

TM240

Ladder Diagram (LD)



선행 및 필요 조건

교육 자료	TM210 – The Basics of Automation Studio TM223 – Automation Studio Diagnostics
소프트웨어	Automation Studio 4.2.5 또는 그 이상 버전
하드웨어	없음

목차

1	소개	4
1.1	학습 목표	4
2	래더 다이어그램	5
2.1	래더 다이어그램의 흥미로운 정보.....	5
2.2	특징과 옵션.....	5
2.3	래더 다이어그램 편집기	6
3	래더 다이어그램의 기본 요소.....	9
3.1	네트워크 Network	10
3.2	실행 순서	10
4	래더 다이어그램 기호.....	11
4.1	접점 Contact.....	11
4.2	코일 Coil.....	14
5	논리 프로그래밍	18
5.1	이진 논리	18
6	프로그램 흐름 제어.....	20
6.1	조건부 점프.....	20
6.2	리턴	20
7	평션, 평션 블록과 액션	21
7.1	평션 블록 사용하기	22
7.2	Compute와 Compare 블록.....	26
7.3	IEC 액션(action).....	27
8	예제	29
8.1	예제- 컨베이어 벨트	29
8.2	예제- 콘크리트 충전 시스템.....	30
9	요약	32
10	부록	33
10.1	편집기에서 활용하는 키보드 단축키.....	33

1 소개

래더 다이어그램(Ladder Diagram)은 원래 유선 릴레이 로직 프로그래밍을 대체하기 위해 개발된 이미지 기반 프로그래밍 언어입니다. 래더 다이어그램은 일반적으로 사용되며 IEC 표준¹에 포함되어 있습니다.

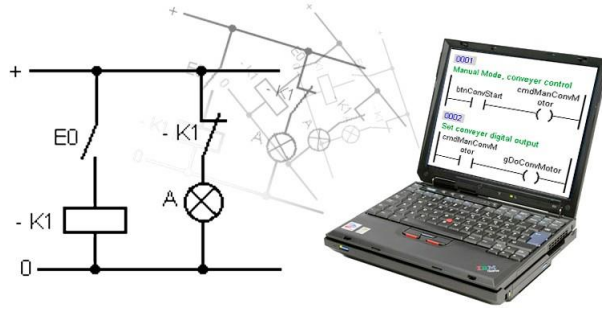


그림 1: 래더 다이어그램

다음 장에서, 래더 다이어그램을 활용한 프로그래밍 기능에 대한 개요를 제공합니다. 각 기능은 예를 사용하여 설명합니다.

1.1 학습 목표

교육 자료는 일반적인 어플리케이션 작업을 예제로 선택하여 래더 다이어그램(LD)으로 작업합니다.

- LD 프로그래밍 언어의 작동 원리를 학습합니다.
- LD 편집기 및 내장 진단 도구 사용법을 학습합니다.
- 액션, 평선, 평선 블록의 차이점과 사용방법을 학습합니다.
- 디지털 및 아날로그 신호를 LD 프로그램에 통합하는 방법과 논리 프로그래밍 기초를 학습합니다.
- LD 프로그램의 흐름 제어에 사용되는 요소를 배웁니다.

¹ IEC 61131-3 표준은 PLC(PROGRAMMABLE LOGIC CONTROLLER)에 사용되는 프로그래밍 언어를 위한 유일한 국제 표준입니다. 표준에는 INSTRUCTION LIST, STRUCTURED TEXT와 FUNCTION BLOCK DIAGRAM도 포함됩니다.

2 래더 다이어그램

2.1 래더 다이어그램의 흥미로운 정보

PLC (Programmable Logic Controller)의 최초 개념은 약 1968년 미국(USA)에서 개발되었습니다. PLC 개념은 배선 기반 시스템을 대신하여 프로그램을 할 수 있도록 마이크로프로세서에 기반을 두고 개발되었습니다.

PLC 자체는 래더 다이어그램을 기반으로 했으며, 릴레이 회로를 기반으로 한 논리 제어 시스템을 개략적으로 나타냅니다. 당시 개념은 비교적 적은 교육만으로 간단한 논리 제어 시스템을 신속하게 설정하고 프로그래밍하는 매우 빠른 방법이 되었습니다.

많은 제조업체들이 프로그래밍 시스템을 래더 다이어그램을 기반으로 합니다. 불행히도 공개 표준이 없다는 것은 각 공급 업체의 시스템이 약간 다르다는 것을 의미했습니다. 많은 제조업체가 기능을 향상시키기 위하여 특수 명령을 추가했습니다.

1990년대 초에는 말 그대로 수천 개의 PLC 제조업체가 자체 프로그래밍 인터페이스 및 명령 세트를 가지고 있었습니다. 다른 시스템에서 개발된 프로그램은 비슷하지만 그 구조와 명령은 종종 크게 달랐습니다.

1979년 국제 전기 표준 회의(IEC, International Electrotechnical Commission)에서 실무 그룹을 구성하여 PLC 공통 표준을 만들었습니다. 실무 그룹은 새로운 표준을 개발하기로 결정했습니다(IEC 61131이 되었습니다).

Part III "Programming Languages for PLCs(PLC 프로그래밍 언어)"는 1993년 출판되었고, PLC 소프트웨어를 위한 규정을 포함했습니다. Part III는 PLC 구성, 프로그래밍, 데이터 저장을 포함합니다.

2.2 특징과 옵션

래더 다이어그램은 그래픽 기반 프로그래밍 언어입니다. 회로도에 사용된 회로도 기호와 일치하는 전기 회로의 상징적 표현이 사용된다. 기호와 연결선은 필요한 논리를 프로그래밍하는 데 사용됩니다.

래더 다이어그램은 다음 특징들을 갖고 있습니다:

- 비주얼 프로그래밍
- 회로도가 90° 회전
- 간단하고 명확한 프로그래밍
- 자기 설명
- 진단하기 쉬움

래더 다이어그램 편집기를 사용하면 다음을 수행 할 수 있습니다:

- 디지털 입력 / 출력 및 내부 부울 변수 사용 (bool)
- 아날로그 입력 / 출력 사용
- 평선 블록, 평선 및 액션 사용
- 프로그램 흐름 제어 (점프, 프로그램 중단)
- 진단 도구 사용

2.3 래더 다이어그램 편집기

편집기의 툴바와 키보드로 래더 다이어그램 목록을 사용하여 모든 래더 다이어그램 기능을 구현할 수 있습니다.

툴바 Toolbar

기본 래더 다이어그램 기호는 래더 다이어그램 편집기에 툴바(Toolbar)를 사용하여 추가합니다. 툴바의 아이콘은 마우스 커서 위치에 따라 활성화되거나 비활성화됩니다.



그림 2: 래더 다이어그램 편집기의 툴바(Toolbar)

툴박스 Toolbox - 래더 다이어그램 목록 Ladder Diagram Catalog

래더 다이어그램 편집기를 열면, 툴박스는 래더 다이어그램 목록을 보여줍니다. 사용 가능한 모든 평선과 평선 블록을 보여줍니다. 필터를 설정하여 목록을 줄일 수 있습니다. 끌어서 놓기를 사용하여 래더 다이어그램 목록의 항목을 프로그램에 추가 할 수 있습니다.

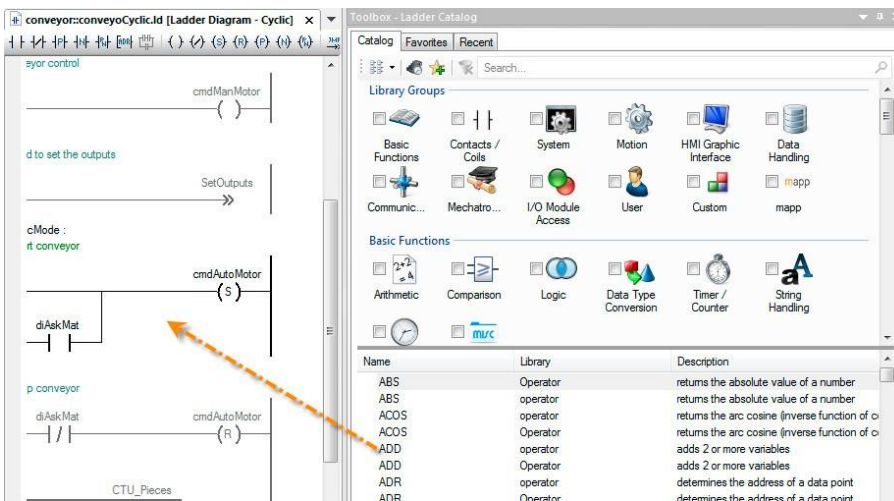


그림 3: 래더 다이어그램 목록을 사용하여 래더 다이어그램 기능 추가하기

보기 옵션

래더 다이어그램 아이콘과 편집기가 표시되는 방법을 사용자가 정의할 수 있습니다. 예를 들어, 래더 다이어그램 편집기에서 바로 가기 메뉴 또는 보기(**View**) 메뉴에서 데이터 유형, 주석, 연결 및 변수의 범위를 표시하거나 숨깁니다.

Tools/Options 메뉴를 사용하여 네트워크 및 래더 다이어그램 기호 크기를 구성 할 수도 있습니다.

네트워크의 표준 너비는 "최소 열 수" 설정으로 구성 할 수 있습니다. 네트워크가 동일한 값 이하의 열 수이면 모든 네트워크 출력이 서로 완벽하게 정렬됩니다.

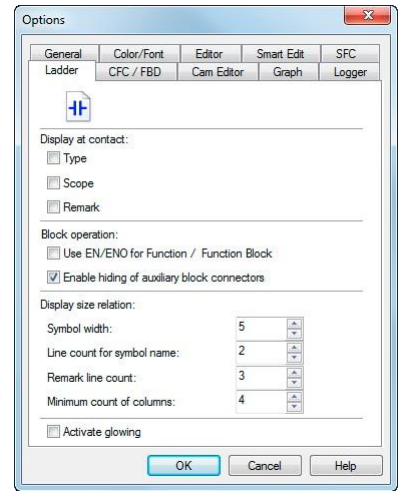


그림 4: 편집기 - 래더 다이어그램 설정

변수 할당

<스페이스 바>를 사용하거나 로지컬 뷰(Logical View)에서 끌어서 놓아 강조 표시된 접점에 변수를 지정할 수 있습니다.

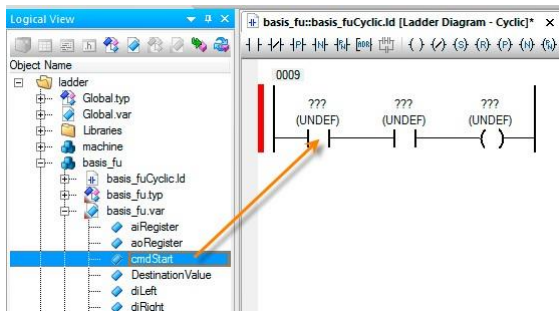


그림 5: 끌어서 놓으므로 변수를 접점에 할당하기

래더 다이어그램 편집기의 모든 기능은 마우스 또는 키보드로 작동 할 수 있습니다(10.1 "편집기에서 활용하는 키보드 단축키").

?

Programming \ Editors \ Graphic editors \ Ladder Diagram editor

- Toolbar
- Ladder Diagram Catalog

Project management \ The workspace \ AS Settings \ Ladder Diagram editor settings

진단

모니터 모드(Monitor mode)와 파워 플로워(Powerflow)는 래더 다이어그램에서 진단을 위해 사용됩니다. TRUE인 모든 논리 경로가 컬러로 표시됩니다. 또한 변수의 툴팁은 프로세스 값과 데이터 타입을 나타냅니다. Automation Studio 변수 모니터링 기능(watch)은 다양한 진단 기능을 완성합니다.

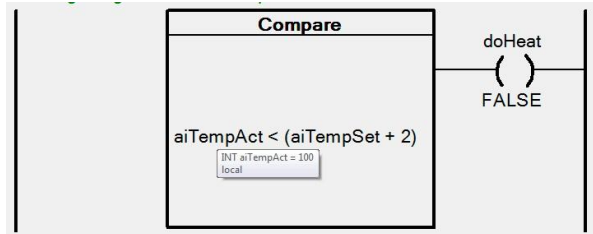
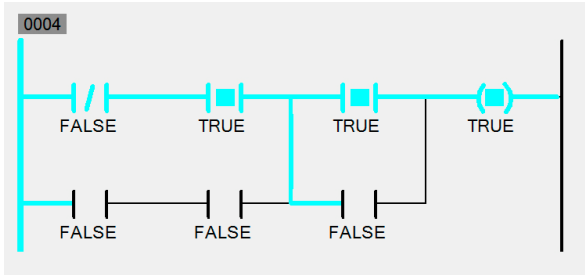


그림 7: 변수 툴팁 표시

그림 6: 파워 플로워(Powerflow)가 활성화된 래더 다이어그램



Diagnostics and Service \ Diagnostics Tool \ Monitors \ Programming languages in monitor mode \ Powerflow

Diagnostics and Service \ Diagnostic Tool \ Variable watch

3 래더 다이어그램의 기본 요소

다음 그림은 래더 다이어그램의 기본 요소를 보여줍니다. 왼쪽에는 영구 "전류 운반" 수직 파워 레일 (1). 오른쪽에는 일반적으로 열린 접점(open contact) (2)이 있으며 그 위에는 컨트롤러 접점 값을 저장하는 프로세스 변수 (3)입니다. 명령 라인 (4) 오른쪽으로 다른 접점이나 코일에 연결됩니다.

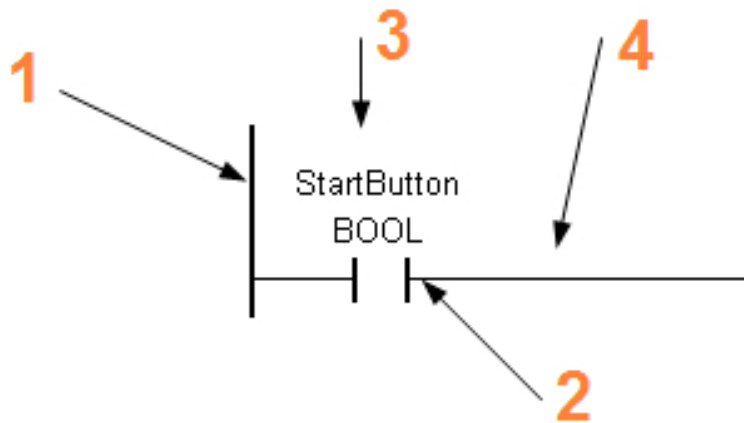


그림 8: 래더 다이어그램의 기본 요소

래더 다이어그램은 기본적으로 두 부분으로 구성됩니다. 왼쪽은 논리를 포함하고 오른쪽으로 결과를 출력합니다. 가장 오른쪽에 있는 요소는 코일(coil)입니다. 예를 들어 코일은 디지털 출력으로 사용될 수 있습니다.

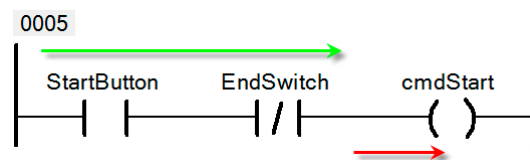


그림 9: 논리(초록색), 출력 또는 스위칭 명령(빨간색)

3.1 네트워크 Network

래더 다이어그램 프로그램은 더 작은 프로그램 단위로 나뉩니다. 이것은 네트워크(network)입니다.

네트워크는 병렬 또는 직렬로 연결될 수 있는 점접(contact)과 코일(coil)로 구성됩니다. 전원 공급 장치는 가장 왼쪽에 있고 참조 전위는 가장 오른쪽에 있습니다.

네트워크는 행 50 개와 열 50 개로 구성 될 수 있습니다. 적어도 코일 하나와 결과 값만 가장 오른쪽에 있더라도 구성이 완성됩니다.

래더 다이어그램은 여러 네트워크로 구성 될 수 있으며 각 네트워크는 오름차순으로 고유 네트워크 번호가 할당됩니다.

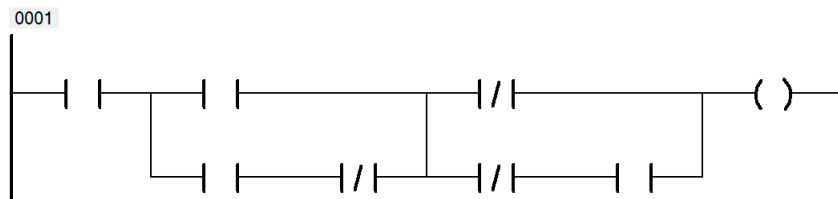


그림 10: 네트워크 구조



각 네트워크에 설명(comment)을 추가 할 수 있습니다. 각 네트워크에 편집기 툴바를 사용하거나 <D> 키를 눌러서 설명을 추가 할 수 있습니다.



[Programming \ Programs \ Ladder Diagram \(LD\) \ Network](#)

[Programming \ Editors \ Graphic editors \ Ladder Diagram editor \ Working with networks](#)

3.2 실행 순서

래더 다이어그램 프로그램에서

래더 다이어그램의 네트워크는 네트워크 번호(network number)에 따라 오름차순으로 실행됩니다. 실행 순서는 점프(jump)를 사용하여 특정 목적지로 향하도록 조작 할 수도 있습니다

네트워크에서

네트워크는 왼쪽에서 오른쪽으로 실행됩니다. 편집기에서 명시적 신호 피드백을 방지합니다. 반대 방향으로 신호 흐름은 불가능합니다.



[Programming \ Programs \ Ladder Diagram \(LD\) \ Execution order](#)

4 래더 다이어그램 기호

4.1 접점 Contact

다양한 기능을 가진 접점(Contact)을 사용할 수 있습니다. 래더 다이어그램 왼쪽에 추가하고 다른 접점을 연결할 수 있습니다. 그러나 맨 오른쪽은 코일(coil) 용으로 예약되어 있으므로 이 곳에 추가 할 수 없습니다. 접점은 데이터 타입이 BOOL이며 데이터 타입이 일치하는 디지털 입력/출력 또는 펄스 블록 매개 변수에 연결할 수 있습니다.

네트워크에서 논리적 접점 연결의 결과는 한 개 이상의 코일에 할당 될 수 있습니다. 각 접점은 변수 선언 창(variable declaration window)에 정의 된 변수 이름으로 표시됩니다.

접점 간 연결은 필요한 제어 로직에 따라 다릅니다. 코일에 에너지를 공급하기 위해 직렬, 병렬, 또는 이 둘의 조합으로 연결 할 수 있습니다.

접점(Contact) 유형	기호
Normally open contact (a 접점)	
Normally closed contact (b 접점)	
Positive edge	
Negative edge	
Both edges	

표 1: 접점 개요

Programming \ Programs \ Ladder Diagram (LD) \ Contacts and coils

4.1.1 a 접점(normally open contact)과 b 접점(normally closed contact)의 차이점

산업 환경에서 "normally open contact"과 "normally closed contact" 용어에 직면합니다 두 용어 모두 접점, 입력 및 출력 범주에 속합니다.

b 접점 (normally closed contact)은 작동되지 않는 한 전류를 전도합니다.

a 접점 (normally open contact)은 작동중인 경우에만 전류를 전도합니다.

b 접점(normally closed contact)을 선택하면 누군가가 초인종 버튼을 누를 때까지 초인종이 울립니다. 버튼을 누르면 접점이 열리고 전기 흐름이 중단됩니다. a 접점(normally open contact) 행동은 정반대입니다.

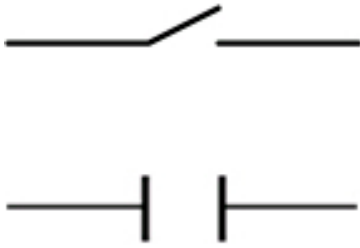


그림 11: Normally open contact (a 접점)

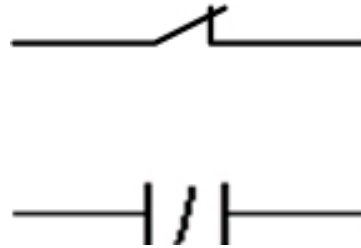


그림 12: Normally closed contact (b 접점)

4.1.2 a 접점 normally open contact

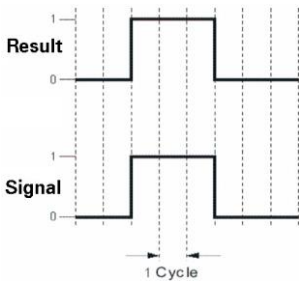


그림 13: 입력 신호와 결과



그림 14: a 접점(normally open contact)

접점이 작동되지 않는 한 전류가 흐르지 않고 로직 상태는 FALSE입니다. 작동되면 물리적 상태가 "ON"으로 바뀌고 결과는 TRUE가됩니다.

4.1.3 b 접점 normally closed contact

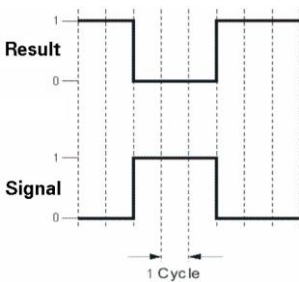


그림 15: 입력 신호와 결과



그림 16: b 접점(normally closed contact)

이 접점은 변수 상태를 반전시킵니다. 출력 설정을 위해 입력 신호가 필요하지 않을 때 사용됩니다. 입력이 TRUE로 설정된 경우 출력 상태는 FALSE로 설정됩니다.

4.1.4 엣지 접점 edge contact

프로그래밍에서 신호 레벨의 상승 및 하강 에지를 평가할 때 항상 도움이 됩니다.

포지티브 엣지

Positive edge

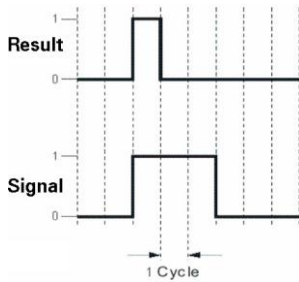


그림 17: 입력 신호와 결과



그림 18: 포지티브 엣지(Positive edge)

이 접점은 포지티브 엣지 신호를 감지하는 데 사용됩니다. 변수 값이 FALSE에서 TRUE로 변경 될 때, 즉 포지티브 엣지가 발생하면 이 접점은 한 주기 동안 TRUE를 반환합니다. 포지티브 엣지 수를 계산할 뿐만 아니라 조건을 설정 또는 재설정하는 데 사용됩니다.

네거티브 엣지

Negative edge

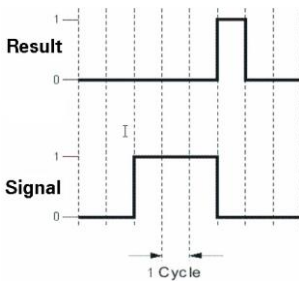


그림 19: 입력 신호와 결과

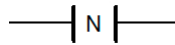


그림 20: 네거티브 엣지(Negative edge)

이 접점은 네거티브 엣지 신호를 감지하는 데 사용됩니다. 변수 값이 TRUE에서 FALSE로 전환되면 결과는 한 주기 동안 TRUE가 됩니다. 예를 들어 출력을 설정 또는 재설정하거나 네거티브 엣지 수를 계산하기 위해 사용할 수 있습니다.

Both edges

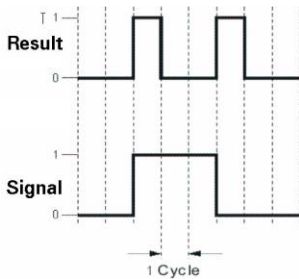


그림 21: 입력 신호와 결과

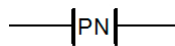


그림 22: 포지티브와 네거티브 엣지

접점은 디지털 신호의 포지티브 및 네거티브 엣지를 형성하는 데 사용할 수 있습니다.

동작은 포지티브와 네거티브 엣지의 병렬 연결에 해당합니다.

4.2 코일 Coil

코일(coil)은 래더 다이어그램의 기본 요소입니다. 항상 래더 다이어그램의 출력으로 오른쪽에 배치됩니다. 코일은 점점 오른쪽에 연결하거나 평선 블록 출력에 연결할 수 있습니다. 네트워크에는 코일이 하나 이상 있어야 합니다. 병렬로 배열 된 여러 코일을 사용할 수도 있습니다.

각 코일은 디지털 출력 또는 프로그램에서 다른 네트워크 입력으로 활용 할 수 있는 내부 변수로 사용될 수 있습니다. 프로그램이 실행되는 동안 모든 접점은 지속적으로 쿼리됩니다. 논리 경로가 발견되면 코일이 TRUE가 됩니다.

코일에는 부울 변수(bool)만 할당 할 수 있습니다.

코일(Coil) 유형	기호
코일(Coil)	—()—
Negated coil	—(/)—
Set coil	—(S)—
Reset coil	—(R)—
Positive transition coil	—(P)—
Negative transition coil	—(N)—
Both edges	—(PN)—

표 2: 코일(coil) 개요

?

Programming \ Programs \ Ladder Diagram (LD) \ Contacts and coils

4.2.1 코일 유형

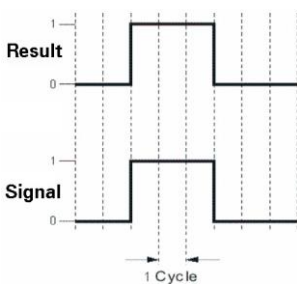


그림 23: 입력 신호와 결과

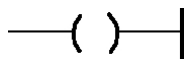


그림 24: 노멀 오픈 코일(Normally open coil)
신호가 TRUE이면 코일이 켜집니다.

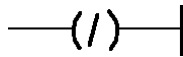
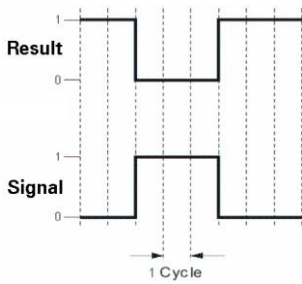


그림 26: 노멀 크로우즈 코일(Normally closed coil)
신호가 TRUE이면 코일이 꺼집니다. 다른 시간에는 켜져 있습니다.

그림 25: 입력 신호와 결과

4.2.2 셋과 리셋

셋 코일 Set coil

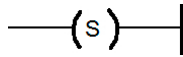
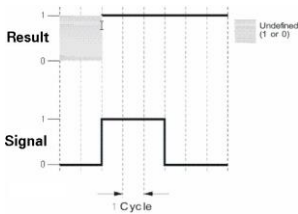


그림 28: 셋 코일(Set coil)
이 코일은 신호가 있을 때 변수를 TRUE로 설정합니다.
변수가 재설정 될 때까지 상태를 유지합니다. 이러한 이유로 코일은 조건부입니다.

그림 27: 입력 신호와 결과

리셋 코일 Reset coil

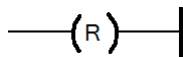
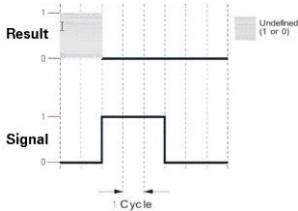


그림 30: 리셋 코일(Reset coil)
이 코일은 신호가 TRUE인 경우 변수를 FALSE로 설정합니다.

그림 29: 입력 신호와 결과

4.2.3 엣지 출력 Edge outputs

포지티브 전이 코일

Positive transition coil

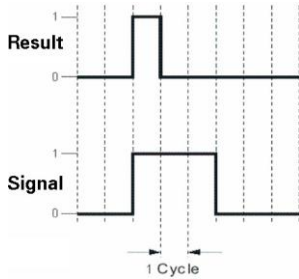


그림 31: 입력 신호와 결과

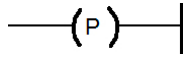


그림 32: 포지티브 전이 코일(Positive transition coil)

이 코일은 신호가 TRUE로 존재하는 경우 한 사이클 동안만 변수를 TRUE로 설정합니다. 동일한 신호 동안의 후속 사이클 출력은 FALSE로 유지됩니다.

네거티브 전이 코일

Negative transition coil

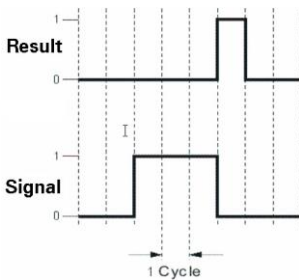


그림 33: 입력 신호와 결과

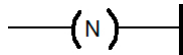


그림 34: 네거티브 전이 코일(Negative transition coil)

이 코일은 신호가 FALSE로 존재하는 경우 한 사이클 동안만 변수를 TRUE로 설정합니다. 동일한 신호 동안의 후속 사이클 출력은 FALSE로 유지됩니다.

포지티브와 네거티브 전이 코일 **Positive and negative transition coil**

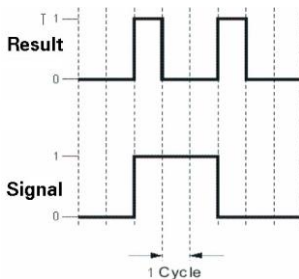


그림 35: 입력 신호와 결과

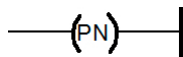


그림 36: 포지티브와 네거티브 엣지의 출력

이 코일은 포지티브와 네거티브 출력의 기능을 통합합니다.

예제: 첫 래더 다이어그램 생성

당신의 첫 래더 다이어그램 프로그램을 생성합니다. 버튼(button)을 누르면, 램프(lamp)가 버튼 누르기를 중단 할 때까지 켜져 있습니다.

변수 Variables	데이터 타입 Data types	설명
diSwitch	BOOL	조명을 키고 끄는데 사용되는 입력
doLight	BOOL	조명에 사용되는 출력

표 3: 입력과 출력 변수 개요

예제: 포지티브와 네거티브 엣지 사용

포지티브 엣지(Positive edge)에서 램프가 켜지고 네거티브 엣지(Negative edge)서 꺼 지도록 이전 예제를 수정하십시오.

예제: 키보드를 사용한 래더 프로그래밍

키보드만 사용하여 다음 래더 다이어그램 프로그램을 작성하십시오. 먼저 래더 다이어그램 기호를 삽입하고 연결선을 만들기 위한 키보드 단축키를 찾으십시오.

그 다음 두 신호를 사용하여 액추에이터를 전환해야 합니다. "bntSwitch1"이 있으면 출력은 "bntSwitch2"에 도달 할 때까지 그대로 유지됩니다.

변수 Variables	데이터 타입 Data types	설명
bntSwitch1	BOOL	조명을 키는데 사용되는 입력
bntSwitch2	BOOL	조명을 끄는데 사용되는 입력
doLight	BOOL	조명에 사용되는 액추에이터

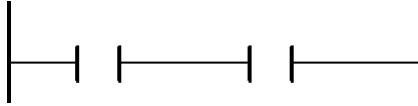
표 4: 입력과 출력 변수 개요

5 논리 프로그래밍

래더 다이어그램은 단순한 스위칭 작업에만 사용되는 것이 아닙니다. 이진 논리를 구현하는 데도 사용할 수 있습니다.

5.1 이진 논리

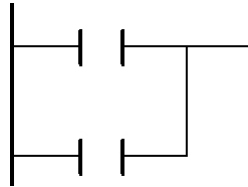
AND 연결



접점이 두 개 이상 직렬로 전환되면 결과는 논리 AND 연결입니다.
모든 조건이 충족되면 출력이 TRUE로 설정됩니다.

그림 37: AND 연결

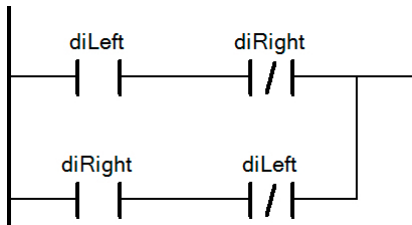
OR 연결



병렬 블록은 OR 연결과 같습니다.
병렬 분기 중 하나 이상이 TRUE이면 출력도 TRUE입니다.

그림 38: OR 연결

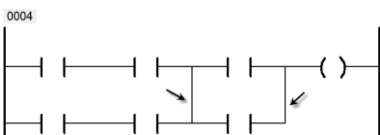
배타적 논리합 연결 Exclusive OR



배타적 논리합 연결은 논리 AND와 OR 연결의 조합입니다.
입력 중 하나가 TRUE이면 출력은 TRUE입니다. 두 입력이 모두 TRUE이면 출력은 FALSE로 유지됩니다.

그림 39: XOR 연결

논리 경로 분기 및 병합



논리 경로는 분기로 수정할 수 있습니다. 병렬 경로를 사용할 수 있습니다. 논리 경로를 닫으려면 분기를 다시 병합해야 합니다. 병합은 다음 병렬 경로에서 분기 될 수도 있습니다 (이미지 참조).

그림 40: 분기 및 병합



편집기 툴바(Toolbar)에서 화살표 아이콘을 사용하거나 <ALT> + <↓>를 눌러 분기를 만들 수 있습니다.



Programming \ Programs \ Ladder Diagram (LD) \ Simple logic structures

Programming \ Editors \ Graphic editors \ Ladder Diagram editor \ Connection lines

예제: 플립 플롭(flip-flop) 프로그래밍

다음 예는 논리 프로그래밍에서 사용 가능한 몇 가지 가능성을 결합한 것입니다. 래더 다이어그램의 프로그램 실행 순서는 응용 프로그램이 올바르게 작동하는 데 중요합니다. 몇 가지 솔루션이 가능합니다.

원하는 프로그램 동작:

- 사용자가 입력을 켜면 출력을 켜야 합니다.
- 입력이 다시 꺼지면 출력은 동일한 상태를 유지해야 합니다.
- 다음에 입력을 켤 때는 출력을 꺼야 합니다.

변수 이름(Variables name)	데이터 타입(Data types)	설명
diSwitch	BOOL	각 포지티브 엣지에서 출력 상태가 변경되는 입력
doFlipFlop	BOOL	입력에 의해 제어되는 출력

표 5: 입력과 출력 변수 개요

6 프로그램 흐름 제어

6.1 조건부 점프 Conditional jump

네트워크 번호 외에도 각 네트워크에는 고유 한 점프 레이블이 제공 될 수 있습니다. 조건부 점프를 점프 레이블이 있는 어느 네트워크 프로그램 순서 에든 배치 할 수 있습니다.

점프 조건이 TRUE이면 점프가 실행됩니다.

점프는 프로그램에서 네트워크를 건너 뛰는 데 사용됩니다. 프로그램 흐름을 보다 효과적으로 제어 할 수 있습니다.

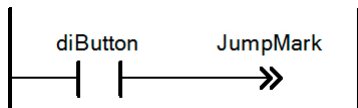


그림 41: "JumpMark" 네트워크로 조건부 점프

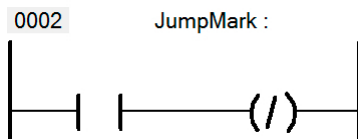




그림 42: 기호 이름이 "JumpMark"인 네트워크

 점프 레이블이 없으면 프로그램을 컴파일시 메시지 창에 오류가 출력됩니다.
Error 1490: Label 'JumpMark' not defined.

 [Programming \ Programs \ Ladder Diagram \(LD\) \ Jump / Jump return](#)

6.2 리턴 Retrun

리턴 **Retrun** 명령은 특정 지점에서 래더 다이어그램을 중단하는 데 사용됩니다. 후속 네트워크는 더 이상 실행되지 않습니다. 다음 프로그램 사이클에서 프로그램은 첫 번째 네트워크에서 활성화 된 경우 리턴 포인트 또는 비활성 된 경우 프로그램 끝까지 실행됩니다.

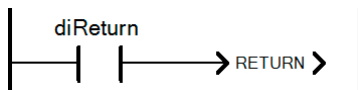



그림 43: 리턴으로 프로그램 중단

 [Programming \ Programs \ Ladder Diagram \(LD\) \ Jump / Jump return](#)

7 평선, 평선 블록과 액션

평선, 평선 블록 및 액션(action)을 사용하면 프로그래밍 언어 기능이 확장됩니다. 프로그램에서 평선과 평선 블록은 여러 번 사용됩니다.

평선(Functions)

... 여러 매개 변수와 리턴 값이 하나만 있습니다. 평선이 호출 된 직후 결과 반환됩니다.

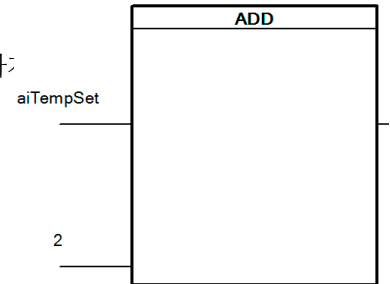


그림 44: 평선 예

평선 블록(Function blocks)

... 보통 여러 개 리턴 값과 인스턴스 변수 한 개가 있습니다. 인스턴스 변수는 평선 블록이 긴 기간, 즉 여러 주기에 걸쳐 작업이 분산 될 수 있기 때문에 필요합니다. 또한 동일한 평선 블록은 다른 입력 매개 변수가 지정 될 때 다른 결과를 반환합니다. 인스턴스 변수는 평선 블록의 "로컬 메모리"를 나타냅니다.

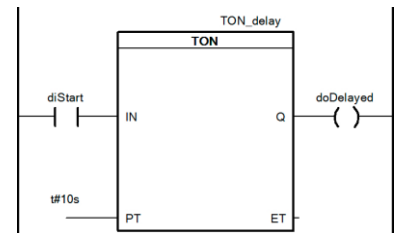


그림 45: 평선 블록 예

액션(Actions)

... 호출 될 수 있는 서브 루틴 또는 이진 활동입니다. 한정자는 호출의 성격, 타이밍 및 지속 시간을 지정합니다. (7.3 "IEC 액션 사용")

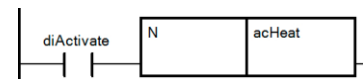


그림 46: "N" 한정자를 지닌 액션(action) 호출



Programming \ Programs \ Ladder Diagram (LD) \ Blocks

7.1 평선 블록 사용하기

라이브러리에서 관리되는 평선과 평선 블록 평선 블록이 삽입되면, 인스턴스 변수가 선언되어야 한다.

편집기의 툴바를 통한 삽입

평선 블록은 편집기 툴바를 통해 프로그램으로 추가 할 수 있다.



그림 47: "Insert function / function block" 아이콘



Automation Studio에서 사용 된 라이브러리만 프로젝트의 일부입니다. 다른 라이브러리의 평선 또는 평선 블록을 사용해야 하는 경우 선택 대화 상자에서 "Show external libraries" 옵션을 활성화합니다.

래더 다이어그램 목록에서 넣기

래더에서 사용 가능한 평선 블록 목록은 전체 텍스트 검색 또는 필터 설정을 사용하여 좁힐 수 있습니다. 끌어서 놓기를 사용하여 평선 블록을 래더 다이어그램에 추가 할 수 있습니다. 그 다음 인스턴스 변수를 선언합니다.

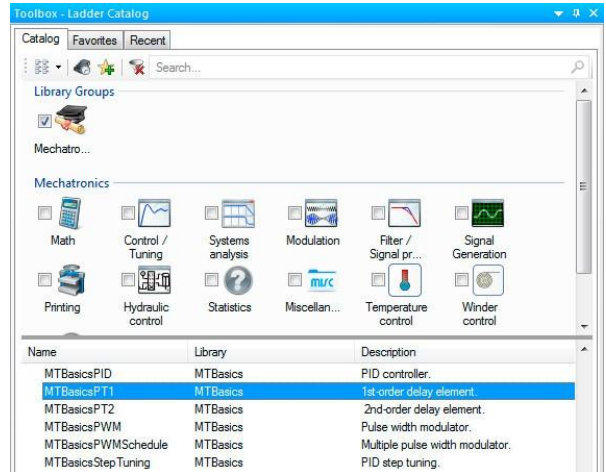


그림 48: 래더 다이어그램 목록(Ladder Diagram Catalog) - 필터 목록 Mechatronic functions



Programming \ Programs \ Ladder Diagram (LD) \ Blocks
 Programming \ Editors \ Graphic editors \ Ladder Diagram editor \ Functions

- Functions
- Ladder Diagram Catalog

Programming \ Editors \ General operations \ Dialog boxes for input support

7.1.1 아날로그 값 사용

데이터 유형 **BOOL**이 아닌 값 (예: 아날로그 값)은 래더 다이어그램 기호가 없습니다. 이 값은 평선 또는 평선 블록에 직접 연결됩니다. 툴바(Toolbar), 스페이스 바를 사용하거나 접점을 두 번 클릭하여 입력 할 수 있습니다.



그림 49: 툴바(Toolbar)를 사용하여 아날로그 값 연결

아날로그 비트 주소 Bit addressing of analog values

아날로그 값이 점점 및 코일과 연관 되어야하는 경우 아날로그의 개별 비트를 연결할 수 있습니다. 아날로그 변수 이름 뒤에 마침표 "."를 배치합니다. 비트(bit)는 0부터 시작하여 오름차순으로 번호가 지정됩니다. 예를 들어 아날로그 두 번째 비트는 aiTemperature.1를 사용하여 접근합니다.

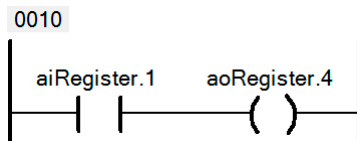


그림 50: "aiRegister"의 비트 2를 "aoRegister"의 비트 5에 할당

? [Programming \ Programs \ Ladder Diagram \(LD\) \ Analog value](#)
[Programming \ Editors \ Graphic editors \ Ladder Diagram editor \ Analog values](#)
[Programming \ Variables and data types \ Variables \ Bit addressing](#)

7.1.2 확장 가능한 평선

일부 평선은 사용자가 확장 할 수 있습니다. 예를 들어 평선 속성에 입력을 추가 할 수 있습니다. 래더 다이어그램에서 다음 평선을 확장 할 수 있습니다:

ADD, AND, SUB, DIV, EQ, GE, GT, LE, LT, MAX, MIN, MOVE, MUL, MUX, OR, XOR

상단에 나열된 평선 외에도 MOVE 기능을 확장 할 수 있습니다.

각 확장을 위해 입력 및 출력이 추가됩니다. 이미지에 표시된 대로 할당이 행 순서대로 실행됩니다.

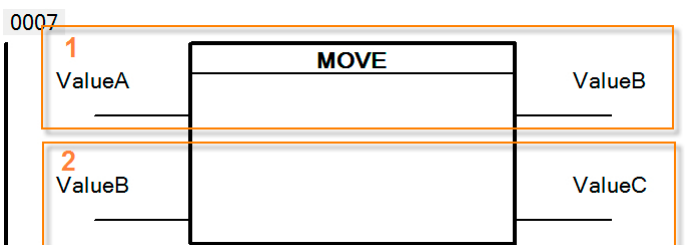


그림 51: 할당은 프로그램 끝에서 ValueC = ValueA와 함께 순서대로 실행됩니다.

? [Programming \ Programs \ Ladder Diagram \(LD\) \ Blocks](#)
[Programming \ Editors \ Graphic editors \ Ladder Diagram editor \ Functions](#)
[Programming \ Libraries \ IEC 61131-3 functions \ OPERATOR](#)

7.1.3 EN / ENO가 있는 블록

단순한 래더 다이어그램을 위해 비트(bit)를 사용하여 평선 블록을 활성화 또는 비활성화 할 수 있습니다. 이 옵션을 "EN / ENO" 라고 하며 각 평선 블록 속성에서 개별적으로 설정할 수 있습니다.

값이 TRUE 인 EN 신호는 평선 블록을 활성화합니다. EN 신호 값만 ENO 출력으로 전달됩니다. 이를 통해 평선 블록을 직렬로 연결하고 비트를 사용하여 활성화 또는 비활성화 할 수 있습니다.

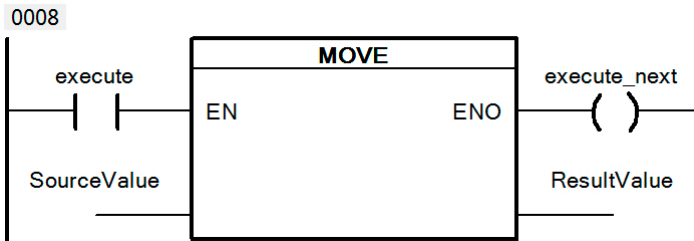



그림 52: "execute"이 TRUE인 경우만 MOVE가 실행됩니다.

평선 블록이 삽입되면 기본적으로 EN / ENO 신호 옵션이 켜집니다. 메뉴 항목 **Tools / Options** 을 사용하여 끌 수 있습니다.

 [Programming \ Programs \ Ladder Diagram \(LD\) \ Blocks with EN/ENO](#)
[Project management \ The workspace \ AS settings \ Ladder Diagram editor settings](#)

예제: 평선 블록 사용하기

이 연습에서는 여러 평선 블록을 호출합니다.

입력 상태는 여러 번 기록됩니다. 수집 된 데이터는 화면 작화 어플리케이션에 사용됩니다. 변수 감시 창 **variable watch window** 에서 화면과 연계된 변수를 조작 할 수 있습니다.

다음 프로그램 동작을 구현합니다:

- 입력이 활성화 된 후 3 초 후에 출력이 켜집니다.
- 세 번 입력 들어온 후 화면 장치에 주의(warning)가 나타난다.
- 화면 장치에서 주의를 확인하면 사라집니다.
- CPU를 다시 시작한 후 켜진 입력 횟수 카운트 값이 유지되어야 합니다.

변수 Variables	데이터 타입 Data types	설명
diSwitch	BOOL	모니터링 되는 입력
doMotor	BOOL	시간 지연 출력
visButtonReset	BOOL	화면 장치에서 확인 버튼

표 6: 입력과 출력 변수 개요

변수 Variables	데이터 타입 Data types	설명
visWarning	BOOL	화면 장치에 주의 표시

표 6: 입력과 출력 변수 개요

예제: 화면 장치를 사용하여 매개 변수 구성

다음 작업을 포함하도록 이전 작업을 확장하십시오.

- 주의가 표시 될 때의 한계 값을 구성합니다.
- 기본 설정된 한계 값은 3 입니다 (**variable watch**).
- 수정 된 한계 값은 CPU가 다시 시작된 후에도 유지되어야 합니다 (**RETAIN**).
- 얼마나 자주 켜 졌는지 출력을 표시합니다.

변수 Variables	데이터 타입 Data types	설명
visParamLimit	UINT	주의 표시에 사용 된 한계 값의 입력 필드
visActivationCount	UINT	출력이 활성화 된 빈도 출력 필드

표 7: 추가할 변수 개요

7.1.4 사용자 평선 생성

Automation Studio에서 자주 사용되는 프로그램 섹션은 사용자 평선 및 평선 블록에 저장할 수 있습니다.

이 평선 및 평선 블록은 래더 다이어그램 프로그램에 직접 할당 할 수 있습니다. 그 다음 평선을 프로그램에서 여러 번 사용할 수 있습니다. 또한 사용자 라이브러리를 생성 할 수도 있습니다. 필요할 때마다 프로젝트 전체에서 사용할 수 있습니다. 두 경우 모두 구현과 호출을 수행하는 프로그램 언어가 다를 수 있습니다.

Object Name	Description
ladder	Standardprojekt
Global.typ	Global data types
Global.var	Global variables
Libraries	Global libraries
Runtime	Internal support library
Operator	Built-in IEC 61131-3 standard fun
AslecCon	This library contains function inte
astime	The AsTime Library supports DA'
standard	This library contains standard fun
userlib	global user library
machine	controlling the machine
machineCyclic.lc	Cyclic code
machine.var	Local variables
local_block.lc	Implementation of local function t
machine.fun	Declaration of local function bloc

그림 53: 프로젝트에서 사용자 라이브러리에 구현된 사용자 평선 블록을 프로그램에 사용



Programming \ Libraries \ Example: Creating a user library

7.2 계산과 비교 블록 Compute and Compare block

논리는 점점 contact과 OPERATOR 라이브러리에서 사용 가능한 평선으로 프로그래밍 할 수 있습니다. 그러나 보다 복잡한 계산 및 비교에는 일반적으로 함수가 두 개 이상 필요합니다. 네트워크를 더 복잡하게 만듭니다.

계산 및 비교 블록을 사용하여 표준화 된 형식으로 식을 입력 할 수 있습니다.

더 큰 표현식을 처리하기 위해 Compute 및 Compare 블록은 다음과 같이 확장 할 수 있습니다(7.1.2 "확장 가능한 평선").



두 블록 모두 모든 지역 및 전역 변수와 상수에 접근 할 수 있습니다. 또한 표현식에는 모든 함수가 포함될 수 있습니다. 입력은 "Structured Text 형식"입니다.

7.2.1 계산 블록 Compute block

계산 블록 Compute block을 사용하여 식을 계산할 수 있습니다. 그 결과는 블록의 출력으로 전달됩니다. 함수 호출 뿐만 아니라 모든 변수와 상수를 사용할 수 있습니다.

0004

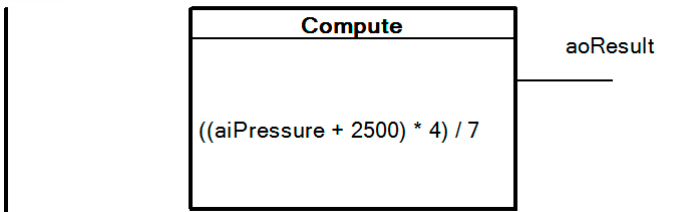


그림 54: 계산 블록 Compute block을 사용하여 표현식 계산



Programming \ Programs \ Ladder Diagram (LD) \ Compute

예제: 평균 값 계산

방 온도는 세 가지 다른 장소 (aiTemp1, aiTemp2, aiTemp3)에서 측정됩니다. 평균 온도 (aoTempAvg)를 결정해야 합니다

7.2.2 비교 블록 Compare block

비교 블록 Compare block을 사용하면 논리 비교 표현식을 사용할 수 있습니다. 표현식이 TRUE이면 다음 표현식이 FALSE가 될 때까지 비교 블록의 출력이 설정됩니다.

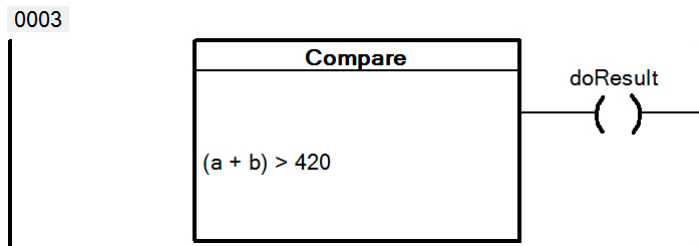


그림 55: 비교 블록 Compare block 사용



Programming \ Programs \ Ladder Diagram (LD) \ Compare

예제: 실내 온도 제어

실내 설정 온도와 실제 온도를 비교합니다. 실제 온도(aiTempAct)가 설정 온도(aiTempSet)보다 낮으면 히터(doHeat)가 작동해야 합니다.

목표 범위에서 히터가 지속적으로 켜지고 꺼지는 것을 방지하려면 실제 온도가 설정 온도보다 2°C 높아질 때까지 계속 가열합니다.

- 1) 변수 선언
- 2) 비교 블록 compare block 삽입
- 3) 비교 식 입력
- 4) 어플리케이션 테스트

7.3 IEC 액션 사용

액션 action은 서브 루틴 및 이진 액션을 구현하는 데 사용 될 수 있습니다. 논리는 액션 블록을 활성화 한 다음 한정자 qualifier를 고려하여 관련 액션을 호출합니다. 예를 들어 한정자를 사용하여 조치 지연 또는 제한 여부를 지정할 수 있습니다.

사용 예

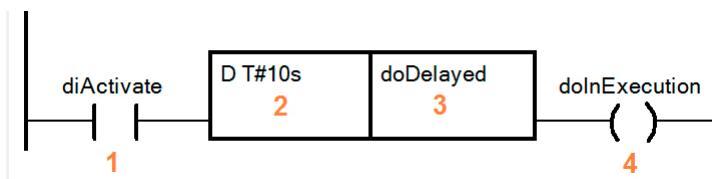


그림 56: "doDelayed"의 지연된 전환

1	"diActivate"는 액션 블록을 호출합니다.
2	"D T # 10s" 한정자는 작업이 10 초 지연되도록 지정합니다.

표 8: 위 그림에 대한 설명

3	이진 액션 "doDelayed"는 시간 지연 후 설정되어 액션 블록 호출 동안 계속 설정됩니다.
4	"doInExecution"은 전달 된 "diActivate"신호에 해당합니다.

표 8: 위 그래프에 대한 설명

중요한 한정자(qualifier) 개요

가장 중요한 한정자 중 일부가 아래 표에 있습니다. 완전한 개요는 Automation Studio 도움말 문서에서 찾을 수 있습니다.

기호	설명
D	액션 지연, 글자로 입력된 시간 사양 필요
L	시간 제한 액션, 글자로 입력된 시간 사양 필요
S	액션 조치 및 R 한정자까지 활성화 상태로 유지
R	액션 리셋
N	액션 블록이 활성화 되어있는 동안 호출된 액션

표 9: 중요한 한정자(qualifier)



[Programming \ Programs \ Ladder Diagram \(LD\) \ Blocks](#)

[Programming \ Actions](#)

[Programming \ Actions \ Action block - Qualifiers](#)

예제: 사용자 평선 블록 생성

래더 다이어그램 프로그래밍의 모든 기본 기능과 가능성을 설명했습니다. 연습에서 평선 블록을 생성합니다. 마지막 연습 내용을 평선 블록에 구현합니다.

입력/출력 IN / OUT	이름 Name	데이터 타입 Data types	설명
IN	Switch	BOOL	모터 활성화를 위한 입력
IN	visParamLimit	UINT	주의를 출력하는 구성 가능한 한도
IN	oldActivationCount	USINT	오래된 활성화 횟수 카운터
IN	visButtonReset	BOOL	화면 장치에서 주의 인지(acknowledge)
OUT	visWarning	BOOL	화면 장치를 위한 주의
OUT	visActivationCount	UINT	총 활성화 된 카운터
OUT	Motor	BOOL	모터 출력

표 10: 평선 블록 입력 및 출력 개요

8 예제

8.1 예제 - 컨베이어 벨트

컨베이어 벨트 운동

레더 다이어그램을 사용하여 컨베이어 벨트를 구동합니다. 어플리케이션 자체에는 수동 및 자동 모드가 있습니다. 자세한 기능 설명과 입출력 목록을 설명합니다.

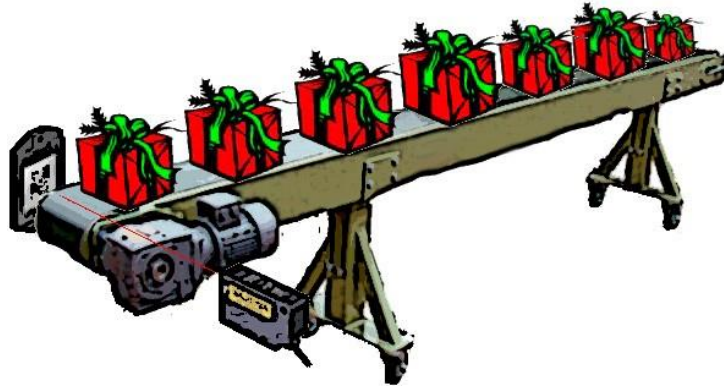


그림 57: 컨베이어 벨트

예제: 컨베이어 벨트

제어 목적으로 다음 기능을 구현합니다:

수동 모드 **Manual mode:**

- 자동 모드가 비활성화되어 있습니다 **"diAutoMode"**.
- **"diManualStart"**가 활성화되면 컨베이어 벨트가 작동합니다.

자동 모드 **Automatic mode:**

- 다음 상황에 컨베이어 벨트를 시작하십시오.
 - 자동 모드 **"diAutoMode"** 활성화 됨
 - 리미트 스위치 **"diConvEnd"**가 작동하지 않음
 - 엔드 스위치 **"diConvEnd"** 및 재료 요청 **"diMachAskMat"**이 활성화 됨
- 다음 상황에 컨베이어 벨트를 중지하십시오.
 - 리미트 스위치 **"diConvEnd"**가 활성화 되고 재료 요청 **"diMachAskMat"**이 비활성화 됨

프로그램 구조:

- 수동 모드에서는 자동으로 건너뛰는 운영 네트워크를 핸들링합니다.

배치 카운터(**Batch counter**):

- CTU 평선 블록을 사용하여 컨베이어로 이동 한 항목 수를 계산합니다.

변수(Variables)	데이터 타입(Data types)	설명
diAutoMode	BOOL	수동과 자동 작동 전환 스위치
diManualStart	BOOL	수동 모드에서 컨베이어 벨트의 수동 이동 시작
diConvEnd	BOOL	컨베이어 벨트 엔드 스위치
diMachAskMat	BOOL	기계 재료 요청
doConvMotor	BOOL	컨베이어 벨트를 구동하는 모터 출력

표 11: 입력과 출력 개요

8.2 예제- 콘크리트 충전 시스템

예제: 콘크리트 충전 시스템

콘크리트 혼합 시스템에서 콘크리트는 컨베이어를 통해 트럭에 적재됩니다. 충전

작업은 On 버튼 (**btnOn**)을 눌러 시작됩니다.

그러나 컨베이어가 5 초 동안 작동하고 트럭이 벨트 아래에 위치 할 때까지 솔레노이드 밸브 (**doValve**)로 제어되는 유압 시스템을 열 수 없습니다 (**diTruck**).

트럭의 총 허용 중량에 도달하자마자 솔레노이드 밸브가 차단됩니다

(**diPressure**). 컨베이어 벨트는 추가로 5 초 동안 더 작동합니다. 꺼짐 버튼 (**btnOff**)을 누르면 전체

시스템이 즉시 종료됩니다.

컨베이어 시스템 (**diConveyorMotorProtection**)에 장애가 있는 경우 솔레노이드 밸브와 컨베이어 벨트 (**doConveyor**)를 즉시 차단해야 합니다. 솔레노이드 밸브 (**diValveProtection**)에 장애가 있으면 즉시 닫히지만 벨트는 추가 5 초 동안 더 작동합니다.

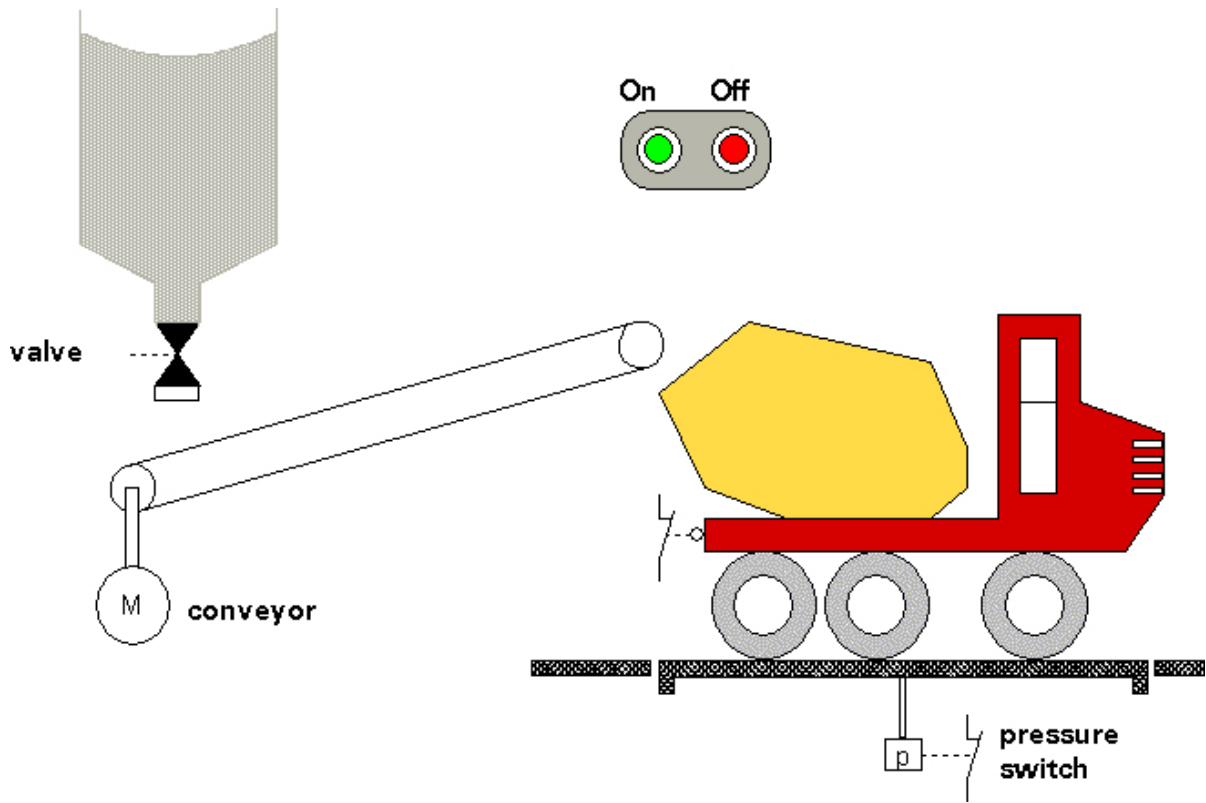


그림 58: "콘크리트 충전 시스템" 예제의 개략도

9 요약

래더 다이어그램은 여전히 매우 인기가 있습니다. 하드 와이어링 릴레이 논리를 로직 스위치로 프로그래밍하기 위해 개발되었습니다.

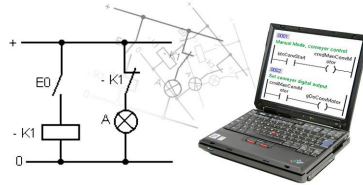


그림 59: 래더 다이어그램

래더 다이어그램에서 아날로그 신호 및 평선 블록을 사용하여 고효율 애플리케이션을 만들 수 있습니다. 프로그램 흐름 제어를 위한 추가 요소는 기능을 확장합니다. Automation Studio에서 Powerflow를 사용하여 프로그램 실행을 추적 할 수 있습니다. 색상은 전기를 공급하는 라인 상태를 표시하는 데 사용됩니다.

10 부록

10.1 편집기에서 활용하는 키보드 단축키



그림 60: 래더 다이어그램 편집기 툴바







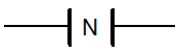





기호	키보드 단축키	기호	키보드 단축키
 그림 61: a 접점, normally open contact	C	 그림 62: 코일 Coil	Shift + C
 그림 63: b 접점, normally closed contact	L	 그림 64: 네거티브 코일 Negated coil	Shift + L
 그림 65: 포지티브 엣지 Positive edge	P	 그림 66: 셋 코일 Set coil	Shift + S
 그림 67: 네거티브 엣지 Negative edge	N	 그림 68: 리셋 코일 Reset coil	Shift + R
 그림 69: 왼쪽 연결 선 추가 / 삭제	ALT + ←	 그림 70: 평선 블록 추가하기	F
 그림 71: 오른쪽 연결 선 추가 / 삭제	ALT + →	 그림 72: 아날로그 값 연결 (숫자, 문자열 등)	스페이스 바 space bar

표 12: 편집기에서 활용하는 키보드 단축키 개요

기호	키보드 단축키	기호	키보드 단축키
 그림 73: 연결선 위쪽으로 삽입 / 삭제	ALT + ↑	 그림 74: 접점의 주소	A
 그림 75: 연결선 아래쪽으로 삽입 / 삭제	ALT + ↓	네트워크(network) 완성 및 확인	엔터 Enter
새 열 삽입, 중간 접점 추가	INS	 그림 76: 설명 추가	D

표 12: 편집기에서 활용하는 키보드 단축키 개요

기존 접점 사이에 새 접점을 추가 할 수 있습니까?

기존 접점 사이에 접점을 추가하려면 <INS> 키를 사용하여 새 열을 추가하십시오. 키보드 단축키로 접점을 선택하면 열린 위치에 추가됩니다.

접점 타입을 변경하는 쉬운 방법이 있습니까?

접점이 강조 표시되거나 커서가 바로 앞에 있는 경우 다른 접점 타입 키보드 단축키를 사용하여 변경할 수 있습니다.

(다른) 변수를 접점에 어떻게 연결할 수 있습니까?

접점을 선택하거나 커서를 바로 앞에 놓고 <스페이스 바>를 눌러 변수 연결 필드를 실행시킬 수 있습니다. <스페이스 바>를 다시 누르면 기존 변수를 선택할 수 있는 변수 목록이 열립니다.

해답

첫 래더 다이어그램 생성

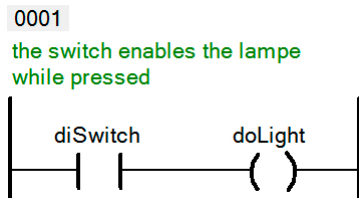


그림 77: 입력이 설정되어 있으면 출력은 TRUE로 유지됩니다.

포지티브와 네거티브 엣지 사용

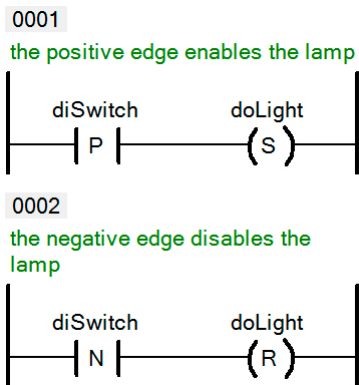


그림 78: 포지티브 에지는 설정에 사용됩니다; 네거티브 에지는 재설정에 사용됩니다.

키보드를 사용한 래더 프로그래밍

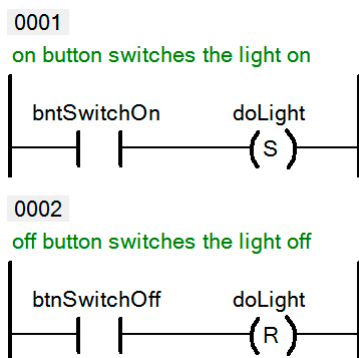


그림 79: 조명을 키고 끄기 위한 별도의 스위치

플립 플롭(flip-flop) 프로그래밍

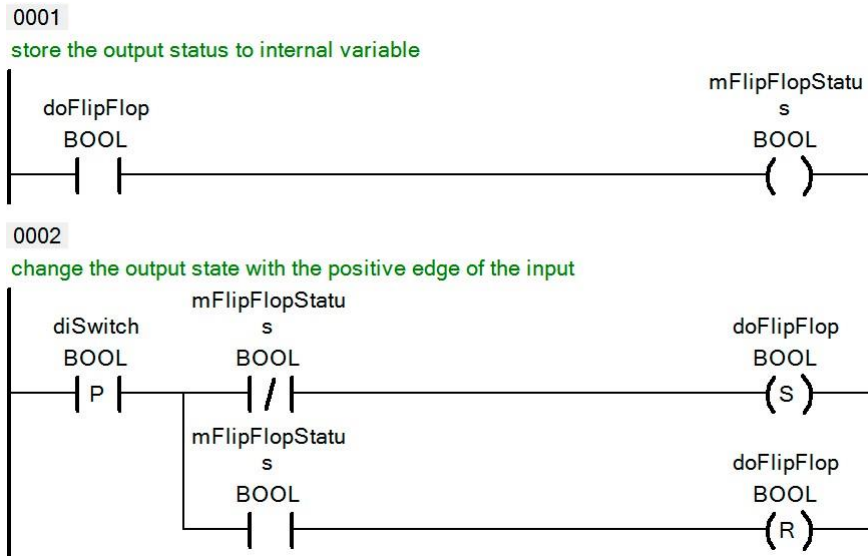


그림 80: 네트워크 1은 초기 상태를 저장하고, 네트워크 2는 입력 엣지에 따라 출력 상태를 변경합니다

평선 블록 사용하기

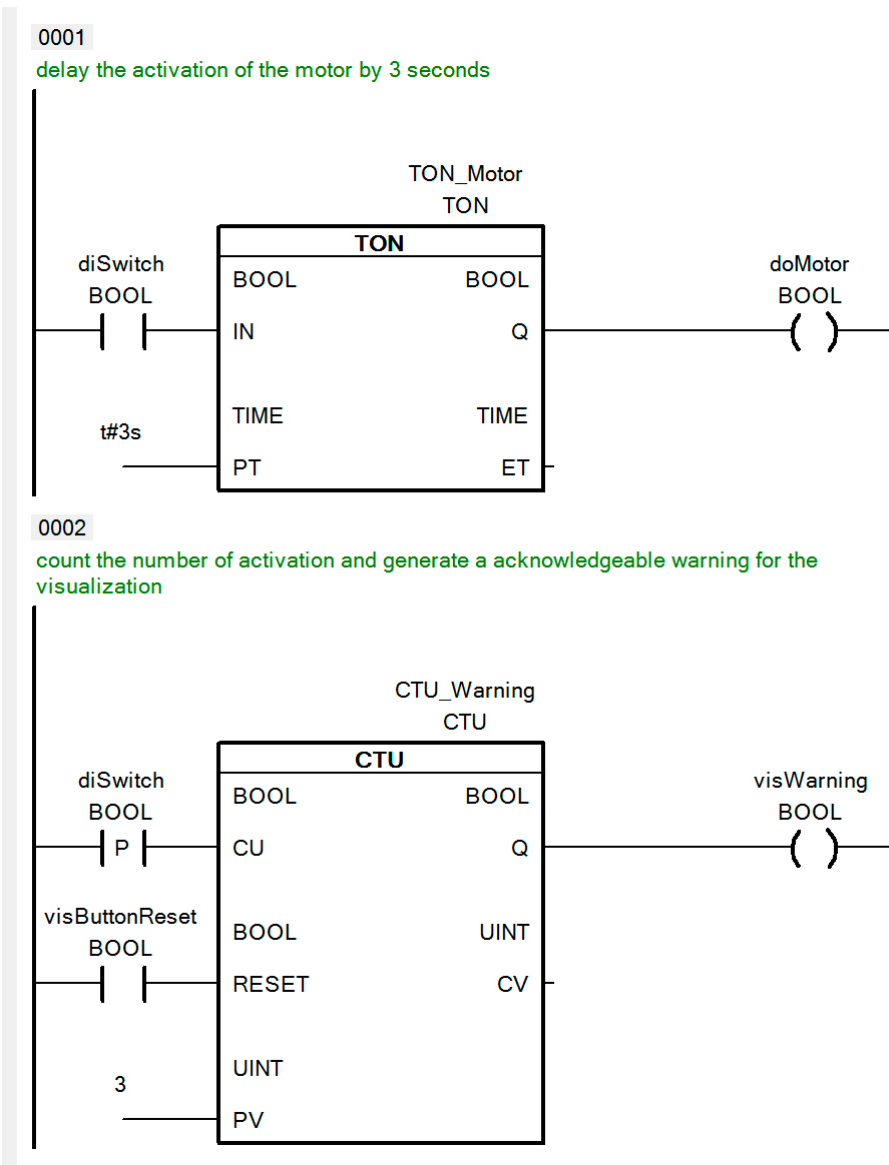


그림 81: turn-on 지연을 위한 TON, 주의 출력 기준값 PV를 가진 CTU

화면 장치를 사용하여 파라미터 구성

Name	Type	& Reference	is Constant	Retain	Value
*COPYRIGHT - Bernecker + Rainer					
* Program use_fubs					
* File use_fubsvor					
* Author brunneh					
* Created: September 22, 2011					
* Local variables of program use_fubs					
TON_Motor	TON		<input type="checkbox"/>	<input type="checkbox"/>	
diSwitch	BOOL		<input type="checkbox"/>	<input type="checkbox"/>	
doMotor	BOOL		<input type="checkbox"/>	<input type="checkbox"/>	
CTU_Warning	CTU		<input type="checkbox"/>	<input type="checkbox"/>	
visButtonReset	BOOL		<input type="checkbox"/>	<input type="checkbox"/>	
visWarning	BOOL		<input type="checkbox"/>	<input type="checkbox"/>	
visParamLimit	UINT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	3
visActivationCount	UINT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	

그림 82: 선언 창에서 미리 설정된 변수 초기화 RETAIN 옵션을 사용하면 재시작 후에도 유지할 수 있습니다.

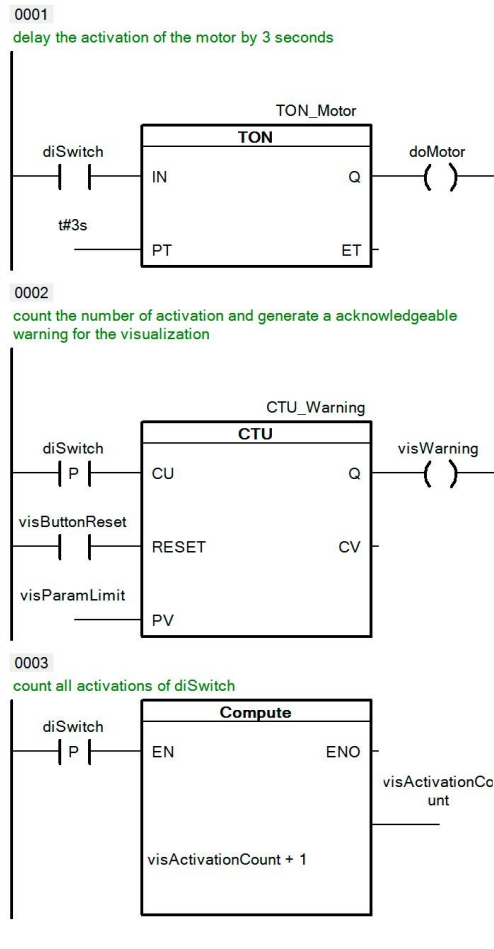


그림 83: CTU에 연결된 구성 가능한 리미트, 계산 블록(Compute block)으로 계산 된 총 개수

사용자 평선 생성

Name	Type	Reference	Constant	Retain	Value
Switch	BOOL	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
visParaLimit	UINT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
visButtonReset	BOOL	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
oldActivationCount	UINT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
visWarning	BOOL	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
visActivationCount	UINT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Motor	BOOL	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
TON_0	TON	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
TON_Motor	TON	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
CTU_Alarm	CTU	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
zzInternalMemory	USINT[0..9]	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

그림 84: .fun 파일에서 평선 블록 인스턴스 선언

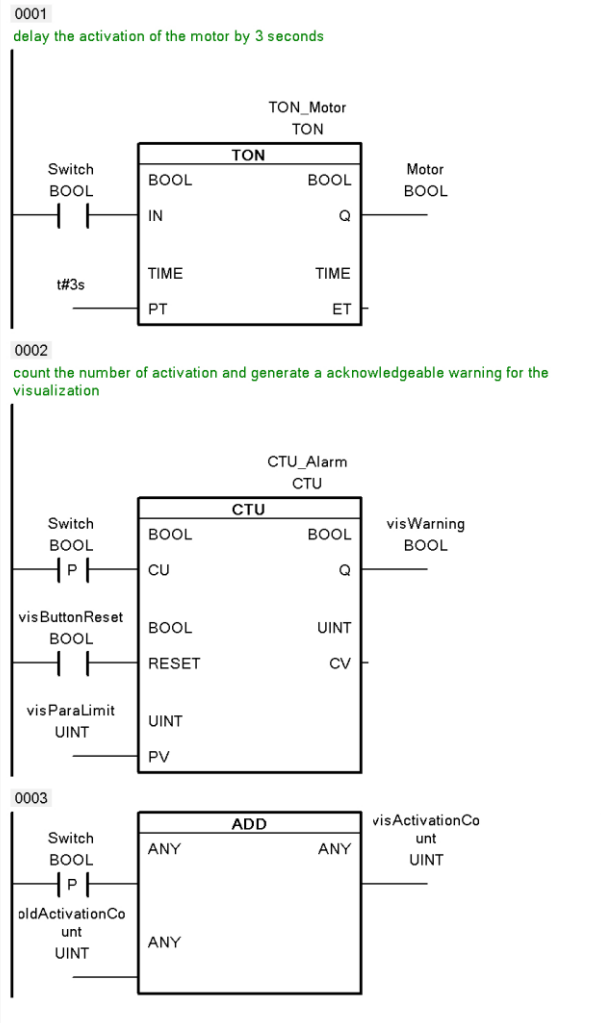


그림 85: 평선 블록을 구현 솔루션

Name	Type	Reference	Constant	Retain	Value
* COPYRIGHT -- Bernecker + Rainer					
* Programm: use_userfub					
* Date: use_userfub.var					
* Autor: tregguser					
* Erstellt: 22. September 2011					
* Lokale Variablen des Programms use_userfub					
alarming_0	alarming	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
dSwitch	BOOL	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
visLimit	UINT	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	3
ActivationCounter	UINT	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
visReset	BOOL	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Warning	BOOL	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
doMotor	BOOL	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

그림 86: 평선 블록을 호출하는 프로그램 변수 선언

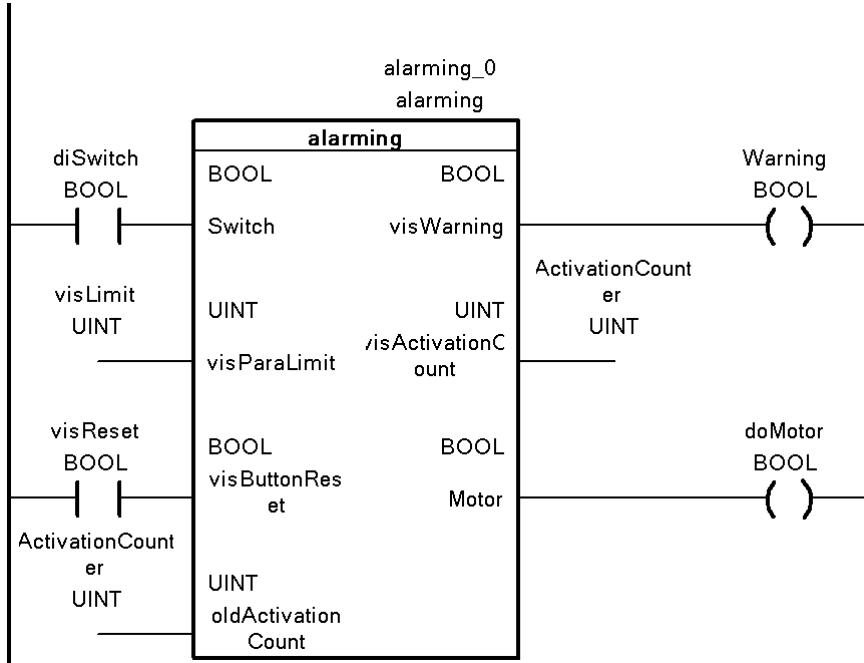


그림 87: 사용자 평선 블록 호출

실내 온도 제어

비교 블록(Compare block) 사용시

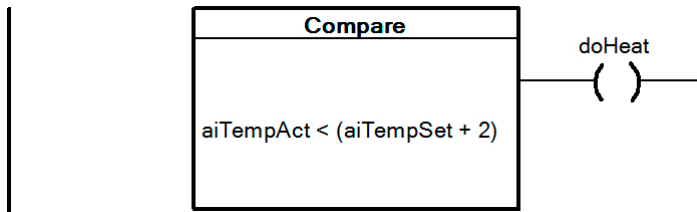


그림 88: 비교 블록(Compare block)을 사용한 솔루션

비교 블록(Compare block) 없이

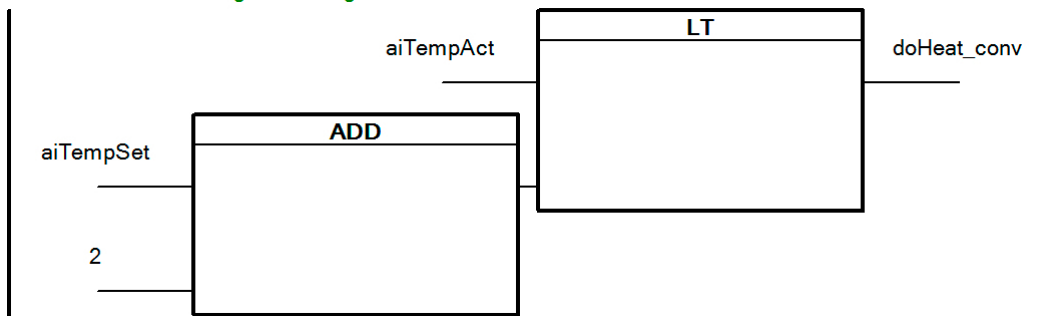


그림 89: 동일한 결과이지만 비교 블록(Compare block)이 없는 솔루션

평균 값 계산

계산 블록(Compute block) 사용시

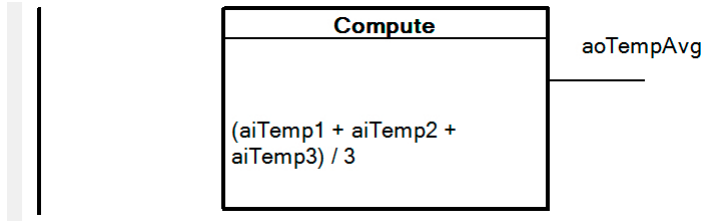


그림 90: 계산 블록(Compute block)을 사용한 솔루션

계산 블록(Compute block) 없이

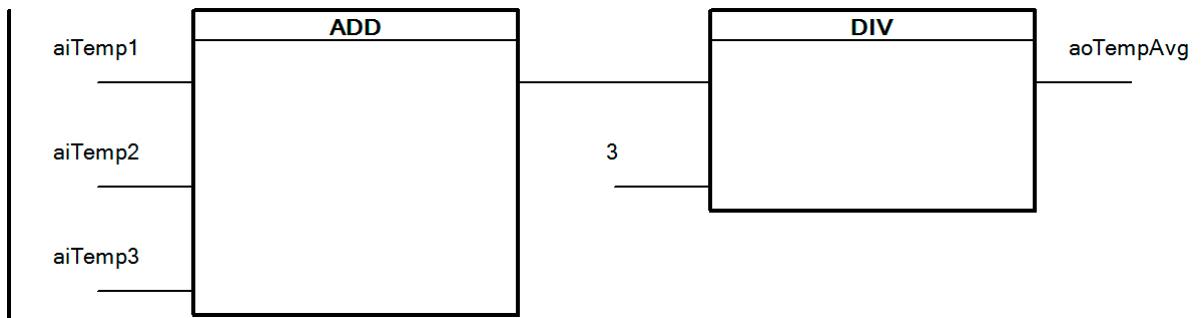


그림 91: 계산 블록(Compute block)이 없는 솔루션

컨베이어 벨트

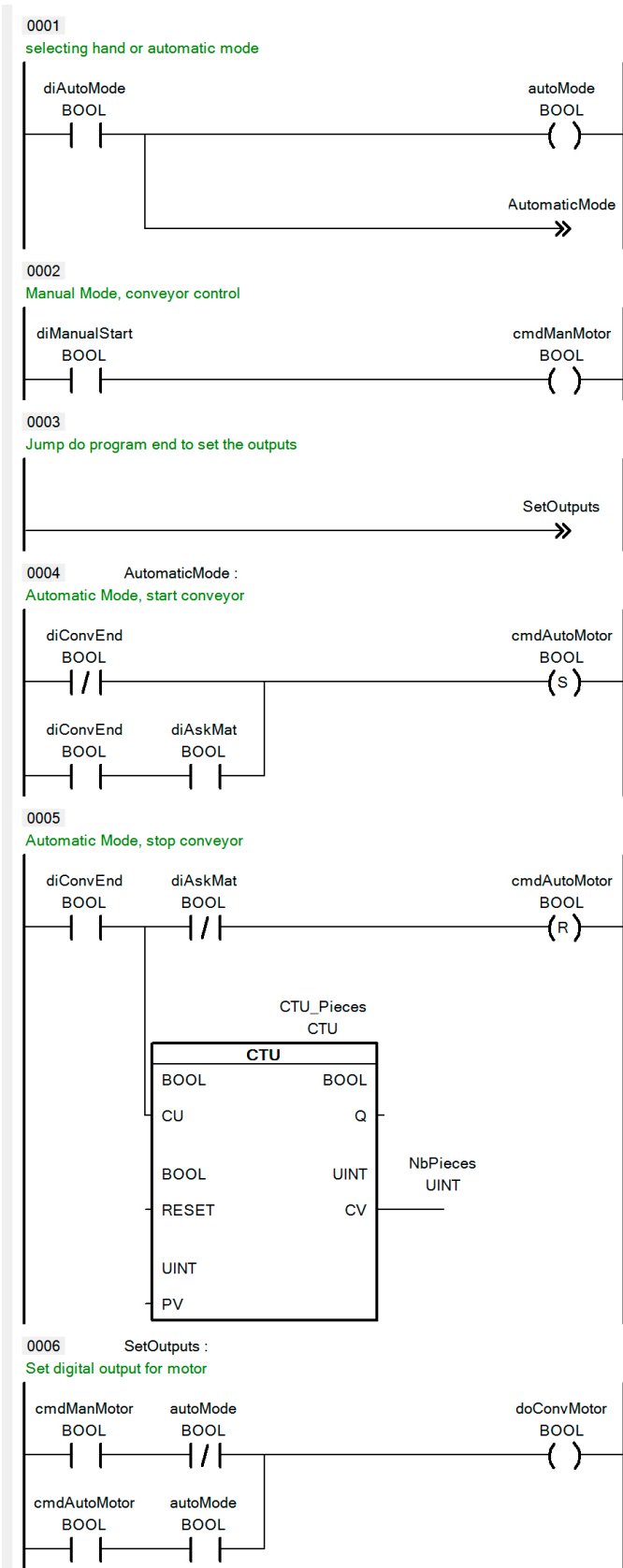


그림 92: 컨베이어 벨트 솔루션

콘크리트 충전 시스템

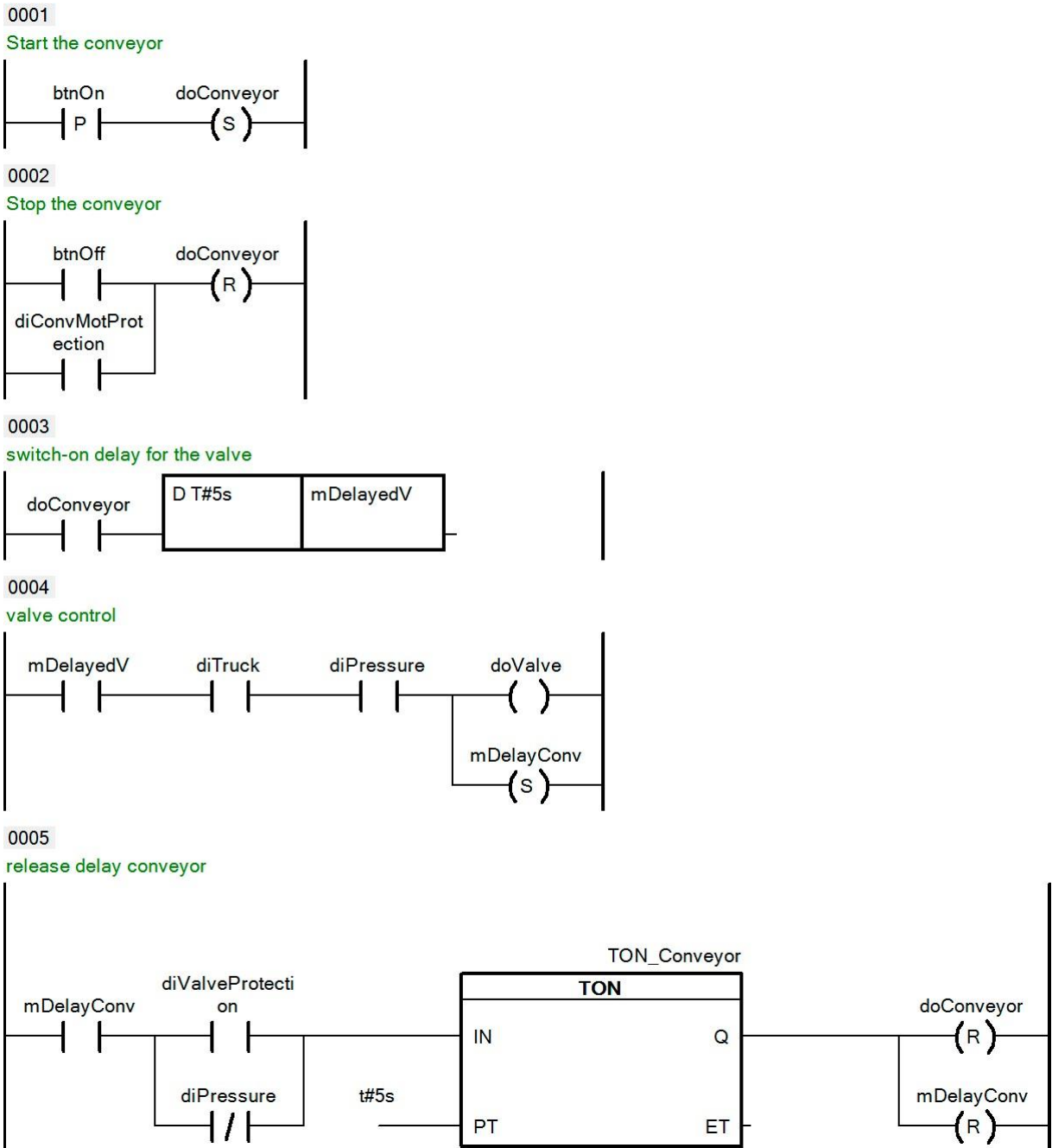


그림 93: 콘크리트 혼합 솔루션

Automation Academy에서 제공하는 것

Automation Academy에서 제공하는 것

우리는 고객뿐만 아니라 직원을 대상으로 한 교육 과정을 제공합니다.

Automation Academy에서, 당신이 필요한 능력을 즉시 향상시킬 수 있습니다.

자동화 엔지니어링 분야에서 필요한 지식 향상을 위해 세미나가 준비되어 있습니다. 한번 이수하면, 당신은 B&R 기술을 이용하여 능률적인 자동화 솔루션을 개발하는 위치에 있을 것입니다. 이를 통해 귀사와 귀사는 끊임 없이 변화하는 시장 수요에 보다 빠르게 대응할 수 있게 됨으로써 결정적인 경쟁 우위를 확보 할 수 있습니다.



세미나



품질 및 관련 성은 세미나의 필수 구성 요소입니다. 특정 세미나의 페이스는 엄격하게 코스 참가자가 직면한 요구 사항과 경험에 근거합니다. 그룹 스터디와 자율 학습에 조합은 학습 경험을 극대화하는데 필요한 높은 수준의 유연성을 제공합니다.

각 세미나는 숙련된 경험이 풍부한 강사 중 한 명이 진행합니다.

교육 자료(Training modules)

교육 자료는 세미나뿐만 아니라 자율 학습을 위한 기초로 제공합니다. 콤팩트 모듈은 일관된 교육 개념에 의존합니다. 상향식 구조는 복잡하고 상호 연관된 주제를 효율적이고 효과적으로 배울 수 있습니다. 광범위한 도움말 시스템이 가장 좋은 보완책입니다. 교육 자료는 다운받을 수 있으며 인쇄된 버전을 주문할 수 있습니다.

카테고리 주제:

- ⇒ 제어 기술
- ⇒ 모션 제어
- ⇒ 세이프티 기술
- ⇒ 화면 작화
- ⇒ 프로세스 컨트롤
- ⇒ 진단 및 서비스
- ⇒ POWERLINK 와 openSAFETY

ETA system



ETA 시스템(ETA system)은 훈련, 교육 및 실험실에서 사용하기 위해 실제와 같은 구조를 제공합니다. 두가지 이상의 다른 기구 구조가 선택될 수 있습니다. ETA light system은 높은 자유도, 공간 절약 및 연구소 작업에 적합합니다. ETA standard system은 튼튼한 기구 구조와 사전에 와이어링된 센서와 액추에이터를 포함합니다.

더 알아보기!

추가 교육이 필요하십니까? B&R Automation Academy가 제공하는 것에 흥미가 있으신가요? 잘 찾아오셨습니다.

상세한 정보는 아래 링크에서 확인하실 수 있습니다:

www.br-automation.com/academy





V1.0.5.1 ©2017/10/17 by B&R, All rights reserved.
All registered trademarks are the property of their respective owners.
We reserve the right to make technical changes.