흐름 제어 언어의 통합 처리

김 태 완† · 장 천 현††

요 약

산업분야에서 자동화 시스템은 제품의 설계, 생산 공정의 제어, 장애 처리, 품질검사 등과 관련된 처리 과정을 자동으로 수행할 수 있도록 하여 생산성을 향상시킨다. 이러한 자동화 시스템에서 감시 및 제어에 대한 처리 과정을 기술하는 언어를 흐름 제어 언어라 한다. 현재 사용되고 있는 흐름 제어 언어는 문자 기반의 IL, ST와 그래픽 기반의 FBD, SFC, LD가 있다. 일반적으로 감시 제어 시스템에서 사용되는 소프트웨어는 사용할 수 있는 흐름 제어 언어를 2종류 이하로 제한하고 있고, 동일한 시스템 환경에서는 언어의 혼용을 통한 통합 시뮬레이션이 불가능하다. 본 논문에서는 흐름 제어 언어의 특성을 분석하고 기존 시스템 환경에서 언어 작성 및 처리 과정에 대하여 분석하고, 언어의 통합 처리를 위하여 고급언어 형태의 ST를 확장한 EST 언어를 제안하였다. 이러한 연구를 기초로 그래픽 언어인 FBD, LD, SFC를 통합 처리하여 EST로 변환하는 그래픽 언어 편집기와 EST를 저급언어인 IL로 변환하는 EST-IL변환기를 구현하였다. 이러한 편집기 및 변환기를 통한 IL 기반의 시스템 구현 및 실험 결과는 흐름 제어 언어의 통합 처리 방안을 제시한 것이다.

Integrate Processing Scheme of Flow Control Language

Tae Wan Kim† · Chun Hyon Chang††

ABSTRACT

Automation systems improve the productivity of works which relate to product design, facilities management, fault processing and quality evaluation. In these systems, the description language for monitoring and control process is called flow control language. These are five flow control languages : IL, ST, FBD, SFC and LD. IL and ST are based on text form. FBD, SFC and LD are based on graphic form. Generally, a software which monitors and controls a system is allowed to use just one flow control language. It is impossible to use more than two languages for simulation in the same system environment. In this paper, we analyzed the characteristics of flow control languages and the process of programming in the legacy system. In addition, for the integrated processing of languages, we propose Extended ST based on the high-level ST language. Based on this research, we implement a graphical language editor and EST-IL convertor. The graphical language editor makes sequence rules, and converts graphical language into EST. EST-IL convertor has a function to convert EST into IL which is similar to assembly language. As the result of this paper, we present a scheme which integrates all the flow control language processing based on IL.

키워드 : 흐름 제어 언어(Flow Control Language), DCS(Distributed Control System), FBD(Function Block Diagram), LD(Ladder Diagram), SFC(Sequence Function Control), EST(Extended Structured Text)

1. 서 론

산업 분야에서 사용되는 컴퓨터 기술은 다양한 분야로 확대되고 있다. 이 중 모든 산업 분야에서 필수로 적용되는 기술 부분은 감시 및 제어 분야이다. 감시 제어 시스템은 작은 규모의 장비에서 발전소, 초대형 선박과 같은 대규모의 시스템 분야에 다양한 방법으로 개발되어 운영되고 있다. 그러나 규모에 상관없이 감시 제어 시스템의 구성 및 처리 과정은 공통된 성격을 띠고 있다. 특정 데이터를 수집 후 분석 과정을 거쳐 결과를 사용자에게 알려주며, 분석 결과를 이용하여 제어 과정을 수행한다. 이러한 특성을 기초로 범용화된 감시 제어 시스템이 개발되었다. 범용화된 감시 제어 시스템은 해당 목적에 맞는 디바이스 연결, 분석, 제어 과정등에 대한 정보만 작성하면 감시 제어의 역할을 수행할 수 있도록 개발되었다.

감시 제어 시스템에는 감시 및 제어에 관련된 정보를 작성하기 위하여 감시 제어용 흐름 제어 언어가 요구된다. 이에 따라서 IEC(International Electrotechnical Commission)에서는 IEC 61131-3에 산업용 감시 및 제어 작성을 위한 IL(Instruction List), ST(Structured Text), FBD (Function Block Diagram), SFC(Sequential Function Chart), LD(Ladder Diagram)의 5종류의 흐름 제어 언어를 규약하였다. 5종류의 언어는 감시 제어의 규칙 작성 및 적용 대상 분야에 따라 선택적으로 사용되고 있다. 이중 IL과 ST는 문자

※ 본 논문은 "2002년도 건국대학교 학술진흥연구비 지원"에 의한 연구결과임.
† 정 회 원 : 건국대학교 컴퓨터공학과 강의교수
†† 종신회원 : 건국대학교 컴퓨터공학과 교수
논문접수 : 2003년 9월 2일, 심사완료 : 2003년 12월 24일

기반의 언어이고, LD와 FBD는 그래픽 기반의 언어이며, SFC는 문자 기반 요소와 그래픽 기반 요소를 모두 포함하고 있는 언어이다. 현재 개발되어 사용되고 있는 시스템은 흐름 제어 언어중 2종류 이하의 언어를 이용하여 사용하는 것이 일반적인 방법이다. 그러나 시스템의 규모가 점차 커지고 대규모 시스템의 통합, 원격 감시 제어와 같은 사용자의 요구에 따라 5종의 언어를 모두 처리할 수 있는 시스템이 필요하게 되었다.

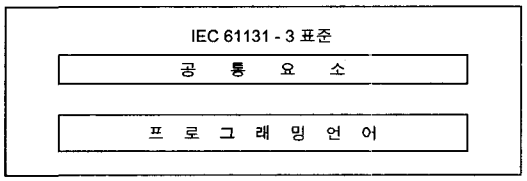
그러나 아직까지 감시 제어 시스템에서 사용되는 소프트웨어에서는 사용할 수 있는 흐름 제어 언어의 종류가 2종 이하로 제한되어 있고, 언어 처리 기능에 대한 한계가 있어 대규모 시스템에 적절한 언어 처리가 불가능하다. 또한 대규모 시스템에 대하여 운영 전에 통합 시험을 할 수 있는 시뮬레이션 방안이 현재로서는 존재하지 않는다.

이에 본 논문에서는 기존에 사용되는 그래픽 기반의 FBD, LD, SFC와, 문자 기반의 ST, IL을 분석하고 언어의 혼용을 위한 통합 방안으로 EST(Extended Structured Text)를 제안한다. 이러한 통합 분석 결과를 기초로 그래픽 기반 언어의 통합 처리 및 문자 기반 언어의 변환 방법을 제시하고, 언어의 분석을 통하여 그래픽 기반 언어와 문자 기반의 언어의 통합화를 통한 통합 처리 방안을 제시한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로 IEC61131-3에서 제안한 흐름 제어 언어에 대한 규약과 표준 언어인 IL, ST, FBD, LD, SFC에 대한 구성 및 개념을 설명한다. 3장에서는 시스템에서의 언어 분석 과정 및 통합화 처리 기준 언어인 EST에 대해 설명한다. 4장에서는 언어 분석 결과를 기초로 흐름 제어 언어 편집기 및 변환기의 설계 및 구현에 대하여 설명하고 흐름 제어용 그래픽 언어 통합 방안을 제시한다.

2. 관련 연구

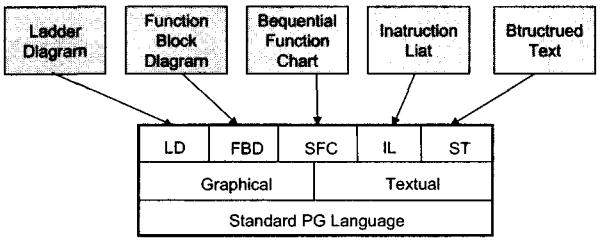
흐름 제어 언어는 표준 규약인 IEC61131-3에서 제안한 언어로서 전체적인 구성은 (그림 1)과 같이 공통 요소와 프로그래밍 언어의 두 부분으로 되어 있다.



(그림 1) IEC 61131-3 구성

공통 요소에는 각 언어에서 범용으로 사용하는 데이터 타입과 변수를 정의하고 있다. 데이터 타입은 논리형, 정수형, 실수형, 문자열형, 사용자 정의 데이터 타입 등 일반 데이터의 표현을 위해 다양한 형태로 정의되어 있다. 변수는 선언한 부분에서만 사용 가능하며 같은 이름으로 다른 부분에서 재사용 가능한 지역변수와 전체에서 참조 가능한 전역변수로 정의되어 있다[1].

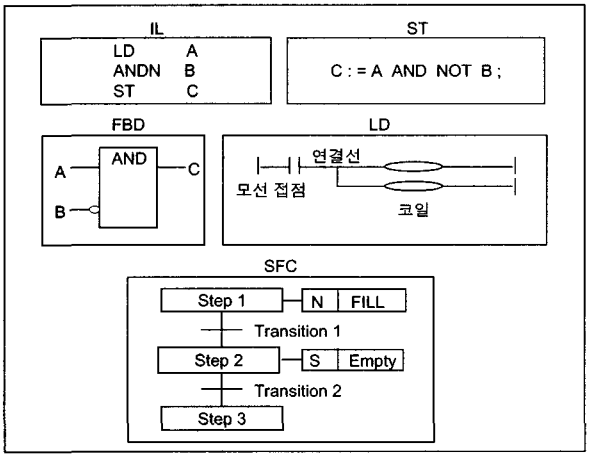
프로그래밍 언어 부분에서는 종류별 흐름 제어 언어의 구성 및 개념에 대하여 정의하고 있는데, (그림 2)에서와 같이 그래픽 기반 언어인 LD와 FBD, 문자 기반 언어인 IL과 ST, 그리고 그래픽과 문자 기반 요소를 모두 가지고 있는 SFC로 구성되어 있다[2, 3].



(그림 2) IEC 61131-3 표준 언어

IL은 컴퓨터 언어인 어셈블리 언어와 유사한 형태로써 연속된 명령어로 이루어져 있으며, 명령어 리스트로 구성되어 있다. 연산자, 변경자, 피연산자 등의 단순화된 언어 형태로 이루어져 있다. ST는 고급 프로그래밍 언어인 파스칼 또는 베이직과 유사한 형태를 가지고 있다. 자동화 과정에 활용되는 고급 언어이며, 컴퓨터 언어의 기본 구조인 조건, 순환, 처리에 대한 내용을 포함하고 있다. 데이터 조작과 수학적 계산을 수행할 수 있는 구조와 명령어 집합으로 되어 있기 때문에 주요 장점은 데이터 처리에 있다.

FBD는 산업용 처리 분야에서 흔히 사용되고 있는데, 표현 방법은 기능 정의 및 동작 정의로 구성되며, 제어 요소들 간의 정보나 데이터의 흐름을 나타내기에 적합하다. 주기적인 작업 수행을 정의하는 언어로 사용되며, 전자 회로처럼 그래픽 블록 형태의 조합으로 구성되어 있다. LD는 코일이나 접점 등의 릴레이 로직을 그래픽 형태로 표현하여 데이터를 검사하고 변경하게 한다. 구성 요소로는 모선, 연결선, 접점, 코일 등이 있다.



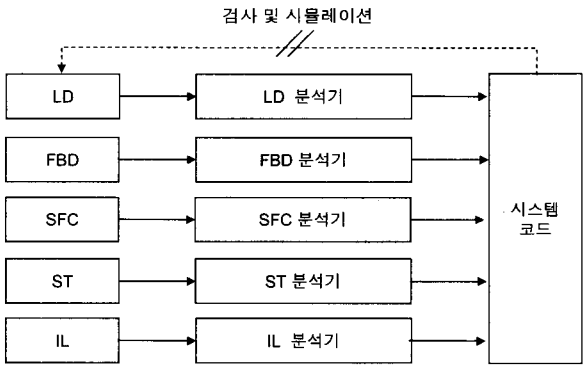
(그림 3) IL, ST, FBD, LD, SFC의 예

SFC는 다른 언어에서 표현된 작업들에 대한 상위 구조로서의 개념을 가지며, 전체 시스템의 수행에 대한 흐름 정의에 적합하다. LD의 대체 언어로 대두되고 있으며 이산 제어 시스템의 순차 논리를 그래픽적으로 표현할 수 있다. 제어 프로그램의 순차적 논리를 명확하게 표시하기 때문에 프로그램의 유지, 작성, 보수와 판독이 용이하다는 장점을 갖고 있다. 스텝, 천이, 액션, 액션 제한자, 흐름선 등으로 구성되어 있다.

3. 시스템 분석 및 언어 확장

3.1 시스템 분석

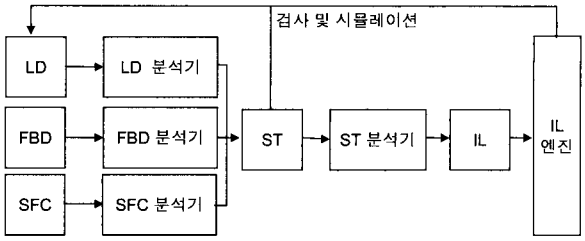
기존 시스템에서는 흐름 제어 언어로 작성된 내용을 시스템에서 수행하기 위하여 (그림 4)와 같이 독립적인 분석기를 이용한다. 시스템에서 2종 이하의 흐름 제어 언어를 사용하기 때문에 각각의 분석기를 두고 사용하는 것은 크게 문제가 되지 않는다. 그러나 점차 시스템의 크기가 대형화되면서 서로 다른 언어로 작성되어 사용하던 시스템들이 하나의 통합된 시스템으로 이루어져야 하며, 이러한 시스템들간의 결합 시뮬레이션이 가능해야 하는 문제가 대두되었다. (그림 4)에서와 같이 언어별 분석기를 통하여 처리된 코드는 검사 및 시뮬레이션이 불가능하여 오류 검사와 같은 중요한 문제를 해결할 수 없으므로, 사용자가 지속적인 감시를 통해 오류를 수정해야 한다.



(그림 4) 기존 시스템의 처리 과정

이러한 문제점을 해결하기 위하여 시스템에서 수행되는 코드를 흐름 제어 언어 중 저급언어 형태로 시스템 코드에 가장 유사한 IL 언어를 기초로 IL 엔진을 구현하고 모든 언어를 IL로 변환함으로써 기존의 시스템 코드 부분을 대처할 수 있다. 또한 IL로 변환하는 과정에서 그래픽 기반의 언어를 단일화된 언어를 통하여 IL로 변환하여 처리함으로써 시스템의 다양한 정보를 통합 처리할 수 있다.

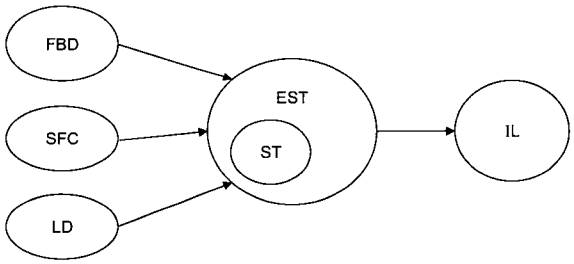
통합 처리가 가능한 구조는 (그림 5)에서와 같이 그래픽 언어로 작성된 정보를 문자 기반의 ST 언어로 변환하고 ST 언어 시뮬레이터를 통하여 그래픽 언어 시뮬레이션이 가능하게 한다. 또한 고급언어 형태의 ST 언어를 IL로 변환하여 시스템에서 구동함으로써 그래픽 언어의 통합을 가능하게 하였다.



(그림 5) 통합 시스템의 처리 과정

3.2 EST(Extended Structured Text)

흐름 제어 언어 중 LD, FBD, SFC는 사용자 편의를 위한 그래픽 처리를 기반으로 하는 언어이므로 시스템에서 이용 시에는 복잡한 번역 과정이 필요하다. 또한 그래픽 기반의 언어를 통합하기 위해서는 중간 언어를 가지고 그래픽 언어를 내부적으로 동일한 언어로 번역하여야 한다. 이에 본 논문에서는 문자 기반의 자유로운 문장을 표현할 수 있는 ST 언어를 확장한 EST 언어를 제안한다.



(그림 6) 4종 언어- EST 확장

EST 언어는 (그림 6)에서와 같이 ST, FBD, LD, SFC을 IL로 변환하기 위한 중간 언어의 역할을 하는데, 3종 언어(FBD, LD, SFC)를 ST로 표현할 수 없는 부분을 ST를 확장하여 새롭게 설계한 것이다[5].

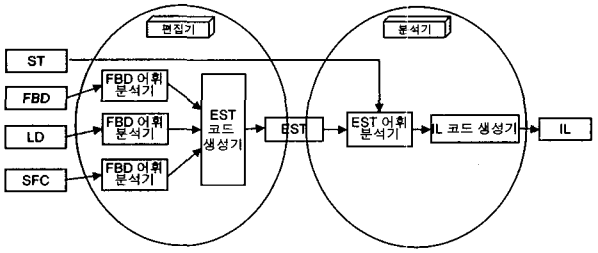
4. 흐름 제어 언어 도구 설계 및 구현

4.1 전체 구조 및 특징

흐름 제어 언어 분석의 전체 구조는 그래픽 편집기 부분과 EST-IL 변환기 부분으로 나눌 수 있다. 그래픽 편집기 부분은 그래픽 언어를 그림으로 편집하는 부분이다. 그래픽 편집기상에서 FBD, LD, SFC는 EST 코드 생성기를 거쳐 EST로 변환된다. 변환된 EST는 EST-IL 변환기를 거쳐 최종적으로 IL로 변환된다. EST 코드를 생성하기 위해서는 FBD, LD, SFC와 EST간의 매핑 테이블을 이용하며, IL로

변환하기 위해서는 EST와 IL간의 매핑 테이블을 이용한다.

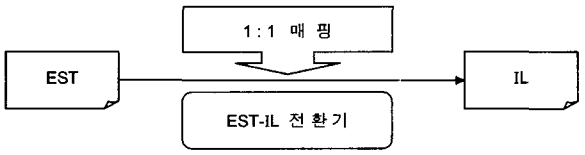
통합 처리 방안은 (그림 7)에서와 같이 중간언어의 역할을 하는 EST를 기준으로 그래픽 언어를 처리하기 위한 편집기 부분과 문자 기반의 언어 처리를 위한 변환기로 분리할 수 있다. 이러한 분리된 구조는 EST 언어 기준으로 그래픽 언어에 대한 검증을 수행할 수 있으며, 문자 언어에 대한 처리를 독립적으로 수행할 수 있기 때문이다. 이러한 시스템의 구조는 흐름 제어 언어의 통합을 EST 언어로 일괄 처리하여 시스템의 수행을 효율적으로 수행할 수 있으며, 기존의 시스템에서 각각의 독립적인 분석기를 사용하여 처리하던 부분을 단일화된 처리기를 이용함으로써 소프트웨어의 크기를 감소시킬 수 있다.



(그림 7) 전체 구조

4.2 EST-IL 변환기

EST-IL 변환기는 그래픽 언어에서 작성되어 생성된 EST 코드 혹은 사용자가 직접 작성한 EST 코드를 IL로 변환한다. EST의 분석은 일반적인 컴파일러 기술을 이용하며 LEX, YACC을 이용한다. LEX를 이용한 어휘분석기와 YACC을 이용한 구문분석기에 의해 분석이 완료된 후 EST-IL 매핑 테이블의 변환 규칙을 이용하여 IL 언어로 변환되게 된다. EST 언어와 IL 언어간의 변환은 언어의 변환 규칙이 일대일이고 언어간의 변화에서 정보의 손실이 발생하지 않기 때문에 매핑 테이블을 이용하여 변환할 수 있다.

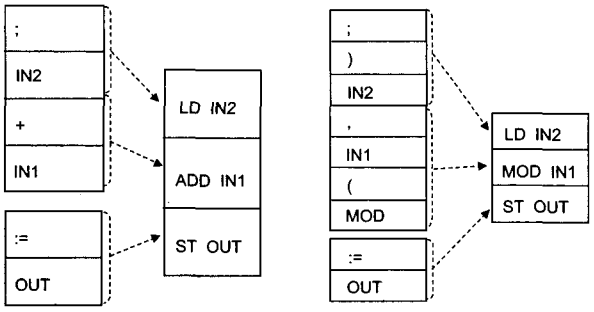


(그림 8) EST-IL 변환

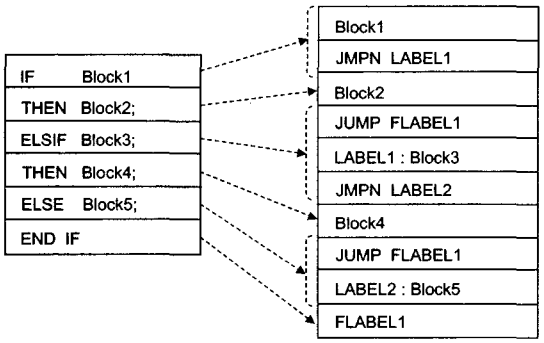
4.2.1 EST-IL 변환 과정

EST 언어는 고급언어의 형태를 띠고 있으므로, IL 언어로 변환시에는 당연히 코드가 증가하게 된다. 그러나 정보의 손실이 발생하지 않기 때문에 IL의 형태가 고정적인 구조를 띠게 되고 실제 변수와 같은 부분만이 변화 가능성을 가지고 있다. 이에 각 코드를 통일화하여 미리 변환된 형태를 작성하고 변화 가능한 부분만을 새로 재정의함으로써 변환 과정을 처리할 수 있다.

(그림 9), (그림 10)은 EST의 형태에 따라 연산자, 함수와 같은 EST 문장이 일대일로 변환되는 과정을 보여준다.



(그림 9) 연산자 및 함수 변환



(그림 10) 조건문 변환

4.2.2 EST-IL 매핑 테이블

EST 코드를 IL로 변환하기 위해서는 언어간 매핑 테이블이 필요하다. EST 언어를 IL로 변환시 연산자와 피연산자 정보 이외에는 고정된 형태를 취하므로 두 언어간의 테이블로 처리한다. <표 1>과 <표 2>는 EST-IL간의 변환규칙을 보여준다.

〈표 1〉 EST-IL 연산자 변환 규칙

ST	IL
a1 : = a2	LD a2 ST a1
a1 + a2	LD a1 ADD a2
a1 − a2	LD a1 SUB a2
a1 × a2	LD a1 MUL a2
ANY_TO_BOOL(a1)	LD a1 ANY_TO_BOOL
ANY_TO_DINT(a1)	LD a1 ANY_TO_DINT
ANY_TO_REAL(a1)	LD a1 ANY_TO_REAL
ANY_TO_TIME(a1)	LD a1 ANY_TO_TIME

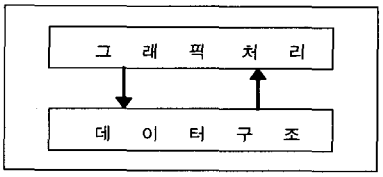
〈표 2〉 EST-IL 조건문 변환 규칙

종 류	EST	IL
IF	IF BLOCK1 THEN BLOCK2 ENDIF BLOCK3 ENDIF	BLOCK1 JUMPN LAVEL1 BLOCK2 JMP ENDLAVEL LAVEL1 : BLOCK3 ENDLAVEL :
CASE	CASE Pa OF NPa1 : BLOCK1 NPa2 : BLOCK2 ELSE BLOCK3 ENDIF	LD Pa EQ NPa1 JMPC LAVEL1LEVEL1 LD Pa2 JMPC LAVEL2LEVEL1 JMP JAVEL3LEVEL1 LABEL1LEVEL1 : BLOCK1 JMP LASTLABEL1LEVEL1 LAVEL2LEVEL1 : BLOCK2 JMP LASTLAVEL1LEVEL1 LABEL3LEVEL1 : BLOCK3 LASTLAVEL1LEVEL1 :

4.3 그래픽 편집기

그래픽 편집기는 사용자가 흐름 제어 규칙에 따라 LD, FBD, SFC의 그림을 작성 후, EST 코드로 변환하는 기능을 가진다. 그래픽 편집기는 그래픽 처리 부분과 EST 생성기 부분으로 나눌 수 있다. 그래픽 처리 부분은 좌표, 속성, 입출력, 파일 입출력, 이벤트 등을 정의한다. EST 생성기 부분은 그래픽 정보를 매핑 테이블에 정의된 내용에 따라 EST 코드로 변환한다. 또, 그래픽 명령에 의해 인터페이스를 호출함으로써 그래픽 처리와 데이터 구조를 분리한다.

4.3.1 편집기의 구조



(그림 11) 전체 구조

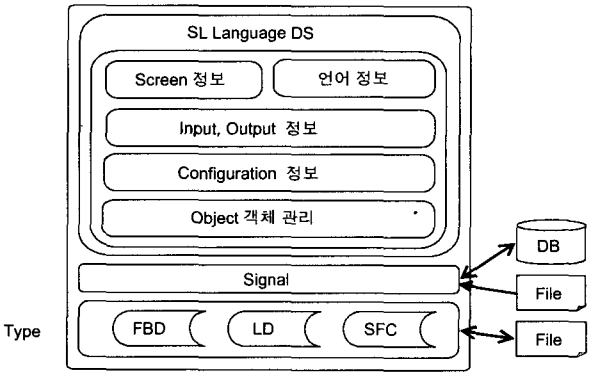
전체 구조는 (그림 11)과 같이 그래픽 기반의 흐름 제어 언어인 LD, FBD, SFC의 데이터 구조 모듈과 그래픽 처리 모듈로 구성되어 있다. 데이터 구조 모듈로부터 필요한 정보가 들어 있는 객체의 포인터를 얻어 그래픽 처리 모듈에서 정보를 처리하여 입출력을 한다. 데이터 구조와 그래픽 처리 기능의 분리된 모듈화 구조는 흐름 제어 언어 기능을 확장할 수 있다. 소프트웨어의 개발 과정에서 독립적인 구조는 단독 시뮬레이션이 가능하고, 데이터 구조는 산업용 임베디드 시스템에 재사용이 가능하다. 그래픽 처리가 데이터 구조의 영향을 받지 않게 함으로써 독립적인 그래픽 처리 기능 개선이 가능하다.

4.3.2 데이터 구조

데이터 구조는 (그림 12)와 같이 각 흐름 제어 언어의 형식에 관한 데이터, 시그널 데이터, 흐름 제어 언어의 정의에 관한 데이터로 구성되어 있다. 흐름 제어 언어의 형식에 관한 데이터에 FBD, LD, SFC를 모두 포함하여 단일 데이터 구조에서 흐름 제어 언어 3종을 구성할 수 있다.

흐름 제어 언어의 형식에 관한 정보는 파일 혹은 데이터베이스에서 처리함으로써 수정을 통해 소프트웨어의 재구축 없이 언어 기능을 조정할 수 있다. 읽어 들인 흐름 제어 언어의 형식에 관한 정보는 신호 정보와 자동으로 결합되어 수행이 가능하다.

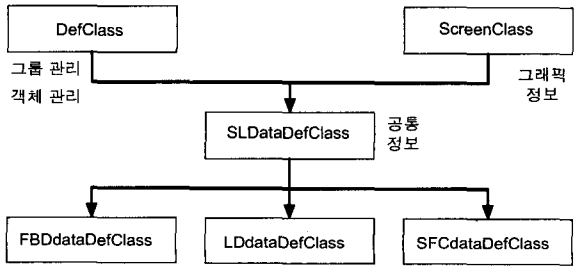
신호 정보의 수집은 단일 감시 제어 시스템에서 사용할 때에는 단순 파일을 통해 처리 가능하며, 대규모 감시 제어 시스템에서는 신호 정보가 포함된 데이터베이스 서버로부터 입출력을 할 수 있다.



(그림 12) 데이터 구조

흐름 제어 언어의 형식에 관한 정보와 신호 정보는 흐름 제어 언어의 구성 요소에 관한 정보와 결합하여 그래픽 처리 모듈에서 처리한다.

데이터 구조의 처리는 (그림 13)과 같이 계층 구조로 되어 있어 단위 객체 구조의 상속만으로 FBD, LD, SFC의 데이터 구조를 처리 가능하게 하고 있다. DefClass는 그래픽 기반 언어인 FBD, LD, SFC의 그룹 관리와 객체 관리를 하며, ScreenClass는 FBD, LD, SFC의 구성 요소가 출력되는 화면에 관한 정보를 관리한다. SLDataDefClass는 흐름 제어 언어의 공통 정보를 처리하며, 하위 객체들은 각 FBD, LD, SFC의 구성 요소에 관한 정보를 가지고 있다.



(그림 13) 데이터 구조 처리

4.3.3 FBD-EST 매핑 테이블

FBD를 EST로 변환하는데 공통으로 사용되는 연산자, 함수, 함수 블록간의 변환 규칙은 <표 3>, <표 4>와 같다. 매핑 테이블의 분류 기준은 입력 변수와 출력 변수의 개수에 따라 분류된다[5].

〈표 3〉 FBD - EST함수 변환 관계

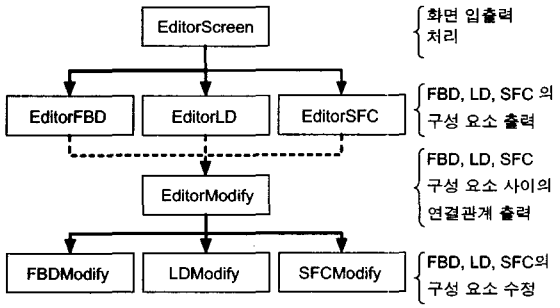
구 분	FBD	EST
1	COS A - IN Q - B	B : = COS(A) ;
2	INSERT A - IN B - Str C - Pos Q - D	D : = INSERT(A, B, C) ;
3	MUX4 A - SEL B - IN1 C - IN2 D - IN3 E - IN4 Q - F	F : = MUX4(A, B, C, D, E) ;

〈표 4〉 FBD-EST 함수 블록 변환 규칙

구 분	FBD	EST
1	F_TRIG A - CLK Q - B	R_TRIG(A) ; B : =R_TRiG.Q ;
2	RS A - SET B - RESET1 Q1 - C	RS(A,B) ; C : =RS.Q1 ;
3	CMP LT - C A - VAL 1 EQ - D B - VAL 2 GT - E	CMP(A,B) ; LT : =CMP.LT ; EQ : =CMP.EQ ; GT : =CMP.GT ;

4.3.4 그래픽 처리

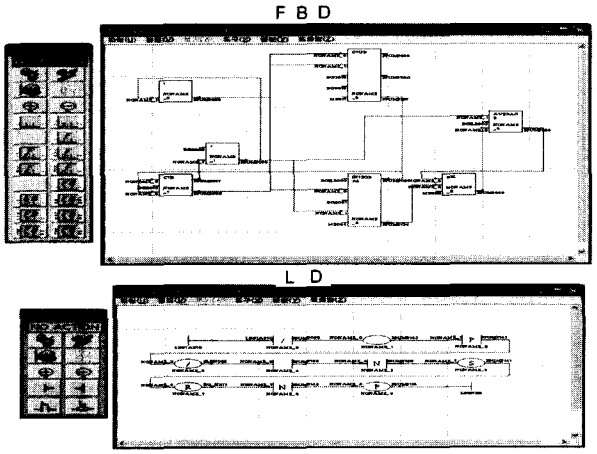
데이터 구조에 저장된 정보를 이용하여 FBD, LD, SFC의 구성 요소를 입출력을 수행하는 모듈은 (그림 14)와 같이 설계되어 언어의 확장이 용이한 구조를 가진다.



(그림 14) 그래픽 처리

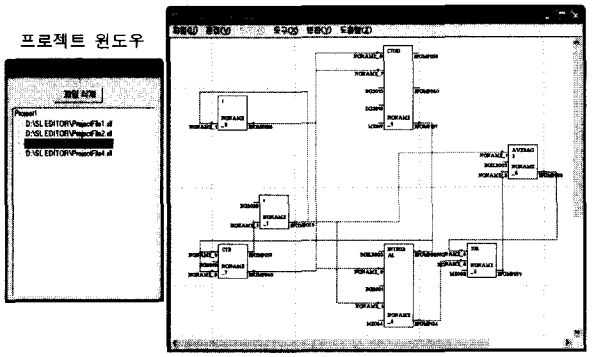
EditorScreen에서 FBD, LD, SFC의 구성 요소에 대한 화면 입출력을 처리함으로써 하위 객체에서 처리된 정보가 출력되는 대상을 바꿀 수 있다. EditorLD, EditorFBD, EditorSFC는 출력되는 각 FBD, LD, SFC 구성 요소의 그래픽 처리에 관한 정보를 가지고 있고, 추가된 구성 요소에 관한 정보를 데이터 구조 모듈에 보낸다.

EditorModify 객체에는 하위 객체에서 사용하는 공통 기능이 정의되며, 상위 객체들의 연결 정보를 가진다. 구성 요소 사이의 연결 관계 출력을 EditorModify 객체가 담당함으로써 각 FBD, LD, SFC 구성 요소를 수정하여 발생하는 연결 관계의 변화를 출력할 수 있게 한다. EditorModify 객체의 하위 객체들은 각 FBD, LD, SFC 구성 요소의 이동, 삭제, 복사에 관한 처리를 하고, 구성 요소의 수정에 관한 정보를 데이터 구조 모듈에 보낸다.



(그림 15) FBD와 LD의 입출력

(그림 15)은 그래픽 처리와 데이터 구조를 이용하여 FBD, LD를 작성하는 편집기의 실행 예이다. 편집기는 언어 선택에 따라 그래픽 처리 부분과 데이터 구조를 변화시킴으로써 단일 소프트웨어로 FBD, LD, SFC 모두 작성 가능하다. 편집하는 흐름 제어 언어의 종류에 따라 서로 다른 입력 인터페이스를 사용하여 각 흐름 제어 언어의 구성요소를 입출력할 수 있다.



(그림 16) 프로젝트 윈도우의 사용

(그림 16)과 같이 프로젝트 처리 기능은 FBD, LD, SFC의 혼용으로 작성된 내용을 통합 관리할 수 있다. 파일 단위의 입출력에 사용한 인터페이스를 사용하여 인터페이스

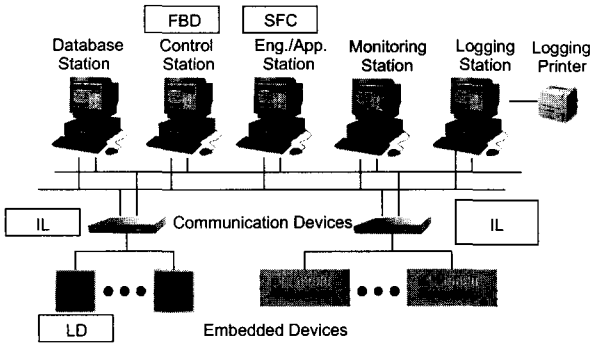
의 일관성을 유지한다. 또한 FBD, LD, SFC의 입출력과 프로젝트 정보는 시스템 정보 관리를 위하여 데이터베이스에 통합 처리된다.

5. 통합 처리 기술 특징과 적용

본 논문에서 제시한 흐름 제어 언어의 통합 처리 방안은 단일 제어 시스템 환경뿐만 아니라 분산 제어 시스템 환경하에서도 효율성을 증대시킬 수 있다. 통합 처리 방안을 적용하면 흐름 제어 언어의 혼용 사용이 가능하고, 시스템 운영 및 적용에서도 융통성을 증대시킬 수 있다. 통합 처리 방안의 특징으로는 운영상의 시스템 구조 개선, 흐름 제어 사용자에 대한 재교육 및 편리성 향상, 기존 분산 제어 시스템과의 결합성 등으로 요약할 수 있다.

5.1 시스템 구조 측면

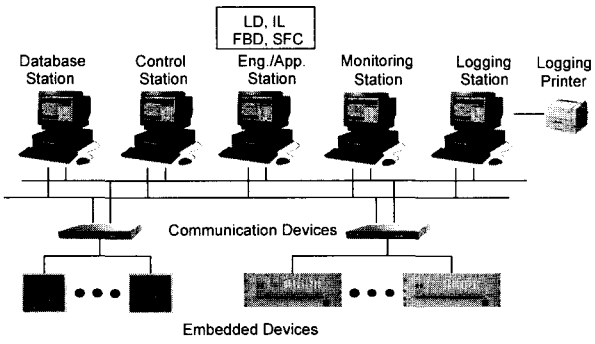
분산 제어 시스템에서는 (그림 17)과 같이 각종 신호 정보를 수집하는 단말장치들과 고유의 역할을 맡고 있는 컴퓨터로 구성된다. 단말장치중 PLC(Programmable Logic Controller)는 LD의 언어를 사용한다. 그러나 분산 환경의 모든 제어를 담당하는 제어 시스템에서는 FBD를 사용하고 응용 부분에서 제어 순서를 정의하는 부분에서는 주로 SFC를 사용한다. 이처럼 다양한 언어를 각 컴퓨터 혹은 단말장치마다 별개로 작성해야 한다. (그림 17)과 같이 기존 분산 제어 시스템에서는 컴퓨터와 단말장치들간의 시스템 결합을 상호 약속된 통신 프로토콜에 의존하기 때문에, 분산 환경 시스템을 구성하는 컴퓨터 혹은 각 고유의 일을 독자적으로 수행하면서 통신 정보만을 서로 공유하고 제어 명령을 수행한다. 기존 시스템 환경에서는 컴퓨터와 단말장치의 독립성을 부여할 수는 있지만 시스템 통합 구조에서는 결합성이 떨어질 수 밖에 없다. 또한 시스템 구축에서도 하나의 장치라도 연결이 안될 경우 시스템 전체 환경에 대한 시험이 불가능하다.



(그림 17) 기존 시스템 환경

본 논문에서 제안한 통합 처리 기술을 적용하면 분산 제어 환경에서 시스템 기준 설정 정보를 다루는 엔지니어링 컴퓨터에서 다른 컴퓨터와 단말장치들에서 수행해야 하는 시스템 명령을 작성한 후 이를 각 컴퓨터와 단말장치들로 전달하는 구조를 가질 수 있다. 이러한 통합 처리 방안을 통해 컴퓨터와 단말장치의 융통성을 높일 수 있다. 즉 엔지니어링 컴퓨터에서 흐름 제어 언어로 작성된 내용에 의하여 시스템에서의 역할을 변경할 수 있다. 이는 단순한 통합 언어 작성 및 처리 이상의 중요한 부분이다. 분산 제어 환경에서 컴퓨터 혹은 단말장치가 연결되지 않았을 경우에도 엔지니어링 시스템을 통하여 해당 단말장치 역할을 대신하는 분산 제어 환경 시뮬레이터가 가능해지기 때문이다.

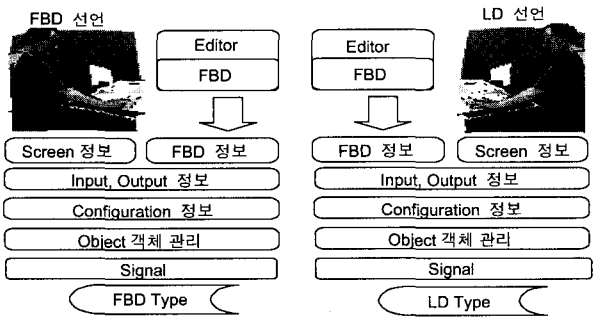
그리고 기존 시스템의 경우 통신 프로토콜에 의한 시스템 통합 방법을 사용하기 때문에 컴퓨터와 단말장치간의 결합성이 떨어지지만, 통합 처리 기술을 적용한 분산 제어 시스템 환경은 통합된 프로토콜을 사용하고 시스템간의 원격 제어 및 가상 시뮬레이터가 가능하게 되므로 시스템의 결합성이 매우 높아진다.



(그림 18) 통합 처리 적용 시스템 환경

5.2 시스템 설치자 및 관리자 측면

분산 제어 환경에서 흐름 제어 언어를 이용하여 시스템 기능을 작성하는 경우, 컴퓨터 혹은 단말장치의 환경이나 언어를 지원하는 도구가 서로 상이하여 대부분 재교육을 필요로 한다. 즉 시스템 엔지니어는 언어마다 별도로 교육을 받아야 사용 가능해진다.



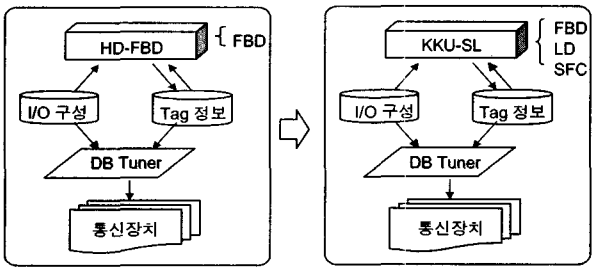
(그림 19) 통합 처리에 의한 개발 지원

본 논문에서 제시한 통합 처리 방안은 (그림 19)와 같이 언어 처리의 통합과 함께 시스템 엔지니어의 도구를 통합

할 수 있다. 이에 따라 도구 재교육 기간 축소 및 전체 시스템 환경에 대한 이해를 높일 수 있다.

5.3 기존 시스템과의 결합

새롭게 제안된 신기술도 기존의 시스템 환경과 응용성이 없을 경우, 그 효용 가치는 떨어진다. 본 논문에서 제시한 통합 처리 방안은 흐름 제어 언어의 통합과 함께 (그림 20)에서와 같이 기존 시스템과의 결합을 가능하게 한다. 본 논문에서 제시한 통합 처리의 최종 언어는 IL이다. 그러나 기존의 시스템 환경에서는 흐름 제어 언어를 분석한 후 개발업체마다 고유의 처리 방법을 이용한다. 이에 반해, 통합 처리 방안은 통합 처리의 결과물을 기존 시스템에서 요구하는 형태로 변경할 수 있는 구조를 지원하고 적용할 수 있으며, 기존 시스템에 언어 처리 기능을 확대할 수 있는 효과를 얻을 수 있다.



(그림 20) 기존 시스템과의 결합

기존 시스템과의 결합 방법은 새롭게 전체 시스템을 개발하지 않고, 현재 분산 제어 시스템 시장에서 요구하는 5종의 흐름 제어 언어에 대한 지원에 대하여 만족시킬 수 있다. 이에 본 논문에서 제시한 통합 처리 방안은 2003년 ㈜현대중공업의 분산 제어 시스템 개발팀에서 기술 적용하고 있으며, 분산 제어 시스템의 핵심 기술로 수자원 공사 프로젝트에 적용할 예정이다.

6. 결론 및 향후 계획

본 논문에서는 감시 제어 시스템 분야에서 사용되는 흐름 제어 언어를 분석하고 이를 기반으로 그래픽 기반의 언어인 FBD, LD, SFC의 언어를 문자 기반으로 처리할 수 있도록 정의한 EST를 제안하였다. 또한 언어의 변환 연구 결과를 기초로 하여 그래픽 언어를 EST 언어로 변환하는 그래픽 편집기 및 EST 언어를 IL 언어로 변환하는 변환기를 구현하여 언어의 통합에 대한 검증을 수행하였다.

본 논문에서 제안한 언어 처리 방법 및 데이터 구조와 그래픽 처리 방법은 그래픽 기반의 언어인 FBD, LD, SFC를 하나의 소프트웨어에서 각 언어의 특성에 맞게 변형될 수 있는 소프트웨어 체계를 제안하였다.

또한 EST를 기준으로 FBD, LD, SFC 언어를 통합하는 통합 분석기 개발은 단일 소프트웨어 상에서 흐름 제어 언어의 통합 시뮬레이션을 가능하게 하였으며, 기존 시스템에서 불가능했던 흐름 제어 언어의 결합 시뮬레이션도 가능하게 하였다.

본 논문의 언어 통합 처리 방안의 응용 분야는 대규모 감시 제어 시스템을 탑재하고 있는 상하수 처리장, 발전소 제어 시스템, 고속 전철 진단 처리 시스템, 전력 감시 시스템, 기타 모니터링 시스템 등의 산업 분야에서 사용될 수 있다. 향후 계획으로는 통합 처리 방안에 의해 개발된 시스템을 이용하여 언어별 시뮬레이션 기능 및 실시간 결합 시뮬레이션 시스템을 개발을 하고자 한다.

참 고 문 헌

[1] International Standard IEC61131-3, First Edition, 1993.
[2] PLCopen, http://www.plcopen.org.
[3] RealGain, Real PLC Manual, 2001.
[4] ISaGraF, ISaGRAF PRO User Guide, 2002.
[5] 정은영, 김선주, 김태완, 장천현, 김문회, 흐름 제어 언어의 통합분석을 위한 확장 ST, 한국정보처리학회 학술발표논문집, 제9권 제1호, pp.1013-1016, 2002.
[6] 김선주, 김태완, 장천현, 흐름 제어 언어 분석 도구 설계 및 구현, 한국정보과학회 학술발표논문집, 제29권 제2호, pp. 634-636, 2002.
[7] 정구회, 김태완, 장천현, 흐름 제어 언어의 그래픽 언어 통합, 한국정보처리학회 학술발표논문집, 제10권 제1호, pp.763-766, 2003.
[8] Goldsec, Software GPPA.

김 태 완

e-mail : twkim@konkuk.ac.kr
1994년 건국대학교 전자계산학과(공학사)
1996년 건국대학교 전자계산학과(공학석사)
1996년~2001년 현대중공업 기전연구소 연구원
1996년~현재 건국대학교 컴퓨터공학과 박사과정
2003년 경민대학 인터넷비즈니스과 겸임교수
2004년~현재 건국대학교 컴퓨터공학과 강의교수
관심분야 : 프로그래밍 언어, 실시간 프로그래밍, 자동화 소프트웨어, 산업기기 감시 진단 제어 시스템

장 천 현

e-mail : chchang@konkuk.ac.kr
1977년 서울대학교 계산통계(학사)
1979년 KAIST 전산학(석사)
1985년 KAIST 전산학(박사)
현재 건국대학교 컴퓨터공학과 정교수
관심분야 : 프로그래밍 언어, 컴파일러, 실시간 시스템