창 립
40주년학술대회
논문 87-B-20-7

GRAFCET을 이용한 프로그램형 제어기의 제어기능 설계 및 모니터링에 관한 연구

°한 승수, 최 돈, 김 현기, 우 광방

연세대학교 전기공학과

A study on programming and monitoring system of the PLC using GRAFCET

Seung-Soo Han, Don Choi, Hyun-Ki Kim, Kwang-Bang Woo

Dept. of Electrical Eng., Yonsei Univ.

Abstract

A structured user-friendly procedures which enable users to program in GRAFCET form directly from sequence specifications have been developed. Using these procedures, we can program the programmable logic controller (PLC) in sequential control and realize easy programming, debugging, and real-time monitoring. GRAFCET has many advantages like parallelism expression, so we can expect higher productivity and easier maintenance than boolean language or relay-ladder diagram specification when adapts them to PLC.

1. 서론

80년대에 이르러 산업이 급격히 발달함에 따라 종래의 단순하던 제어 시스템은 점차 복잡해 졌고 따라서 체계적이고 계층적인 방법으로 설계되어지는 것이 요구되어지고 있다. 순차제어[1]의 초창기에는 릴레이를 사용한 제어반이 주종을 이루었으나 70년대 초 마이크로 프로세서의 출현으로 프로그램형 제어기(Programmable Logic Controller : PLC)가 개발되어 현대에 이르렀고 이러한 프로그램형 제어기의 등장은 논리함수를 프로그램으로 실현해서 필요에 따른 저장과 반복수행을 가능하게 하였다.

프로그램형 제어기를 프로그램 하는데는 보통 boolean 언어나 relay-ladder diagram을 이용하나[2] 이러한 방법은 새로 구성한 시스템으로 부터 직접 프로그램 하기가 어려우며 프로그램에 오류가 생겼을 때에는 이를 발견하여 디버깅 하는데에도 많은 시간과 어려움이 따른다. 이러한 단점을 보완하기 위해서 보다 간편하고 손쉽게 프로그램하고 관리하기 편리한 도구를 개발하기 위한 연구가 많이 이루어지고 있다 [3] [4].

본 논문에서는 프랑스에서 연구가 이루어진 GRAFCET (Le Graphe de Commande Etape - Transition) 을 사용하여 프로그램형 제어기를 제어하고 작업의 효율상태의 실시간 모니터링을 실현 하고 이러한 작업의 수행을 기존의 범용 언어인 PASCAL을 사용하여 PASCAL자체의 구조적인 장점을 이용할 수 있도록 하였다.

2. GRAFCET의 기본 이론

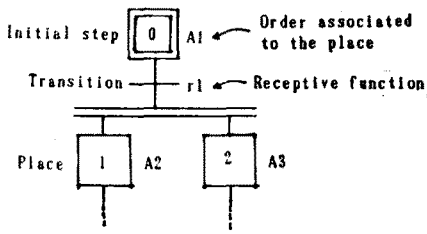
GRAFCET는 프랑스에서 발전했으며 이것은 DIN기준인 DIN 497 - 1916 과 DIN 407819/6 [5]에 잘 정의 되어 있다 이러한 제어 시스템 기술 기준은 어떤 종류의 동력을 사용하던간에 잘 적용될 수 있다. GRAFCET의 기본적인 정의와 기술방법을 설명한다.

2.1 GRAFCET의 정의

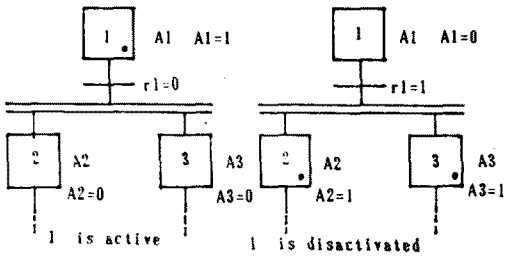
GRAFCET이란 제어 시스템을 그래프적으로 기술한 function diagram 으로서 place (정방형으로 표시), transition (bar로 표시) 과 이들을 연결시켜 주는 oriented arc로서 구성된다 (그림 1). 각 place는 수행해야 할 일련의 action이나 전기, 전자 시스템의 하나의 제어 모듈에 해당되고, transition은 다음 place 로의 천이 (evolution) 를 가능하게 해 주는 logic condition (receptive function)과 관계된다. GRAFCET는 다음과 같은 천이 규칙을 따른다 :

- Initialization은 시스템의 최초의 action을 명시한 active step 으로서 이중정방형으로 표시한다 (그림 1 (a) 참조).

- Place는 active 되어 있거나 혹은 inactive 되어 있다. 어느 순간에 모든 active place에 의해 순차제어기능 (sequential control function)의 상태가 완전히 정의된다. Action은 각 place와 결합될 수 있다.
- Transition은 직전의 place가 active 되고 이 transition과 관련된 logic condition이 참이 되면 해당 transition을 뛰어 넘는다.
- 뛰어넘은 결과로 해당 transition 직전의 place들이 inactive되고, 직후의 place들이 active 된다.
- 동시에 뛰어넘음이 가능한 다수의 transition들은 동시에 뛰어넘는다.
- 동일한 place가 active와 inactive가 동시에 이루어질 경우 이 place는 active된다.



(a) GRAFCET 구조



(b) Place 1 에서의 상태천이

그림 1 GRAFCET의 심볼 및 천이

2.2 예제

그림 2 에 보인 순차 제어 모델은 GRAFCET표현의 한 예이다. 용기는 콘베이어에 의해 이동되며 첫번째 스테이션에서 용기가 채워지고 다음 스테이션에서 봉해진다. 그러나 용기의 공급은 때때로 불규칙적이고 또한 빈곳이 나타나기도 한다. 그러므로 두 스테이션에는 용기의 존재유무를 감지하는 장치가 되어 있어야 한다.

이 시스템의 GRAFCET형태의 순차 제어 흐름 diagram이 그림 3 에 나와 있다. 콘베이어가 한 스텝 나아갈때 두 스테이션 모두 동시에 동작하기 시작한다. 각각의 스테이션에서는 용기의 존재유무를 조사하여 용기를 채우는 작업과 봉하는 작업을 동시에 수행하게 된다.

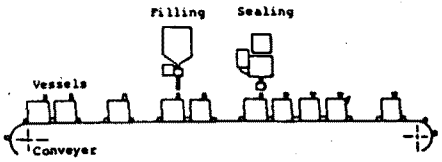


그림 2 순차 제어 모델

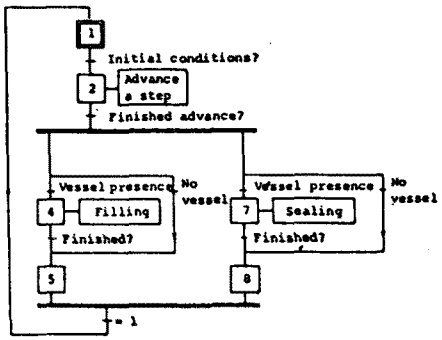


그림 3 모델의 GRAFCET 표기

3. GRAFCET 프로시듀어의 구성

3.1 Divergence와 Convergence에 의한 구성

GRAFCET의 각 unit는 divergence나 convergence의 개념을 이용하여 구현 할 수 있다. 이때 각 스테이지는 보통 2라인으로 나타내어 진다.

첫번째라인 : 주어진 스테이지와 관련된 내부변수 Xn의 활성화. 이를 위해 두개의 조건이 만족되어야 한다.

(1) 바로 직전의 모든 스테이지 X(n-1)가 transition을 위해 active되어 있어야 한다.

(2) 실제 transition과 관련된 receptive function R(n-1)이 만족되어야 한다.

두번째라인 : 내부변수 Xn의 기억과 관련된 출력 An의 활성화 (activation).

주어진 스테이지 Xn의 활성화를 위해서 바로 앞의 선행하는 스텝인 X(n-1)이 완료되어 있어야 하므로 첫번째 라인에 완료조건이 있다. 그렇게 해서 다음 스테이지인 X(n+1)이 active되지 않는한 보유함수 (retentive function)가 영향을 주는 두번째 라인을 위한 필요조건이 이루어 진다. 이상의 개념에 따라 다음과 같이 쓸수 있다.

$$X(n-1)\ R(n-1) = X(n)$$

$$X(n)\ X(n+1) = X(n)\ A(n) \qquad (1)$$

식(1) 에 기초한 receptive function의 GRAFCET의 수행은 그림 4 와 같다.

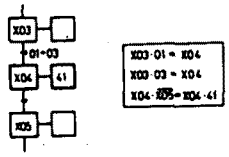


그림 4 Receptive function의 GRAFCET 프로그램

부시스템을 포함하는 복잡한 시스템일 경우에는 판단 (decision making) 프로세스가 포함되어 있는데 이 경우에는 divergence와 convergence를 이용한다. 예를들어 alternative path가 있는 경우의 프로그램에는 OR - divergence 와 OR - convergence 가 포함된다 (그림 5). 반면에 parallel path가 있는 경우에는 AND - divergence 와 AND - convergence 로 프로그래밍 된다 (그림 6).

점프 스테이지와 반복 스테이지의 프로그래밍은 그림 5 의 경우를 이용하면 효과적으로 프로그램 할 수 있다. 이와같이 AND, OR - divergence, - convergence 를 이용하면 앞에서 언급한 5개의 요소를 모두 프로그램 할 수 있게 된다.

3.2 PASCAL 프로시듀어의 PLC에의 적용

원하는 순차제어 시스템의 분석이나 합성에 GRAFCET를 적용하기 위해서는 sequence의 기본적인 형태와 그의 GRAFCET 형태를 알아야 한다. GRAFCET에서 sequence의 흐름의 구조는 5개의 요소로 되어 있으며 이러한 구조는 구조적 프로그래밍을 가능하게 해주며 divergence나 convergence의 개념을 이용하여 프로그래밍을 하는것 보다 손쉽게 GRAFCET형태의 프로그램을 작성할 수 있다. 5개의 요소를 살펴보면 다음과 같다 (그림 7).

(1) 단순 transition (Simple transition)

이 구조는 몇몇 transition조건을 포함하는 sequence unit로 이루어져 있다. 실행 스텝은 transition조건이 만족되면 위에서부터 아래로 진행한다. 여기서 Transition (transition condition) 명령은 transition조건을 나타내준다. Transition 명령의 operand는 복잡한 logical expression 뿐만 아니라 입력 포트에서 입력된 조건 (예를 들면 외부 센서로 부터의 입력등)을 포함한다.

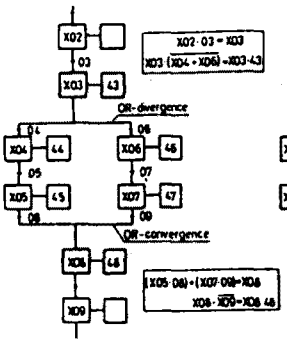


그림 5 OR-divergence, convergence의 GRAFCET 프로그램

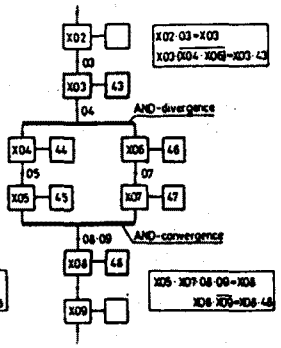


그림 6 AND-divergence, convergence의 GRAFCET 프로그램

(2) 조건 점프 (Conditional jump)

이 구조는 점프조건이 만족되면 sequence unit의 한 부분으로 실행 스텝이 점프한다. 조건 점프는 선택적 수행의 한 부분이므로 선택적 수행을 이용하여 프로그램할 수 있다.

(3) 반복 (Repeat)

이 구조에서는 반복조건이 만족되면 sequence unit의 특정한 부분을 반복하여 수행한다. 반복구조는 RepeatStart ~ RepeatEnd (Repeat condition) 으로 나타내며 반복할 프로시듀어는 두 명령어 사이에 프로그램 한다.

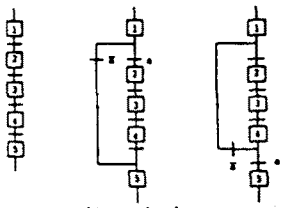
(4) 선택적 수행 (Selective execution)

이 구조는 2개의 sequence unit를 갖는다. 한번에 오직 하나의 sequence unit만이 선택 조건에 따라 수행한다. SelectiveStart 명령에 의해서 선택적 수행의 시작을 나타내고 Select (sequence unit no, condition) 에 의해 조건에 맞는 sequence unit를 선택해서 수행한다. 선택된 sequence unit이 모두 수행되면 SelectiveEnd 명령어 이하 부분이 수행된다.

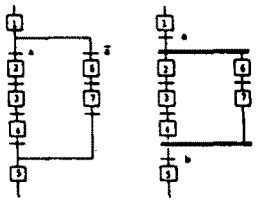
(5) 동시 수행 (Simultaneous execution)

이 구조는 동시에 수행되는 몇몇 sequence unit들로 구성된다. 모든 sequence unit이 완전히 수행되면 프로그램은 다음 스텝을 수행한다. ParallelStart (number of simultaneous execution units) 명령어는 동시 수행해야 할 sequence unit의 수와 동시 수행 작업의 시작을 알린다 동시 수행의 각 부분은 Parallel (sequence unit no)로 선택해서 프로그램 하고 ParallelEnd 명령어는 모든 sequence unit이 완전히 수행될때까지 수행이 대기상태에 있어야 함을 나타낸다.

이러한 프로시듀어들은 표준 PASCAL 하에서 수행 되므



(a) 단순 transition (b) 조건 점프 (c) 반복



(d) 선택적 수행 (e) 동시 수행

그림 7 GRAFCET의 5가지 구조적 요소

로 시스템에서 요구되는 데이타 처리 명령, 주변장치 제어 명령 들이 있으며 시스템을 실시간 모니터링 하는데 요구되는 모니터와 PLC의 초기화 명령인 GRAFCET, InitGRAFCET, EndGRAFCET, LeaveGRAFCET 등이 있다. 이러한 명령어를 사용함으로써 사용자는 단순한 on/off 순차제어 프로그램뿐만 아니라 순차제어 및 정보처리를 포함한 복잡한 제어 프로그램도 디자인 할 수 있다.

4. PLC에의 적용 및 시뮬레이션

4.1 PLC에의 적용

PLC의 프로그래밍에는 보통 relay - ladder diagram을 사용하는데 이는 그 구조가 단지 AND와 OR logic expression으로만 되어 있어 그 구조가 단순하기 때문이다

GRAFCET 프로시듀어를 PLC에 적용하기 위해서는 transition 조건이나 place의 출력을 ladder diagram으로 표현 했을때의 코드로 변환 시켜야 한다. Ladder diagram의 각 명령어의 코드 예가 표 1 에 나와 있다.

GRAFCET의 각 명령을 만나면 GRAFCET 프로시듀어에서 명령어에 해당되는 조건을 분석하여 찾는 프로시듀어를 실행하여 표 1 에서 그 코드를 찾아 코드변환을 실행한다. 복잡한 logic expression일때는 각 조건에 맞는 코드를 찾아 결합하므로 여러개의 코드가 생성된다. 이 코드가 PLC와 연결된 포트를 통해 PLC로 입력되며 PLC는 입력된 코드에 따라 실행한다.

표 1 Ladder diagram 명령어 코드 예

명령어	코 드	명령어	코 드
R	X0001XXX	W.TMR	00110000
A	X0010XXX	W.CTR	10110001
O	X0011XXX	W.SR	10110010
W	X0100XXX	A.MRG	00110011
R.N	X1001XXX	O.MRG	10110100
A.N	X1010XXX	W.NRG	00110101
O.N	X1011XXX	CLR	00110110
W.SC	X0101XXX	DS	10110111

4.2 시뮬레이션 및 결과

그림8 과 같이 3곳의 작업대를 갖는 회전 플레이트에서 첫번째 작업대에서는 가공할 물건을 채우고 두번째에서는 drilling을 하고 세번째에서는 가공된 물건을 검사한 후 비운다. 테스터에의해 검사가 행해지며 정확하게 drilling 됐으면 테스터가 low position까지 내려간다. 만일 low position까지 내려가지 못하면 시스템은 정지되며 이때 테스터는 high position에 머문다. 이 사이에 작업자는 불량품을 꺼내고 다시 재가동 시킨다. 이러한 시스템의 GRAFCET기술은 그림9 에 나타나 있고 각 접점의 의미는 표2 와 같다.

표 2 Action과 logic condition의 의미

Actions

01	Push charger	02	Pull charger
03	Push holder	04	Descend drill
05	Raise drill	06	Pull holder
07	Descend tester	08	Raise tester
09	Push evacuator	10	Pull evacuator
11	Rotate plate		

Logic conditions

X01	Start	X02	Charger in back
X03	Drill in high	X04	Tester in high
X05	Evacuator pull	X06	Piece released
X07	Piece charged	X08	Piece holded
X09	Drill in low	X10	Tester in low
X11	Condemantion	X12	Piece evacuated
X13	Restart	X14	End of rotation

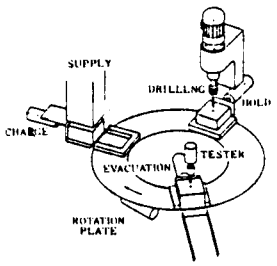


그림 8 Drilling machine 작업 모델

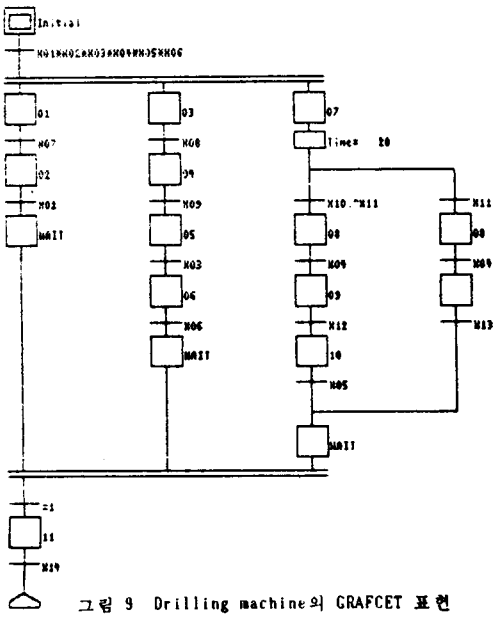


그림 9 Drilling machine의 GRAFCET 표현

'Start' 명령과 각 부분이 정위치에 있다는 초기조건 정보에 따라 세개의 sequential unit이 동시에 active 된다. 이 세개의 sequence는 각각 독립적으로 천이되며 각 sequence의 마지막 place에서 다른 sequence가 모두 완료되기를 기다리는 'WAIT STEP'이 있다. 이 세개의 WAIT STEP에서는 어떠한 action도 이루어 지지 않고 다만 병렬 sequence의 재동기화 (re - synchronization) 역할만을 하며 이것이 모두 active되고, 직후의 logic condition이 참이 되어야만 한다. 이를 위해 항상 참을 의미하는 조건 '=1' 을 사용한다. 세번째 sequence에서는 테스터가 low position에 있어야 하고 이 상태에서 2초가 경과해야 함을 고려했으며 시험결과 불량일 경우와 아닐경우에 따라 선택적 수행을 하게 했다.

5. 결론

본 논문에서는 PLC를 프로그램 하는데 GRAFCET 기술형식으로 프로그램을 실현하는 방법에 대해 고찰했다. 복잡한 산업 sequence들은 GRAFCET sequence의 기본적인 형태를 결합함으로써 처리될 수 있다.

대규모의 입출력 변수 및 내부 상태 변수가 필요한 산업 자동화에서 병렬 수행 기능을 기술할 수 있고 또한 기술결과로부터 손쉽게 직접 프로그램할 수 있는 방법을 제시 했으며 이는 ladder diagram과 같은 고전적인 방법으로 행하는것보다 높은 생산성을 유지할 수 있다 [3]. 또한 이를 이용하면 디자인한 제어 시스템의 시각화와 높은 유연성, 손쉬운 수리와 유지를 이룰수 있다.

Reference

1. O.P. Kuznetsov, "Program realization of logical functions and automata ; Part I : Analysis and synthesis of binary process," Automata. Remote Contr., Vol. 38, pp. 1077 - 1087, 1977.
2. O.J. Struger, et. al, "Language for programmable controllers," Proc. IECON '84, pp. 362 - 366.
3. Kichie Matsuzaki, et. al, "Prtri - net structured sequence - control language with GRAFCET - like graphical expression for programmable controllers," Proc. IECON '85, pp. 433 - 438.
4. Ljubomir Jacic, "Flexible industrial control system design - A GRAFCET approach," Japan - USA Symposium on Flexible Automation, pp. 847 - 850, 1986.
5. Anonymous, "Le GRAFCET - Outil des presentation du cashier des charges d'un automatisme logique," Rapport final de la Commission AFCET, AFCET - Paris, 1977.