제 4 절

# STEP 5 연산 집합
# 논리 연산

제 4 절에서는

- 기본적인 STEP 5 동작의 이치 또는 부울(Boole) 대수 AND와 OR 논리 동작에 대해 기술하고 전형적인 적용에 대해 설명 한다.
- 프로그램을 입력할 때 뒤따르는 절차를 설명한다.
- PLC가 프로그램을 어떻게 검색(scan)하는지 설명한다.
- 프로그램에 따른 STATUS(상태) 표시에 대해 설명한다.

이 절을 모두 마친 교육생은

- 기본적인 AND와 OR 동작 그리고 프로그램에서 그 중요성을 이해한다.
- PLC와 프로그램 작성기 내부에서의 이치(또는 부울 대수) 명령문을 세가지 표현 방법으로 입력 할 수 있다.
- "입력", "전송", "출력", "수정" 기능을 응용할 수 있다.
- 상태(STATUS)의 표시를 이용하고 평가할 수 있다.

STEP 5의 연산 기능은 기본 연산, 추가 연산, 시스템 연산 동작으로 나누어진다. 이 과정에서는 기본 연산 동작만을 다룬다.

**기본 연산**

이 기본 연산은 모든 형식의 소프트 웨어 블록에서 간단한 이치 디지탈 기능을 프로그램하기 위하여 사용된다. 이미 제 2 절에서 언급한 대로 STEP 5 언어에는 3 가지의 서로 다른 방법으로 입력되고 표시될 수 있다. 즉, 사다리선도(LAD), 제어 시스템 흐름도(CSF) 그리고 명령문 목록(STL)이다.

**예 외**

다음의 동작들은 이 규칙에서 예외인데, 오직 STL 형식에서만 프로그램된다.

- 블록 호출
- 적재 및 전송 동작
- 산술 연산

**추가 연산 및 시스템 연산**

추가 연산과 시스템 연산에 대한 사항은 상위 교육 과정에서 함수 블럭의 프로그래밍에 관한 주제를 다룰때 설명될 것이다.

**연산 목록**

각각의 PLC에는 개별적인 연산이나 명령문에 대한 자세한 정보를 포함하는 자체의 연산 동작 목록이 있다.

**STEP 5 Operation Set**

| | Representation | | | Type of Block | | | |
|---|---|---|---|---|---|---|---|
| | CSF | LAD | STL | PB | OB | FB | SB |
| Basic operations | X | X | X | X | X | X | X |
| Supplementary operations | | | X | | | X | |
| System operations | | | X | | | X | |

그림 4.1

AND 연산

AND 연산은 회로도에서 접점의 직렬 연결과 같다.
AND 기능의 출력 Q 4.0과 Q 4.1은 입력(I 1.0, I 1.1)이 모두 "1" 일때 출력이 "1"이 된다. 둘중의 하나가 "0"으로 검색되면 출력은 "0"이 된다. 즉, 스위치가 OFF 된다.

기본적으로 입력을 검색하여 얻어진 논리 동작의 결과는 몇개의 동작으로 지정된다. 그림 4.2의 실습 예제에서 이런 목적으로, 두개의 Q 4.0과 Q 4.1로 지정하였는데 하나씩 차례로 프로그램된다.

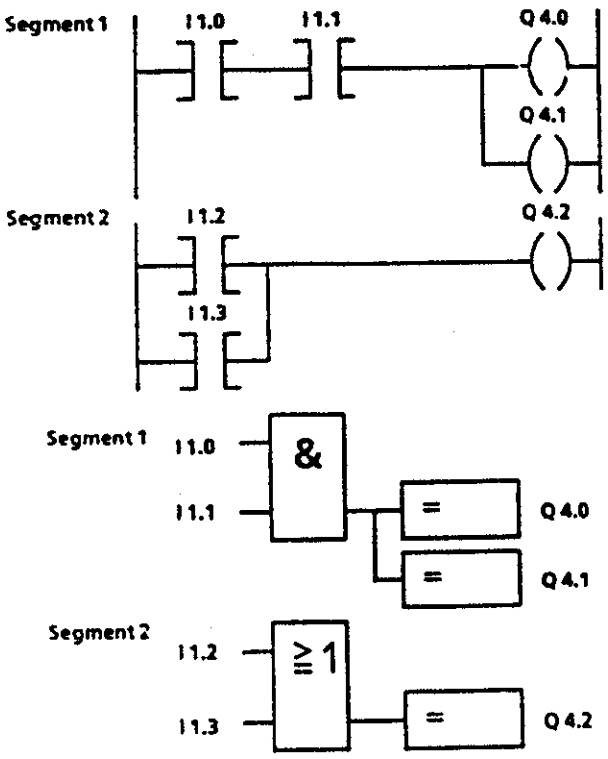
OR 연산

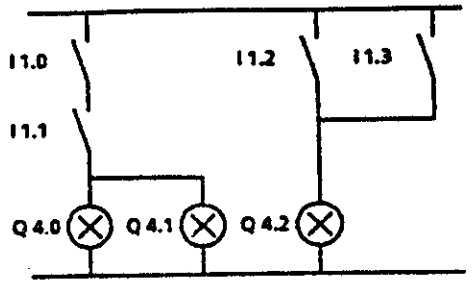
OR 동작은 회로도에서 접점의 병렬 연결과 같다.
출력 Q 4.2 는 OR 동작에 의해 입력 (I 1.2 , I 1.3)의 두개중 하나가 "1" 일때 "1"이 출력된다. 모든 입력이 "0"일때 출력은 "0"이 되고 스위치는 OFF 된다.

참 고

검색되는 연산수의 갯수는 프로그램 작성기의 메모리에 저장되어 있는 명령어의 숫자에 의해 제한된다. 그러나 도식적 방법(LAD, CSF)에서 기호의 갯수는 화면의 폭에 의해 또한 제한된다 :

- 사다리선도(LAD)에서는 직렬로 7개의 입력의 검색과 1개의 출력 지정이 가능하다.
- 제어 시스템 흐름도(CSF)에서는 혼합 연산의 그래픽 요소를 6개까지 직렬로 연결할 수 있다.




| | | | |
|---|---|---|---|
| Segment 1 | A | I | 1.0 |
| | A | I | 1.1 |
| | = | Q | 4.0 |
| | = | Q | 4.1 |
| | * * * | | |
| Segment 2 | O | I | 1.2 |
| | O | I | 1.3 |
| | = | Q | 4.2 |
| | B E | | |

그림 4.2

AND-뒤에-OR
(AND-before-OR)
논 리

AND-뒤에-OR 논리는 여러개의 직렬 연결한 접점을 병렬로 연결한 것이다.

출력 Q 4.0은, 직렬 및 병렬로 연결된 가지(지로)로 구성되어 있고, 여러개의 병렬 가지 중에서 하나의 직렬 연결이 모두 닫히거나 또는 단일한 접점 I 0.5이 닫혀져야, "1"이 된다.

AND-뒤에-OR 연산은 STL에서 괄호 없이 프로그램 될 수 있다. 그러나 병렬 가지는 프로그램의 O(=OR 기능) 문자를 사용하여 하나 하나 분리시켜 작성하여야 한다.

먼저, AND 기능이 수행되고 AND 연산 결과에 따라 OR 기능의 결과가 생긴다. 첫번째 AND 기능(I 0.0, I 0.1, I 0.2)은 두번째 AND 기능(I 0.3, I 0.4)으로 부터 O (= OR 기능)에 의해 분리되며, 단일한 연산수(I 0.5)는 "O I 0.5"로써 기능 "O"의 끝에 바로 프로그램할 수 있고, 그 다음에 출력을 지정하는 프로그램을 작성한다.

**주 의 !**

- 단일 O는 AND 기능과 OR 동작되는 AND 기능 사이에 항상 있어야 한다.
- AND-뒤에-OR 끝부분에 단일 OR 게이트는 O I... 와 같이 프로그램될 수 있다.

**실습 문제**

* AND-뒤에-OR 동작에 대하여 STL로 프로그램 작성하시오.

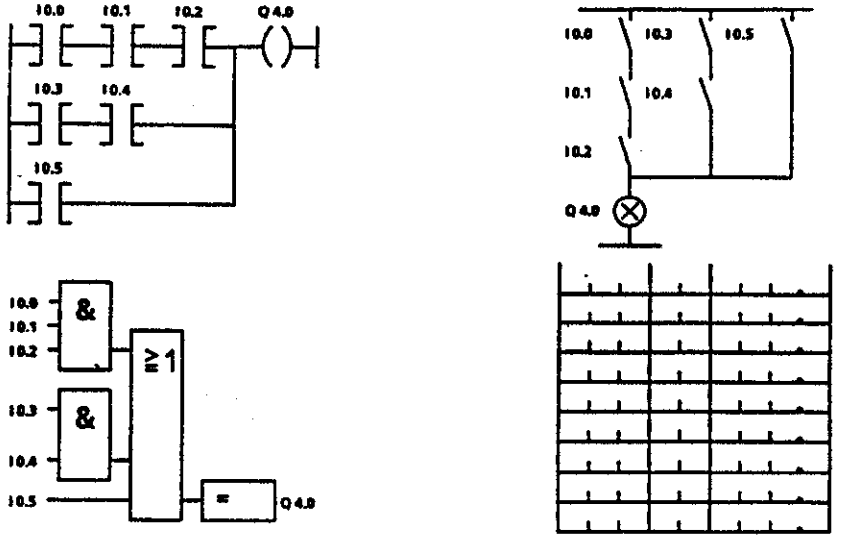



그림 4.3

OR-뒤에-AND
(OR-before-AND)
논 리

OR-뒤에-AND 논리 동작은 여러개의 병렬로 연결된 접점을 직렬로 연결한 것이다.
출력 Q 5.4는 병렬 및 직렬 결선으로 구성된 OR-뒤에-AND 논리의 출력이며, 각 병렬 가지에서 여러개의 병렬 접점중 하나가 "1"이고 하나의 접점 I 1.2가 닫혀져야만 출력이 "1"이 된다.

**주 의 !**

**OR-뒤에-AND 기능의 OR 동작은 반드시 괄호에 의해 닫혀져야 한다. 이렇게 하면 AND 기능을 수행하기 전에 OR 기능이 확실하게 수행된다.**

괄호 기능(STL)

이 논리 명령은 괄호에 의해 연산 동작으로 구성된다. 가능한 명령어는 다음과 같이 두 개이다 :

A( "왼쪽 괄호"
) "오른쪽 괄호"

"왼쪽 괄호"의 갯수와 "오른쪽 괄호"의 갯수는 같아야한다. 즉, 괄호를 열면 반드시 닫아야 한다.

괄호에 의한 표현 A(... ...)은 복잡한 OR 또는 OR-뒤에-AND 동작을 포함한다.

실습 문제

* OR-뒤에-AND 동작에 대하여 STL로 프로그램 작성하라.

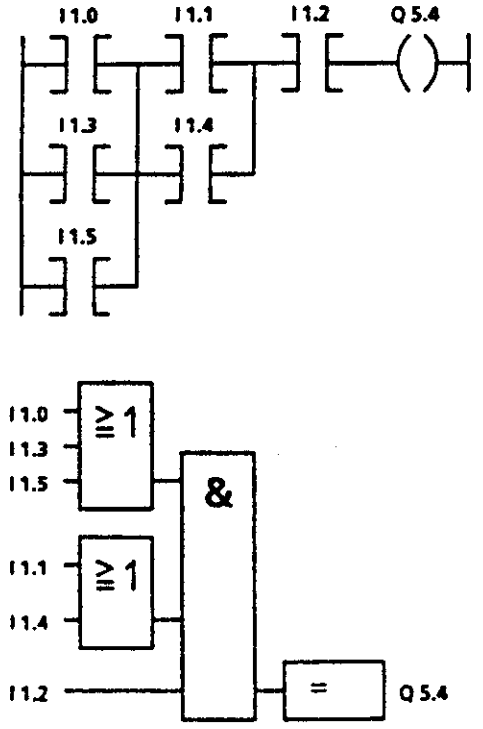



그림 4.4

SIMATIC S5 PLC는 여러개의 중첩된 괄호(네스트) 연산을 사용할 수 있다.

**괄호의 중첩 갯수**

S5-101U, S5-100U 그리고 S5-115의 PLC에서는 최대 6개의 괄호까지 동시에 열 수 있다. S5-135U, S5-150U 그리고 S5-155U에서는 한꺼번에 7개까지 괄호를 열 수 있다.

CSF나 LAD에서는 화면의 폭때문에 최대의 괄호 갯수까지 프로그램할 수 없다. CSF에서는 6개, LAD에서는 8개까지 표현할 수 있다.

**실습 문제**

STL에 따라 CSF 프로그램을 작성하라.

**참 고**

각 괄호의 쌍을 표시할 것.

STL

```
:A(
:A(
:O I1.0
:O I1.1
:)
:A(
:O I1.2
:O I1.3
:)
:O I1.4
:)
:A(
:O I1.5
:O I1.6
:)
:= Q 4.0
```

CSF

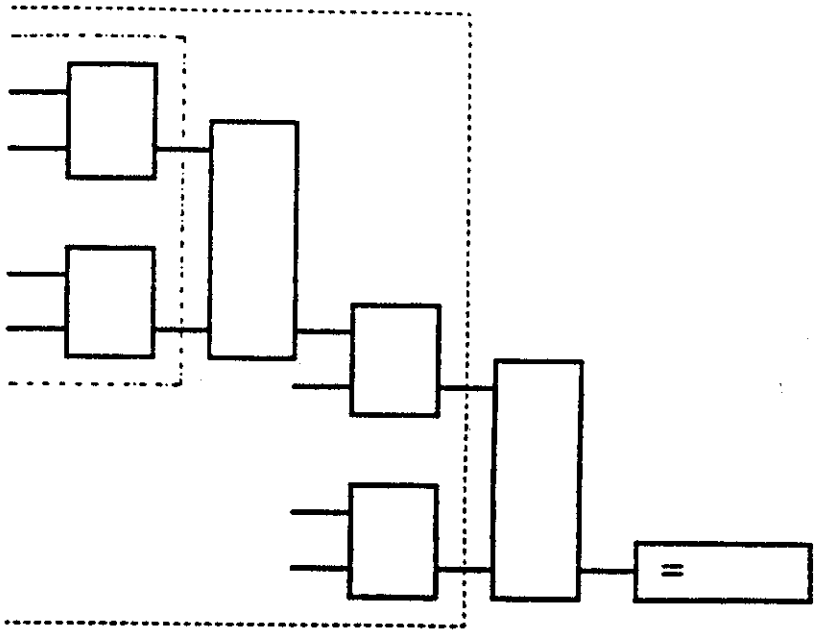



그림 4.5

Notes :

프로그램 작성기에는 소프트웨어 블록을 입력하기 위해 여러가지의 정교한 편집 기능들이 있다. 아래에서는 STL 표현으로 소프트웨어 블록의 입력을 설명한다. 우선 첫째로 세개의 세그먼트로 구성된 PB 1 블록을 하드 디스크상의 파일 S15A에 작성하시오.

**생략시의 형식**

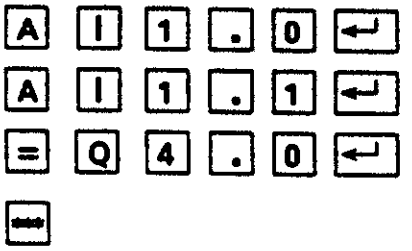
* LAD, CSF, STL 팩키지를 선택하라. 생략시(presetting)의 형식을 완성하라.
  PROGRAM FILE : B:*S15A*@@ST.S5D
  LANGUAGE : STL
  COMMENTS : YES
  기능 키 F6으로 생략시 형식의 입력을 확인하라.

**블록의 입력**

* 입력 기능을 호출하라.
  이를 위해 F1 = INPUT 키를 누르고, 다시 한 번 F1 = BLOCK 키를 누른다.
  화면에는 다음과 같은 명령어가 나온다.
  INPUT DEVICE : BLOCK :
  다음과 같이 명령어 문장을 완성하라 :
  INPUT DEVICE : *FD* BLOCK : *PB1*
  그리고 Enter 키를 누른다.
  화면에는 다음과 같이 표시된다 :

```
PB 1          B:S15A@@ST.S5D          LEN = 0
  SEGMENT 1                          INPUT
      •
      •
```

* 이제, 다음의 키들을 이용해 프로그램을 세그먼트 1에 입력한다 :




**세그먼트의 종료**


화면에는 다음과 같이 표시된다 :

```
PB 1          B:S15A@@ST.S5D          LEN = 9
  SEGMENT 2                          INPUT
      •
      •
```

PB1

```
SEGMENT 1                                   B:S15A@@ST.S5D    LEN=26
      :A   I   1.0                                            INPUT
      :A   I   1.1
      :=   Q   4.0
      :...
SEGMENT 2
      :O   I   1.2
      :O   I   1.3
      :=   Q   4.2
      :...
SEGMENT 3
      :A(
      :O   I   1.0          01
      :O   I   1.3          01
      :O   I   1.5          01
      :)                    01
      :A(
      :O   I   1.1          01
      :O   I   1.2          01
      :O   I   1.4          01
      :)                    01
      :A   I   1.2
      :=   Q   5.4
      :BE
```

그림 4.6

* PB 1의 세그먼트 2와 3을 그림과 같이 입력한다.

* 마지막 명령어(BE)가 입력된 다음에 이 블록은 하드 디스크에 전송된다. 그리고나면, 메뉴가 나타난다.

참 고

세그먼트 3에서 오른 쪽에 표시된 숫자 01은 그 순간에 열려진 괄호의 갯수를 나타낸다.

오류 수정에 대한 참고

프로그램 작성 도중에 어떤 실수를 했을 경우 커서 이동용 키를 사용하여 커서를 잘못된 글자위에 옮겨 올바로게 다시 겹쳐 쓸 수 있다.

프로그램 블록 PB 1이 올바르게 입력되었는지를 블록을 화면에 표시하여 확인 할 수 있다. 그리고 수정도 할 수 있다.

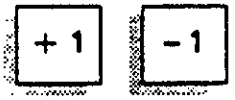
**블록의 화면 표시**

* 메뉴에서 F2를 두번 눌러서(F2 = OUTPUT, F3 = BLOCK) OUTPUT(디스플레이) 기능을 선택하라.

* 블록 PB 1을 읽거나 화면에 디스플레이하라.
  이를 위해, 다음과 같이 명령문을 완성하라 :
  OUTPUT DEVICE : *FD* BLOCK : *PB1* SEARCH : PTR :
  그리고 Enter 키를 누른다.
  화면에는 PB 1의 세그먼트 1이 표시된다.

* PB 1을 점검하라.

**다음/이전 네트워크**




세그먼트 1의 프로그램을 전번의 실습 문제(4-13쪽)에서 작성한 프로그램 블록을 화면에 표시하여 비교하라. "Scroll up"('+1') 키를 사용하여 세크먼트 2와 3을 하고 점검하라.

이제, 화면에 표시된 PB 1을 수정하라 :

명령어 = Q 4.1을 세그먼트 1에 추가한다.

* PB 1의 세그먼트 1을 선택하라.

**수 정**


세그먼트 내부에서는, CORRECTION(수정) 모드로의 전환만 가능하다. 이는 Corr 키를 이용하여 호출한다.

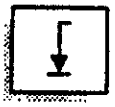
* Corr 키를 누른다.

화면 오른쪽 윗편에 표시는 현재 CORRECTION 모드에 들어가 있음을 나타난다.

**명령어 삽입**

* 'segment end' 앞에 명령문 = Q 4.1를 삽입한다.

**수직 확장**



수직으로 확장하기 위해서, 커서를 'segment end'에 놓는다. 즉,

: * *

그리고 'expand vertical'키를 눌러 새롭게 입력할 명령어에 대한 자리를 확보하라.

* Enter 키를 눌러 수정을 완료하라. 수정이 완료되면 출력 기능으로 다시 되돌아 온다.

PB1

```
SEGMENT 1                          B:S15A@@ST.S5D     LEN=26
      :A  I   1.0                                     CORRECTION
      :A  I   1.1
      :=  Q   4.0
      :=  Q   4.1   <---------- Insert
      :...
SEGMENT 2
      :O  I   1.2
      :O  I   1.3
      :=  Q   4.2
      :...
SEGMENT 3
      :A(
      :O  I   1.0        01
      :O  I   1.3        01
      :O  I   1.5        01
      :)                 01
      :A(
      :O  I   1.1        01
      :O  I   1.2        01   <---------- Delete
      :O  I   1.4        01
      :)                 01
      :A  I   1.2
      :=  Q   5.4
      :BE
```

그림 4.7

**명령문 삭제**

세그먼트 3의 명령문 O I 1.2를 삭제하고자 한다.

* '+1' 키를 이용해 세그먼트 3을 선택하라.

**문자 삭제**

* 커서를 O I 1.2 명령문의 왼쪽의 콜론(:)에 두고 'Delete character' 키를 눌러 명령문을 완전히 지워라.

**수정 기능의 종료**

* Enter 키를 눌러 수정을 끝낸다(출력 기능으로 되돌아 온다).

* Enter 키를 다시 누른다.
'PB 1 ALREADY IN DESTINATION FILE, OVERWRITE?' 라는 질문이 나온다. Enter 키를 눌러 확인하라. 수정된 PB 1은 하드 디스크로 전송된다.

각각의 프로그램은 프로그램 기억 장치에 차례차례로 저장되어 있는 논리 연산의 순서로 구성되어 있다. 논리적 순서는 입력, 출력, 플래그, 타이머 및 카운터를 검색하기 위한 하나 또는 여러개의 명령어 그리고 출력, 플래그, 타이머 및 카운터를 동작시키기 위한 하나 또는 여러개의 명령어 등으로 구성된다.

**검색의 결과**

PLC의 프로세서가 프로그램을 검색하면, PLC의 프로세서는 먼저 각각의 검색 연산의 결과를 얻는다. 만약 검색 연산이 참이면 결과는 "1"이고, 그렇지 않으면 "0"이 된다.

**논리 등작의 결과**

첫번째 검색(first scan)의 결과는 논리 연산의 결과(RLO)로서 프로세서에 저장된다. 다음의 검색 연산에 대한 결과는 첫번째 검색의 RLO와 AND 또는 OR 연산을 하여 새로운 RLO를 생성한다.

마지막 검색 연산이 끝나거나 논리적 순서의 명령문의 처리가 완료되면, 논리 연산의 결과는 변하지 않고 유지된다. 따라서 동일한 RLO (논리 연산의 결과)로써 여러개의 출력 지정을 처리할 수 있다. 즉, RLO는 여러개의 출력에 지정된다. RLO는 마지막 지정 명령문이 처리될 때까지 유지된다. RLO는 조건부 동작(지정)과 검색 동작(첫번째 검색) 사이의 경계에서 그 효력이 상실된다.

**첫번째 검색**

PLC의 프로세서는 첫번째 검색를 통하여 새로운 RLO를 생성한다. 첫번째 검색의 결과는 논리 연산없이 저장되며, 첫번째 검색에 대한 AND 또는 OR 연산은 의미가 없다. 그러나 프로그램과 동작의 형식상, 항상 포함되어야 한다.

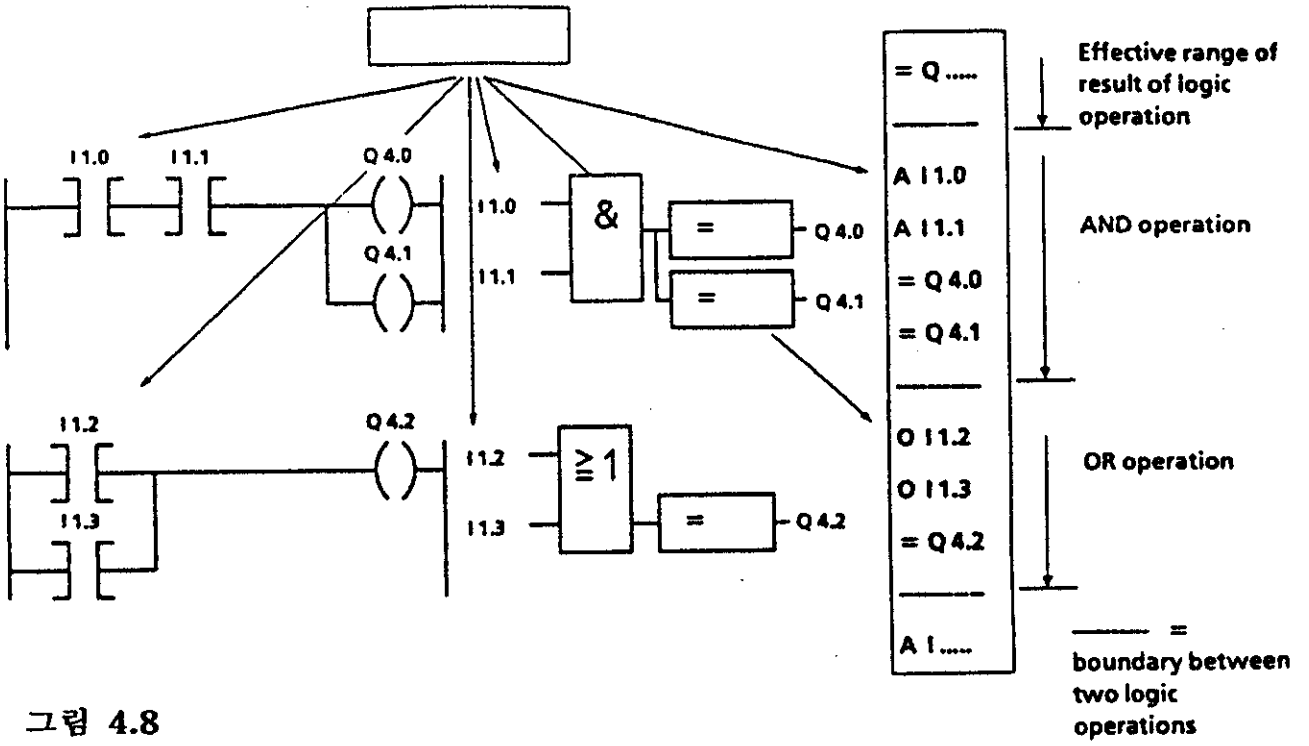



그림 4.8

이 실습 문제의 목적은 신호 상태, 또는 연산수의 상태 그리고 논리 연산의 결과 사이의 차이점을 설명하기 위해서이다.

기억할 사항 :

RLO = "1"

논리 연산의 조건이 충족되면, 그 결과는 RLO = "1"이 된다; 조건부 연산이 실행된다.
예 제 : RLO = "1"이면 출력의 지정(= Q 4.0)은 출력, 플래그 등에 의해 설정된다.

RLO = "0"

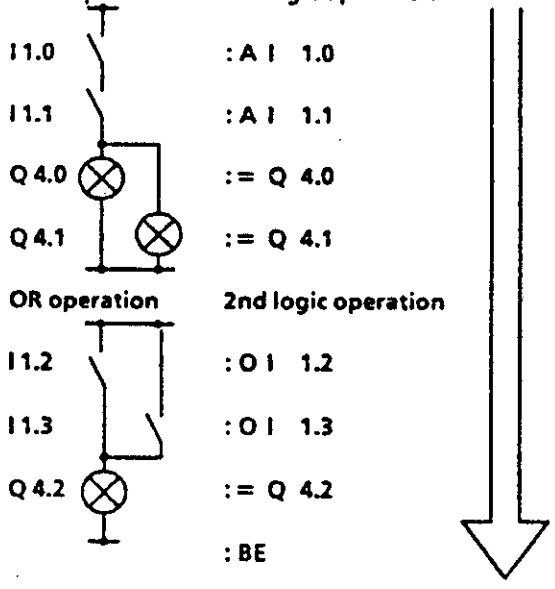
논리 연산이 조건을 충족시키지 못하면 RLO = "0"이 된다; 조건부; 연산이 실행되지 않는다.
예 제 : RLO = "0"이면 출력의 지정(= Q 4.0)은 해당되는 출력, 플래그 등을 재설정시킨다.

두 논리 연산 사이의 경계가 교차하면, 새로운 RLO가 생성된다.

**실습 문제**

* 주어진 신호 상태에 따라 어떤 RLO가 생성 되는가?

* 출력에 전류가 흘러 램프가 켜지기 위한 모든 조건을 테이블로 작성하라.




| 1st status | | 2nd status | | 3rd status | | 4th status | |
|---|---|---|---|---|---|---|---|
| RLO | STAT | RLO | STAT | RLO | STAT | RLO | STAT |
| 0 | 0 | 1 | | 0 | | 1 | |
| 0 | 0 | 0 | | 1 | | 1 | |
| 0 | 0 | | | | | | |
| 0 | 0 | | | | | | |
| | | | | | | | |
| 0 | 0 | 1 | | 0 | | 1 | |
| 0 | 0 | 0 | | 1 | | 1 | |
| 0 | 0 | | | | | | |

RLO: Result of logic operation; STAT: STATUS or signal state

그림 4.9

적 용

프로그램에 따른 신호 상태의 표시는 각각의 연산수의 현재의 신호 상태를 보는데 사용된다.
프로그램 작성기의 출력(DISPLAY)기능에 의해 프로그램 작성기의 화면에 한 블록 세그먼트에 대한 STL, CSF, LAD가 표시된다.
프로그램 작성기의 STATUS기능은 소프트웨어 블록의 오류 수정(debugging) 또는 고장 발생때에 자주 사용된다. 이 기능은 기능 선택 메뉴에서 F3 = TEST를 다시 F3 = STATUS를 눌러서 선택한다.

STATUS STL

각각의 연산수와 논리 연산의 결과에 의해 다음의 내용이 STL의 오른쪽에 테이블 형태로 표시된다 :

RLO = 논리 연산의 결과
STATUS = 신호 상태

STL형식에서의 STATUS 기능으로 주어지는 다른 정보들은 이 교육 과정의 나중 부분에서 또는 더 상위 과정에서 다룬다.

STATUS CSF

각각의 연산수와 논리 연산의 결과에 따른 신호 상태에 따라서, 기능 기호(신호 상태)의 입력과 출력 그리고 각 기호 사이의 연결(논리 연산의 결과)은 다음과 같이 표시된다 :

= = = = 신호 상태나 논리 연산 결과가 "1"
- - - = 신호 상태나 논리 연산 결과가 "0"

STATUS LAD

LAD 형식에서는, 기호는 "0" 또 "1"을 검색(scan)한 결과만을 나타내며, 이것은 각 연산수의 신호 상태에 의하여 결정된다. 다시말하면, 신호 상태는 다음과 같이 표시된다 :

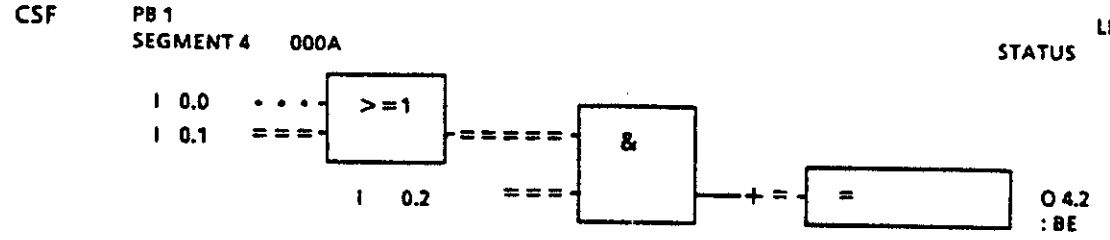
= = = = 신호 상태나 논리 연산 결과가 "1"
- - - = 신호 상태나 논리 연산 결과가 "0"

이런 이유로, STATUS의 표시 상태를 판독할 때 주의하여야 한다.

STL

```
PB 1                                                    DBADR = 0000                    LEN = 30
SEGMENT 4          STL STATUS          RLO  STATUS/ACCU1   ---ACCU2---   STATUS        SAC
   : A (                                  1                             00000011      7122
   : O   I0.0                             0              0              00000000      7124
   : O   I0.1                             1              1              00000110      7126
   : )                                    1                             00000110      7128
   : A   I0.2                             1              1              00000110      712A
   : =   Q 4.2                            1              1              00000111      712C
```

CSF




LAD

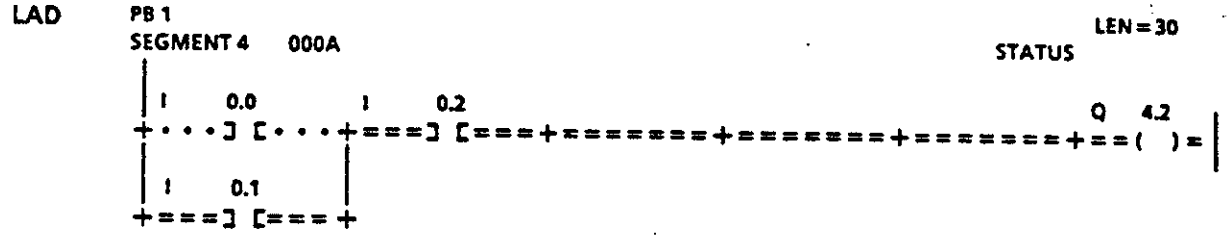



그림 4.10

**Notes :**

소프트웨어 블록은 PLC의 프로세서에 의해 다음과 같을 때에만 처리하고 시험할 수 있다.

1. 프로그램 블록이 PLC의 메모리에 저장되어 있을때
2. 조직 블록 OB 1에서 프로그램 수행이 호출되어 주기적인 프로그램 수행으로 통합되었을 때

수정된 프로그램 블록 PB 1을 모의 실험 장치로써 시험하여 보자.

**블록 전송**

* 프로그램 블록 PB 1을 하드 디스크로부터 PLC의 메모리로 전송하라.

프로그램 작성기의 TRANSFER 기능은 기능키 F7 AUXILIARY FUNCTIONS(보조 기능)을 사용하여 선택한다.

* F7 = AUX.FCT 그리고 F1 = TRANSFER 키를 누르고 아래의 명령어 문장을 완성하라 :
  TRANSFER FROM SOURCE: *FD* BLOCK: *PB1* TO DEST.: *PC* BLOCK :

  Enter 키를 누른다. 프로그램 블록은 원래의 기억 장치 FD로부터 PLC의 메모리로 전송한다. 메뉴가 프로그램 작성기의 화면에 다시 나타난다.

**블록 검사**

* 모의 실험 장치를 이용해 프로그램 블록을 시험하라.
  모의 실험 장치의 I 1.2 또는 I 1.3 스위치를 동작시켜라. 출력 Q 4.2에 불이 켜질 것이다.
  관찰한 바를 기록하시오. .....

  .....

* 입력 기능을 호출하여 다음의 명령문을 완결시킨다.
  INPUT DEVICE : *PC* BLOCK : *OB1*
  그러면 4-23쪽의 그림처럼 프로그램 OB 1이 나타난다.

**참 고**

INPUT 기능은 "FUNCTION SELECT" 메뉴에서 호출할 수 있다. 이 메뉴는 F8 = RETURN 키에 의하여 호출된다.

* Enter 키를 눌러서 입력된 내용을 확인할 때, 다음의 메세지가 나타난다 :
  OB 1 ALREADY IN PC, OVERWRITE ?

* Enter 키를 다시 누른다(이전의 OB 1이 방금 입력한 내용으로 바뀐다).

* 스위치 I 1.2 또는 I 1.3을 모의 실험 장치에서 전류를 흘릴 때 Q 4.2가 출력 되는가 ?

  yes ☐ no ☐

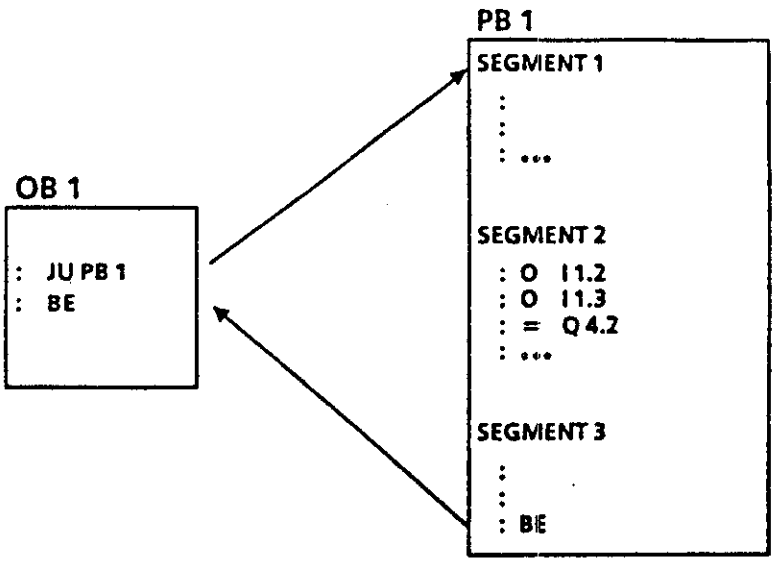

PB 1
SEGMENT 1
: •••
SEGMENT 2
: O I1.2
: O I1.3
: = Q4.2
: •••
SEGMENT 3
: BE
OB 1
: JU PB 1
: BE

그림 4.11

**Notes :**

프로그램에 종속된 신호 상태 표시 기능을 이용하여, STL/CSF/LAD의 세가지 표현 방법으로 프로그램 블록 PB 1을 검사하라.

**STATUS 기능 호출**

* 프로그램 작성기에서 STATUS 기능을 호출하라.

  이렇게 하기 위하여 F3 = TEST 그리고 F3 = STATUS 키를 누른다.
  그리고 STATUS(신호 상태)모드에서 보고자하는 블록을 명령어 문장으로 나타내시오.
  STATUS BLOCK : *PB1* SEARCH :

* STL 표현 모드에서 Enter 키를 눌러 PB 1의 3개의 세그먼트의 신호 흐름을 관찰하라.

* 세그먼트 1과 2의 상태 표시를 4-19쪽에서 작성한 표의 신호 상태와 비교하라.

또한 도식적 형식인 CSF와 LAD 형태로 PB 1의 신호 흐름을 관찰하라.

**표현 방법의 변경**

F7 키를 이용해 모드를 전환하라(그림 4.12a 참조).

* 세그먼트 1을 선택하고, F7을 누르시오. 세그먼트 1이 사다리 선도(LAD)로 화면에 표시된다.

* F7 키를 다시 누른다. 세그먼트 1이 CSF로 화면에 표시된다.

* LAD 그리고 CSF 형태로 3개의 세그먼트를 모두 검사한다.

* F7 키를 이용해 STL 모드로 되돌아 오라.

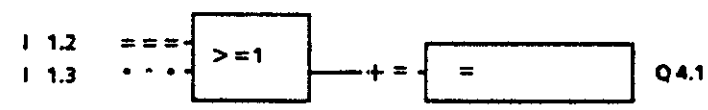
**STATUS 기능의 종료**

* Break 키를 두번 눌러 FUNCTION SELECTION(기능 선택) 메뉴로 되돌아 오라.

STL

```
PB 1                                        DBADR=0000      LEN=26        ABS
SEGMENT 2      STL STATUS          RLO  STATUS/ACCU1  ---ACCU2---   STATUS      SAC
   :O   I1.2                        1          1                   00000000    71BE
   :O   I1.3                        1          0                   00000110    71C0
   :=   Q4.1                        1          1                   00000111    71C2
   :***
```

CSF




LAD

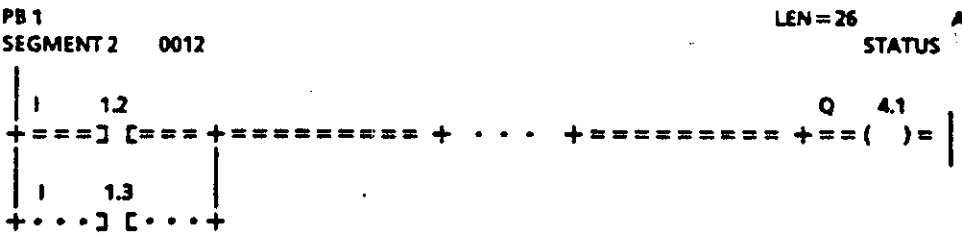



그림 4.12

| F1 DSP SYM | F2 REFER-ENCE | F3 SEARCH | F4 DIAG-NOSIS | F5 ADDRE-SSES | F6 LIB. NO. | F7 → LAD | F8 |
|---|---|---|---|---|---|---|---|

그림 4.12a *STATUS Menu*

**주 의 !**

만약 STATUS 기능이 호출된 다음에, 프로그램 작성기의 화면에 아래와 같은 메세지가
**BLOCK DOES NOT EXIST IN PC**
표시되면 PLC의 메모리에는 검사될 프로그램이 없는 것이다.
메세지가
**INSTUCTION(or BLOCK) CANNOT BE PROCESSED**
이면 검사하려는 주기적 연산(OB 1?)에 통합되지 않았음을 나타낸다.

하나의 블록을 프로그램 작성기의 STATUS 기능으로 검사하는 동안에도, OUTPUT(디스플레이) 기능과 똑같은 방법으로 수정할 수 있다. 세그먼트 내에서 수정될 수 있을뿐 아니라, 세그먼트를 완전히 지우거나 추가할 수 있다.

**위 험 !**

STATUS 모드에서 프로그램의 수정은 전원이 연결된 상태로(VDE 0105!!!) 결선을 수정하는 것과 같다. 오동작의 위험이 클 뿐 아니라 이 때문에 인명에 위해를 주거나 제어대상 또는 장치를 파괴하는 수가 있다.

PLC가 RUN 모드에 있을때 프로그램 작성기의 STATUS 기능에 의해 프로그램 블록 PB 1을 수정하시오.

**STATUS 호출**

* STATUS PB 1의 세그먼트 3을 호출하라.
* 모의 실험 장치에서 스위치 I 1.2 만을 동작 시킨다. 출력 Q 5.4는 출력되지 않을 것이다.
* 수정 키를 누르고 커서를 A I 1.2의 명령문으로 이동시킨다.

**RUN 모드에서의 수정**

* 4-27쪽의 그림과 같이 명령문을 수정하라.
  A(=AND)를 O(=OR)로 겹쳐쓰고 Enter 키를 누르시오.
* PLC의 PB 1이 수정되었을 때 모의 실험 장치의 출력 Q 5.4를 관찰하라 :

  출력은 ..... !

**주 의 !**

수정 작업은 PLC가 STOP 모드에 있을 때에만 실시할 것 !

수정된 블록은 실제적으로 새로운 블록으로 다루어야 한다.
수정된 블록은 처음에는 개별적으로 검사하고 마지막에 전체의 프로그램을 다시 합친다.

**참 고**

RUN 모드에서 원하는 않는 프로그램 수정을 피하려면 생략시(PRESETTING) 형식으로 RUN 모드에 [MOD IN STOP] 속성을 지정하면 된다.

STATUS PB 1

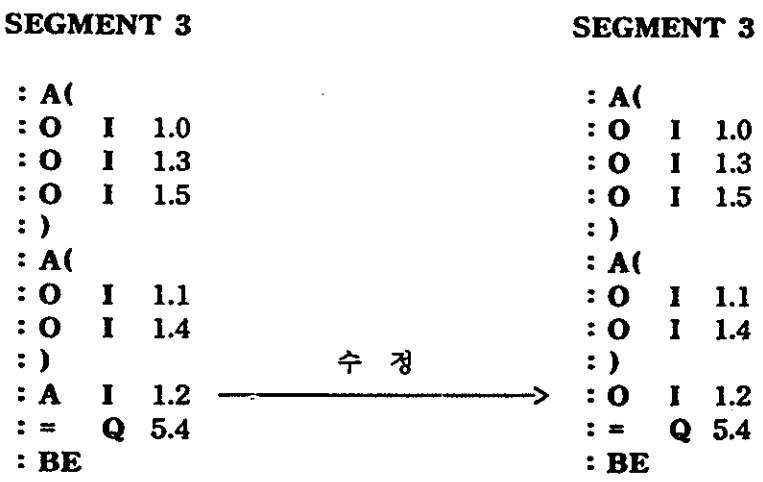



그림 4.13

Notes :

사용자 프로그램의 주기적 검색(cyclic scanning)은 PLC의 프로세서의 시스템 프로그램(system program)에 의하여 제어된다.

**디지탈 입력 데이타 기억장치 PII**

검색 시간이 시작된 후, PLC의 디지탈 입력의 신호 상태들은 검색되어 다음의 검색 주기까지 디지탈 입력 데이타 기억 장치(process input image) PII에서 검색되고 저장된다. OB 1이 호출됨으로써 그 다음에 시작하는 프로그램 검색 동안에 PII로부터 입력 신호의 상태가 얻어진다.

**디지탈 출력 데이타 기억장치 PIQ**

하나의 검색 주기 동안에 하나 또는 몇 개의 출력의 신호 상태가 새로 성립하면, 새로운 신호 상태는 먼저 디지탈 출력 데이타 기억장치(process output image) PIQ에 저장된다. 프로그램이 조직 블록 OB 1의 마지막 명령(BE)까지 검색되면, 디지탈 출력 데이타 기억장치 PIQ의 내용이 출력 모듈(output module)로 전송된다. 한 검색 주기 동안에 하나의 출력에 프로그램의 여러 지점에서 상반된 ON/OFF 명령이 지정되더라도, 직전의 검색 주기 동안에 디지탈 출력 데이타 기억장치 PIQ에 저장되었던 신호 상태 만이 유효하게 지속된다.

디지탈 입/출력 데이타 기억장치(process I/O image)는 PLC 프로세서 내부의 RAM 영역이며, 이 영역 안에 디지탈 입/출력을 기억하는 구역이 지정되어 있다.

**장 점**

입/출력의 신호의 상태를 디지탈 입/출력 데이타 기억장치에 완충 기억(buffering) 시키면, 검색 주기 동안에 입력의 신호의 변화가 생기더라도 PLC의 기능의 순서(functional sequence)에 나쁜 영향을 주지 않으며, 출력이 "진동(chattering)" 즉, 짧은 시간에 출력의 ON/OFF 스위치 동작이 발생하는 것을 방지할 수 있다.

더욱이, 입/출력의 신호 상태를 완충 기억(버퍼링)에 의하여 검색 시간(scan time)이 짧아진다. 왜냐하면 디지탈 입/출력 데이타 기억장치와 같은 PLC의 프로세서의 시스템 데이타 기억장치(system data memory)에 접근하는 것이 입/출력 모듈에 직접으로 접근하는 것보다 훨씬 짧은 시간이 소요되기 때문이다.

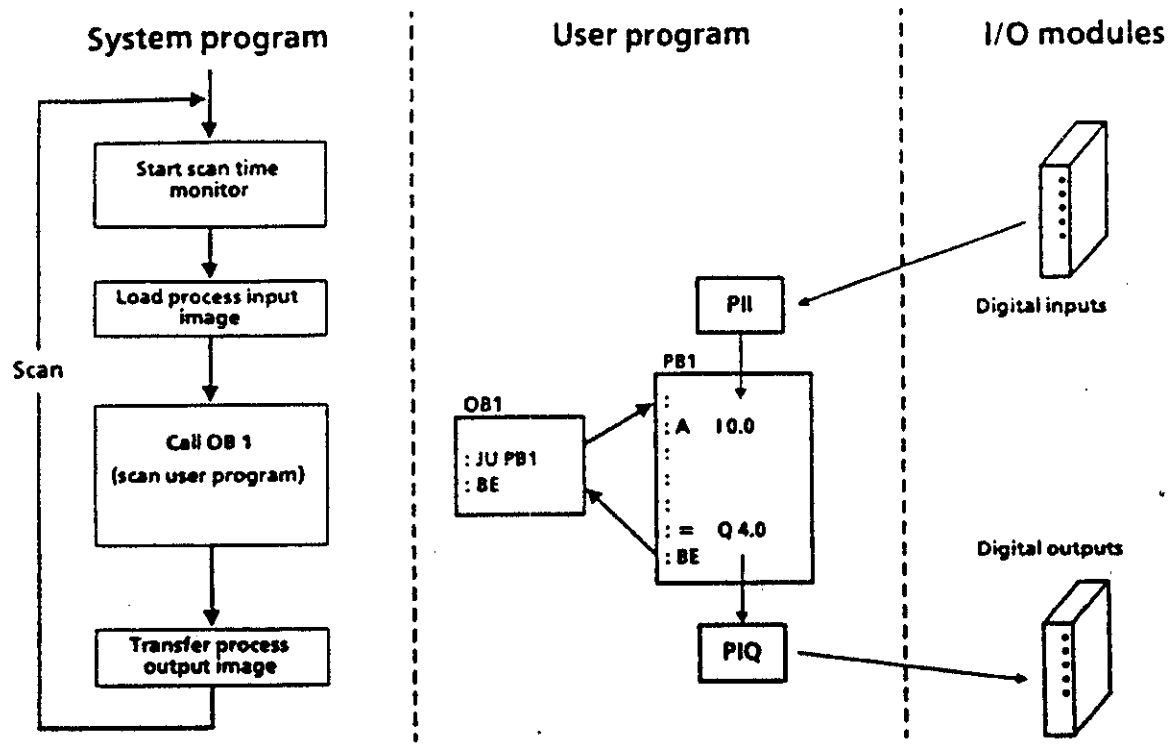

System program
User program
I/O modules
Start scan time monitor
Load process input image
Scan
Call OB 1
(scan user program)
Transfer process output image
PII
Digital inputs
PB1
OB1
: JU PB1
: BE
: A I0.0
: = Q 4.0
: BE
Digital outputs
PIQ

그림 4.14

Notes :

앞의 몇 쪽에서는, PLC의 입력의 신호 상태, 즉 살아있는 상태(energized) 또는 죽은 상태(deenergized)를 검색하는 것에 관하여 설명하였다. 검색은 제어 프로그램으로 정한 지정된 순서에 의해 수행된다. 제어 대상에 설치된 센서가 상시 개방형이든지 상시 접촉형인지는 고려하지 않는다.

그러나 센서의 형식이나 기술적 특성은 프로그램을 작성할 때에 미리 인지하고 있어야 한다.

**상시 개방형(NO) 접점**

입력에 연결한 센서가 상시 개방(NO) 접점이면, 센서가 살아있을 때 입력은 "1"이 된다.

**상시 폐로형(NC) 접점**

상시 폐로형 센서에 입력을 연결하면, 센서가 살 때 입력은 "0"이 된다.

PLC는 입력에 연결된 센서가 NO인지 NC인지 결정할 수 없다. 나중에 알게 되겠지만, PLC는 입력 신호가 "1" 또는 "0"의 상태 여부만을 검색할 뿐이다.

**신호 상태 "1"**

입력 상태가 "1"로 검색되면, 이 신호 상태가 살아있는(energized) NO 접점 또는 죽은(de-energized) NC 접점에 의한 것인지를 상관하지 않는다.

**신호 상태 "0"**

입력 상태가 "0"으로 검색되었다 하더라도, 이 신호 상태가 죽은 NO 접점 또는 살아있는 NC 접점에 의한 것인가는 무관하다.

제어 대상 공정에서 NC 또는 NO 접점중 어느 것을 사용할 것인가를 안전 규정(전선의 절단, 동작 가능등)을 토대로 결정하여야 한다.

**주 의 !**

독일 공업 규격 DIN VDE 0113, 1/02.86 부분, 5절에 의하면, 기계는 가능하다면, 전원을 차단하였을때 정지하여야 한다. 이 유형의 정지가 안전하고 신뢰성이 있는데, 왜냐하면 센서 회로의 합선(short- circuit), 전선의 절단 또는 전원 고장이 발생하여도 기계가 정지 기능을 할 수 있기 때문이다. 이런 이유 때문에, OFF(차단) 스위치와 리미트 스위치에서 정상적으로 사용되는 NC 접점은 결코 신호 상태가 "0"으로 검색되는 NO 접점을 교체하면 안된다 !

| Sensor contacts and their functions | | | Effect of the sensor contact on the program |
|---|---|---|---|
| The sensor contact is | The sensor is | Voltage signal at input | Signal state at input |
| normally open | energized | present | 1 |
| | de-energized | not present | 0 |
| normally closed | energized | not present | 0 |
| | de-energized | present | 1 |

그림 4.15

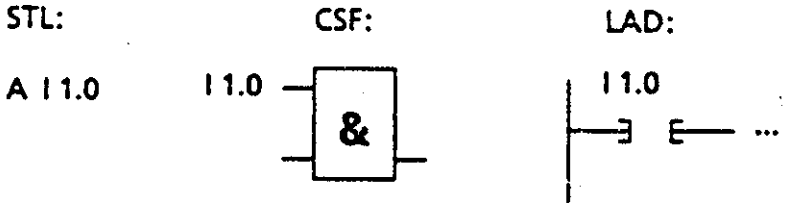
**Notes :**

배타적-OR 연산

4-33 쪽의 그림은 배타적-OR 기능을 보이고 있는데, 출력 Q 5.0은 스위치 I 1.0 또는 I 1.1이 살아날 때에만, 살게된다. 하드와이어 결선의 시퀀스 제어 회로에서, 배타적-OR 기능은 NO 및 NC 특성을 가진 기계적인 연동 스위치 요소(mechanically linked switch element )로서만 실현 가능할 뿐이다.
그러나 PLC에서는 오직 한 가지의 접점만을 가진 스위치를 정상적으로 사용하는데, NC 또는 NO 어느 것이든지 사용할 수 있다. 그러므로 PLC에서는 연산수를 "1"이나 "0"으로 검색할 수 있다.

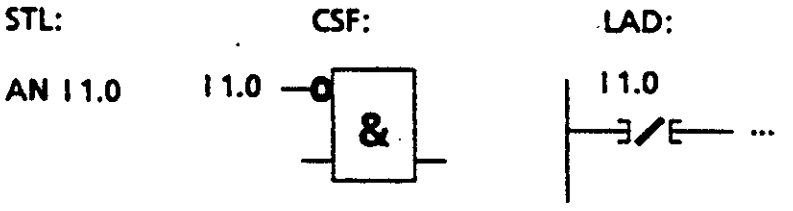
"1"의 검색

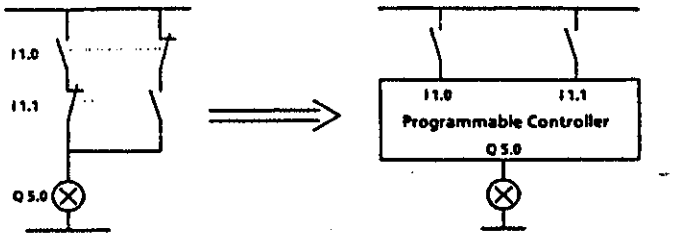
NO 접점이 닫히거나(동작하였을때), 또는 NC 접점이 닫히는(동작하지 않았을때) 경우에 입력은 "1"이 된다. "1"의 상태는 다음과 같이 3가지 방법으로 표현될 수 있다 :




"0"의 검색

NO접점이 열리거나(동작하지 않았을때) 또는 NC 접점이 열리는(동작하였을때) 경우에 입력은 "0"이 된다. "0"의 상태는 다음과 같이 3가지 방법으로 프로그램될 수 있다 :



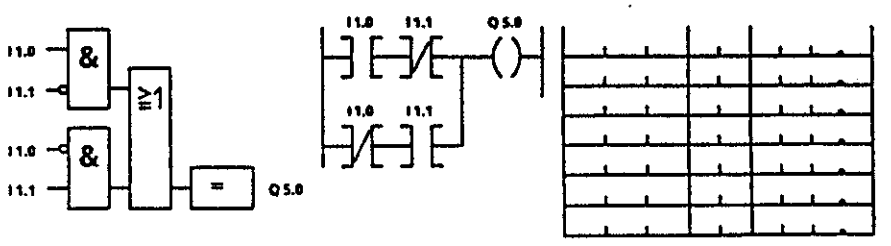




그림 4.16

실습 문제

* STL 작성 용지에 배타적-OR 연산을 추가 작성하라.

4-35 쪽의 그림은, 센서 S1은 동작하고 센서 S2가 동작하지 않을때 접점 K1에 전류가 흐르게 하고자 한다.
NO 접점이나 NC 접점은 센서 S1과 S2로 사용된다.

**실습 문제**

* 센서 S1과 S2에 대하여 서로 다른 스위치 기능(NC, NO)에도 불구하고 아래의 조건을 만족하도록 "1" 또는 "0" 의 신호 상태를 검색하는 프로그램을 3가지 표현으로 완성하라 :

  **S1이 동작하고 S2가 동작하지 않으면 K1 = ON 이다.**

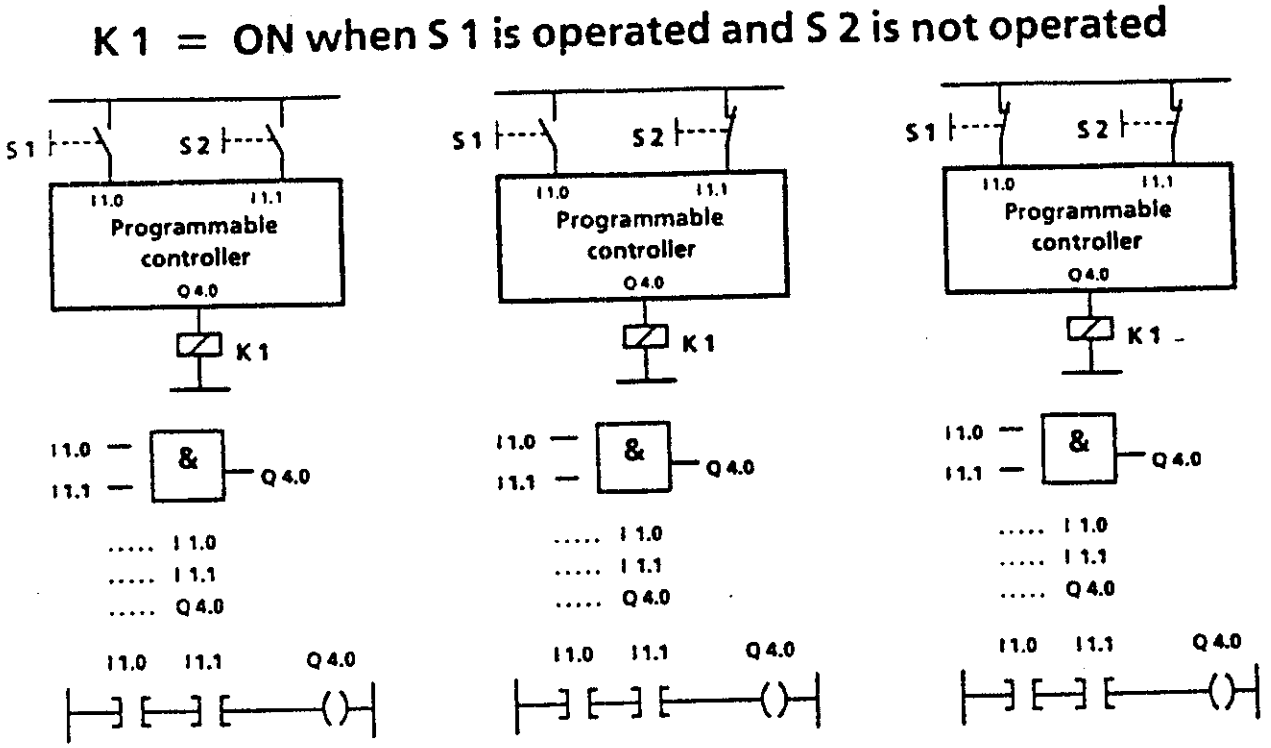



그림 4.17

다음의 실습 문제에서는 도식적 표현 방법인 CSF와 LAD의 편집 절차를 예시하고자 한다.

**생략시(default)의 LAD 표현**

* 프로그램 작성기에서 PRESETTING(생략시, default) 형식을 호출하고, LAD를 선택한 다음 새로운 설정 상태를 확인하라.

**블록의 입력**

* 입력 기능(F1/F1)을 선택하고 다음의 명령문을 완성하라 :
  INPUT DEVICE : ***FD*** BLOCK : ***PB2***

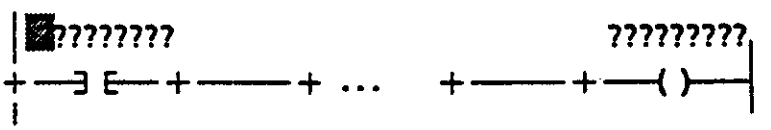
**참 고**

LAD나 CSF 형식을 입력할 때 적절한 도식 기호는 메뉴에서 F1-F8 키를 이용해 선택할 수 있다. PG 685에서 이러한 기능 키나 전용의 키를 사용해 편집할 수 있다.
PG 750은 오직 편집용 기능 키만이 이용 가능하다.

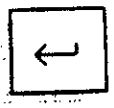
* PB 2의 세그먼트 1을 입력하시오.

**신호 상태 "1"의 검색**

* 이렇게 하기 위해 "Scan for 1" 키를 누른다.
  다음과 같은 화면이 나타난다 :




**CR 키**



* 물음표가 있는 자리에 연산수 I 0.0과 Q 4.0으로 바꾸시오.
  (CR 키를 각각 눌러 입력을 마친다).

**신호 상태 "0"의 검색**

* 첫번째 입력 검색 다음의 (+) 기호 위에 커서를 놓는다.
  "Scan for 0" 키를 누르고 I 0.1를 입력한다.

**세그먼트 종료**

* 'Segment end' 키를 눌러 세그먼트 1의 입력을 마친다.

* 그림과 같이 PB 2의 세그먼트 2를 입력한다.
  화면 왼쪽에서 "Scan for 0" 키를 눌러 입력 I 0.3의 병렬 가지(지로)(OR 연산)를 수직으로 벌려 놓는다.

**병렬 가지를 닫음**

* "Close parallel branch" 키를 입력 I 0.3을 붙이기 전에 누르시오!

**참 고**

병렬 가지 사이에 충분한 공간이 없을 때, 프로그램 작성기에 입력이 들어가지 않는다. 병렬 가지 사이에 너무 많은 공간이 존재할 경우, 프로그램 작성기는 병렬 가지를 바른 위치로 옮겨 놓는다.

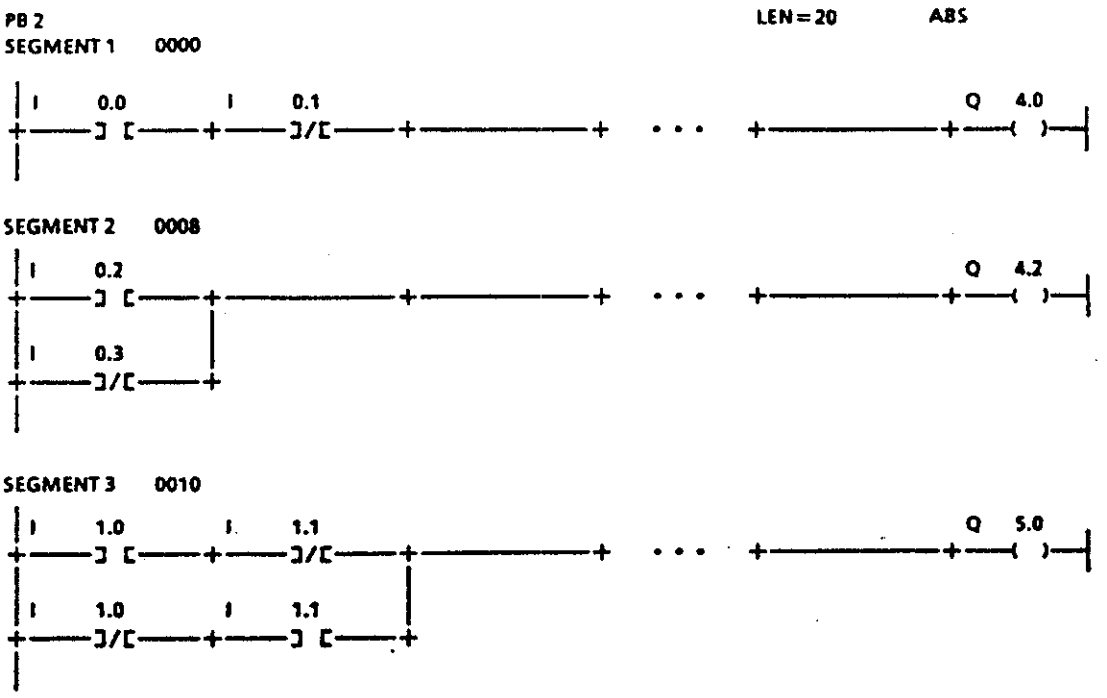



그림 4.18

* 세그먼트 3(배타적-OR)를 입력하고 Enter 키를 누르시오.

| 주 의 ! | CSF나 LAD 표현 방법에서는 INPUT(입력) 모드에서 제어 명령문 "블록의 끝"(BE)를 입력할 수 없다. 이 명령문은 프로그램 작성기에 의해 자동으로 입력된다. |
|---|---|

**오류 수정에 대한 참고**

각각의 연산수에 이름을 붙이면서 저지른 오류는, STL에서 처럼 간단하게 다시 겹쳐쓸 수 있다.
검색용 기호가 잘못된 경우 커서를 잘못된 기호 앞에 (+) 위에 놓고 올바른 기호 키를 눌러 겹쳐쓰도록 한다.

**문자의 삭제**

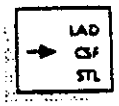
검색용 기호를 삭제하기 위해서 커서를 기호 위에 위치시키고 "Delete character" 키를 눌러 지운다.

앞의 실습 문제에서 입력된 PB 2 블록을 이제 도식적인 방법으로 수정하고자 한다. 두 표현 방법에서는 프로그램 작성기의 같은 키를 사용해 지우거나, 추가, 수정의 기능을 시작할 수 있으나, 다음의 실습 문제에서 수정은 CSF의 표현에만 국한하였다.

**세그먼트 선택**

OUTPUT(출력) 모드에서 SEARCH란에 세그먼트 숫자를 입력하여 바로 선택할 수 있다.
또한 STATUS 기능에 의해서도 세그먼트를 선택할 수 있다. "DELETE?" 물음에 대하여 아래와 같이 조작하여 응답하라.

**CSF로 블록 디스플레이**




* 출력 기능을 호출하고(F2/F2), 다음의 명령문을 완성시킨다.
  OUTPUT DEVICE: ***FD*** BLOCK: ***PB2*** SEARCH: ***2*** PTR.:

  F7 키를 사용하여 CSF 모드를 선택하라.

**세그먼트 삭제**

* 원래의 PB 2의 세그먼트 2를 삭제하라.
  "Delete segment" 키에 의해 세그먼트를 지울 수 있다.

  Enter 키를 누른다.
  화면에는 PB 2의 세그먼트 3(배타적-OR)를 새로운 세그먼트 2로 보여준다.

**세그먼트 삽입**

* 세그먼트 2 앞에 다른 세그먼트를 끼워넣을 수 있다.
  그렇게 하기 위해 세그먼트 2를 선택하고, "Insert segment" 키를 눌러 INSERT 모드를 입력하고, 그림처럼 새로운 세그먼트 2를 입력한다.

INSERT 모드는 화면의 오른쪽 윗편에 표시된다.

새로운 세그먼트 2를 회로를 적절히 구성하여 입력하라. 즉, 이번 실습 문제에서는 **AND** 연산으로 시작하여, 아래의 오른쪽으로부터 윗쪽의 왼쪽으로가는 회로를 구성하라.

**AND 기호**


* 첫째 "AND" 키를 누르고, 다음에 "OR" 키를 눌러 AND 기호 윗쪽의 입력에 OR 연산을 추가하라.

* OR 연산자의 입력에 연산수를 붙이라. 각각의 연산수의 입력을 마친후 CR 키를 누르시오.

**OR 기호**


* AND의 아래쪽 입력에 "OR"키를 다시 한번 눌러 2번째 OR 연산을 추가하라.

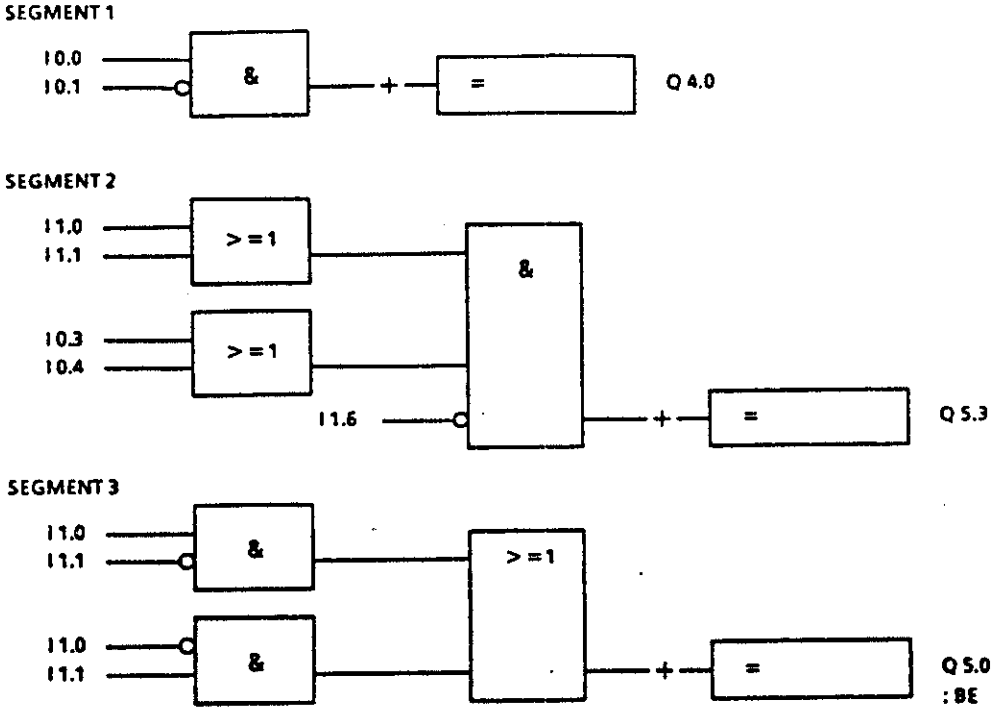
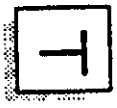
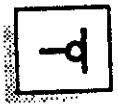



그림 4.19

**신호 상태 1 의 검색**



**신호 상태 0 의 검색**



* 다음과 같이 AND 기호에 커서를 옮겨서 입력을 추가하라 :

그리고 "Scan for 1" 키를 누르시오.

* 그림과 같이 입력에 차례로 연산수를 붙이되, 하나의 연산수를 입력시킬 때마다 CR 키를 눌러서 입력을 종료하라. CR 키 대신에 "Scan for 0" 키를 눌러 입력의 부호를 반전시킬 수 있다. 또한 "Scan for 1" 키로 반전된 입력을 바꿀 수 있다.

* 마지막으로 회로의 출력에 연산수를 입력하고, Enter 키를 누르시오.

이제 다시 출력 기능으로 되돌아 간다.
("Segment end" 키를 이용해 세그먼트를 더 입력할 수 있다).

실습 문제(계속)

* PB 2 의 세그먼트 1을 선택하고 수정하시오.

신호 상태 1의 검색

* 첫째로 커서를 연산자 I 위에 위치시키고 "Scan for 1" 키를 눌러 입력 I 0.1의 검색 동작을 "0"에서 "1"로 바꾸시오.

수 정


CORRECTION(수정) 모드에서만 세그먼트는 수정될 수 있다. Corr 키를 이용해 수정 모드를 호출하라. 수정 모드는 화면의 위쪽에 나타난다.

* 이제 그림 4.20 처럼 AND 연산 다음에 OR 연산을 삽입하라.

수평 확대

이렇게 하기 위해, 커서를 Q 4.0 앞의 (+) 위에 놓고 "Expand Horizontal" 키를 누르시오.

* OR 연산을 삽입하고 OR 연산 기호의 두번째 입력에 연산수를 붙이시오.

* Enter 키를 눌러 수정을 종료하라.

그러면, OUTPUT(출력) 기능으로 되돌아 간다. 이제, 수정된 블록이 프로그램 작성기의 기억장치에 저장되어 있다! Enter 키를 두 번 누르면 PB 2는 하드 디스크에 겹쳐 수록된다. 화면에는 FUNCTION SELECTION(기능 선택) 메뉴가 나타난다.

오류 수정에 대한 참고

잘못된 기호가 입력된 경우 이위에 커서를 놓고 올바른 기호 키를 눌러 수정할 수 있다.

검색의 결과는 병렬 출력에 추가로 지정할 수 있는데, 출력(Q 4.0)의 아래에 커서를 옮기고, 두번째의 연산수를 붙이고 CR 키를 누르면 된다.

커서를 지울 부분(연산자, 연산수)의 오른쪽에 놓고 "Delete character" 키를 눌러 세그먼트 부분을 지울 수 있다. 커서의 왼쪽에 있는 모든 것이 지워진다. 그러나 병렬 출력은 예외이다. 병렬 출력은 출력의 왼편에 있는 (+) 위치에 커서를 놓고 "Delete character"키를 눌러 지울 수 있다.

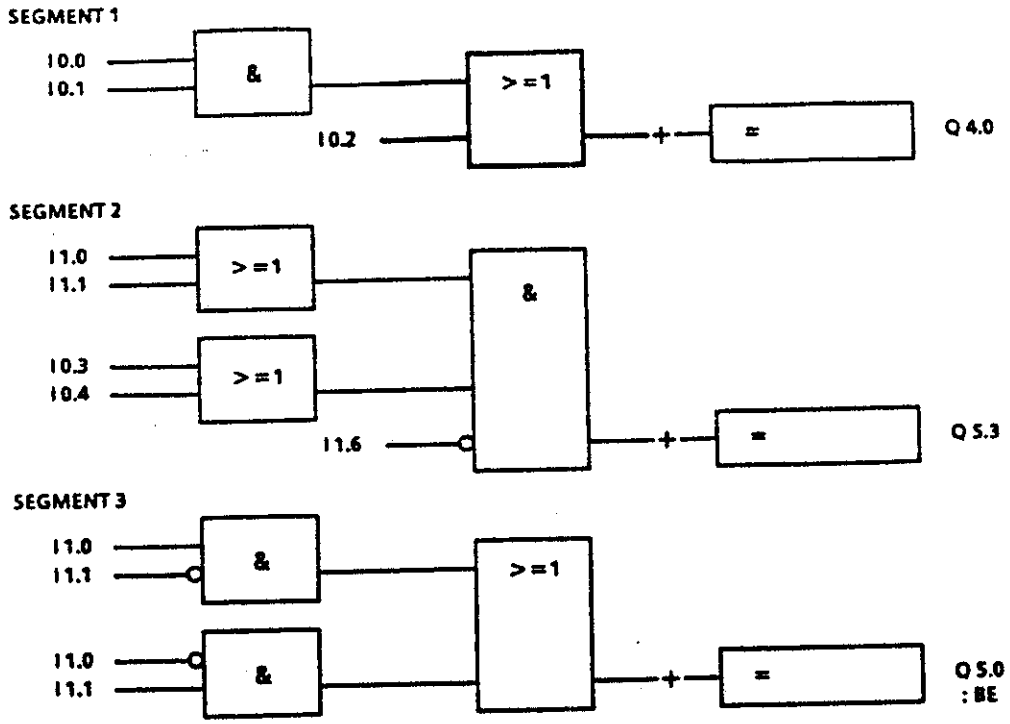



그림 4.20

INPUT(입력) 모드에서 방금 입력한 세그먼트를 지우려면, Break 키를 한 번 누르시오. 화면은 비워지고 세그먼트를 다시 구성하여 입력할 수 있다.

**주 의 !**

모든 표현 방법에서, 새로운 세그먼트를 화면에 표시된 세그먼트의 앞쪽에 삽입할 수 있다. 마지막 세그먼트 이후에 다른 세그먼트를 추가하려면, "Insert segment" 키 대신에 "Segment end" 키를 누르시오.

**삽 입**

각각의 개별 세그먼트는, 프로그램 전송 목적 장소(PLC, EPROM, 하드 디스크, 플로피 디스크)와 표현 방법(STL, CSF, LAD)의 선택 등과는 관계없이, 프로그램 작성기의 기억 장치(programmer memory)에 기록되며, 동시에 현재 처리되고 있는 세그먼트가 화면에 표시된다. 입력 절차가 끝나면, BE(Block End) 명령문 이후에, 방금 만들어진 블록의 사본이 지정된 목적 장소로 전송된다.

**디스플레이**

OUTPUT(출력) 기능이 선택되면, 프로그램 블록은 특정한 원천(PLC, EPROM, 하드 디스크, 플로피 디스크)으로부터 호출되고 프로그램 작성기의 기억 장치에 복사된다. 화면에는 OUTPUT 모드에서 특별히 지정하지 않으면, 항상 블록의 첫번째 회로(가지)가 디스플레이된다. 화면에 표시된 각 세그먼트의 사본은 프로그램 작성기의 중앙(주) 기억 장치에 저장된다.

**수 정**

프로그램 블록은 OUTPUT 모드에서 수정할 수 있다.
OUTPUT 모드에서 블록이 수정되면 어떤 일이 생기는가?

Corr 키를 눌러서 하나의 세그먼트 안에서 수정하는 것과 "Insert segment" 키 또는 "Delete segment" 키로써 시작되며 세그먼트 전체를 포함하여 수정하는 사이에는, 구별이 있어야 한다.
수정용 키를 누르면, 프로그램 작성기의 기억 장치(programmer memory)[프로그램 블록 저장용 RAM]는 프로그램 작성기의 중앙(또는 주) 기억 장치와 격리된다. 수정, 삭제 또는 삽입은 이제 프로그램 작성기의 중앙(또는 주) 기억 장치 안에서 안전하게 수행되어, 세그먼트 안의[일부분] 또는 하나의 세그먼트 전부가 삽입, 삭제 또는 수정될 수 있다.
Enter 키를 눌러서 수정이 끝나면 프로그램 작성기의 기억 장치[프로그램 블록 저장용 RAM]와 다시 연결된다. 수정된 세그먼트가 프로그램 작성기의 기억 장치[프로그램 블록 저장용 RAM]로 전송되고 관계가 있는 프로그램 블록으로 통합된다.
이제 다른 세그먼트를 선택하여 수정한다. 모든 세그먼트가 처리되면, 입력 키를 다시 한번 눌러서 블록을 원래의 기억 장치로 전송시킬 수 있다.

**참 고**

입력 기능 모드에서는 항상 새로운 블록이 만들어지므로 이미 존재하고 있는 블록을 수정할 수 없다.

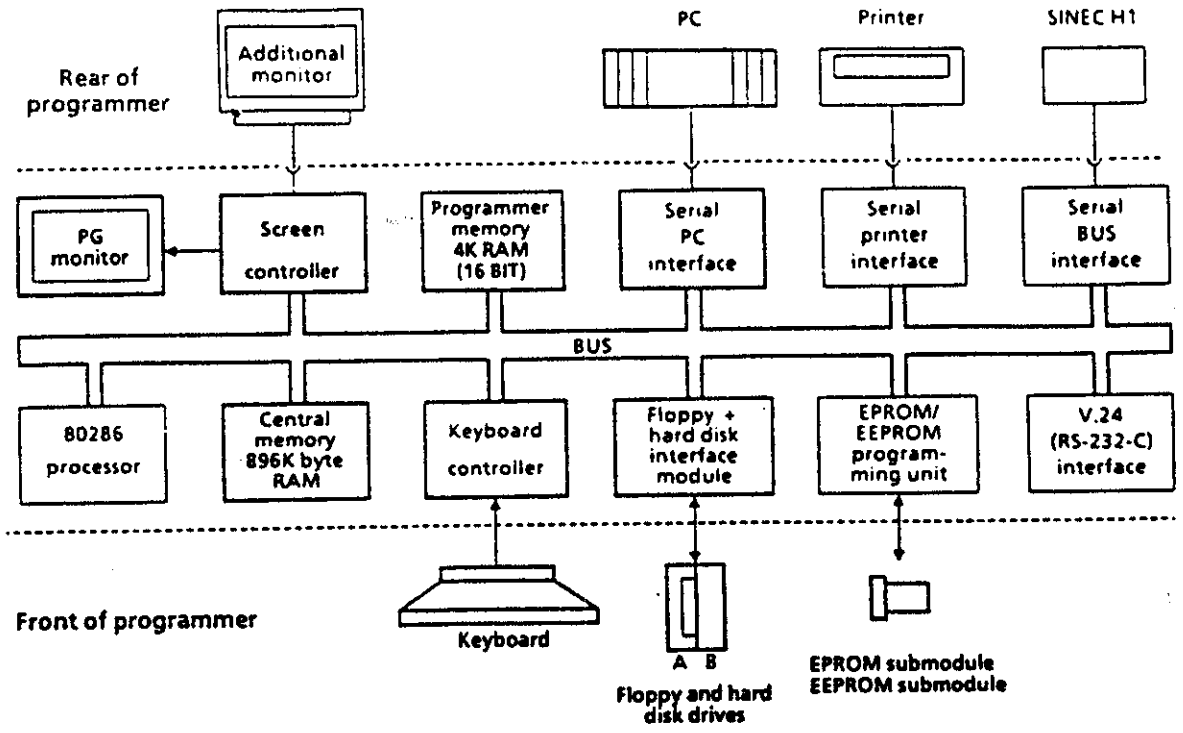



그림 4.21

전 송

전송 명령은 다음과 같이
- 하나의 블록
- 하나의 블록 영역
- 6개 까지의 임의의 블록
- 원천 기억 장치에 저장된 모든 블록
- 하나의 프로그램 파일 전부

등을 하드 디스크, 플로피 디스크, 보조 기억 모듈, PLC 기억 장치 사이에 전송된다. 블록은 개별적으로 하나씩, 원천 기억 장치로부터 프로그램 작성기의 기억 장치[블록 저장용]로 전송되고, 그런 다음에는 프로그램 작성기의 기억 장치[블록 저장용]로부터 목적하는 기억 장치로 전송된다.

**주 의 !** 오직 하나의 블록만이 프로그램 작성기의 기억 장치[블록 저장용]로부터 또는 프로그램 작성기의 기억 장치로 전송될 수 있다.

자세한 내용은 14절에서 다루었다.