현장 기술자를 위한

소형 PC에 의한 시퀀스 제어

현·장
기·술
7

제6장 시퀀스 제어내용의 변경

6.1 제어내용의 변경은 일반적인 것

시퀀스 제어내용을 소프트웨어(프로그램)로 만들거나 하드웨어(릴레이, IC)로 만들거나 그 근본이 되는 제어내용의 설계 시방이 필요하다. 이것은 달리 말하면 시퀀스 동작의 표현이라고 해도 되는 것이다. 이 설계 시방이나 동작의 표현에 입각해서 구체적인 프로그램이나 회로가 만들어진다.

그런데 시퀀스 제어의 특징으로 이 설계 시방을 완전히 표현하는 것은 상당히 어려운 일이다. 그 때문에 일단 설계 시방이 결정되어도 그 후 설계 시방의 변경은 항상 붙어다니게 된다.

일단 설계 시방이 정해지고 다음에 이것에 입각한 제어내용을 만들게 된다. 그러나 실제로 기계를 움직여 시운전을 하여 보면 책상 위에서 제어내용을 계획하고 설계할 때는 생각지도 못하던 여러 가지 문제가 생기게 된다. 이 때문에 제어내용의 변경이 필요해진다. 그리고 또 현장에서 제어 시스템이 운전상태에 들어가고 나서도 변경시키고 싶은 것이 생기게 된다. 그것은 제어 시스템의 사용성이나 안전상에서 오는 것이다.

이상과 같이 제어 시스템은 그것을 만들고 운용해 나가는 모든 단계에서 내용의 변경이 수반되는 것이라고 할 수 있다.

6.2 PC는 제어내용 변경에 유리

제어내용의 변경에 있어서 종래의 전자 릴레이나 무접점 릴레이를 사용하여 만든 제어회로는 배선 변경이나 기구의 추가(보조 릴레이의 추가)와 같은 하드웨어의 변경으로 대처하지 않으면 안되게 된다.

이 하드웨어 변경의 번거러움에서 어떻게 탈피하지 않을 수 없을까 생각하게 되었다.

그리고 종래의 제어 시스템에 대해 근본적으로 재검토를 하게 된 것이다. 이 속에서 PC가 탄생되었다.

소프트웨어(프로그램)의 변경으로 제어내용을 바꿀 수 있는 새로운 제어장치가 출현한 것이다.

PC를 사용하면 설계부터 현장 운전의 모든 단계에서 일어나는 제어내용의 변경에 대해서 제어장치를 개조하지 않고 프로그램 변경만으로 신속하고 용이하게 대처할 수 있게 된다. 이것은 대단히 유리한 것이다. 또한 프로그램의 변경은 현장의 PC 조작원, 보전원이 충분히 할 수 있는 것이다.

물론 프로그램의 변경만이라고 하지만 변경내용이 입출력 기기에 관계될 때는 하드웨어의 추가가 필요하다.

6.3 무효 명령(NOP)과 점프 명령(JMP)

프로그램의 변경은 우선 반드시라고 할 수 있을 정도로 있는 것을 알았다. 지금 취급하고 있는 PC는 변경에 대해서 대체로 다음 두가지 방법으로 대응을 하게 된다.

우선 첫번째는 무효 명령의 채용이다. 무효 명령은 NOP(No Operation)라는 리모닉 코드로 표시되듯이 이 명령이 나가면 PC는 아무런 동작을 하지 않고 다음 스텝으로 진행한다. 프로그램을 작성하는 단계에 의미를 가지는 프로그램 블록마다의 중간틈 부분에 이 NOP 명령을 넣어 둔다. 이렇게 하여 두면 후에 프로그램에 변경이 생겼을 때 이 무효 명령을 넣은 범위에서 프로그램을 추가할 수 있다.

그러나 무효 명령은 최초에 프로그램을 만들 때 미리 예상하여 어느 정도 넣어 두는가를 정한 것으로서, 대폭적인 추가가 생긴 경우는 이것으로는 대응할 수 없게 된다.

두번째는 점프 명령을 사용하여 추가된 프로그램의 스텝에 원래의 주 프로그램의 흐름을 보내는 방법이다. 점프 명령은 JMP(Jump)라는 뉴모닉 코드로 표시되며, 이 명령이 나가면 JMP의 다음 ADD(어드레스) 항에 기록된 스텝으로 가서 그 스텝 명령을 실행한다.

추가시키려는 프로그램은 주 프로그램 뒤에 추가 기록해 나간다. 추가코자 하는 개소에서 점프 명령으로 이 추가부분으로 가서 이 부분의 실행이 끝나면 재차 점프 명령으로 주 프로그램의 다음 스텝에 되돌린다.

이 방법은 상당히 편리한 것이다. 그러나 프로그램의 흐름이 이리 저리로 가게 되기 때문에 이해하기 힘든 프로그램이 되는 것이 결점이다.

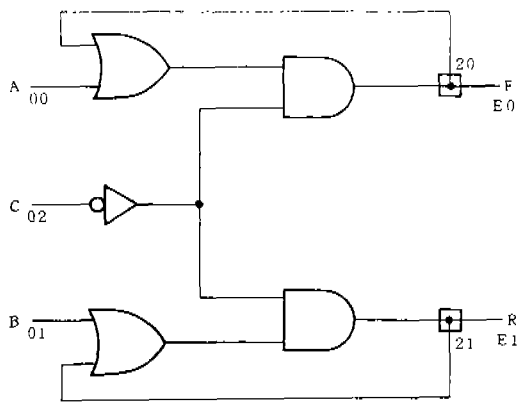
6.4 프로그램 변경방법

무효 명령과 점프 명령의 각각을 사용하여 프로그램 변경을 어떻게 하는가를 구체적으로 다음 두가지 예를 가지고 알아보기로 하자. 예 1은 무효 명령 NOP를 미리 프로그램 블록 중간틈 부분에 넣어 둔 것, 예 2는 점프 명령을 사용한 것이다. 여기서는 설명의 편의상 두 가지 방법을 따로 따로 기술하고 있지만 실제로는 두 가지 방법을 병용하면 그 효과가 커지는 것이다.

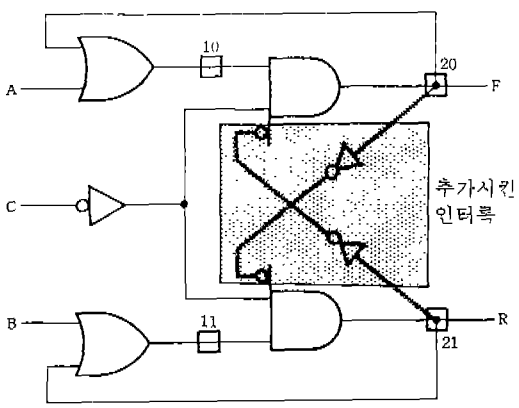
[예 1] NOP를 미리 넣어 두는 방법

그림 6.1과 같은 회로가 있다. 이 회로는 제5장에서 이미 기술한 인터록 회로와 비슷하다. 그러나 A, B가 동시에 입력됐을 때 출력 F, R가 양쪽 모두 ON이 된다. 그러므로 이 회로를 모터의 정역운전 제어회로에 사용하면 주회로측에서의 단락사고를 일으키게 된다.

이와 같은 불편이 생기지 않게 하기 위해 A를 입력하여 출력 F가 ON하고 있을 때는 B를 입력하여 R를 ON시키려고 해도 안되는 것 같은 안전장치가 필요하다. 이것은 R가 ON하고 있을 때 동일하다. 그래서 이에 대한 안전대책으로서 그림 6.2에 나타내는 인터록을 추가하도록 한다.



<그림 6.1> 원회로



<그림 6.2> 변경 후의 회로

<표 6.1> 프로그램 리스트

스 텝 No.	명 령 OP	명 령 ADD	비 고
00	LD	00	
01	OR	20	
02	AND	02	
03	STO	E0	
04	OUT	00	
05	NOP	00	NOP로 매워둔다
06	NOP	00	
07	NOP	00	
08	NOP	00	
09	NOP	00	
0A	LD	01	
0B	OR	21	
0C	AND	02	
0D	STO	21	
0E	OUT	E1	
0F	NOP	00	NOP로 매워둔다
10	NOP	00	
11	NOP	00	
12	NOP	00	
13	NOP	00	
14	|	|	
15	|	|	
16	|	|	
17			
18			
19			

(a)

스 텝 No.	명 령 OP	명 령 ADD	비 고
00	LD	00	
01	OR	20	
02	STO	10	변경분
03	LD	21	
04	INV	00	
05	AND	10	
06	AND	02	
07	STO	20	
08	OUT	E0	
09	NOP	00	
0A	LD	01	
0B	OR	21	
0C	STO	11	변경분
0D	LD	20	
0E	INV	00	
0F	AND	11	
10	AND	02	
11	STO	21	
12	OUT	E1	
13	NOP	00	
14	|	|	
15	|	|	
16	|	|	
17			
18			
19			

(b)

그림 6.1의 원래의 회로를 프로그램으로 표현한 것이 표 6.1 프로그램 리스트(a)이다. 00 스텝에서 04 스텝까지가 하나의 의미를 가지는 블록이므로 이 블록 뒤의 5 스텝을 NOP 00로 매워 둔다. 다음의 0A에서 0E까지가 다음 의미가 있는 블록이며, 이 뒤도 동일하게 하여 둔다.

표 6.1 프로그램 리스트(b)가 인터록을 추가시킨 변경 후의 것이다. 이 부분만을 보면 원래의 프로그램 부분 보다도 추가시킨 부분 쪽이 많아져 이렇다면 전부를 고쳐 쓰는 편이 좋을 것 같다. 그러나 긴 프로그램의 일부분인 경우는 이 정도의 변경으로 끝나면 전부의 내용을 이동시키기 보다

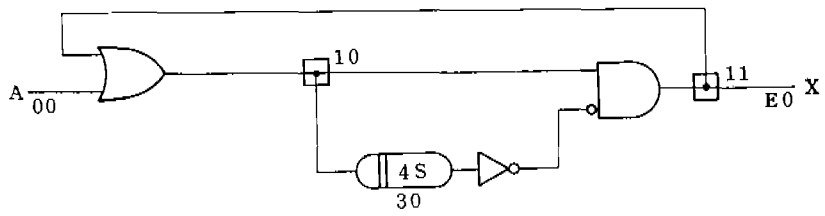


그림 6.3> 원회로

<표 6.2> 프로그램 리스트

스 텝 No.	명 령 OP	명 령 ADD	비 고
00	LD	00	
01	OR	11	
02	STO	10	
03	LD	10	
04	TIM	80	4초
05	LD	30	
06	INV	00	
07	AND	10	
08	STO	11	
09	OUT	E0	
0A			
0B			
0C			
0D			
0E			
0F			

(a)

스 텝 No.	명 령 OP	명 령 ADD	비 고
00	LD	00	
01	OR	11	
02	JMP	0B	
03	LD	10	
04	TIM	80	
05	LD	30	
06	INV	00	
07	AND	10	
08	STO	11	
09	OUT	E0	
0A	JMP	00	
0B	AND	01	
0C	STP	10	
0D	JMP	03	
0E			
0F			

(b)

훨씬 영향이 적은 것이 된다. 이 프로그램도 긴 프로그램의 일부로 생각하기 바란다. 이 변경 프로그램을 만들기에 앞서서 연산결과 대기를 위해서 10번지와 11번지에 새로 일시기억을 설정할 필요가 있다.

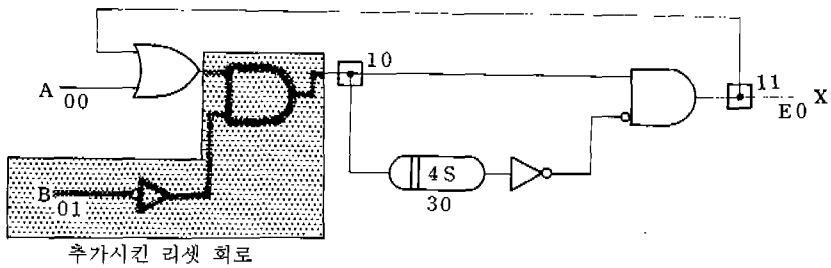
[예 2] JMP에 의한 변경

그림 6.3과 같은 회로가 있다. 이 회로는 이미 제5장에서 기술한 간격동작회로와 비슷하다. 그러나 비상정지(리셋)가 없다. 그래서 그림 6.4와 같이 리셋을 추가하기로 한다.

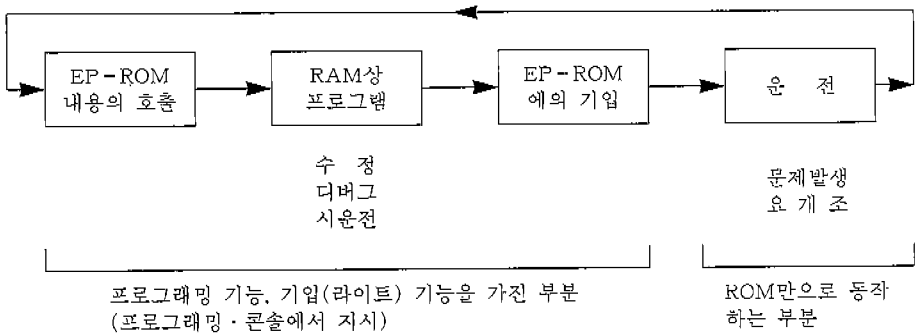
그림 6.3의 회로를 프로그램으로 표현한 것이 표 6.2 프로그램 리스트(a)이다. 이 프로그램을 그림 6.4와 같은 회로의 내용을 가지는 프로그램으로 변경시킨다.

우선 표 6.2 프로그램 리스트(b)와 같이 0A 스텝에 프로그램 END를 나타내는 JMP 00를 쓴다. 지금까지의 프로그램에는 이 명령은 쓰지 않았지만 JMP 명령을 사용하는 경우는 반드시 이 명령을 프로그램 마지막에 써두지 않으면 프로그램이 목적대로의 동작을 하지 않게 된다.

다음에 STO 10 앞에 AND 01을 삽입하고 싶은데 그렇게 하려면 STO 10을 JMP 0B로 바꾼다. 이것으로 프로그램이 가는 곳이 0B 스텝이 된다. 이렇게 하고 삽입하고 싶은 명령을 0B 스텝부터 써나가면 되는 것이다.



<그림 6.4> 변경 후의 회로



<그림 6.5> 프로그램 변경순서

0B 스텝 AND 01
0C 스텝 STO 10

가 그것이다. 그리고 마지막으로 프로그램의 복귀 스텝 No.를 지시한다. 이것이 0D 스텝의 JMP 03 이다. 프로그램의 흐름은 03 스텝 LD 10으로 복귀한다.

6.5 프로그램의 변경과 RAM, ROM 사용방법

소형 PC를 사용하여 시퀀스 제어 시스템을 구성하는 경우 EP-ROM에 제어내용을 넣어 EP-ROM의 프로그램으로 동작을 시키는 경우가 많다. 이 때문에 프로그래밍을 하거나 프로그램 내용을 EP-ROM에 기록하거나 하는 이른바 프로그램 개발용 PC를 운전용 PC와는 별도로 갖게 된다. 이 개발용 PC로 프로그램을 개발하여 시운전까지를 하고 ROM화한 프로그램을 운전용 PC 안에 넣어 기계를 운전하게 된다.

프로그램을 변경하는 경우는 운전용 PC의 EP-ROM 내용을 프로그램 개발용 PC의 RAM 상에 일단 호출하여 프로그램 수정, 디버그, 시운전을 한다. 이것으로 결과가 변경 시방과 맞으면 앞의 EP-ROM 내용을 지우고 변경 후의 새로운 내용을 EP-ROM에 기입한다.

이 일련의 순서를 그림 6.5에 들었다. 그리고 EP-ROM 내용의 호출, EP-ROM에의 기입방법은 후술하므로 그것을 참조하기 바란다.

☞ 다음호에 계속 ☜

일본어투 생활용어 순화

순화대상용어	순화한 용어	순화대상용어	순화한 용어	순화대상용어	순화한 용어
다시(dash)	줄표, 대시	다시(出, 出汁)	맛국물	다오루(towel)	수건, 타월
다이(台)	대, 받침(대)	다이루(tile)	타일	다이알(dial)	번호판, 다이얼
다꾸앙(澤庵)	단무지	다데(縦)	세로	당고(團子)	경단
단도리(段取)	채비, 단속	단스	장롱, 옷장	당가(擔架)	들것
담뿌도랏쿠	덤프트럭, 덤프차	닭도리탕(一鳥湯)	닭볶음탕	대절(貸切)	전세
데나오시(手直)	재공사	데모도(手許)	곁꾼, 보조공	데스리(手摺)	난간
데꼬(挺子)	지렛대	데코보코(凹凸)	울록불록, 요철	덴마배(傳馬船)	거룻배
덴조(天井)	천장	덴찌(電池)	손전등	덴찌(大地)	상하
뎀뿌라(天婦羅)	튀김	뎃빵(鐵板)	우두머리, 철판	도끼다시(研出)	갈기