論文95-32B-6-1

고급 제어 알고리즘을 위한 공정 제어 언어에 관한 연구

(A Study on the Process Control Language for Advanced Control Algorithms)

金 聖 雨 *, 徐 暢 浚 *, 金 炳 國 *

(Seong Woo Kim, Chang-Jun Seo, and Byung Kook Kim)

요 약

본 논문에서는 고급 제어 알고리즘들을 포함한 다중 루프 제어 시스템의 구성을 위한 공정 제어 언어를 제안하였다. 제어기 구성을 위해 이 제어 언어는 특정 동작을 하는 기능 블럭들을 이용하였다. 그래서, 전체 제어 알고리즘은 이런 기능 블록들의 집합이며, 각 블럭은 알기 쉬운 ASCII 코드들의 라인인 기능 코드를 나타내는데, 이 기능코드로부터 기능, 입력, 출력, 파라미터들을 알 수 있다. 임의의 블럭의 입출력 포트로써 문자 변수를 사용하는 것이 가능하다. 기능 블럭 개념을 이용하는 다른 언어와 비교해볼때, 제안된 언어에서는 퍼지, 뉴럴 넷, 예측 제어기 등의 고급 제어 알고리즘들의 자유로운 사용이 가능하다. 이는 벡터, 행렬 등을 입출력 변수로 자유롭게 사용할 수 있기 때문이다. 유연성을 높이기 위해, 기능 코드와 타겟 의존적인 운용 코드 사이에 중간 단계인 C 언어 코드를 두었다.

Abstract

This paper presents a process control language for constructing multiloop control system, which include advanced control algorithms. In order to make controller, this language uses function blocks that do specific operations. Then, the total control algorithm is a set of function blocks, of which each block is represented as a function code. The function code is a line of simple ASCII codes denoting function, input, output, parameters. It is possible to use variables as input/output port of any block. Compared with other language using function block concept, the proposed one enables to use advanced control algorithms undefinitely, such as fuzzy, neural network, predictive controller, etc., because vector and matrix variables as input/output can be used freely in this language. To raise flexibility, we put an intermediate level, which is C-language code, between function code and target-dependent operation code.

I. 서 론

현대의 컴퓨터 기술 발달은 여러 용도의 많은 장치를 자동화시켰으며, 그 파급효과가 산업 공정에까지 미쳐 공정 제어 시스템의 모듈화, 시스템화에로의 발달이 가속하게 되었고 최근에는 디지탈화된 분산 제어 시스템(Distributed Control System) 에 대한 연구가 활발히 진행되고 있는 실정이다. 기본적인 분산 제어 시스템의 구성은 직접적으로 플랜트의 제어를 담당하는 PCS(Process Control System), PCS를 위한 엔지니어링 작업을 담당하는 EWS(Engineering Work-Station), 그리고 PCS가 보낸 플랜트의 공정 상태 정

* 正會員, 韓國科學技術院 電氣 및 電子工學科
(Korea Advanced Inst. of Science and Tech. Elec. Eng. and Elec.)

接受日字:1994年10月14日, 수정완료일:1995年6月5日

보, 경보를 그래픽 처리하여 운영자에게 보여주는 OIS(Operator Interface Station)로 이루어진다. 다음 그림 1은 연구 목적에 적합한 간이 분산 제어 시스템의 한 예이다. 여기서, PCM(Process Control Module), SIM(Simulation Module)은 각각 공정을 제어하는 부분, 공정을 모사하는 부분을 나타낸다. 그러므로, 이 시스템은 대규모 공정 제어 시스템으로의 확장이 매우 용이하다.

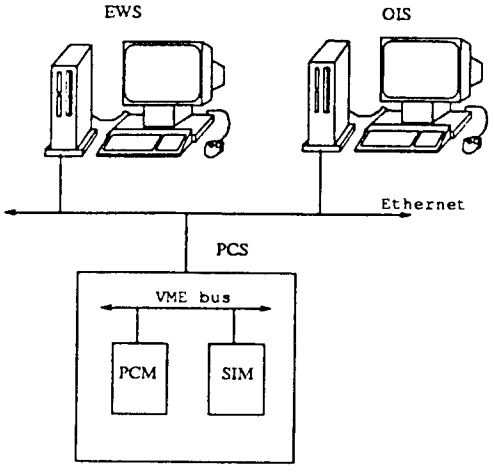


그림 1. 간이 분산 제어 시스템
Fig. 1. Mini distributed control system.

오늘날, 실시간 처리가 가능한 고속 프로세서로 무장된 분산 제어 시스템의 하드웨어는 비교적 견실하게 구성되게 되었고, 효율성, 신뢰성을 포함한 전체 시스템의 성능은 대부분 소프트웨어(S/W)에 좌우하게 되었다. 발전소와 같이 대규모이면서 복잡한 구조를 갖는 산업 공정에 적합한 분산 제어 시스템의 소프트웨어는 그 크기에 걸맞게 분야도 방대하다.

본 논문에서는 이 소프트웨어 분야 중 공정 제어 언어에 관련된 부분만을 다루기로 한다. 실제 공정을 제어하게 되는 제어 프로그램은 EWS에서 작성되어 PCS로 옮겨 실행되게 되는데, 보다 효과적인 프로그램의 개발 과정으로 컴퓨터 언어를 사용하게 되고, 나아가 아예 공정 제어 시스템을 위한 여러 가지 제어 언어가 개발되었다. 최근, 대부분의 제어 언어들은 복잡한 공정 제어 시스템의 구성을 용이하게 하기 위하여 모듈화된 기능 블럭 개념을 도입하였다. 그 예로 외국에서는 Bailey Function Code[2], Block Diagram Language[4] 등등이 있으며, 국내에서도 PCL1(Process Control Language ver.1)[1] 등이 개발된 바 있다.[5][6]

한편, 비선형 다입력 다출력 공정에 대한 다중루프(multiloop) 제어기 형태인 대부분의 산업 공정 제어 시스템의 제어 성능 향상을 위해서는 고전적인 PID 제어기 외에 적응 필터, PID 자동동조, 신호 처리, 신경회로망, 퍼지 제어, 예측 제어 등의 다양한 고급 제어 기능을 내장하여야 한다. 이러한 고급 제어 알고리즘에서 벡터나 행렬들의 연산이 일반적인데 비해, 이미 개발된 제어 언어들은 벡터, 행렬을 사용하지 않거나, 제한적으로 사용하고 있어서, 현대 제어 이론을 기반으로 한 고급 제어 알고리즘을 적용하기에는 대부분 적합하지 않다.

본 논문에서는 공정 제어 언어에 적합한 기능블럭 개념을 도입함과 동시에, 고급 제어 알고리즘의 적용이 용이한 제어 언어(PCL2, Process Control Language ver.2)를 제시한다. 제시한 언어에서는 입 출력,파라미터와 내부변수 등에서 벡터나 행렬들의 사용이 자유롭다. 벡터, 행렬의 구현 방법은 각각 일차원 배열, 다차원 배열로 이루어진다. 벡터나 행렬은 복잡한 수식의 간단한 표현을 가능하게 하므로, 전체 기능블럭갯수가 감소하여 사용자의 편의를 도모할 수 있으며, 벡터나 행렬식이 대부분인 고급 제어 알고리즘의 폭넓은 사용으로 성능 향상이 이루어질 수 있다. 벡터, 행렬을 많이 쓰는 제어 알고리즘으로 MIMO 시스템이나 상태 공간에서의 제어기를 예로 들 수 있다. 또한, 이 언어의 장점으로는 타겟 하드웨어(target hardware)에 독립적인 구현 방법을 채택하여 대부분의 시스템에 적용 가능하다는 것이다.

본 논문의 구성은 1장 서론에 이어 2장에서 제안한 제어 언어에 대하여 설명하고, 3장에서 그 구현 방법을 제시하며, 4장 결론 및 추후 연구 과제로 논문을 끝맺는다.

Ⅱ. 고급 제어 언어

일반적인 공정 제어 알고리즘은 다입력 다출력 구조이며 내부적으로 여러개의 local loop가 상호 연관성을 갖고 이루어진 다중루프 제어기이다. 그래서, 제어 알고리즘 전체를 한꺼번에 취급할 수 없으므로, 모듈화에 의한 확장성을 갖고 복잡한 제어 시스템의 구성이 용이하게 하는 것이 필요하다. 이를 가능하게 하는것으로 configurable 제어기를 생각할 수 있다. Configurable 제어기는 우선 제어 알고리즘을 구성하는 데 있어 자주 사용되거나 유용한 그리고 기본이 되는 기능들을 추출하여 기능블럭(function block)으로 정의한다. 그리고 나서 원하는 제어 알고리즘이 되게끔 기능블럭들을 유기적으로 결합(configuration)하면 된

다. 이러한 방법으로 제어기를 구성할 경우 구성자는 각 기능블럭 내의 구조를 알 필요가 없으며, 단지 그 입력, 출력, 기능만을 이해하면 된다. 이러한 개념에 기초하여 보면 사용된 언어는 일종의 problem oriented language(POL)이다.[11] 만약 기능블럭에 의한 제어 프로그래밍 언어가 복잡한 구조를 가지고 쉽게 이해할 수 없는 코드들을 사용하여 정의되었다면 이러한 언어는 사용가치가 없을 것이다. 이러한 POL 형식의 제어 언어는 Bailey 같은 회사에서도 이미 구현되었으며, 본 연구기관에서도 분산 제어 시스템 용 제어 언어(PCL1)를 개발한 바 있다. 그러나, 기존의 PCL1은 문자 변수를 허용하지 않고 입출력의 크기가 제한적인 단점이 있다. 본문에서는 PCL1의 단점을 극복하고 보다 강력히 개선된 제어 언어(PCL2)를 구현하였다.

PCL2의 특징을 간단히 나열하면 다음과 같다.

- 기존의 PCL1과 완벽하게 호환된다.
- 입출력 문자 변수의 사용으로 사용자가 이해하기 쉽다.
- 출력,입력, 내부 파라미터에 사용되는 변수 타입은 정수, 실수, 정수벡터, 정수행렬, 실수벡터, 실수행렬이다.
- 한 기능코드에 대해 출력이 여러개인 경우를 고려했다.
- 입력에서 상수(integer constant) 사용이 가능하다.
- 내부 파라미터 값들을 임의 file로부터 받을 수 있다.
- 초기화 루틴을 따로 둘 수 있다.

1. 기능코드(function code)

기능블럭들은 기능코드(function code)들로 표현된다. 기능코드는 기능블럭이 갖는 블럭들 사이의 입출력 관계와 내재된 파라미터들을 사용자가 알기 쉽게 ASCII문자로 표현한 것이다. 한개의 기능블럭에 하나의 기능코드들이 대응되며, 전체 제어기의 구성은 이러한 기능코드들의 나열로 이루어질 수 있다. 다음 그림 2은 기능 블럭의 구성 요소를 나타낸다.

새로 제안한 제어 언어에서 사용되는 기능코드의 표현형식은 다음과 같다.

블럭번호 출력문자변수 = 기능코드이름(입력 리스트: 파라미터값 리스트)

여기서 PCL1과의 호환성을 고려하여 출력문자변수 항목이 없을 수도 있으며, 리스트의 의미는 해당 항목이 하나 혹은 여러개가 있거나 전혀 없음을 의미하는데, 여러개일 경우는 컴마(,)로 항목들을 구분하게 되고, 항목이 존재하지 않을 경우는 비워둔다.

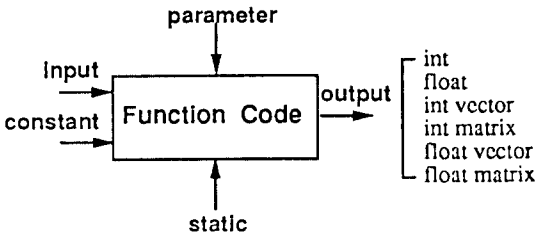


그림 2. 기능블럭의 구성 요소
Fig. 2. Components of function block.

각 부분에 대한 구체적인 규칙 및 설명은 다음과 같다.

1) 블럭 번호: 기능블럭의 나열로써 제어기를 구성할 때, 각 블럭에 할당되는 음이 아닌 정수인데, 수행 순서를 나타낸다. 또한, 내부 파라미터들의 메모리 할당을 위한 index로써 꼭 필요하다.
2) 출력 문자변수: 출력 포트를 나타내는 문자 변수이며 정수, 실수 뿐만 아니라 벡터나 행렬도 사용할 수 있다. 기존의 PCL1과 호환이 가능하도록 하기 위해, 출력변수가 없을 경우에는 블럭번호를 참조하여 전역변수로 할당된다. 또한, 기능코드에 따라 여러개의 출력 포트를 갖는 것이 가능한데, 문자 변수가 없을 경우 블럭번호 차례대로 전역 변수에 할당된다.
3) 기능코드이름: 기억하기 쉽고 기능을 적절히 표현할 수 있는 문자열로써, 각각의 기능을 갖는 기능코 드에 고유한 이름을 갖는다.
4) 입력 리스트: 기능코드가 받게 될 입력들을 나타낸다. 입력들은 크게 두가지로 구분되는데, 하나는 다른 기능블럭으로부터 온 입력포트를 나타내며, 또 하나는 상수를 나타낸다. 입력 리스트의 첫번째 항목부터 출력포트의 크기를 나타내는 상수를 정의하며, 파라미터 중에 벡터나 행렬이 있는 경우 그 크기를 나타내는 상수를 마지막 항목에 표시한다. 그리고, 나머지 항목들도 미리 정의된 각각의 순서에 맞게 배열되어야 한다. 여기서도 입력포트의 경우, PCL1과의 호환성을 위해 문자, 숫자 변수 둘 다 허용하며 벡터나 행렬이 가능하다.
5) 파라미터값 리스트: 기능코드가 수행되기 위해 사용하는 파라미터 값들을 말한다.정수, 실수, 벡터, 행렬들의 형을 가지며, 각각의 기능코드에 따라 갯수가 다르다. 특정 파라미터가 있는 데도 불구하고 항목에 기재하지 않았다면 default값

이 사용되며, 여러 파라미터가 있는 경우 값을 기재하지 않더라도 각각을 구분하는 컴마(,)는 있어야 된다. 또한, 임의의 파일로부터 파라미터를 읽을 경우 문자 스트링으로 파일이름을 입력할 수 있다.

파라미터는 기능코드 수행이 끝나도 그 값을 유지하며, 운전 도중 운전자에 의해 변경이 가능하다. 내부 변수는 특정값으로 초기화되며, 기능코드 수행이 끝나도 그 값을 유지하나, 운전도중 운전자에 의해 변경이 불가능하다.

다음은 첫번째 입력에 첫번째 파라미터 값을 곱하여 얻은 값과 두번째 입력에 두번째 파라미터 값을 곱해 얻어진 값을 더하는 기능을 갖는 기능블럭의 기능코드의 예를 나타낸다.

```
3 z = sumII ( 1, x, y, 2 : 10.0, 15.0)
or 3   = sumII ( 1, 1, 2, 2 : 10.0, 15.0)
```

위의 두 기능 코드는 같은 기능을 가진다. 여기서, 두 기능 코드에 공통적으로 첫번째 입력 항목인 1은 출력변수의 크기를 나타내고, 네번째 항목인 2는 파라미터 갯수를 나타낸다. 다만, 차이는 첫번째 기능코드에서는 입출력포트를 변수로 나타낸 것이고,두번째 기능코드는 입출력포트를 번호로 나타낸 것이다.

기존의 PCL1의 형식으로 된 기능코드들을 여기서도 사용할 수 있게 변환하여야 하며, 또한, 고급 제어 알고리즘을 포함한 새로운 기능코드를 구성하여 등록하면 효과적인 제어 언어로써의 기능을 가지는 것이다.

2. Configuration file

Configuration file은 기능블럭으로 구성된 제어기를 나타내는 화일이며, 각각의 기능블럭에 대응하는 기능코드들의 나열로 이루어져 있다. 이 화일의 구성형식과 요소는 다음과 같다.

1) Configuration file: ASCII문자로 이루어진 텍스트 화일인데 인식자, 기능코드, 주석(comment)문, 빈 라인(blank line)으로 구성되어 있다.
2) 인식자(identifier): 인식자는 다른 인식자가 나올 때까지의 나열된 기능코드의 수행을 구별한다. 다음은 구현될 인식자들을 나타낸다.
   - INITIALIZE : 메모리 할당이나 내부 파라미터/변수값 설정을 위한 것이다. 처음 한번만 수행한다.
   - CONTROL : 매 샘플링마다 반복적으로 수행되는 제어 루틴이다.
   - END : 수행의 끝을 표시한다. 이 다음에 나오는 모든 코드들은 무시된다.
3) 기능 코드: 앞에서 설명한 바와 같게 구성된다.
4) 주석(Comment)문: '!' 다음부터 라인의 끝까지의 문자열로써 이해를 돕거나 설명을 위해 사용될 수 있다.
5) 빈 라인(Blank line): ASCII문자가 없는 라인이 여럿 존재하는 것을 허용한다.

Configuration file의 예는 다음과 같다.

```
! This is a example of configuration file
INITIALIZE
1 a = setvec ( 2 : 100., 2. )
2 b = setvec ( 2 : 0., 0. )
CONTROL
3 x = ain ( : 0, 1, 1)
4 b = ftov ( 2, x, 1 : )
5 b = ftov ( 2, a[1], 2 : )
6 c = addvec (2, a, b : )
END
```

여기서, 각각의 기능코드를 간단히 설명하면 setvec는 실수터를 설정하고, ain은 아날로그 입력을 포트로부터 받아들이고, ftov는 실수값을 벡터의 한 요소에 입력하고, addvec는 실수 벡터끼리의 합을 구하는 기능을 한다.

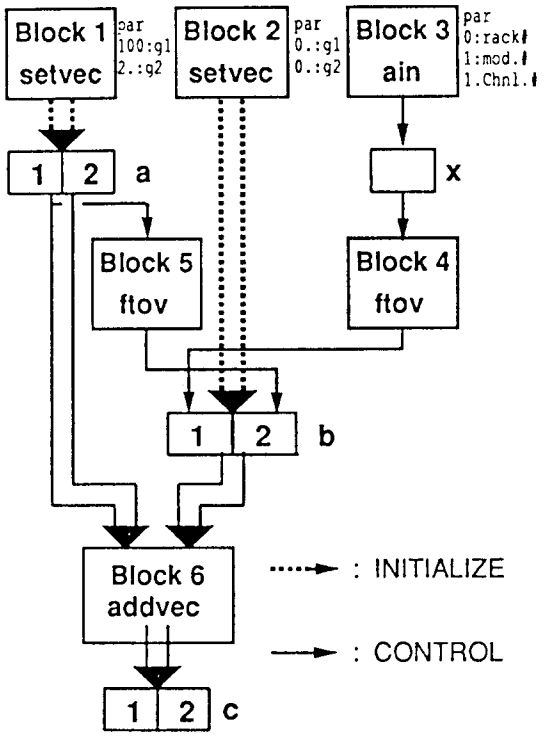


그림 3. 예제 configuration file에 대한 블럭 선도
Fig. 3. Block diagram of a configuration file.

기능블럭 1, 2와 3는 초기에 한번만 실수벡터들을 설정하는데, 벡터들의 메모리 할당을 겸한다. 포트넘버 (0,1,1)으로부터 아날로그 입력을 받아 x에 저장하고, b 벡터의 첫번째, 두번째 성분에는 각각 x, a 벡터의 첫번째 성분을 대입한다. 그리고 나서, a,b 두 벡터의 합을 c에 저장한다. 그림 2는 위의 configuration file에 대한 블럭선도를 나타낸다.

다음 표는 지금까지 구현된 기능코드들의 리스트를 나타낸다. 이외에도 성능이 우수한 고급 제어 알고리즘에 관련된 기능 코드들을 작성, 추가 등록하여 손쉽게 공정 제어기를 만들 수 있다.

표 1. 구현된 기능 코드들
Table 1. Developed function codes.

기능코드 이름	내 용
setvec	실수 벡터 초기화
setmat	실수 행렬 초기화
setivec	정수 벡터 초기화
setimat	정수 행렬 초기화
ftov	실수를 벡터의 한 요소로 세팅
itov	실수를 벡터의 한 요소로 세팅
addvec	실수 벡터의 합
addmat	실수 행렬의 합
mulmat	실수 행렬의 곱
addivec	정수 벡터의 합
addimat	정수 행렬의 합
mulimat	정수 행렬의 곱
trfun	임의의 차수 전달 함수(플랜트) 모사
rlsl	1차 플랜트의 계수 추정
rls2	2차 플랜트의 계수 추정
rlsn	임의의 차수 플랜트의 계수 추정
gpc1	1차 플랜트의 예측 제어
gpc2	2차 플랜트의 예측 제어
gpcn	임의의 차수 플랜트의 예측 제어
flc	퍼지 제어기
nn	신경회로망 제어기

일례를 들자면, 일반형 예측 제어기(GPC 제어기)를 다음과 같이 구현할 수 있다.

```
! This is the GPC example
INITIALIZE
1 p1 = setvec (4: 0.893, -0.17377, 0.2807,
                  0.0 )
7 p2 = setvec (4: 1.38533,-0.475367,
                  0.024465, 0.01905 )
CONTROL
! Plant
2 b  = first (u1: 2.0, 2.0, 0.0)
3 y1 = first (b: 5.0, 5.0, 0.0)
8 y2 = trfun (u2, 2, 1, 2: 3., 2., 1., 0., 0.)
! GPC controller
4 w  = msc (: 0.0)
5 sw = msc (: 0.0 )
6 u1 = gpc2 (w, y1, p1, sw, 0, 10, 1:
                  0., -10., 10., 0.1 )
9 u2 = gpcn (w,y2,p2,sw,0.3,2,10,1:
                  1.5.,-10.,10.,0.1 )
END
```

본 예제는 2개의 플랜트에 대해 2개의 일반형 예측 제어기를 각각 구현한 것이다. 먼저 2, 3번 블럭에 해당하는 첫번째 플랜트는 2/(s+2) 와 5/(s+5) 가 cascade로 연결된 형태이다. 각각 step-invariance method를 써서 이산화하였다. first 기능코드는 1차 시스템을 이산화시키는 기능을 갖는다. 시스템 파라미터는 1번 블럭에서 초기화되어 6번 블럭에서 예측 제어를 하게 된다.

8번 블럭에 해당하는 두번째 플랜트는 Runge-Kutta method를 이용하여 다차의 플랜트를 모사하는 trfun 기능코드를 이용하였으며, 1/(s+1)(s+2) 이다. 시스템 파라미터는 7번 블럭에서 초기화되어 9번 블럭의 예측 제어에 이용된다. 다음 그림 4는 set point (w)를 적절히 변화시켰을 때, 각각의 플랜트 제어 결과를 나타낸다.

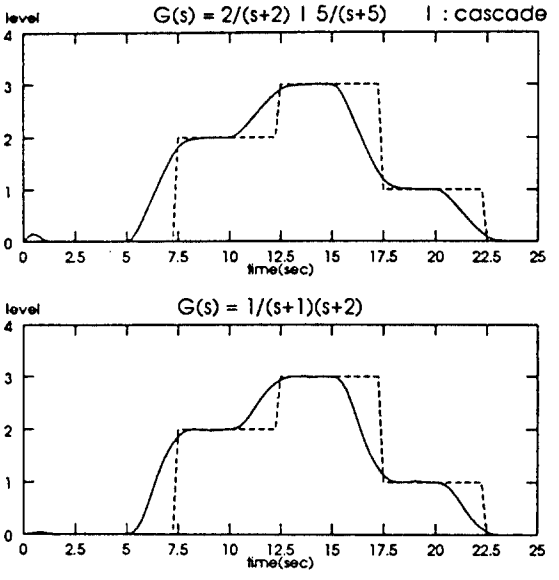
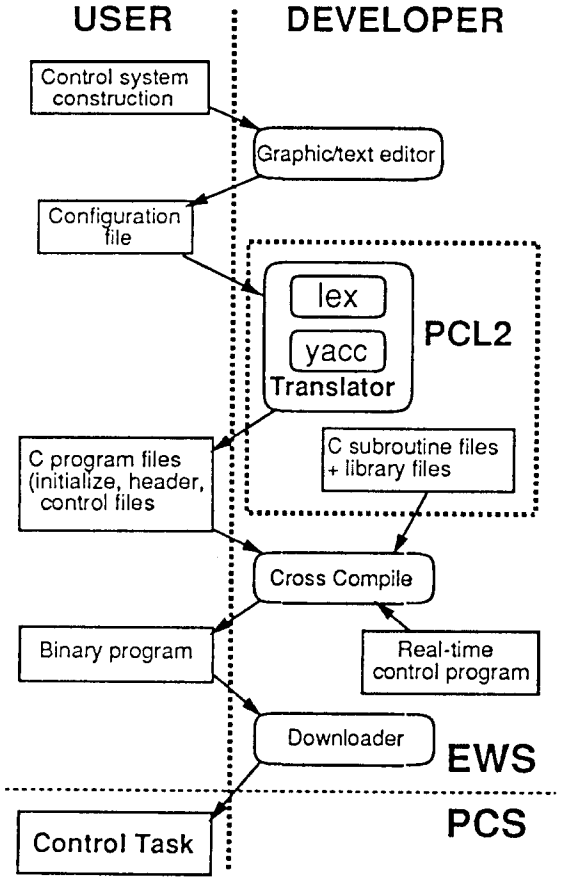


그림 4. GPC 제어 시뮬레이션 결과
Fig. 4. GPC control simulation result.

Ⅲ. 고급 제어 언어의 구현

본 논문에서 제안한 제어 언어의 구현 과정은 기존의 PCL1과 크게 다르지 않다. 기능코드들의 나열은 타겟 하드웨어(target hardware)에서 직접 이해할 수 없으므로, 그 하드웨어에 맞게 변환되어야 한다. 타겟 프로세서의 종류에 따라 독립적인 컴퓨터 코드가 필요하므로, 이식성을고려해 볼때, 프로세서 종류에 무관한 중간 변환 단계를 두어야 한다. 대표적인 예가 고급 언어의 일종인 언어이며, C 언어에서 여러 종류의 타겟 하드웨어에 맞는 코드로 바꿔주는 cross-compiler은 쉽게 구할 수 있다. 결국, 수행 순서는 일단 기능코드들의 나열로 된 파일에서 C 언어로 된 파일들을 생성해주며, 생성된 C 프로그램 파일들을 타겟 프로세서에 맞게 cross compile하여 최종적으로 타겟 프로세서에서 수행되게 된다.

그림 5. 제어 태스크 생성의 전체 흐름도
Fig. 5. Flow diagram about control task creation.



기능코드들의 나열로 된 파일을 C 언어 프로그램으로의 변환은 UNIX 제공 유틸리티인 어휘 분석기(lexical analyzer) lex와 구문 및 어의 해석기(syntax/semantic analyzer) yacc을 사용한다. 또한, 생성되는 언어 프로그램과는 별도로 각 기능코드에 알맞는 기능을 하는 서브루틴(subroutine)과 내부 변수를 사용할 경우 이를 초기화하는 루틴을 미리 개발자가 작성하여 제어 태스크 생성에 이용하게 된다. 제어 태스크 생성의 전체 흐름도는 그림 5과 같다.

Ⅳ. 결론 및 추후 연구 과제

본 논문에서는 고급 제어 알고리즘을 포함한 다중루프 제어 시스템의 구성을 위한 공정 제어 언어를 제안하였다. 제안된 언어는 공정 제어 알고리즘의 작성이 용이하고 새로운 기능코드들의 추가 등록이 가능한 configurable language를 이용한 것이 장점이다. 또한, C 언어 변환의 중간과정을 거치므로 타겟 하드웨어에 상관없는 독립적인 구조를 가져 flexibility가 향상되었다. 또한 입출력 변수의 사용, vector, matrix의 사용으로 인하여 진보된 제어 알고리즘의 구현에 매우 적합하다. 결국, 본 언어는 제어성능 향상이 요구되는 실제 시스템에 적용하여 좋은 결과를 낼 수 있으리라 기대된다. 그리고, 많은 고급 제어 알고리즘들을 추가하는 연구가 필요하다.

참 고 문 헌

[1] 한국 전력 공사 기술 연구원, "분산 제어 시스템의 고장 대처 기능 및 제어언어의 구현(최종보고서)", 1993. 2

[2] Bailey Controls, "Bailey Network 90 - Function Code Reference Manual".

[3] Bailey Controls, "C Language Implementation Guide for the Multi-Function Controller (NMFC03) - Software Manual".

[4] Eeldeink, "A Block-Diagram Language to Implement Controllers in a Distributed Computer Control Network," *Proc. 2nd IFAC/IFIP Symposium on Software for Computer Control*, 1979

[5] 김병국, "기능 블록 구성에 의한 공정 제어 언어의 개발", 대한전자공학회 논문지, Vol. 29, pp598-608, Aug. 1992

[6] 정현규, "제어언어를 이용한 다중루프 프로그램

형 제어기에 관한 연구", KAIST, 석사논문, 1988

[7] 김성우,서창준,김병국, "고급 제어 알고리즘을 위한 공정 제어 언어에 관한 연구", 제어 계측 연구회, 1994

저 자 소 개

金 聖 雨(正會員) 第 31卷 第 7號 B編 參照.
현재 한국과학기술원 전기 및 전자 공학과 박사과정 재학중

徐 暢 俊(正會員) 第 29卷 第 12號 B編 參照.
현재 한국과학기술원 전기 및 전자공학과 박사과정 재학중

金 炳 國(正會員) 第 29卷 第 8號 B編 參照.
현재 한국과학기술원 전기 및 전자 공학과 부교수