技術論文 한국정밀공학회지 제13권 제4호 (1996년 4월)
Journal of the Korean Society of Precision Engineering Vol. 13. No.4. April. 1996.

그래픽 조직 언어를 이용한 순차 제어용 프로그래밍 시스템 개발

국 금 환*

Development of a Programming System for Sequential Control Using a Graphic Organization Language

Kum-Hoan Kuk*

ABSTRACT

PLCs are vital components of modern automation systems, which have penetrated into almost every industry. Many industries have a demand for facilitation of PLC programming. In this study, a programming system for sequential control is developed on a personal computer. This programming system consists of two main parts, a GRAFCET editor and a GRAFCET compiler. The GRAFCET editor enables us to model an actual sequential process by a GRAFCET diagram.This GRAFCET editor is developed by the menu-driven method based on specific menus and graphic symbols. The GRAFCET compiler consists of two parts, a GRAFCET parser and a code generator. The possible errors in a drawn GRAFCET diagram are first checked by the GRAFCET parser which generates finally an intermediate code from a verified GRAFCET diagram. Then the intermediate code is converted into a control code of an actual sequential controller by the code generator. To show the usefulness of this programming system, this system is applied to a pneumatically controlled handling robot. For this robot, a Z-80 microprocessor is used as the actual sequential controller.

Key Words : Sequential Control(순차 제어), PLC, Graphic Organization Language(그래픽 조직 언어), Programming System (프로그래밍 시스템), Z-80

1. 서 론

최근에 PLC(programmable logic controller)는 공정 및 공장 자동화를 위한요소로서 그 중요성이 더욱 커지고 수요도 계속 확대되고 있다.[1] 또한 컴퓨터 발전에 힘입어 성능 향상과 소형화 및 저가격화도 계속 추진되고 있다. 현재는 PLC H/W 발달에 비해 오히려 PLC S/W 발달이 뒤진 실정이다.[2-3] 타 자동화 요소와 마찬가지로

* 경상 대학교 제어계측공학과, 자동화 및 컴퓨터 응용 기술 연구소(정회원)

PLC 상호간의 H/W, S/W 측면의 호환성이 점차 문제점으로 부각되고 있으며, PLC 의 고기능화로 프로그래밍 작업이 복잡해 지고 PLC 프로그램 용량이 커짐에 따라 PLC 프로그램의 작성, 테스트, 보수 및 관리 문제가 점차 커지게 되었다. 기존의 PLC 프로그래밍 관련 연구는 다음의 세 가지 접근으로 분류 할 수 있다. 첫 번째 접근은 PLC 프로그래밍을 비교적 손쉽게 접속 가능한 독립된 소형 전용 장치로 수행하는 것이다.[4] 이 장치는 단순한 연산부터 다양한 제어 알고리즘 처리까지, 진단 기능부터 타 자동화 설비에 데이터를 전송하는 기능까지 그 기능이 무척 다양하다.

이러한 장치는 복잡한 프로그래밍 작업을 단순한 파라메터 설정 작업으로 대치시킨다. 두 번째 접근은 통일된 언어와 하나의 범용 프로그래밍 장치에 의하여 서로 상이한 PLC 들의 제어 프로그램을 작성하는 방법이다.[5-6] 이 때의 PLC 프로그래밍 작업은 단순히 마크로 명령들을 정의해 주는 작업이 된다. 이 방법을 위하여 먼저 이러한 마크로 형태의 표준 명령들을 정의해 주는 작업이 필히 선행되어야 하며 또한 이 표준 명령들로 작성된 제어 프로그램을 개별 PLC 가 읽을 수 있는 프로그램 형태로 바꾸어 줄 수 있는 개별 PLC 전용의 포스트프로세서들이 요구된다. 세 번째 접근은 범용 컴퓨터 지원에 의하여 순차 제어기 선정(개발)부터 프로그래밍 및 도큐멘테이션까지 통일적으로 작업하는 방법이다.[7] 이 방법의 경우 주어진 제어 대상에 대한 최적의 순차 제어기를 신속히 선정하는 작업이 선정된 제어기의 프로그래밍 작업 못지 않게 중요하게 취급된다.이때 순차 제어 작업내용은 컴퓨터 모니터 상에서 각종 그래픽 언어로서 표현된다.

첫째와 두 번째 접근은 순차 제어기 생산 업체에서 주로 추진해 온 방법이고 세 번째 접근은 순차 제어기 이용 업체(각종 자동화 기계 제작 업체, 자동차 생산 업체 등)들이 업체 자체 전용의 합리적 도구 개발로서 추진 해 온 방법이다.

본 연구의 목적은 순차 제어 작업 흐름을 PC(personal computer) 를 이용하여 그래픽조직 언어들 중 하나인 GRAFCET 언어로 모델링한 후 작성된 모델로부터 순차 제어기의 제어 프로그램을 직접 생성해 주는 프로그래밍 시스템을 개발하는 것으로, 상기 세 번째 접근 방법의 연구 범위 중 최적 제어기 선정(개발) 부분을 제외한 컴퓨터 응용 순차 제어기 프로그래밍 연구 부분에 해당된다.

현재 래더다이아그램을 이용한 PLC 프로그래밍과 관련 S/W 개발이 주류를 이루고 있으며 GRAFCET, SFC 등의 그래픽 조직 언어는 아직 제어 관련 기술자들에게 친밀한 언어가 아니다.[8-10] 그러나 점차 순차 제어 작업의 규모와 복잡도가 커지고 있으므로 래더다이아그램에 비해 모델링 능력에서 보다 유연성이 크고 주로 순차 제어 대상 작업 흐름에 대한 지식만 요구하는 그래픽 언어인 GRAFCET, SFC 가 PLC 프로그래밍 언어로 더욱 광범위하게 보급될 것이 예상된다.[11]

2. 그래픽 조직 언어와 페트리 네트

기존 PLC 언어는 명령어를 직접 이용하는 텍스트 형식 언어와 그래픽 심벌을 이용하는 그래픽 언어로 크게 나뉘어지며 그래픽 언어는 다시 래더다이아그램과 같은 그래픽 기본 언어와 GRAFCET(Graphic de Commande Etape Transition), SFC(sequential function chart)등과 같은 플로 차트 형식의 그래픽 조직 언어(GOL:graphic organization language)로 나뉘어진다. 현재의 언어 개발 추이는 컴퓨터 지원이 보다 효과적이고 순차 제어 작업을 포괄적으로 기술 할 수 있는 그래픽 심벌을 활용하는 언어 개발이 주가 되고 있다. 즉, 프로그래밍 작업이 제어의 기술적 실현성을 직접 기술하는 것보다는 제어 기술과 독립된 제어 환경의 기술이 주가 된다. 상기 세 가지 언어를 상호 비교해 보면,

순차 제어기 자체에 대한 전문 지식 요구의 최소화, 제어 대상의 작동 내용 으로부터의 독립성, 제어 흐름 이해의 용이성, 제어 흐름과 복잡한 논리 결합 표현의 용이성 등의 측면에서 텍스트 형식 언어, 래더다이아그램, 플로 차트 형식 언어 순으로 기능이 점차 우수해진다.

그 동안 페트리 네트(Petri Net)는 자동화의 경우 생산 시스템의 모델링과 해석에 주로 이용되어 왔으나 최근 생산 시스템의 제어를 위한 도구로도 이용되고 있다.[12-15] 페트리 네트는 한 시스템의 동적 특성과 구조 분석을 위하여 이용 할 수 있는 유방향 그래프로서 지금까지 순차 제어 프로그래밍시 주로 사용해 온 래더다이아그램에 비해 순차 흐름이나 비동기 공정을 직접 표현할 수 있는 장점을 가지며 이론적으로도 정립되어 있다. GRAFCET 은 페트리 네트를 변형한 것으로서 이산 사건 흐름의 모델링 능력에서는 페트리 네트와 동일하며 순차 제어 흐름의 모델링에 보다 적합하도록 GRAFCET 고유의 전용 기능으로서 각 스텝(step)에 액션(action)을 부가하고 트랜지션(transition)에 조건 함수로서 불리안 표현을 부가 할 수 있는 장점을 가지고 있다.

GRAFCET와 페트리 네트를 상호비교하면 GRAFCET의 기본 요소(step, transition, link)와 페트리 네트의 기본 요소(state, event, arc) 사이에 하나의 대응 관계가 존재하며, GOL의 전개상 기본 요소(single sequence, simultaneous sequence, divergence, convergence)와 페트리 네트의 전개상 기본 요소 사이에도 일정한 대응 관계가 존재한다. 또한 타이머 기능과 같은 특수한 기능도 대응하는 페트리 네트로 나타낼 수 있다. 이러한 대응 관계에 의하여 임의 GOL(GRAFCET,SFC 등) 다이아그램도 페트리 네트 그래프로 치환 할 수 있다. 본 연구에서 개발하는 프로그래밍시스템의 기본 데이터 구조는 페트리 네트 모델링시 이용되는 보다 일반적인 데이터 구조를 채택함으로서 기존 GOL의 모델링뿐만 아니라 필요시 최소의 비용으로 래더다이아그램 작성 프로그램, 유공압 회로 작성 프로그램과 접속시킬 수 있는 가능성를 내장시켰다 (Fig. 1). 또한 페트리 네트와 같은 형태의 모델링은 동적 메모리 할당 기능을 갖고 망상 관계 표현이 손쉬운 포인터를 갖는 PASCAL, C 등이 적합하여 본 연구에서는 C를 개발 언어로 선정하였다.

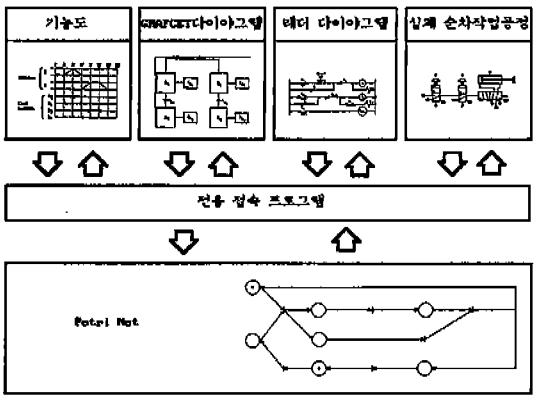

Fig. 1 Petri Net as a common internal model

3. 프로그래밍 시스템의 구조와 그래픽 심벌

3.1 프로그래밍 시스템의 구조

GRAFCET 다이아그램을 PC 모니터 상에서 작성한 후 작성된 이 다이아그램을 컴파일하여 중간 코드를 생성하고 이 중간 코드로부터 개별 순차 제어기의 제어 코드를 생성 할 수 있는 프로그래밍 시스템의 구조를 Fig. 2와 같이 설계하였다. Fig. 2 의GRAFCET Editor 는 개별 순차 제어기와는 무관하고 오로지 제어 과제 자체와 제어 흐름 내용에 의존한다. GRAFCET Compiler 는 개별 순차 제어기와는 독립인 Parser 와 개별 순차 제어기에 직접 의존하는 전용 Code Generator 로 구성된다.

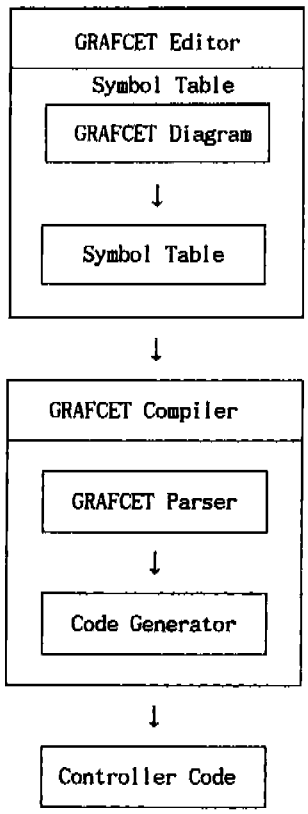

Fig. 2 Structure of the programming system

3.2 GRAFCET 언어의 그래픽 심벌

GRAFCET은 기본적으로 step, action, transition, link 로 구성된다. Step 은 정사각형으로 표시하며 동작의 단계를 구분하는데 사용된다. Action 은 직사각형으로 표시하며 임의의 step 에서의 복수 action 은 복수 개의 직사각형을 계속 연결하여 나타낸다. Transition 은 십자 형태로 표시하며, 바로앞 step 이 활성화되었을 때 그 step 에 상당하는 action 의 완료나 상태 변화를 감지하여 다음 step 으로 진행하기 위한 천이 조건 감지에 이용된다. 즉 천이는 오로지 활성화된(active) step 으로 부터 발생하며, 일단 천이가 발생하면 다음 step 이 활성화

되고 직전의 step 은 비활성화(inactive) 된다. Link 는 step 과 transition 을 연결하는 요소로서 루프의 형성, 조건분기 및 동시진행등을 나타내는데 사용된다.

Fig. 3 는 GRAFCET 으로 작성된 프로그램의 한 예를 보여준다.

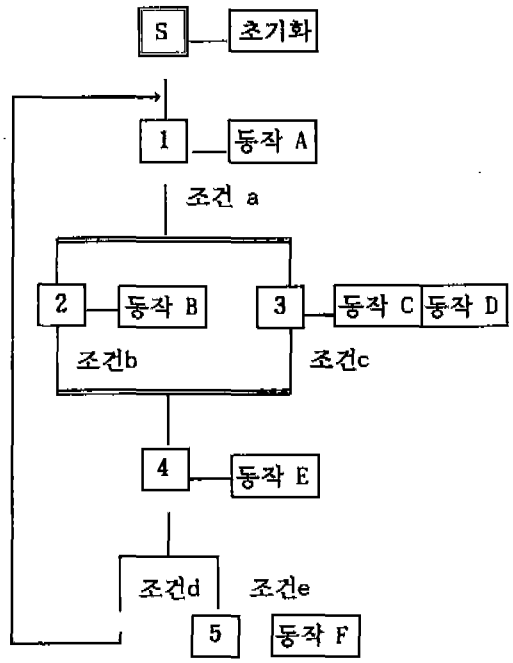

Fig. 3 Program example described in GRAFCET language

프로그래밍 시스템의 개발을 위하여 일차적으로 GRAFCET 다이야그램 구성요소들 (step, action, transition, link, 여러 접속 자들)이 대응하는 화면상 그래픽 심벌로 정의되어야 하며, 정의된 그래픽 심벌들을 컴퓨터 내부 모델로 나타내기 위하여 전술한 페트리 네트 데이터 구조를 기초로 각 요소에 대응하는상세한 데이터 구조가 확정되어야 한다.이를 위해 GRAFCET 에디터로 다양한 GRAFCET 다이야그램들을 PC 화면상에서 손쉽게 작성하기 위하여 Fig. 4 와 같은 15 개의 그래픽 기본 심벌들을 정의 하였다. 이것들은 각각 다음과 같은 목적에 이용된다.

- INIT : 초기화의 step에 이용됨
- STEP, ACTION, TRAN : 각각 처리단계, 실제의 작업내용 및 천이 조건을 나타냄

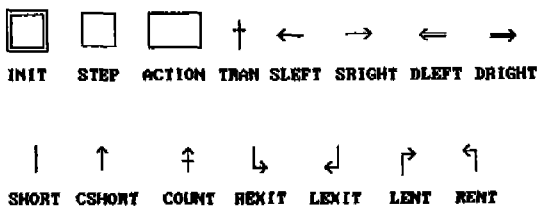

Fig. 4 Graphic symbols for GRAFCET editor

- SLEFT, SRIGHT : 분기(branch)와 경로(path)를 나타냄
- DLEFT, DRIGHT : 동시 진행 과정에서 상하의 과정을 연결하는데 이용됨
- CSHORT, COUNT : 루프(loop)를 형성하기 위해 위로 계속 연결하는데 이용되고, COUNT는 루프의 계속 진행 조건을 나타내는데 이용됨
- LEXIT, REXIT : 루프의 시작을 나타내기 위한 곳에 이용됨
- LENT, RENT : 루프 형성의 목적지를 나타내는데 이용됨

이들 심벌들 중 SLEFT, SRIGHT, DLEFT, DRIGHT 에는 화살표가 부착되어 있으나 이것은 단지 그 방향을 구분하는 것에 불과하고 실제 화면상에는 나타나지 않는다.

3.3 그래픽 심벌의 자료 구조

그래픽 심벌들은 이진 트리(binary tree)의 자료 구조를 갖고 Table 1 과 같이 정의하였다. 이 자료 구조의 각 변수가 갖는 의미는 다음과 같다.

- TYPE : 심벌의 이름을 나타냄
- X, Y : 화면상에서 심벌의 기준 점의 위치로서 화면의 절대 좌표로 나타냄
- PAGE, WIDTH : 심벌을 화면에 나타낼 때의 구간 블록으로서 실제의 (PAGE×WIDTH)는 (30×10)으로 한정함
- PRX1, PRY1, PRX2, PRY2 : 오른쪽 자노드(child node)와 왼쪽 자노드의 좌표값
- MESSAGE : 심벌에 기입되는 메시지를 저장
- LCHILD, RCHILD : 왼쪽과 오른쪽의 자노드를 가리키기 위한 포인터

각 그래픽 심벌마다 오른쪽 자노드(child node)와 왼쪽 자노드가 부착되는 일정한 위치를 갖는다.

Table 1 The Data structure

TYPE	
X	Y
PAGE	WIDTH
PRX1	PRY1
PRX2	PRY2
MESSAGE	
LCHILD	RCHILD

3.4 그래픽 심벌의 연결

GRAFCET 은 순차 제어 흐름을 기술하기 위한 하나의 언어로서 각 그래픽 심벌들이상호 연결 될 때 일반적인 컴퓨터 언어와 같이 각 심벌들 사이의 연결 문법이 존재 한다. 구체적으로 어떤 심벌은 하나의 자노드만 가지도록 제한되며, 또 심벌들간의연결들도 실제적으로 허용되는 연결만 Fig. 5 와 같이 한정된다. 프로그래밍 시스템이 이러한 정보를 이용하여 프로그래밍 시스템 사용자가 GRAFCET 에디터로 하나의 GRAFCET 다이야그램 작성 할때 실수로 연결 불가능한 심벌들간의 연결을 시도하는 경우 이러한 시도를 억제한다.

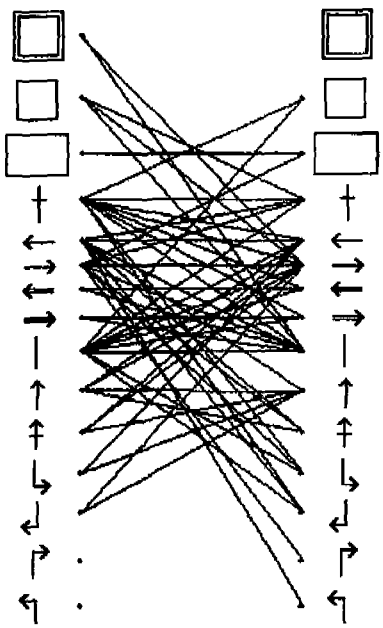

Fig. 5 Possible connection between the graphic symbols

4. GRAFCET 에디터

GRAFCET 다이야그램 그리기 위한 대화형 그래픽 에디터의 전체적인 화면구성은 Fig. 6 과 같다. DRAW FIELD 는 GRAFCET 다이야그램이 그려지는 부분으로 한 화면에 나타나는 분량을 한 페이지라 정의 할 때 (30

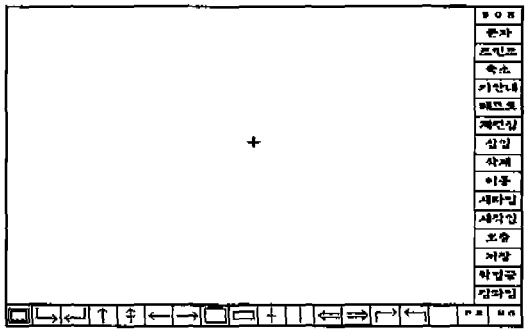

Fig. 6 Menu design of graphic editor

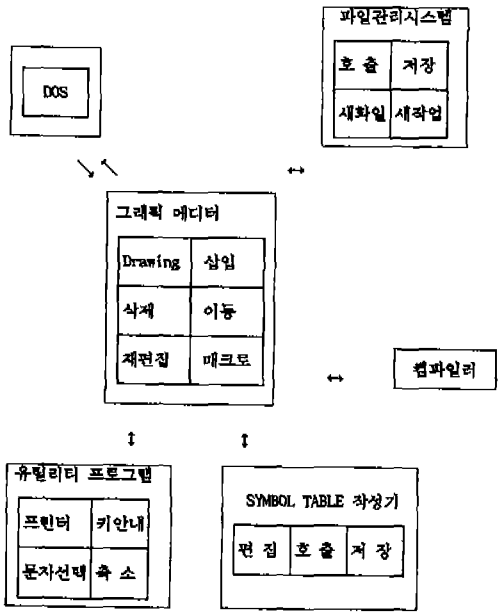

Fig. 7 Working environment of GRAFCET editor

× 10) 페이지의 분량을 스크롤(scroll)하며 그릴 수 있다. 이때 STATUS FIELD 는 화면에 나타난 페이지를 나타낸다.

DRAW SYMBOL FIELD 는 GRAFCET 다이야그램 그리기 위한 기본 심벌들을 그려넣은 부분으로 사용자가 키조작에 의해 선택 할 수 있으며, 이때 선택된 심벌은 SELECTION FIELD 에 표시된다. MENU FIELD 에는 사용 할 수 있는 모든 명령들이 표시된다.

GRAFCET 다이야그램 그리기 위한 대화형 그래픽 에디터의 작업환경은 Fig. 7 과같다. 그림과 같이 GRAFCET 다이야그램의 저장과 호출은 화일 관리 시스템이 담당한다. 또한 새작업 기능에 의해 새로운 GRAFCET 다이야그램을 추가 작성 할 수 있으며 파일 이름도 새파일 기능에 의해 변경 될 수 있다. 작업시 유틸리티 프로그램의 도움으로 문자선택, 키안내, 전체 GRAFCET 다이야그램 축소가 가능하고 필요시 프린트도 가능하다.

에디터로 GRAFCET 다이야그램 작성 작업을 완료한 후에 심벌 테이블을 작성해야 한다. 이 심벌 테이블에는 GRAFCET 다이야그램 ACTION 과 TRAN DRAW SYMBOL 의 동작코드에 대한 동작 내용이 한글과 영문으로 기술된다. 심벌 테이블에는 GRAFCET 의 ACTION 에 상당하는 기기의 동작을 기술하여 순차 제어기에서 각 작동기(actuator)에 신호를 전송하거나 TRANSITION 에 상당하는 센서의 신호를 검출 할 수 있도록 표로 만들어 참조 할 수 있도록 한 것이다. 심벌 테이블 각 행의 콜론(colon) 우측에는 I/O 번호를 기입한다. 이곳의 +/- 기호는 센서나 스위치의 정상 상태에서의 ON/OFF 를나타낸다. 따라서 이러한 심벌 테이블을 이용하기 위하여 먼저 I/O 번호를 정의한 후 I/O 포트(port)와 각 포트의 비트(bit) 를 입/출력으로 구분하는 I/O 정의 표가작성되어야 한다. 심벌 테이블 작성용 에디터는 Fig. 8 과 같다.

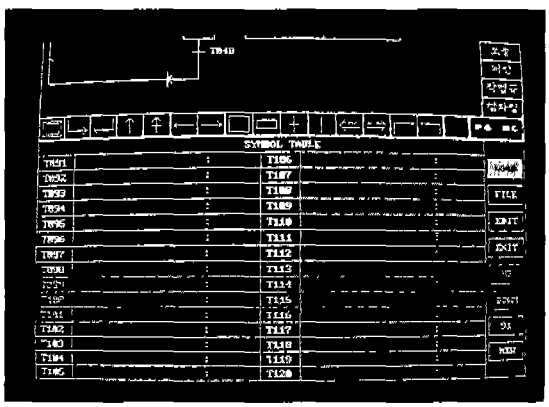

Fig. 8 Symbol table editor

5. GRAFCET 컴파일러

GRAFCET 컴파일러는 작성된 GRAFCET 다이야그램 전체적 구조를 파악하고 에러를 검출한 후 중간 코드를 생성해 내는 파저(parser)와 이 중간 코드로 부터 순차 제어기의 제어 코드를 생성하는 코드 발생기(code generator)로 구성된다(Fig. 9). 파저는 GRAFCET 다이야그램에 대한 이진 트리 를 탐색하여 중간 코드 생성시 필요한 분기의 목적지에 대한 레이블(label) 을 생성하고 심벌간의 연결 관계에서 체크되지 못한 의미상 에러를 검출한다. 코드 발생기는 먼저 시스템 초기화 코드를 생성한 후 중간 코드 차례로 해석하여 I/O 정의 표와 심

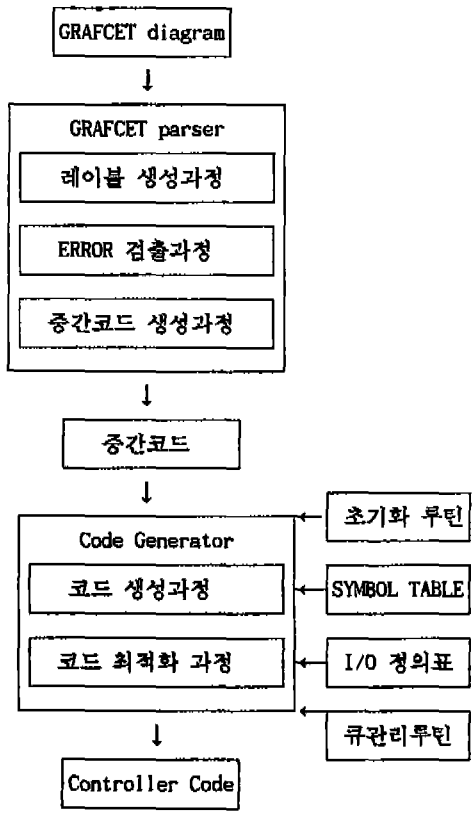

Fig. 9. Structure of GRAFCET compiler

벌 테이블에 수록된 I/O 포트 번호 및 센서 등의 정보를 이용하여 각 동작 코드에 상당하는 순차 제어기의 제어 코드를 생성해 낸다. 또한 동시진행 과정의 시작과 병합을 큐를 이용하여 올바르게 처리하도록 큐를 초기화시키는 과정 및 큐 작동 루틴 등을 추가한다. 본 연구에서는 PC 와의 제어 파일 전송 용이성을통한 프로그래밍 시스템 개발을 위한 실험의 용이성 과 순차 제어기로서의 범용성 때문에 기 상품화된 특정 PLC 가 아닌 Z-80 마이크로 프로세서(모든 기술적 정보가 개방됨)를 순차 제어기로 선정하고 일차적으로 이 제어기를 위한 전용 코드 발생기를 개발하였다. Z-80 의 명령어 집합(논리/산술연산, 입출력, 저장/호출)이 PLC 의 주요 기본 명령들을 모두 포함하기에 이 코드 발생기는 필요시 특정 PLC 전용 코드 발생기로 개조 할 수 있다.

6. 공압 핸들링 로봇에 적용

Fig. 10 과 같은 구조의 공압 핸들링 로봇과 작업물의 반입/반출을 순차 제어하기 위한 제어 코드를 생성하기

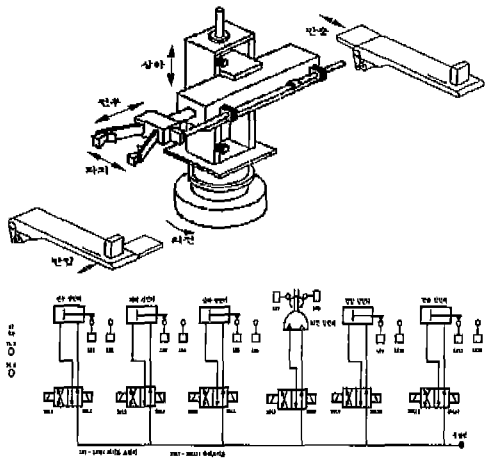

Fig. 10 Handling robot system

Table 2 Time chart of the handling robot system

동작 / 실린더	1	2	3	4	5	6	7	8	9	10
전후 실린더										
파지 실린더										
상하 실린더										
회전 실린더										

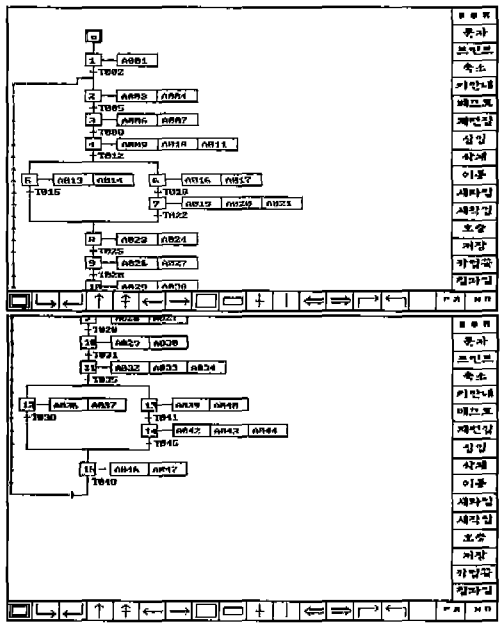

Fig. 11 GRAFCET diagram of the handling robot system

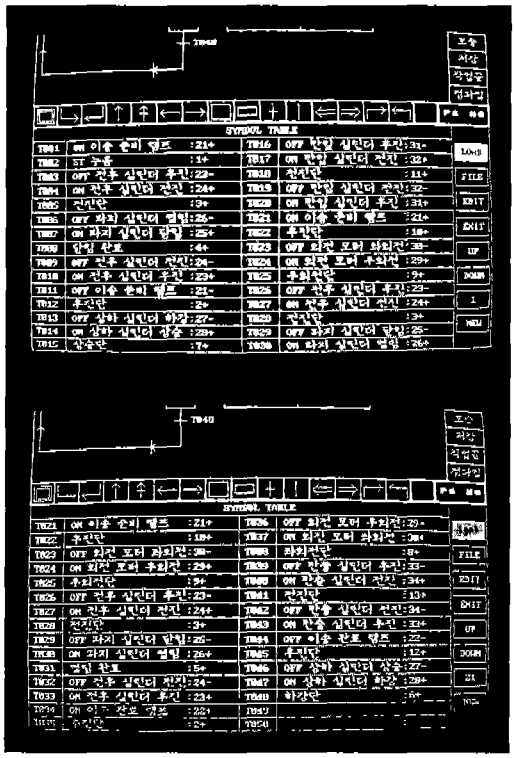

Fig. 12 Symbol table of the handling robot system

(중간 코드)

```
        A001
        T002
JMP0:
        A003
        A004
        T005
        A006
        A007
        T008
        A009
        A010
        A011
        T012
        S015
        JP    BRCH0
        A023
        A024
        T025
        A026
        A027
        T028
        A029
        A030
        T031
        A032
        A033
        A034
        T035
        S038
        JP    BRCH1
        A046
        A047
        T048
        JP    JMP0
BRCH1:
        S
        A039
        A040
        T041
        A042
        A043
        A044
        T045
BRCH1:
        S
        A016
        A017
        T018
        A019
        A020
        A021
        T022
```

(제어 코드)

```
;--------SYSTEM INITIALIZE
        RORG 0000H
        LD   SP,2800H
        LD   IX,2000H
        LD   (QFRONT),IX
        LD   (QREAR),IX
;--------INITIALIZE ROUTINE
        LD   A,9BH
        OUT  (13H),A
        OUT  (23H),A
        LD   A,80H
        OUT  (33H),A
        LD   A,FFH
        OUT  (31H),A
        OUT  (32H),A
        OUT  (30H),A

        IN   A,(12H)
        SET  4,A
        OUT  (12H),A
WAIT0:  IN   A,(10H)
        BIT  0,A
        JP   Z,WAIT0
JMP0:   IN   A,(32H)
        RES  7,A
        OUT  (32H),A
        IN   A,(12H)
        SET  7,A
        OUT  (12H),A
WAIT1:  IN   A,(10H)
        BIT  2,A
        JP   Z,WAIT1
        IN   A,(20H)
        RES  1,A
        SET  0,A
        OUT  (20H),A
WAIT2:  IN   A,(10H)
        BIT  3,A
        JP   Z,WAIT2
        IN   A,(12H)
        RES  7,A
        OUT  (12H),A
        IN   A,(32H)
        SET  7,A
        OUT  (32H),A
        IN   A,(12H)
        RES  4,A
        OUT  (12H),A
WAIT3:  IN   A,(10H)
        BIT  1,A
        JP   Z,WAIT3
        IN   A,(10H)
        BIT  6,A
        JP   Z,BRCH0
        IN   A,(20H)
        RES  5,A
        SET  4,A
        OUT  (20H),A
WAIT4:  IN   A,(11H)
        BIT  0,A
        JP   Z,WAIT4
        IN   A,(32H)
        RES  7,A
        OUT  (32H),A
        IN   A,(12H)
        SET  7,A
        OUT  (12H),A
WAIT5:  IN   A,(10H)
        BIT  2,A
        JP   Z,WAIT5
        IN   A,(20H)
        RES  0,A
        SET  1,A
        OUT  (20H),A
WAIT6:  IN   A,(10H)
        BIT  4,A
        JP   Z,WAIT6
        IN   A,(12H)
        RES  7,A
        OUT  (12H),A
        IN   A,(32H)
        SET  7,A
        OUT  (32H),A
        IN   A,(12H)
        SET  5,A
        OUT  (12H),A
WAIT7:  IN   A,(10H)
        BIT  1,A
        JP   Z,WAIT7
        IN   A,(10H)
        BIT  7,A
        JP   Z,BRCH1
        IN   A,(20H)
        RES  2,A
        SET  3,A
        OUT  (20H),A
WAIT8:  IN   A,(10H)
        BIT  5,A
        JP   Z,WAIT8
        JP   JMP0
BRCH1:  IN   A,(12H)
        BIT  7,A
        JP   NZ,END
        IN   A,(21H)
        RES  0,A
        SET  1,A
        OUT  (21H),A
WAIT9:  IN   A,(11H)
        BIT  4,A
        JP   Z,WAIT9
        IN   A,(21H)
        RES  1,A
        SET  0,A
        OUT  (21H),A
        IN   A,(12H)
        RES  5,A
        OUT  (12H),A
WAIT10: IN   A,(11H)
        BIT  3,A
        JP   Z,WAIT10
BRCH1:  IN   A,(11H)
        BIT  3,A
        JP   Z,END
        IN   A,(20H)
        RES  6,A
        SET  7,A
        OUT  (20H),A
WAIT11: IN   A,(11H)
        BIT  2,A
        JP   Z,WAIT11
        IN   A,(20H)
        RES  7,A
        SET  6,A
        OUT  (20H),A
        IN   A,(12H)
        SET  4,A
        OUT  (12H),A
WAIT12: IN   A,(11H)
        BIT  1,A
        JP   Z,WAIT12
END:    HALT
;
; < QUEUE ADDITION ROUTINE >
; INPUT DATA SHOULD BE GIVEN
; IN REGISTER BC.
;
QFRONT  EQU  2100H
QREAR   EQU  2102H
QADD:   PUSH HL
        PUSH AF
        LD   HL,(QREAR)
        INC  L
        INC  L
        LD   A,10H
        CP   L
        JP   NZ,Q1
        LD   L,00H
Q1:     LD   A,L
        LD   (QREAR),HL
        LD   HL,(QFRONT)
        CP   L
        JP   NZ,Q2
        JP   EMGNCY
Q2:     LD   HL,(QREAR)
        LD   (HL),C
        INC  L
        LD   (HL),B
        POP  AF
        POP  HL
        RET
;
; < QUEUE DELETE ROUTINE >
; DATA RETURN IN REGISTER IY
;
QDEL:   PUSH HL
        PUSH DE
        PUSH AF
        LD   HL,(QREAR)
        LD   A,L
        LD   HL,(QFRONT)
        CP   L
        JP   NZ,Q3
        JP   EMGNCY
Q3:     INC  L
        INC  L
        LD   A,10H
        CP   L
        JP   NZ,Q4
        LD   L,00H
Q4:     LD   E,(HL)
        INC  L
        LD   D,(HL)
        PUSH DE
        POP  IY
        POP  AF
        POP  DE
        POP  HL
        RET
;
; QUEUE OPERATION ROUTINE
;
QOP:    CALL QADD
        CALL QDEL
        JP   (IY)
```

Fig. 13 Intermediate and control code of the handling robot system

위하여 개발한 프로그래밍 시스템을 적용한 예를 보이고자 한다.

이 핸들링 시스템의 동작 순서는 다음과 같다. 먼저 작동 준비 램프(PL R)가 켜지면 ST 스위치를 눌러 공압 로봇을 작동시킨다. 로봇의 네 개 실린더의 동작은 각각 2 개의리미트 스위치를 통해 확인되며 Table 2 의 타임 차트에 의하여 동작한다. 작업물 반입 실린더와 반출 실린더의 동작도 소속된 두 개의 리미트 스위치를 통해 확인된다. ST 스위치와 모든 리미트 스위치는 제어기 입력 포트와 연결되고 두 개의 파이롯트램프(PL R, PL E)와 모든 솔레노이드는 제어기 출력 포트와 연결된다. Fig. 11 은 주어진 핸들링 시스템의 동작에 대응하는 GRAFCET 다이야그램이고 Fig. 12 는 GRAFCET다이야그램에 대응하는 심벌 테이블이다. Fig. 13 은 GRAFCET 컴파일러의 출력인 중간 코드와 순차 제어기의 제어 코드이다.

7. 결 론

본 연구에서 순차 제어 작업 흐름을 PC(personal computer) 을 이용하여 그래픽조직 언어들 중 하나인 GRAFCET 언어로 모델링한 후 작성된 모델로부터 순차 제어기의 제어 코드를 직접 생성해 주는 프로그래밍 시스템을 개발하였다. 이 시스템은 VGA 그래픽 보드가 장착된 IBM PC/AT 이상의 컴퓨터에서 이용될 수 있다. 프로그래밍시스템의 사용자는 순차 동작 내용, I/O port 번호, I/O bit 번호, 각 스위치 동작 방식 등을 심벌 테이블과 I/O 정의 표에 정의해 준 후 GRAFCET 다이야그램은 이 테이블과 정의 표만을 의지하여 작성 할 수 있다. 심벌 테이블과 I/O 정의 표 작성은 사용자에게 순차 제어기 자체와 제어공정 흐름에 대한 어느 정도의 기술적 지식을 요구하지만 이러한 요구를 현 개발한 프로그래밍 시스템에서 피할 수 없는 실정이다. 물론 이러한 지식들은 래더다이야그램 작성 시에도 필히 요구되는 지식이다. 반면 GRAFCET 다이야그램 작성시 래더다이야그램처럼 요소 기호 상호간의 의존성을 고려할 필요가 없고 내장된 GRAFCET 의문법(syntax) 체크 지원을 받으므로 보다 손쉽게 GRAFCET 다이야그램 작성 할 수 있다는점이 기존 래더다이야그램 방법 보다 상대적으로 우수한 본 연구의 특징이다. GRAFCET parser 가 작성된 GRAFCET 다이야그램 의미상 에러를 체크해 주므로 이상의과정을 통해 모든 가능한 에러를 제거한 후 코드 generator 에 의하여 요구되는 순차제어 코드가 생성된다. 그러나 프로그래밍 시스템에 현재 내장된 기능 자체가 제어 코드의 실용성(비상 정지, 안전 운전 등)을 높여 줄 수는 없다.

본 연구를 통해 국내 PLC 생산 업체 측면에서 PLC의 국내 고유 모델 개발시 PLC의 man/machine interface 기능을 GRAFCET을 이용하여 자체 개발 할 수 있는 가능성을 보였으며, PLC나 타 순차 제어기 사용자 측면에서 순차 제어기의 작업 프로그램 작성 작업이 주로 개별 작성자의 경험과 습관에 크게 의존함에 따라 후속 제어기 보수 및 유지 작업이 어려운 현재의 문제점을 그 자체가 documentation을 갖는 GRAFCET을 사용함으로서 제어기 작업 프로그램 작성 작업이 보다 효율적으로 추진 될 수 있음을 보였다. 특히 본 연구는 이를 통해 순차 제어기 제작자나 사용자 모두에게 순차 제어기 제작 기간과 비용을 절감 할 수 있는 하나의 방법이 될 수 있다.

앞으로의 과제는 사용하는 순차 제어기와 순차 제어 대상 공정에 대한 기술적인 지식이 부족한 사용자를 도와주는 기능과 실제 경험이 적은 사용자도 보다 실용적인제어 코드를 생성 할 수 있도록 도와주는 기능을 추가하는 것과 GRAFCET 다이야그램 래더다이야그램으로 번역해 주는 기능을 추가 개발하는 것이다. 또한 컴퓨터 지원에 의한 최적 순차 제어기 선정 및 개발 지원 프로그램 시스템의 개발도 미래의 과제이다.

후 기

본 논문은 1993 년도 교육부 지원 한국학술진흥재단의 신진교수과제 학술연구조성비에 의하여 연구되었으며, 연구비를 지원해 주신 교육부에 대하여 감사의 뜻을 표합니다.

참 고 문 헌

1. D. W. Pessen, "Industrial Automation", John Wiley & Sons, pp.360-384, 1990.
2. N.U. Gunasena, E.A. Lehtihet, "Automatic Program Generation for PC-Based Sequence Control Problems", Journal of Manufacturing Systems, Vol. 10, No. 2, pp. 109-120, 1991.
3. 임 동진, "자동화 기기용 Embedded System Software 의 개발 동향", '92 로보틱스 및 자동화

연구회 Workshop 논문집, 1992.
4. User' Manual, "Samsung Programmable Controller, PGM300A"
5. "GRAFCET: a Functional Chart for Sequential Processes", ADEPA, France, 1979.
6. R. Mittmann, "Unabhaengiges Programmieren", Markt & Technik, 1982.
7. T. Oestreicher, "Rechnergestuetzte Projektierung von Steuerungs-systemen", Dissertation, Uni. Karlsruhe, 1986.
8. 조 영조, 윤 태웅, 이 준수, 오 상록, 최 익, 김 광배, "연속 공정자동화를 위한 Function Block Diagram 형 제어 언어의 설계 및 구현", ' '91 KACC 논문집, pp. 226-231, 1991.
9. 안 재봉, 유 지훈, 신 영민, 송 인창, "Ladder Diagram 과 Instruction List 와의 상호 변환 알고리즘", '90 KACC 논문집, pp. 629-633, 1990.
10. 국 금환, "조립 자동화 시스템 개발(I)", 과학기술처, 1991.
11. M. Parent, C. Laurgeau, "Logic and Programming", Kogan Page,pp. 23-62, 1985.
12. S. H. Teng, J. T. Black, "Cellular Manuacturing Systems Modeling: The Petri Net Approach", Journal of Manufacturing Systems, Vol. 9, No. 1, pp. 45-54, 1990.
13. H. Kodate, K. Fujii, K. Yamanoi, "Representation of FMS with Petri Net Graph and Its Application to Simulation of System Operation", Robotics & Computer-Integrated Manufacturing, Vol.3, No. 3, pp. 275-283, 1987.
14. E. Kasturia, F. DiCesare, A. Desrochers, " Real Time Control of Multilevel Manufacturing Systems Using Colored Petri Nets", Proc. of IEEE Int. Conf. on Robotics and Automation, 1988.
15. M. Hasegawa, M. Takata, T. Temmyo, H. Matsuka, "Modelling of Exception Handling in Manufacturing Cell Control and Its Application to PLC Programming", IEEE, pp. 514-519, 1990.