코드는 다른 기종의 코드와 호환되는 것이 바람직 할 것이다.

5.2 데이터 타입 및 변수

IEC1131-3에서는 다음과 같은 20가지의 기본 데이터 타입을 제시하고 있다.

- 이진 데이터 타입: BOOL.
- 정수형 데이터 타입: SINT, INT, DINT, LINT, USINT. UINT, UDINT, ULINT.
- 실수형 데이터 타입: REAL, LREAL.
- 시간, 날짜형 데이터 타입: TIME. DATE, TIME_OF_DAY, DATE_AND_TIME.
- 문자열 데이터 타입: STRING.
- 이진문자열 데이터 타입: BYTE, WORD. DWORD, LWORL.

이상의 데이터 타입은 최대 64비트의 크기를 지원하고 있으나 제작하는 PLC의 용도에 맞게 데이터 형을 정리할 필요가 있다. 가령, 소형 또는 소프트웨어 중심의 PLC의 경우, 너무 많은 데이터 타입은 사용자에게 혼란을 줄 수 있다. 이런 경우에는 실수형 데이터 타입이나 정수형 데이터 타입의 종류를 한 가지만으로 결정하는 것도 좋은 방안이 될 수 있다.

좀 더 까다로운 사용자의 경우에는 어떤 표준 함수의 데이터 타입을 결정해서 사용하는 것 자체를 부담스러워 할 수도 있다. 예를 들어, 제조업체에서 ADD_INT(정수의 합)와 ADD_REAL(실수의 합)의 두 가지 함수를 제공한다고 할 때, 사용자가 이 둘 중 하나를 결정하는 작업을 없게 해 준다면 더욱 효과적일 것이다. 이 경우에는 사용자는 ADD(일반적인 합)라는 함수를 그냥 사용하게 하고 입력이 정수 또는 실수인지를 제어 프로그램 컴파일러가 판단하여 적절한 코드를 생성하는 것이 바람직하다.

이를 위해 IEC1131-3에서는 기본 데이터 타입 이외에도 다음과 같이 포괄 데이터 타입(generic data types)을 정의하고 있다.

```
ANY
   ANY_NUM
      ANY_REAL
         LREAL
         REAL
      ANY_INT
            LINT, DINT, INT, SINT, ULINT, UDINT,
            UNIT, USINT
   ANY_BIT
      LWORD, DWORD, WORD, BYTE. BOOL
   STRING
   ANY_DATE
      DATE_AND_TIME
      DATE
      TIME_OF_DAY
   TIME
```

포괄 데이터 타입으로 정의된 변수에 대해서는 그 처리 시에 자동으로 데이터 타입이 결정되도록 제어 프로그램 컴파일러가 지능적으로 작업을 수행해야 한다.

파생 데이터 타입(derived data types)은 IEC1131-3 프로그래밍 모델의 일환으로써 사용자가 데이터 타입을 새로 정의할 수 있게 함으로서 생겨나는 데이터 타입이다.

변수는 하나의 데이터를 표현하는 단일 변수 외에도 여러 개의 데이터를 표현하는 배열 변수가 정의되어 있다. 변수마다 CONST, AT, RETAIN 키워드를 이용하여 특수한 기능을 할당할 수 있다. 여기서 AT은 백분율 기호(%)를 이용해서 특정 어드레스를 가리키게 하는 직접 표현 변수(directly represented variables)인데 사용자의 편의를 위해서 특정 어드레스를 표현하는 방식을 좀 더 용이하게 제조업체 나름대로 정의할 수도 있다.

변수가 정의됨으로써 이를 기반으로 하는 프로그램 조직 단위들을 다음과 같이 정의할 수 있다.

5.3 프로그램 조직 단위I: 함수

함수는 실행될 때 하나의 결과값을 넘기는 프로그램 조직 단위이다. 함수 블록과 구분되는 함수의 가장 큰 특징은 함수는 어떠한 내부 상태 정보도 포함하지 않는다는 것이다. 즉, 입력 값이 같으면 출력 값도 항상 같다.

함수는 제조업체에서 제공하는 기본 함수와 사용자가 기본 함수들을 조합하여 생성하는 사용자 정의 함수의 두 가지 종류가 있다. 기본 함수들을 제공할 때는 포괄 데이터 타입을 지원하는 것이 사용자에게 편리하다.

제어 프로그램 편집기는 제조업체 제공 함수를 쉽게 열람할 수 있고 사용자 정의 함수를 쉽게 등록하여 사용할 수 있게 제작되어야 경쟁력을 갖게 된다. 다음은 IEC1131-3에 기술된 주요 함수의 종류이다.

- 데이터 타입 변환 명령어: 데이터 타입을 변환하기 위한 함수들.
- 수치 명령어: 수치연산을 위한 함수들.
- ST 등가 명령어: ST 언어에서 사용하는 연산자들과 같은 기능을 하는 함수들.
- Bit String 명령어: 쉬프트, 로테이트 등 이진 문자열 처리를 위한 함수들.
- Boolean 논리 명령어: AND, OR 등과 같은 이진 논리 연산을 위한 함수들.
- 선택 명령어: 다수의 입력 중 원하는 것을 선택하게 하는 함수들.
- 비교 명령어: 두 개의 입력을 비교하는 함수들.
- 문자열 명령어: 문자열 처리를 위한 함수들.
- 날짜, 시간 명령어: 날짜, 시간 처리를 위한 함수들.

이밖에도 사용자가 자주 사용할 만한 함수들이 지원되어야 한다.

5.4 프로그램 조직 단위II: 함수 블록 과 프로그램

함수 블록은 내부에 메모리를 가지고 있어서 과거의 상태를 저장하는 동적 프로그램 조직 단위로서의 역할을 한다. 엣지 검출, 카운터, 타이머 등이 함수 블록의 예이다. 함수 블록이 여러 개 사용될 경우 각자의 메모리가 필요하므로 함수 블록을 구현하기 위해서는 그 이름 이외에 각각의 인스탄스가 지정되어야 한다. 한 제어 프로그램에 연관된 전체 인스탄스의 관리와 작성이 용이해야 함은 물론이다

함수 블록은 프로그램 또는 다른 함수 블록 내에서만 사용되지만, 프로그램은 리소스 내에서 사용된다. 리소스 내부에는 다수의 태스크와 프로그램들이 존재하며 이들이 하나씩 연동되어 각각 프로그램 인스탄스를 생성한다.

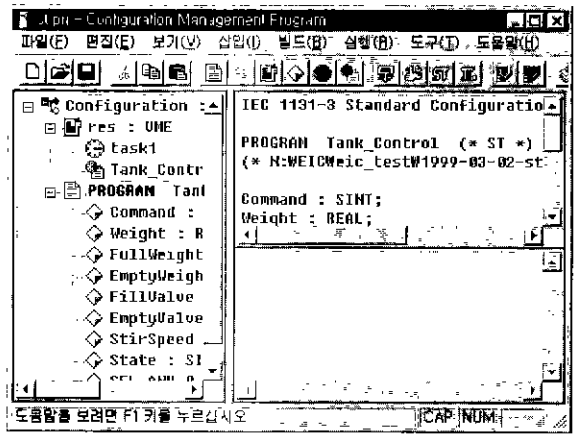

그림 3. 제어 프로그램 편집기의 예

제어 프로그램 편집기는 컨피규레이션 정의 및 컨피규레이션 구현, 리소스, 테스크, 사용자 정의 함수/함수 블록, 프로그램, 프로그램 인스탄스를 편집할 수 있는 사용자 인터페이스를 제공해야 한다. 그림 3은 전술된 요소들을 편집할 수 있는 제어 프로그램 편집기의 예이다

6. 프로그래밍 언어의 종류

6.1 개요

IEC1131-3에는 ST, FBD, LD, IL, SFC의 다섯 가지 언어가 정의되어 있다. 이 중 SFC는 순차 제어 프로그램(sequential control program)을 작성하기 위해 사용된다. 따라서 SFC 프로그램은 컴파일 되면 독특한 형태의 코드로 변환된다. 반면에 나머지 언어로 작성된 프로그램들은 단순 코드로 배열로 표현된다.

PLC의 개발자 입장에서는 개발할 시스템에 대한 사용자의 요구를 정확히 파악하고 해당하는 언어를 지원해야 한다. 다섯 가지의 언어를 모두 개발할 경우, 완성도가 높은 제품이 될 수 있으나 다른 사항들도 고려해야 한다. 예를 들어, 복잡한 프로그램보다는 프로그램 방식의 편이성이 요구되는 소형 시스템에 많은 언어를 지원하는 것은 그 개발 과정에도 많은 투자가 필요할 뿐만 아니라 사용자에게는 오히려 부담이 될 수도 있다.

6.2 프로그래밍 언어 I: ST

ST는 PASCAL과 매우 유사한 상위 레벨 프로그래밍 언어이다. 하지만 PASCAL과는 달리 제어 프로그램 작성을 위해 정의되었다. ST 프로그램은 값의 대입, 함수 호출, 복잡한 수식의 생성 및 조건/선택/반복 연산을 위해 설계되었다. 이같은 특징때문에 특히 ST는 복잡한 수식과 조건을 사용하는 프로그램에 알맞은 언어이다. 심지어 순차 제어 프로그램

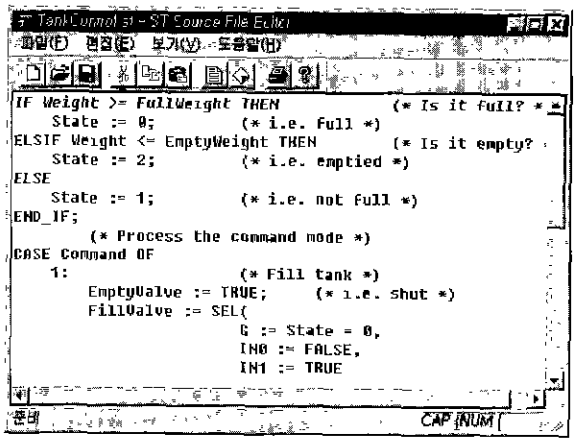

그림 4. ST 프로그램의 예

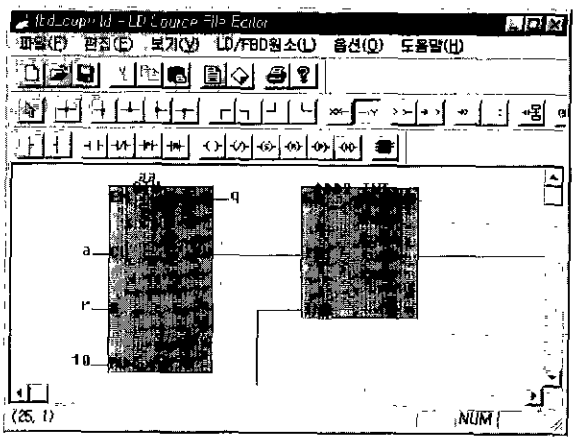

그림 5. FBD 프로그램 편집기의 예

용인 SFC 프로그램조차 ST로 변환이 가능하다.

최근에는 ST 언어만으로 구성된 제어 프로그램 편집기도 개발되고 있는데 이것은 ST가 그 강력한 기능에 덧붙여 텍스트 형식의 언어로 C나 PASCAL의 사용에 익숙한 사람이면 쉽게 익혀서 사용할 수 있기 때문이다. 예전의 PLC가 LD 위주의 프로그래밍 환경을 제공한 반면, 앞으로는 ST 언어의 지원이 필수적인 요구 사항이 될 것으로 전망된다.

개발자 입장에서는 ST 언어를 지원하기 위해서 ST 컴파일러를 개발해야 한다. 다른 언어 컴파일러에 비해 개발이 까다롭지만 lexer나 yacc 같은 도구를 이용하면 특정 제품에 알맞은 ST 컴파일러를 개발할 수 있다. ST 프로그램의 예는 그림 4와 같다.

6.3 프로그래밍 언어 II: FBD

FBD는 대표적인 그래픽 기반 언어이다. FBD로 작성된 프로그램은 그 흐름을 이해하기가 용이하므로 주로 제어 블록 간의 신호의 흐름으로 이루어진 프로그램을 작성하는데 사용된다. FBD는 제어 루프로 구성된 논리 전기 회로도와 유사한 형태를 가지기 때문에 전기회로에 익숙한 프로그래머들이 선호하는 언어이다.

FBD에서 함수 또는 함수 블록은 왼쪽 입력 단자와 오른쪽 출력 단자를 갖는 사각형으로 표현되며 그 내부에 이름, 그 위쪽에 인스탄스 이름이 표시된다. 각 입/출력 단자의 이름 역시 사각형의 내부에 표시된다. FBD 프로그램 편집기 개발자의 입장에서는 그림이 복잡해 지는 것을 피하기 위해서 입/출력 단자의 이름 대신 번호를 사용하는 것이 바람직하다. 가령, 풀다운 메뉴 등을 지원하면 사용자가 입/출력 단자의 이름을 쉽게 확인할 수 있다. FBD 프로그램 편집기의 예는 그림 5와 같다.

6.4 프로그래밍 언어 III: LD

LD는 릴레이 논리에 기반을 둔 고전적인 PLC의

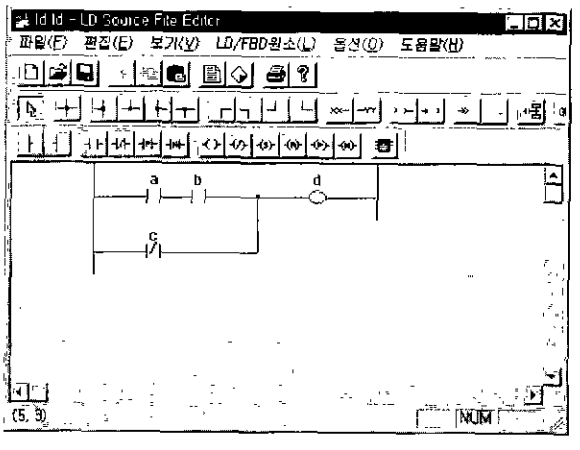

그림 6. LD 프로그램 편집기의 예

언어이다. LD 언어가 지원되므로 릴레이 레더에 익숙한 사람에게도 IEC1131-3이 그렇게 생소한 것만은 아니다. LD 언어는 왼쪽 전원 레일의 전원이 입력 접점 들을 통과할 경우, 출력 접점에 공급되는 물리적인 현상을 그대로 그림으로 표현한 것이다. 각 입력 접점은 read only 상태, 출력 접점은 write only 상태의 이진 변수들을 나타낸다.

FBD와 LD는 함께 사용하는 경우가 많으므로 두 언어의 편집기를 통합하는 추세이다. LD 프로그램 편집기의 예는 그림 6과 같다.

6.5 프로그래밍 언어 IV: IL

IL은 어셈블러와 유사한 하위 레벨 프로그래밍 언어이다. IEC1131-3 IL의 명령어들은 표준화 이전에 존재했던 명령어들을 모두 취합하여 모든 다른 언어의 기본이 되도록 설계되었다. 몇몇 개발자들이 ST를 기본 언어로 사용하긴 하지만 아직까지도 많은 개발자들이 다른 언어들을 IL로 변환하여 코드를 생성한다. IL 프로그램의 각 실행 단위는 직접 하나의 코드 단위로 연결되기 때문에 개발자 입장

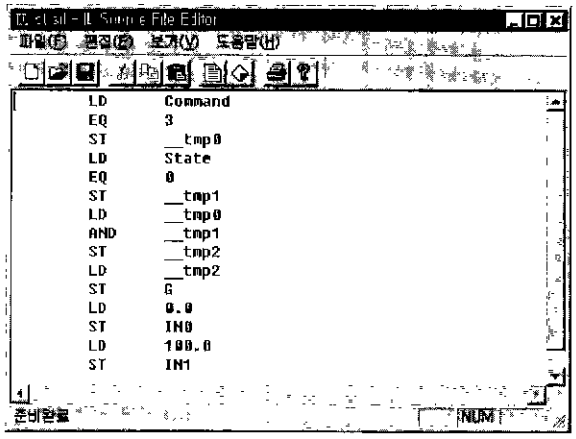

그림 7. IL 프로그램의 예

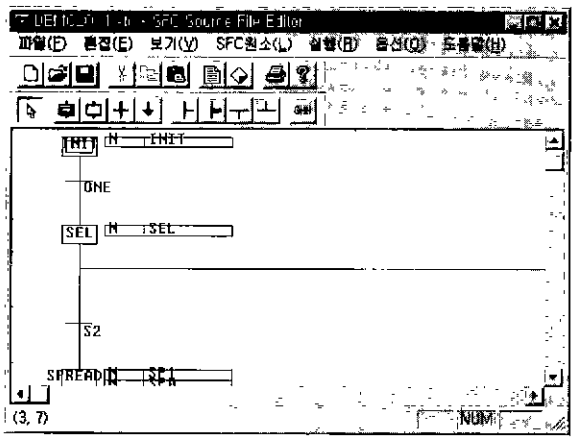

그림 8. SFC 프로그램 편집기의 예

에서는 구현이 용이하다.

IL 언어는 조건 연산이 거의 없고 실행 흐름의 변화가 적은 짧은 프로그램의 최적화에 알맞은 언어이다. 반면에 IL 언어는 해석이 어렵고 디버깅도 사실상 불가능하다는 단점이 있다. IL 프로그램의 예는 그림 7과 같다.

6.6 프로그래밍 언어 V: SFC

SFC 언어는 순차 제어 프로그램을 작성하기 위해 설계되었다. 표준안이 재정되기 이전에도 대부분의 PLC 제품은 SFC의 형식과 유사한 순차 제어가 가능한 양식을 제공하고 있었는데 이것이 SFC로 통일되었다.

페트리넷(Petri-nets)으로부터 출발한 SFC는 시스템의 동작을 상태(states)와 전이(transitions)로 표현한다. 페트리넷은 이산 이벤트 시스템(discrete event systems)을 표현하는 훌륭한 도구로 사용되고 있으며 그 해석을 위해 현재도 많은 연구가 진행중이다.

SFC의 시퀀스는 사각형 형태의 스텝(steps)의 연결로 표현되며 각 스텝은 제어하고자 하는 시스템의 특정 상태를 나타낸다. 스텝들 사이의 연결선인 전이는 이전 스텝의 토큰(token)을 어떤 조건이 만족되면 다음 스텝으로 전달한다.

각 스텝은 하나 또는 그 이상의 액션들과 연동되는데 어떤 스텝이 활성화된 상태에서는 그 스텝에 해당하는 액션들이 실행된다. 마찬가지로 전이도 ST, FBD, LD, 또는 IL로 작성된 코드가 전이 조건을 표현한다. 이런 방식으로 SFC를 이용하면 손쉽게 순차 제어 프로그램을 작성할 수 있다. SFC 역시 대부분의 PLC에서 지원하는 언어로, 그 프로그램 구조의 복잡성 때문에 얼마나 편리한 사용자 인터페이스를 제공하느냐가 제품 경쟁력의 관건이 된다. SFC 프로그램 편집기의 예는 그림 8과 같다.

7. 결론

본 논고에서는 표준화에 중점을 둔 PLC의 최근 기술 동향, PLC 프로그래밍 언어의 표준 안으로 자리잡고 있는 IEC1131-3의 내용, 그리고 PLC 프로그래밍 환경을 효율적으로 개발하기 위한 방안을 개발자의 입장에서 살펴 보았다.

이러한 표준화 규격에 힘입어 관련 하드웨어 개발 업체들은 특정 모듈의 집중 개발이 가능하게 되었고 소프트웨어 개발 업체 역시 PLC의 소프트웨어 중심화에 고무되어 있다. 네트워크의 발달을 충분히 활용할 수 있는 제품과 사용자 입장에서 손쉽게 사용할 수 있는 제품 또한, 나름대로의 시장성을 가지고 있음을 PLC 개발자들은 참고해야 한다.

예를 들어, 사용자가 PLC에 대한 전문지식이 없어도 손쉽게 프로그램을 편집할 수 있는 기능의 지원은 필수적인 것이며 기능 다양화, 기기간의 호환성 및 MMI 기능의 강화, 프로그래밍 언어의 다양화, 사용자 인터페이스 강화 등에 총력을 기울여야 할 것이다 이것이 국내 PLC 제품이 선진국의 제품을 추월하기 위한 필요 조건이다.

원 진 명

1972년 11월 13일생 1995년 포항공과대학교 전자전기공학과 (공학사). 1997년 동 대학원 (공학석사). 1997년-현재 동 대학원 박사과정 재학 중. 관심분야는 공장 자동화 및 지능 제어.

신 동 민

1970년 7월 18일생. 1994년 포항공과대학교 전자전기공학과 (공학사). 1996년 동 대학원 (공학석사). 1996년-현재 동 대학원 박사과정 재학 중. 관심분야는 공장 자동화 및 로봇 제어.

이 진 수

1953년 3월 1일생. 1975년 서울대학교 전자공학과 (공학사). 1980년 미국 UC-Berkeley 전기공학과 (공학석사). 1984년 UCLA System Science (공학박사). 1981년-83년 Union Oil Company R&D Center 연구원. 1984년-85년 AT&T Bell Lab 연구원. 1985년-89년 GE Aerospace 고등기술연구소 선임연구원. 1989년-현재 포항공과대학교 전자전기공학과 조교수, 부교수, 정교수, 주임교수. 관심분야는 컴퓨터 제어, 공장 자동화, 적응제어 및 지능제어.