현장 기술자를 위한

소형 PC에 의한 시퀀스 제어

현·장 기·술 4

역 / 박 한 종(협회 교육홍보위원)

제3장 프로그램 작성의 기초로서의 논리회로

5. 정논리와 부논리

여기서는 그림 4.6과 같은 리셋 신호 B의 입력 푸시버튼 스위치 취급방법을 알아 보기로 하자. PC에 의해 제어를 하는 경우도 입력의 세트, 리셋 신호를 부여하는 것은 사람과의 관계를 고려하여 유접점 푸시버튼 스위치를 사용한다.

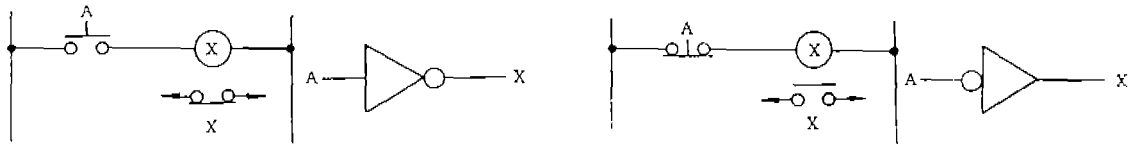
정지용 푸시버튼 스위치는 주지하는 바와 같이 일반적으로 안전상(페일·세이프*가 되도록) b접점으로 부여하게 되어 있다.

a접점으로부터도 내부에서 조작을 가하면 입력된다. 입력을 a접점에서 가하는가 b접점에서 가하는가는 앞에서 잠깐 언급한 논리회로에 있어서의 정논리(正論理), 부논리(負論理)의 문제이다. 그림 5.1에 그림 2.1의 부정의 유접점, 무접점 회로를 재차 들고 이것에 대해서 생각해 보자(그림 5.1은 그림 2.1의 입력측 접점 A를 자동복귀형의 푸시버튼 스위치로서 생각한다).

무접점 심볼도에서는 입력, 출력신호의 유무를 전압 레벨의 고저로 정하고 있는 것은 앞에서 본 것과 같으며 높은 전압 레벨을 「H」, 낮은 전압 레벨을 「L」로 나타내고 있다.

한편, 불식의 연산치로서 취할 수 있는 2개의 값은 「1」과 「0」이다. 논리회로에서는 「1」을 능동(active)이라고 하며 그림 5.1의 유접점 회로의 푸시버튼 스위치로 생각하면 푸시버튼을 조작하는 것(누름)으로 대응시키고 있다. 또 「0」을 비능동(non-active)이라고 하며 푸시버튼을 조작하지 않고(누르지 않음) 대응시키고 있다.

그런데 무접점 논리회로 표기법에서는 「H」와



〈그림 5.1〉 NOT의 정논리·부논리

* fail safe, 기계의 고장, 사고, 오조작 등이 일어나도 위험한 상태에 빠지지 않도록 하는 안전 시스템.

「L」 어느 쪽을 「1」로 하고 어느 쪽을 「0」으로 하는가는 자유로 되어 있다. 이것을 어떻게 정하는가로 그림 5.1의 NOT의 무접점 기호의 2개 그림이 나오게 된다.

그림 5.1(a)와 같이 푸시버튼 스위치가 a접점으로 부여되고 있으면 「능동」=「1」은 푸시버튼 스위치가 눌리고 접점이 「닫히는」 것에 대응하고 「비능동」=「0」은 푸시버튼이 눌리지 않고 접점이 「열려 있는」 것에 대응한다.

그러나 푸시버튼 스위치가 (b)와 같이 b접점으로 부여되고 있으면 「능동」=「1」은 푸시버튼 스위치가 눌리고 접점이 「열리는」 것에, 「비능동」=「0」은 푸시버튼 스위치가 눌리지 않고 접점이 「닫혀 있는」 것에 각각 대응하게 된다. 이 설명은 그림의 유접점과 무접점 회로도를 보면 충분히 이해될 것으로 생각된다.

(a)에서는 「H」가 「1」이 되어 있으므로 입력은 정논리이다. 그러나 한편 (b)는 「H」가 「1」로 되어 있으므로 입력은 부논리이다.

정리하면 정논리란 「H」를 「1」로, 「1」을 「0」으로 정한 논리이다. 한편 부논리란 「H」를 「0」으로, 「L」을 「1」로 정한 논리다.

여기서 다시 한번 그림 5.1에 있어서의 NOT의 무접점 심볼을 살펴 본다. 이 안의 버퍼 기호의 앞 또는 뒤에 달려 있는 작은 원은 상태표시기호라고 해서 동일한 논리회로도내의 정논리와 부논리를 구별하고 있는 것이다.

그림 5.1(a)에서는 입력 A는 푸시버튼 스위치를 조작하는 것(누르는 것, 능동 「1」)은 유접점에서는 접점이 「닫히는」 것에, 무접점에서는 전압이 높은 레벨 「H」가 된다. 입력 A를 능동 「1」로 하면 출력 X는 이 경우 b접점으로 부여되고 있으므로 유접점은 「개」, 무접점은 「L」 상태이다. 이와 같이 입력은 정논리(「1」→「H」)로 논리가 구성되어 있으므로 무접점 심볼도 입력측에는 상태표시기호(○)표를 달지 않고 출력측에 달게 된다.

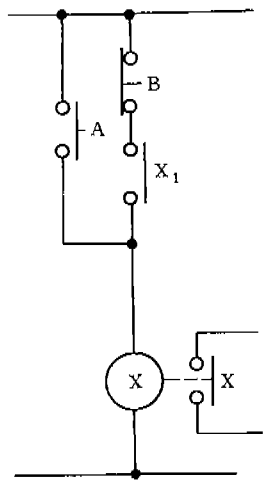
한편 (b)는 입력 A를 조작하는 것(능동 「1」)은 유접점 회로에서는 접점이 「열리는」 것으로, 그리고 무접점에서는 「L」이 되었다. 그러나 출력 X는 이 경우 a접점으로 부여되고 있으므로 유접점은 「개」, 무접점은 「L」 상태이다. 이와 같이 입력이 부논리이므로 무접점 심볼도의 입력측에 상태표시기호를 단다.

이상 정논리와 부논리에 대해서 좀 상세히 기술하였다. 이것은 PC를 사용해서 시퀀스 제어를 하는 경우에 입력기기부터의 접점 부여방법이 PC의 프로그램에 직접 관계하기 때문이다. 예를 들면 입력을 프로그램 상에 데이터로서 넣는데 입력 데이터의 취급(정논리, 부논리의 처리)을 잘못하면 전혀 다른 동작을 하게 된다.

유접점의 리셋 우선자기유지회로(優先自己維持回路)에 대해서 이것을 무접점화하는 방법을 설명하였다. 그러면 여기서 연습으로 세트 우선자기유지회로를 무접점화하는 문제를 들어 본다. 우선 자기 힘으로 시험해 보기 바란다. 그리고 나서 마지막에 해답을 보기 바란다.

(연습문제 5.1)

다음의 유접점 회로를 무접점화하라.



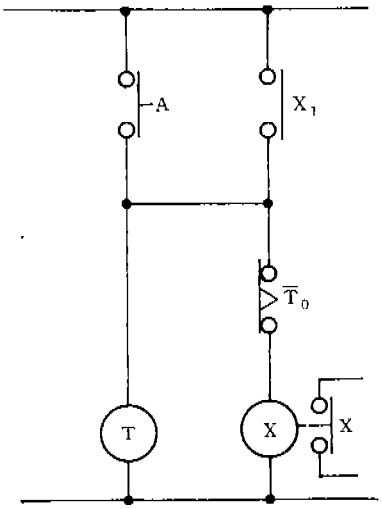
〈그림 5.2〉

6. 타이머 요소를 포함한 유접점 회로의 무접점 회로화

여기서는 타이머 요소를 포함한 유접점 회로를 무접점 회로로 변환하는 요령에 대해서 설명한다.

(1) 회로 동작

회로 동작에 대해서 독자들은 이미 알고 있겠지만 무접점 회로에의 변환에 대해 확인하는 의미에서 간단히 언급해 두기로 한다. 그림 6.1이 유접점



〈그림 6.1〉
간격동작회로
(유접점)

의 간격 동작회로이다. 자기유지회로와 다른 것은 타이머 요소를 포함하고 있는 것이다.

기동 푸시버튼 스위치 A를 순시(瞬時) 누르면 출력접점 X가 즉시 「닫힘」이 되고 동시에 유지접점 X_1이 「닫힘」이 되며 자기유지가 걸린다. 타이머 설정시간이 경과하면 타이머의 T_1 접점이 「열림」이 되고 전자 코일 Ⓧ가 비여자가 되며 출력접점 X가 「열림」이 된다.

(2) 불식

무접점 변환의 포인트는 타이머 취급이다. 불(Bool)식으로는 타이머 고유의 표현을 할 수 없으므로 유접점 회로를 불식으로 변환할 때 타이머를 분별해 두어야 한다. 여기서는 유접점의 타이머 코일과 타이머 순시동작 접점을 기호 T, 한시동작 접점을 기호 T_1으로 하여 구별하고 있다.

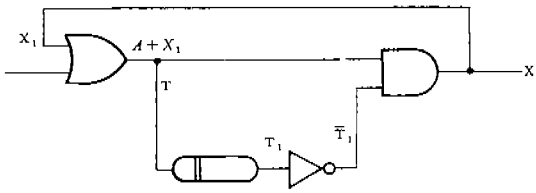
불식 표현은 다음과 같다.

$$X=(A+X_1)\cdot\overline{T_1} \qquad (6.1)$$

$$T=\overline{A}+X_1 \qquad (6.2)$$

여기서 $\overline{T_1}$은 Ⓣ 코일에 의해 작동하는 시한동작 b접점, 그리고 T는 A와 X_1의 OR가 되므로 식 6.2가 성립된다.

Ⓣ 코일의 출력은 한시동작 접점 T_1만이고 순시동작 접점 T는 그림 6.1에서는 사용하고 있지 않



〈그림 6.2〉 간격동작회로(무접점)

다. 그런데 왜 T로 하는가 하면 Ⓣ 코일의 여자상태와 출력접점 T의 동작이 동일하기 때문에 Ⓣ 코일의 여자상태를 T로 하여 표시하고 있다. 이것에 대해서는 다음의 무접점 회로도를 보면 더 확실해질 것으로 생각된다.

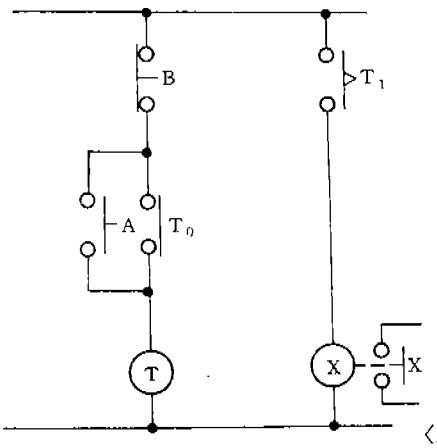
(3) 무접점 회로 표현

식 6.1, 6.2를 무접점 심볼도로 표현한 것이 그림 6.2이다. 유접점 회로의 Ⓣ 코일과 순시동작 접점 T, 한시동작 접점 T_1의 관계는 식 6.1, 6.2의 불식으로는 표현할 수 없었다. 그 때문에 한시동작은 T와 구별을 두어 T_1으로 한 것이다.*

그것이 그림 6.2에서는 A와 X_1의 OR회로($A+X_1$)의 출력이 T가 되고 그 T가 타이머(타임 딜레이)를 통해서 오프딜레이 동작의 T_1이 되고 있는 것을 한 눈에 알 수 있다.

(연습문제 6.1)

다음 유접점 회로를 무접점 회로로 표시하라.



〈그림 6.3〉

* 물론 T와 구별하여 T_1으로 기록하는 것만으로는 한시동작으로 되어 있는가의 여부를 판단할 수 없다. 이것이 불식 표현의 한계이다.

제4장 논리연산과 프로그래밍

제2장에서 자기유지회로를 PC를 사용하여 프로그램화하고 그 프로그램을 실행시켜 보았다. 그때 프로그램에 대한 대체적인 것을 알았을 것으로 생각된다. 여기서 정리를 하여 요약해 보면 「프로그램」이란 PC에 「일을 시키기 위한 명령을 어느 법칙에 따라 순서대로 바르게 기술한 것」이라고 할 수 있다.

자! 제3장에서는 유접점 릴레이 시퀀스 제어회로를 논리적으로 보고 그것을 무접점 논리회로로 변환하였다. 이 과정을 통해서 논리회로의 기본적인 사고방식을 기술하였다.

이 제4장에서는 논리회로를 구성하는 기본 논리연산 기능을 PC의 프로그램으로 표현하려면 어떻게 하면 되는가에 대해서 생각해 본다.

PC는 명령내에 논리연산 명령을 가지고 있다. 이 논리연산 명령과 기타 몇가지 명령을 사용해서 하드웨어의 논리회로는 논리연산을 실행시키는 명령의 집합, 즉 프로그램으로 바꾸어 놓을 수가 있다. 그리고 이 프로그램을 실행시키면 유접점 시퀀스 회로나 무접점 시퀀스 회로와 등가의 논리연산 동작을 PC에 시킬 수가 있는 것이다. 이것이 PC로 시퀀스 제어를 실현할 수 있는 원리이다.

1. 명령과 그 기술방법

(1) 명 령

시퀀스 제어회로는 프로그램에 의해 표현할 수 있는 것을 알았지만 프로그램 만들기는 PC의 명령을 알지 않으면 안된다. 그리고 명령을 어떠한 순서로 기술하면 되는가도 알아 둘 필요가 있다.

명령은 전술한 바와 같이 니모닉 코드로 기술하지만 이 수는 마이크로 컴퓨터에 있어서는 이백수십종류나 된다. PC는 시퀀스 제어 전용기로 명령의 수는 이것보다 감소된다. 그러나 그래도 많은 것은 백 가까이 있다. 이들 명령을 잘 이해하여 사용하는 것은 용이한 일이 아니다.

PC를 용이하게 사용할 수 있게 하기 위해 본고에서는 사용하는 명령의 수를 가급적 적게 하였다. PC의 특징은 전선으로의 배선작업을 대신해서 제어내용을 프로그램으로 소프트 와이어드로 구체화하는 것에 있었다. 이것을 「프로그램 코딩」이라고 했는데 이 코딩이 배선작업에 비해서 어려워지는 것은 허용되지 않는다. 배선작업을 할 때와 같이 사람이 기계적으로 코딩할 수 있게 되어 있어야 한다.

이상의 것을 구체화하기 위해 본고에서는 사용하는 명령을 다음과 같이 하였다.

첫째, 논리 연산명령을 다음 4종류로 한정

AND : and ; 논리적

OR or ; 논리화

INV* : inverter ; 부정

TIM : timer ; 한시

전술한 바와 같이 대부분의 시퀀스 제어회로는 이 4개의 요소를 조합함으로써 만들 수가 있으므로 연산명령은 이것만으로 하였다.

둘째, 데이터의 전송명령을 다음 3종류로 한정

LD : load ; 판독

STO : store ; 일시 기억

OUT : out ; 출력

입력 유닛측에서 들어오는 데이터를 판독하는 것이 LD 명령. 그것에 논리연산 처리를 하여 출력측에 출력하는 것이 OUT 명령이다. STO 명령은 메모리붙이 전자계산기의 메모리와 동일한 것으로 중간에서의 기억이라고 생각해도 되는 것이다.

이제부터 당분간 사용하는 명령은 합계해서 이상의 7개 종류이다. 이 정도의 명령이 있으면 명령에 대한 특별한 학습은 없어진다. 다음은 회로를 보면서 코딩해 나가는 요령을 익히면 이들 명령은 자연히 알아진다.

(2) 프로그램 기술방법

다음에 명령을 어떠한 순서로 기술하는가에 대

* 부정 NOT로서, 이 니모닉 코드를 사용한다.

〈표 1.1〉 16진수와 그 표시

10진법	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16진법	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
표시	0	1	2	3	4	5	6	7	8	9	A	b	C	d	E	F

해서 기술한다. 간단히 말하면 「메모리불이 전자계산기를 사용해서 계산하는 요령으로 기술하면 된다」는 것이다. 이것은 무접점 회로도가 시퀀스 제어회로를 나타내 주는 동시에 제어내용을 프로그램화하기 위한 원시정보*도 확실하게 제공해 주기 때문이다. 이것이야말로 무접점 회로도를 채용한 이유이다.

구체적인 기술방법은 프로그램 코딩시에 특히 다음 제5장의 기본 시퀀스 제어회로의 프로그램 코딩에서 기술한다.

(3) 16진수

프로그램은 프로그램 콘솔에서 니모닉 코드 및 숫자 키로 키인한다. 그러나 표시는 전부 숫자**이다. 이 숫자는 16진수라고 하는 것을 사용한다. 이 16진수는 PC를 사용해 나가는데 있어 반드시 필요한 것이므로 빨리 익숙해 지도록 해야 한다.

16진수란 한마디로 말하면 16까지 가면 1자리 올라가게 만들어진 수이다. 보통 우리들이 사용하고 있는 것은 10진수인데 이것은 1자리째는 9까지이고 다음의 수는 10이 되며 2자리째(십의 자릿수)에 자리 올림의 1이 쓰인다. 이것과 마찬가지로 16진수에서는 10진수에서 말하는 15까지의 수를 한자리째에 쓰고 16이 되어 비로서 두자리째로 자리올림이 있는 것이다.

16진 표시를 할 때 0~9까지의 수는 문제가 없지만 10~15까지의 수는 한자리로는 나타낼 수가 없다. 그래서 10진수의 10 이상을 알파벳의 A~F로 대용시키고 있다. 그러므로 이 A~F의 영문자는 16진수에서는 숫자로 취급되고 있는 것을 잊지 않도록 하는 것이 중요하다.

〈표 1.2〉 자기유지회로 프로그램

스텝 No.	명령 OP	명령 ADD	비 고
00	LD	00	
01	OR	10	
02	AND	01	
03	STO	10	
04	OUT	E0	

(4) 명령의 구성

명령이 어떠한 구성으로 되어 있는가를 보기 위해 앞에서 든 자기유지회로 프로그램을 재차 표 1.2에 들었다.

이것에서 알 수 있듯이

명령=OP+ADD

로 구성되어 있다.

OP는 Operation의 약자로서 조작을 말한다. PC에 「무엇을 시키는가」를 정하는 부분으로서, AND나 LD와 같은 생략형의 니모닉 코드로 표기한다. 단, PC의 표시는 16진수이다(기종에 따라서는 그대로 AND, LD로 표시하는 것도 있다).

ADD는 Address의 약자로서 번지를 뜻한다. 입출력 번지***, 일시기억 번지****를 설정하는 부분이다. 타이머의 동작시간 설정시에만 그 데이터를 ADD부에 기록하게 된다.

☞ 다음호에 계속 ☜

* 컴퓨터의 플로자트와 같은 것으로, 프로그램의 원안.
** 기종에 따라서는 니모닉 코드로 영문자로 직접 표시하는 것도 있다.
*** 입출력 유닛의 번호를 할당하여 부르는 번지
**** 메모리 격납장소에 번호를 할당하여 부르는 번지