SIEMENS

SCE 교육 커리큘럼

Siemens Automation Cooperates with Education | 05/2017

TIA Portal Module 032-200

FB 프로그래밍 기초

Cooperates with Education

Automation

SIEMENS

교육 커리큘럼에 따른 적합한 SCE 트레이너 패키지

SIMATIC 컨트롤러

- **SIMATIC ET 200SP Open Controller CPU 1515SP PC F 및 HMI RT SW**
  주문 번호: 6ES7677-2FA41-4AB1
- **SIMATIC ET 200SP Distributed Controller CPU 1512SP F-1 PN Safety**
  주문 번호: 6ES7512-1SK00-4AB2
- **SIMATIC CPU 1516F PN/DP Safety**
  주문 번호: 6ES7516-3FN00-4AB2
- **SIMATIC S7 CPU 1516-3 PN/DP**
  주문 번호: 6ES7516-3AN00-4AB3
- **SIMATIC CPU 1512C PN(소프트웨어 장착) 및 PM 1507**
  주문 번호: 6ES7512-1CK00-4AB1
- **SIMATIC CPU 1512C PN(소프트웨어 장착), PM 1507 및 CP 1542-5 (PROFIBUS)**
  주문 번호: 6ES7512-1CK00-4AB2
- **SIMATIC CPU 1512C PN(소프트웨어 장착)**
  주문 번호: 6ES7512-1CK00-4AB6
- **SIMATIC CPU 1512C PN(소프트웨어 장착) 및 CP 1542-5 (PROFIBUS)**
  주문 번호: 6ES7512-1CK00-4AB7

교육용 SIMATIC STEP 7 소프트웨어

- **SIMATIC STEP 7 Professional V14 SP1 - 단일 라이센스**
  주문 번호: 6ES7822-1AA04-4YA5
- **SIMATIC STEP 7 Professional V14 SP1 - 강의실 라이센스 (최대 인원 6명)**
  주문 번호: 6ES7822-1BA04-4YA5
- **SIMATIC STEP 7 Professional V14 SP1 - 업그레이드 라이센스 (최대 인원 6명)**
  주문 번호: 6ES7822-1AA04-4YE5
- **SIMATIC STEP 7 Professional V14 SP1 - 학생 라이센스 (최대 인원 20명)**
  주문 번호: 6ES7822-1AC04-4YA5

위 트레이너 패키지는 필요 시 후속 모델 패키지로 대체가 된다는 점에 유의하십시오. 현재 출시된 SCE 패키지에 대한 개요는 siemens.com/sce/tp에서 제공됩니다.

보충 교육

지멘스의 지역별 SCE 보충 교육에 대한 내용은 해당 지역의 SCE 고객 센터로 문의하시기 바랍니다.
siemens.com/sce/contact

SCE 관련 추가 정보

siemens.com/sce

사용 관련 정보

통합 자동화 솔루션인 TIA(Totally Integrated Automation)를 위한 SCE 교육 커리큘럼은 공교육 시설 및 R&D 기관 교육 목적의 "SCE(Siemens Automation Cooperates with Education) 프로그램을 위해 마련된 것입니다. Siemens AG는 프로그램의 내용을 보증하지 않습니다.

본 문서는 지멘스 제품/시스템을 초기 교육하는 용도로만 사용되어야 합니다. 따라서 교육 범위 내에서의 사용 목적으로 전체 또는 일부를 복사하여 교육생들에게 제공할 수 있습니다. 본 문서는 공공 교육 및 고등 교육 시설 내에서의 교육을 위한 목적으로의 배포, 복사 및 내용의 공유가 가능합니다.

예외적인 경우에는 Siemens AG 담당자의 서면 동의가 필요합니다. Roland Scheuerer roland.scheuerer@siemens.com.

해당 규정의 위반 시에는 그에 대한 책임이 부과될 수 있습니다. 특히 특허가 부여되었거나 실용신안 또는 의장등록이 된 경우, 번역을 포함한 제반 권리는 지멘스의 소유입니다.

산업체 고객을 위한 교육 과정의 사용은 명시적으로 금지됩니다. 지멘스는 교육 커리큘럼의 상업적 이용을 거부합니다.

드레스덴공대(TU Dresden), 특히 공학 박사 Leon Urbas 교수와 Michael Dziallas Engineering Corporation, 그리고 본 교육 커리큘럼을 준비하는 과정에서 도움을 주신 모든 관계자들께 감사의 말씀을 전합니다.

목차

TOC

FB 프로그래밍 기초

1 목표

이 챕터에서는 제어 프로그램의 기본 요소인 *오거나이제이션 블록(OB), 펑션(FC), 펑션 블록(FB), 데이터 블록(DB)*에 대해 알아보겠습니다. 뿐만 아니라, 라이브러리 호환 펑션 및 펑션 블록 프로그래밍에 대해서도 소개를 하겠습니다. *펑션 블록 다이어그램(FBD)* 프로그래밍 언어에 대해 알아보고 이를 이용해 펑션 블록(FB1)과 오거나이제이션 블록(OB1)을 프로그래밍해보겠습니다.

제3장에 기술된 SIMATIC S7 제어 장치를 사용할 수 있습니다.

2 전제 조건

이 챕터에서는 SIMATIC S7 CPU1516F-3 PN/DP의 하드웨어를 구성해 보겠습니다. 그러나 디지털 입력 및 출력 카드가 포함된 다른 하드웨어 구성을 사용할 수도 있습니다. 이 챕터에서는 예를 들어 SCE_EN_012_101__Hardware_Configuration_CPU1516F.zap13라는 프로젝트를 사용할 수 있습니다.

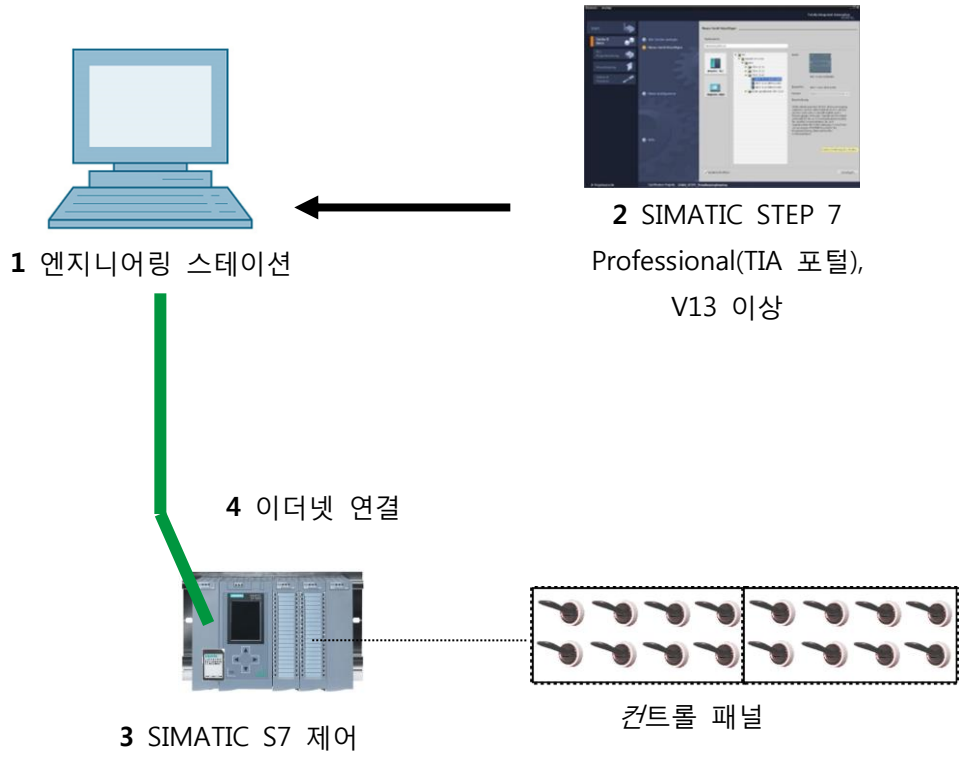
3 필요한 하드웨어 및 소프트웨어

1 엔지니어링 스테이션: 하드웨어 및 운영 시스템이 필요합니다(자세한 정보는 TIA 포털의 설치 DVD Readme/Liesmich를 참조하세요).

2 TIA 포털의 소프트웨어 SIMATIC STEP 7 Professional – V13부터

3 SIMATIC S7-1500/S7-1200/S7-300 제어 장치, 예: CPU 1516F-3 PN/DP – 펌웨어 버전 V1.6 이상, 메모리 카드와 16DI/16DO 및 2AI/1AO 포함 참고: 디지털 입력은 컨트롤 패널에서 실행되어야 합니다.

4 엔지니어링 스테이션과 제어 장치 간 이더넷 연결



4 이론

4.1 운영체제 및 애플리케이션 프로그램

모든 컨트롤러(CPU)에는 운영체제가 포함되어 있는데, 특정한 제어 작업과 연관이 없는 CPU의 모든 기능과 프로세스를 조직하는 역할을 합니다. 운영체제는 다음과 같은 작업을 수행합니다.

- 웜 리스타트 수행
- 입력 및 출력의 프로세스 이미지 업데이트
- 주기적 사용자 프로그램 호출
- 인터럽트 감지 및 인터럽트 OB 호출
- 오류 감지 및 처리
- 메모리 영역 관리

운영체제는 CPU를 구성하는 핵심 요소로서 사전 설치가 되어 있습니다.

사용자 프로그램에는 특정한 자동화 작업을 수행하는 데 필요한 모든 기능들이 포함되어 있습니다. 사용자 프로그램은 다음과 같은 작업을 수행합니다.

- 스타트업 OB를 이용한 웜 리스타트를 위해 기본적으로 필요한 기능을 확인
- 프로세스 데이터 처리 (예: 입력 신호 상태의 한 기능으로서 출력 신호 활성화)
- 인터럽트 및 인터럽트 입력에 반응
- 프로그램이 정상 실행되는 동안 오류 처리

4.2 오거나이제이션 블록

오거나이제이션 블록(OB)은 컨트롤러 (CPU)의 운영체제와 애플리케이션 프로그램 사이의 인터페이스를 형성합니다. 이들 블록은 운영체제에서 호출이 되어 다음과 같은 작업들을 제어합니다.

- 주기적 프로그램 처리 (예: OB1)
- 컨트롤러의 스타트업 구성
- 인터럽트 중심(interrupt-driven)의 프로그램 처리
- 오류 처리

프로젝트는 최소한 *주기적 프로그램 처리를 위한 오거나이제이션 블록*은 가지고 있어야 합니다. 그림 1에서와 같이 *시작 이벤트*에 의해 OB가 호출됩니다. 뿐만 아니라 각각의 OB에는 우선순위가 정의되어 있는데, 예를 들어 오류 처리를 위한 OB82는 주기적 OB1을 인터럽트할 수 있습니다.

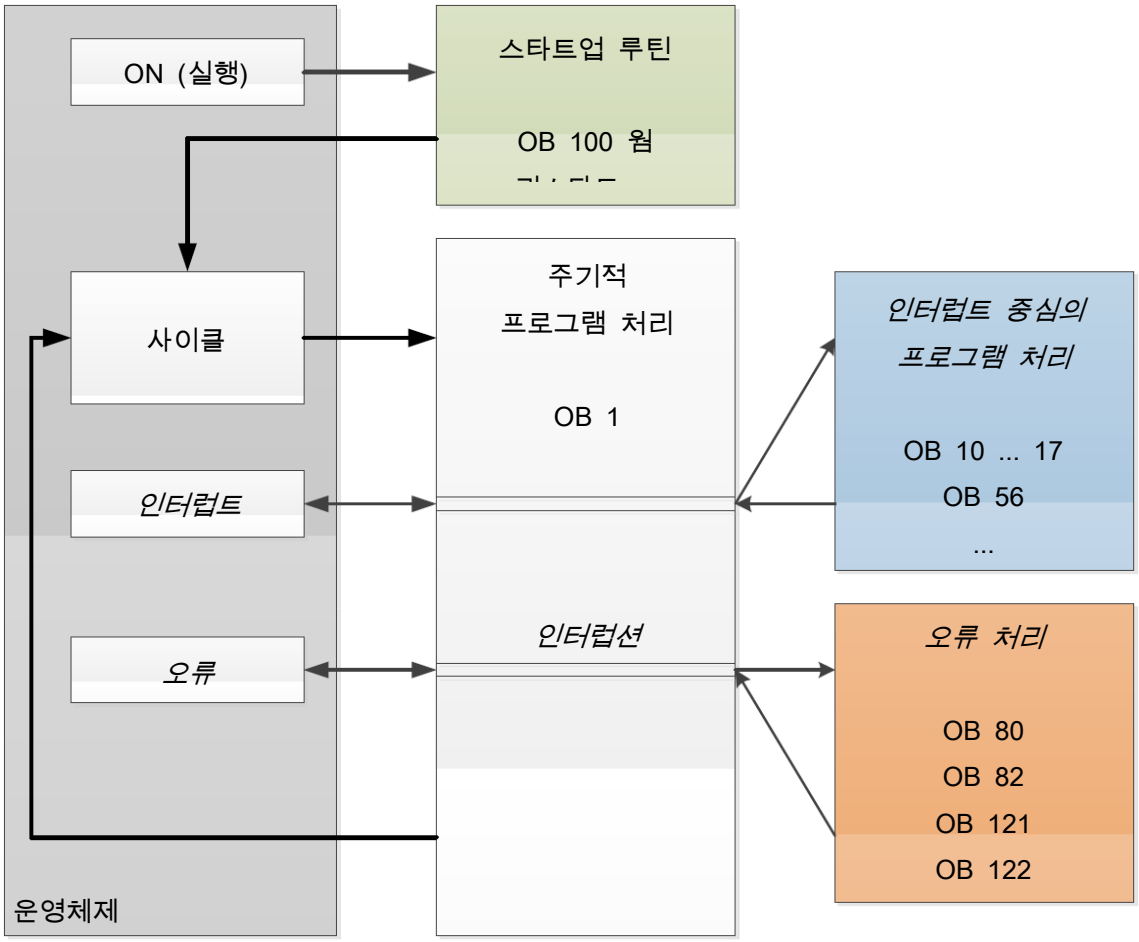


그림 1: 운영체제 및 OB 호출에서의 시작 이벤트

시작 이벤트가 발생하면 다음과 같이 반응할 수 있습니다.

- OB가 이벤트에 지정이 된 경우에는 이러한 이벤트가 지정된 OB의 실행을 개시합니다. 지정된 OB의 우선순위가 현재 실행 중인 OB의 우선순위보다 높으면 이 OB가 즉시 실행됩니다 (인터럽트). 그렇지 않은 경우에는 우선순위가 더 높은 OB가 끝까지 실행이 될 때까지 지정된 OB가 대기합니다.
- OB가 이벤트에 지정되지 않은 경우에는 기본 설정된 시스템 반응이 수행됩니다.

표 1에는 SIMATIC S7-1500에 대한 시작 이벤트와 가능한 OB 숫자, 오거나이제이션 블록이 컨트롤러에 존재하지 않은 경우 기본 시스템 반응의 예가 몇 가지 나와 있습니다.

시작 이벤트	가능한 OB 번호	기본 시스템 반응
스타트업	100, ≥ 123	무시
주기적 프로그램	1, ≥ 123	무시
TOD(Time-Of-Day) 인터럽트	10 ~ 17, ≥ 123	-
업데이트 인터럽트	56	무시
초과된 주기적 모니터링 시간 스캔	80	정지
진단 인터럽트	82	무시
프로그래밍 오류	121	정지
I/O 액세스 오류	122	무시

표 1: 다양한 시작 이벤트를 위한 OB 번호

4.3 프로세스 이미지 및 주기적 프로그램 처리

주기적 사용자 프로그램은 입력(I) 및 출력(O) 신호를 처리할 때 입력/출력 모듈에서 직접 신호 상태를 쿼리하지 않습니다. 그 대신 CPU의 메모리 영역을 액세스합니다. 이러한 메모리 영역에는 신호 상태의 이미지가 포함되어 있는데, 이를 프로세스 이미지라고 합니다.

주기적 프로그램 처리 순서는 다음과 같습니다.

1. 주기적 프로그램이 시작되면 각각의 입력에 전압이 인가되었는지 아닌지를 확인하기 위한 쿼리가 전송됩니다. 이러한 입력의 상태는 입력의 프로세스 이미지(PII)에 저장됩니다. 이 과정에서 전압이 인가된 입력에 대해서는 정보 1 또는 "High"가 저장되고, 무전압 상태의 입력에 대해서는 정보 0 또는 "Low"가 저장됩니다.

2. 이제 CPU는 주기적 OB에 저장된 프로그램을 실행합니다. CPU는 이전에 읽어들인 입력의 프로세스 이미지(PII)를 액세스하여 필요한 입력 정보를 얻고, 논리 연산의 결과(RLO)가 소위 출력의 프로세스 이미지(PIQ)에 기록됩니다.

3. 사이클이 끝나면 출력의 프로세스 이미지(PIQ)가 신호 상태로서 출력 모듈로 전달되고, 전원이 인가되거나 전원이 끊깁니다. 항목 1부터 다시 시퀀스가 계속 반복 진행됩니다.

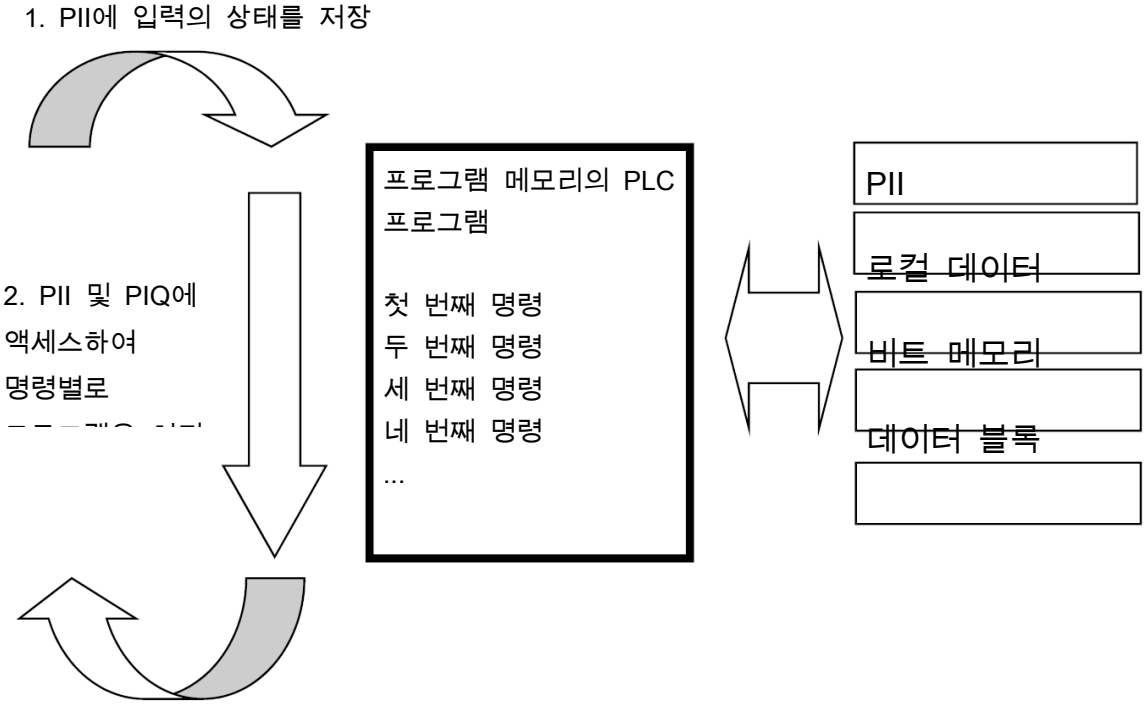


그림 2: 주기적 프로그램 처리

참고: 이 시퀀스를 위해 CPU에서 필요로 하는 시간을 사이클 시간이라고 합니다. 사이클 시간은 명령의 수와 유형, 그리고 컨트롤러의 프로세서 성능에 따라 결정됩니다.

4.4 펑션

펑션(FC)는 메모리가 없는 로직 블록입니다. 이들은 블록 파라미터의 값들을 저장할 수 있는 데이터 메모리가 없습니다. 따라서, 펑션이 호출될 때 모든 인터페이스 파라미터를 연결해 주어야 합니다. 데이터를 지속적으로 저장하려면 글로벌 데이터 블록을 미리 생성해야 합니다.

펑션에는 또 다른 코드 블록에서 펑션이 호출될 때마다 실행되는 프로그램이 포함되어 있습니다.

이러한 펑션은 예를 들어 다음과 같은 용도로 사용할 수 있습니다.

- 수학 펑션 – 입력 값에 따라 결과를 반환

- 기술 펑션 - 바이너리 논리 연산을 통한 개별 제어 등의 펑션

또한 펑션은 한 프로그램 내의 서로 다른 위치에서 여러 차례 호출이 가능합니다.

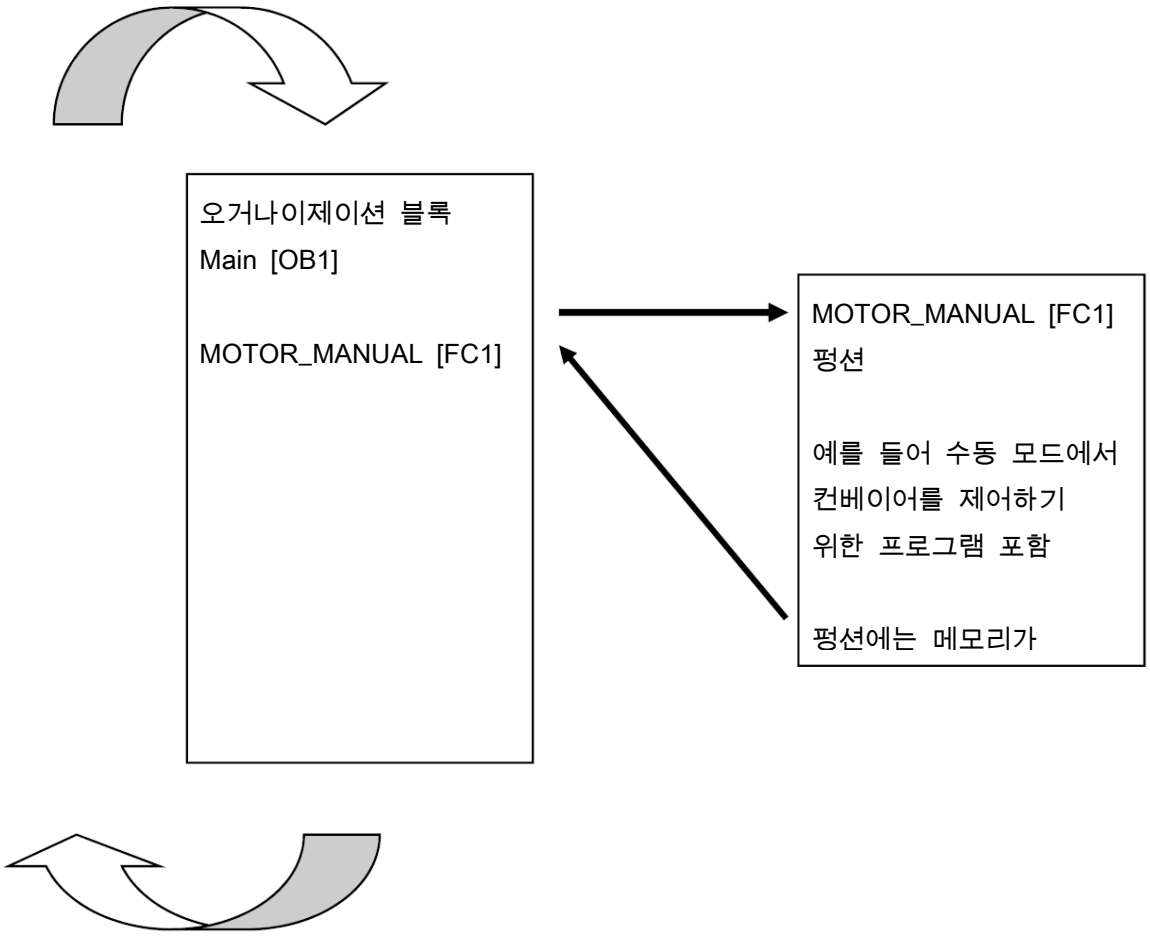


그림 3: 오거나이제이션 블록 Main [OB1]에서 호출되는 펑션

4.5 펑션 블록 및 인스턴스 데이터 블록

펑션 블록은 입력, 출력 및 입력-출력 태그를 비롯해 정적 태그를 인스턴스 데이터 블록에 지속적으로 저장하는 코드 블록이기 때문에 *블록이 실행된 이후에도 데이터의 사용이* 가능합니다. 이러한 이유로 펑션 블록은 "메모리"를 갖춘 블록으로 불리기도 합니다.

펑션 블록은 임시 태그에서도 작동이 가능합니다. 그러나 임시 태그는 인스턴스 DB에 저장되지 않습니다. 대신에 한 사이클 동안에만 사용이 가능합니다.

펑션 블록은 다음과 같은 펑션을 통해 구현할 수 없는 작업에 사용됩니다.

- 블록에서 타이머와 카운터가 필요한 모든 경우

- 프로그램에 정보를 저장해야 하는 모든 경우(예를 들어 버튼을 통한 운전 모드의 사전 선택)

펑션 블록은 다른 코드 블록에서 호출이 될 때마다 실행됩니다. 또한 펑션 블록은 한 프로그램 내의 서로 다른 위치에서 여러 차례 호출이 가능합니다. 따라서 자주 반복되는 복합 펑션을 신속하게 프로그래밍할 수 있습니다.

펑션 블록의 호출을 인스턴스라고 합니다. 펑션 블록의 각 인스턴스에는 펑션 블록이 사용하는 데이터가 포함된 메모리 영역이 할당되어 있습니다. 이 메모리는 소프트웨어에 의해 자동 생성되는 인스턴스 데이터 블록입니다.

또한, 멀티 인스턴스 형태로 하나의 데이터 블록 내에 여러 개의 인스턴스가 가능하도록 메모리를 제공하는 것이 가능합니다. 인스턴스 데이터 블록의 최대 크기는 CPU에 따라 다릅니다. 펑션 블록에서 선언된 태그가 인스턴스 데이터 블록의 구조를 결정합니다.

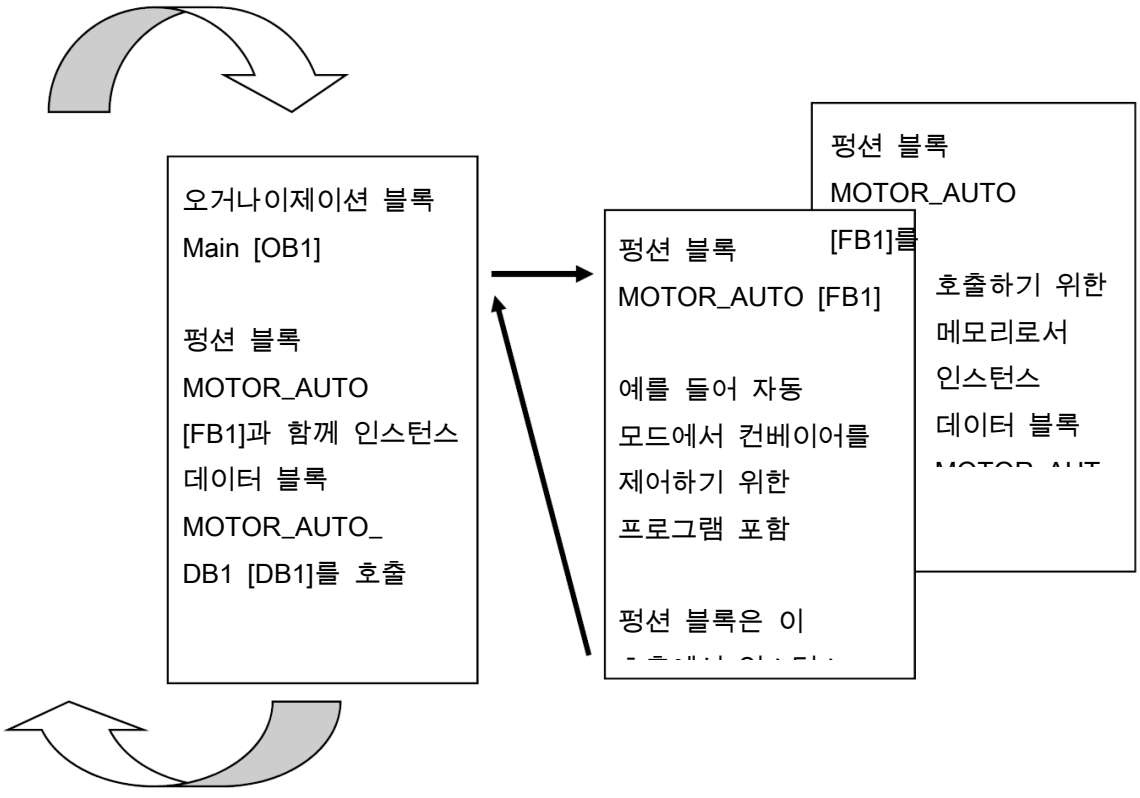


그림 4: 오거나이제이션 블록 Main [OB1]에서 호출된 펑션 블록 및 인스턴스

4.6 글로벌 데이터 블록

로직 블록과 대조적으로 데이터 블록에는 명령어가 포함되어 있지 않습니다. 그 보다는 사용자 데이터를 위한 메모리 역할을 합니다.

따라서 데이터 블록에는 사용자 프로그램에서 사용되는 가변 데이터를 저장합니다. 필요에 따라 글로벌 데이터 블록의 구조를 정의할 수 있습니다.

글로벌 데이터 블록에는 기타 모든 블록들에서 사용이 가능한 데이터가 저장되어 있습니다(그림 5 참조). 인스턴스 데이터 블록은 관련된 펑션 블록에서만 액세스해야합니다. 데이터 블록의 최대 크기는 CPU에 따라 다릅니다.

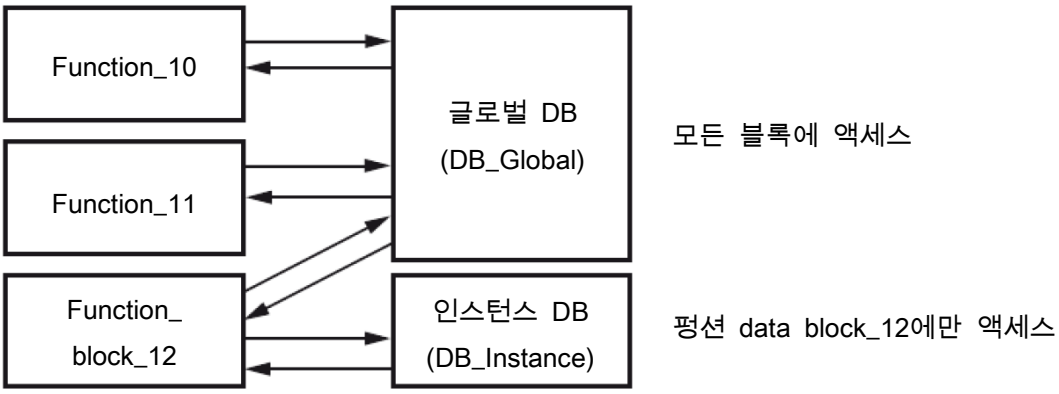


그림 5: 글로벌 DB와 인스턴스 DB 간의 차이

글로벌 데이터 블록의 애플리케이션 예는 다음과 같습니다.

- 스토리지 시스템에 대한 정보 저장 (예: "어떤 제품이 어디에 위치해 있는가?")
- 특정 제품에 대한 레시피 저장

4.7 라이브러리 호환 코드 블록

사용자 프로그램은 선형 또는 구조적 프로그래밍을 통해 생성할 수 있습니다. 선형 프로그래밍은 사이클 OB에서 전체 사용자 프로그램을 기록하지만, LOGO! 같이 보다 저비용 제어 시스템을 사용하는 단순한 프로그램에만 적합합니다.

보다 복잡한 프로그램에서는 *구조적 프로그래밍*을 권장합니다. 이는 전체 자동화 작업을 펑션과 펑션 블록과 같은 작은 단위 작업으로 나누어 자동화 솔루션을 구현하도록 합니다.

이 경우에는 라이브러리 호환 로직 블록을 우선적으로 생성해야 합니다. 즉, 펑션 또는 펑션 블록의 입력/출력 파라미터가 일반적으로 정의가 되고, 블록이 사용될 때 현재의 글로벌 태그(입력/출력)를 통해서만 제공됩니다.

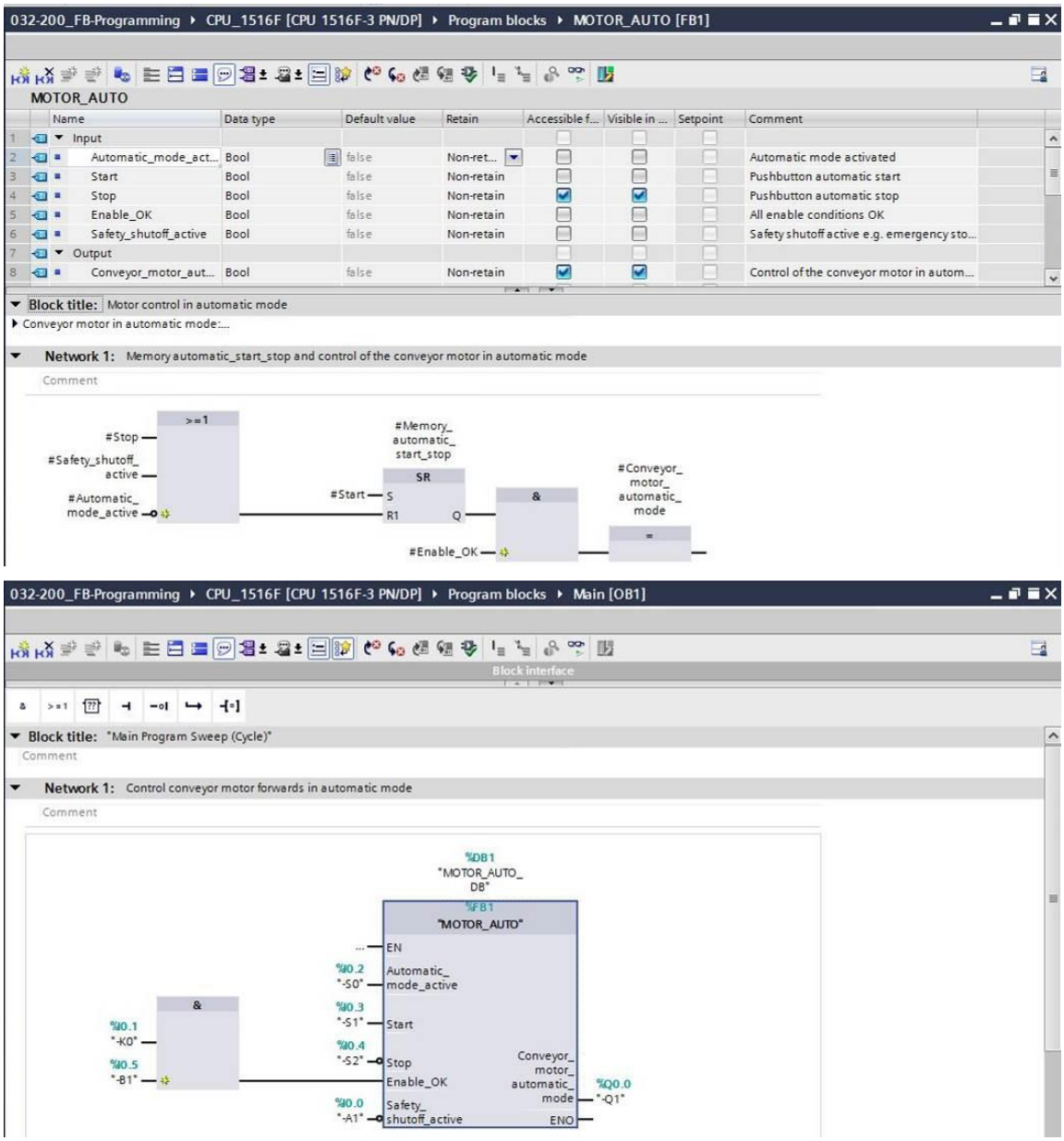


그림 6: OB1에서 호출된 라이브러리 호환 펑션

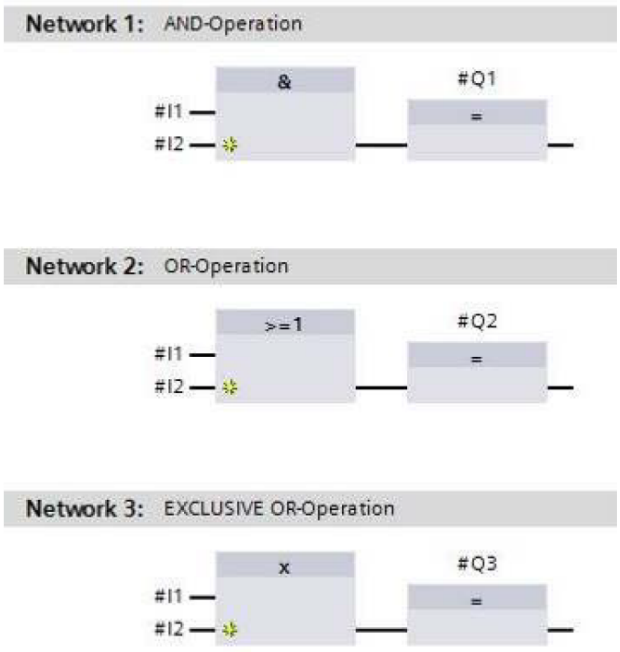
4.8 프로그래밍 언어

펑션을 프로그래밍하는 데 사용할 수 있는 프로그래밍 언어로는 펑션 블록 다이어그램(FBD), 래더 로직(LAD), 구문 목록(STL), Statement List(SCL)가 있습니다. 펑션 블록의 경우에는 GRAPH 프로그래밍 언어를 그래픽 스텝 시퀀스를 프로그래밍하는 데 추가적으로 사용할 수 있습니다.

펑션 블록 다이어그램(FBD) 프로그래밍 언어의 특징은 다음과 같습니다.

FBD는 그래픽 프로그래밍 언어입니다. 전자 스위칭 시스템을 토대로 표현이 됩니다. 프로그램은 네트워크에 매핑됩니다. 네트워크에는 1개 이상의 논리 연산 경로가 포함되어 있습니다. 바이너리 및 아날로그 신호는 박스로 연결이 됩니다. Bool 대수로부터 파악된 그래픽 로직 심볼은 바이너리 로직을 표현하는 데 사용됩니다.

바이너리 펑션을 이용해 바이너리 오퍼랜드에 대해 쿼리를 수행하고 이들의 신호 상태를 논리적으로 결합할 수 있습니다. 바이너리 펑션으로는 "AND operation", "OR operation", "EXCLUSIVE OR operation" 같은 명령이 있습니다. 이들은 그림 7에 나와 있습니다.



#e1	#e2	#a1
0	0	0
0	1	0
1	0	0
1	1	1

#e1	#e2	#a2
0	0	0
0	1	1
1	0	1
1	1	1

#e1	#e2	#a3
0	0	0
0	1	1
1	0	1
1	1	0

그림 7: FBD의 바이너리 펑션 및 관련 로직 테이블

예를 들어 간단한 명령을 이용해 바이너리 출력을 제어하기 위해 엣지가 발생할 때, 프로그램에서 점프 기능을 실행할 수 있습니다.

IEC 타이머 및 IEC 카운터 같은 프로그램 요소들은 복합 명령을 제공합니다.

빈 박스는 필요한 명령을 선택할 수 있는 플레이스홀더 역할을 합니다.

입력 파라메터 EN(enable)과 출력 파라메터 ENO(enable output) 메커니즘은 다음과 같습니다.

- EN/ENO 메커니즘이 없는 명령은 박스 입력의 신호 상태에 무관하게 실행
- EN/ENO 메커니즘이 있는 명령은 입력 파라메터 "EN"의 신호 상태가 "1"인 경우에만

실행됩니다. 박스가 올바르게 처리되면 출력 지원 "ENO"가 신호 상태가 "1"이 됩니다. 처리 동안 오류가 발생하면 그 즉시 "ENO"가 리셋됩니다. 입력 파라메터 EN에 아무 것도 연결 되지 않은 경우에는 항상 박스가 실행됩니다.

5 과제

이 챕터에서는 다음과 같이 선별기 프로세스 설명에 대한 펑션들을 계획, 프로그래밍 및 테스트해보겠습니다.

- 자동 모드 - 컨베이어 모터

6 계획 수립

명확성 및 재사용성을 이유로 OB1의 모든 기능을 프로그래밍하는 것은 권장하지 않습니다. 따라서 프로그램 코드의 대다수는 펑션(FC) 및 펑션 블록(FB)에서 작성 됩니다. FB로 이동시킬 펑션과 OB1에서 실행할 펑션은 아래와 같이 결정이 됩니다.

6.1 비상 정지(EMERGENCY STOP)

비상 정지를 위해서 별도의 펑션이 필요하지 않습니다. 마치 운전 모드처럼 비상 정지 릴레이의 현재 상태를 블록에서 직접 사용할 수 있습니다.

6.2 자동 모드 - 컨베이어 모터

컨베이어 모터의 자동 모드는 펑션 블록 (FB) "MOTOR_AUTO"에 캡슐화가 됩니다. 한편으로 캡슐화는 OB1의 명확성을 지켜줍니다. 다른 한편으로는 또 다른 컨베이어 벨트가 스테이션에 추가된 경우에 재사용을 지원합니다. 표 2에는 계획된 파라미터가 나와 있습니다.

입력	데이터	코멘트
Automatic_mode_active	BOOL	자동 모드 활성화
Start	BOOL	푸시버튼 자동 시작
Stop	BOOL	푸시버튼 자동 정지
Enable_OK	BOOL	모든 시작 조건 OK
Safety_shutoff_active	BOOL	안전 전원 차단 활성화 (예: 비상 정지 푸시버튼을 누름)
출력		
Conveyor_motor_automatic_mode	BOOL	자동 모드의 컨베이어 모터 제어
Static		
Memory_automatic_start_stop	BOOL	시작/정지 자동 모드에 사용되는

		메모리

표 2: FB "MOTOR_AUTO"를 위한 파라미터

Memory_automatic_start_stop은 리셋 조건이 충족되지 않을 경우, 시작 시 랫치되어 값이 1이 됩니다.

정지 기능이 작동되거나, 안전 전원 차단이 활성화 되어 있거나, 자동 모드가 활성화 되어 있지 않은 경우(수동 모드)에는 Memory_automatic_start_stop이 리셋됩니다.

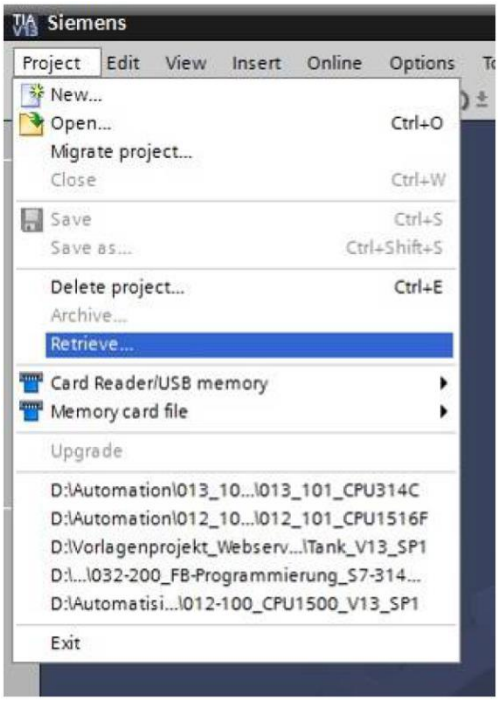
Memory_automatic_start_stop이 셋(set)되어 있고 시작 조건이 충족되면 이에 따라 Conveyor_motor_automatic_mode 출력이 제어됩니다.

7 단계별 따라 해보기

아래에는 계획을 수립하는 방법에 대한 지침이 나와 있습니다. 모든 내용을 이미 충분히 숙지했다면 숫자가 표시된 단계로 넘어가도 좋습니다. 그렇지 않다면, 아래에 나와 있는 지침의 단계를 따라가시면 됩니다.

7.1 기존 프로젝트 압축 풀기

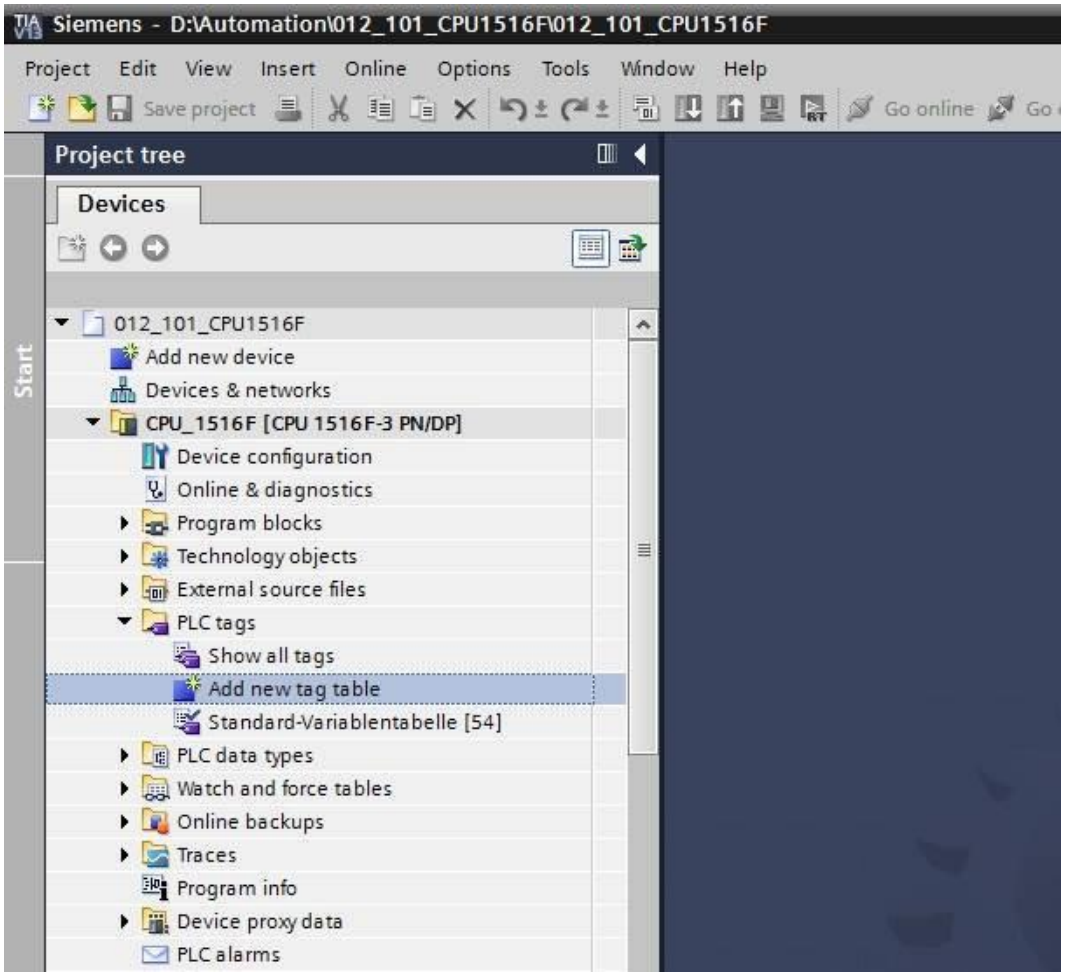
→ 펑션 블록(FB) "MOTOR_AUTO"의 프로그래밍을 시작할 수 있으려면 하드웨어 구성(예: SCE_EN_012_101_Hardware_Configuration_S7-1516F_R1502.zap)을 가진 프로젝트가 필요합니다. 아카이브된 기존 프로젝트의 압축을 풀려면 "Project"의 "Retrieve"으로 가서 해당되는 아카이브를 선택하고 "Open"을 클릭해 선택을 확정합니다. (→ Project → Retrieve → .zap 아카이브 선택 → Open)



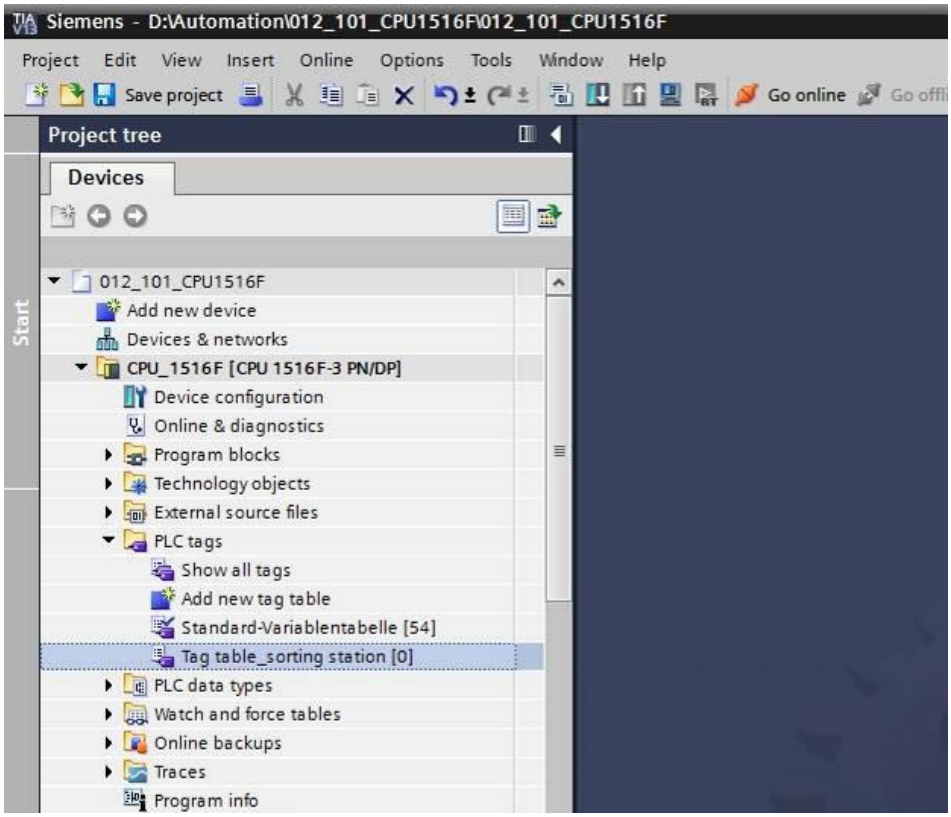
→ 그 다음으로 이 프로젝트가 저장될 대상 디렉토리를 선택합니다. "OK"를 눌러 선택을 확정합니다. (→ Target Directory → OK)

7.2 새 테이블 생성

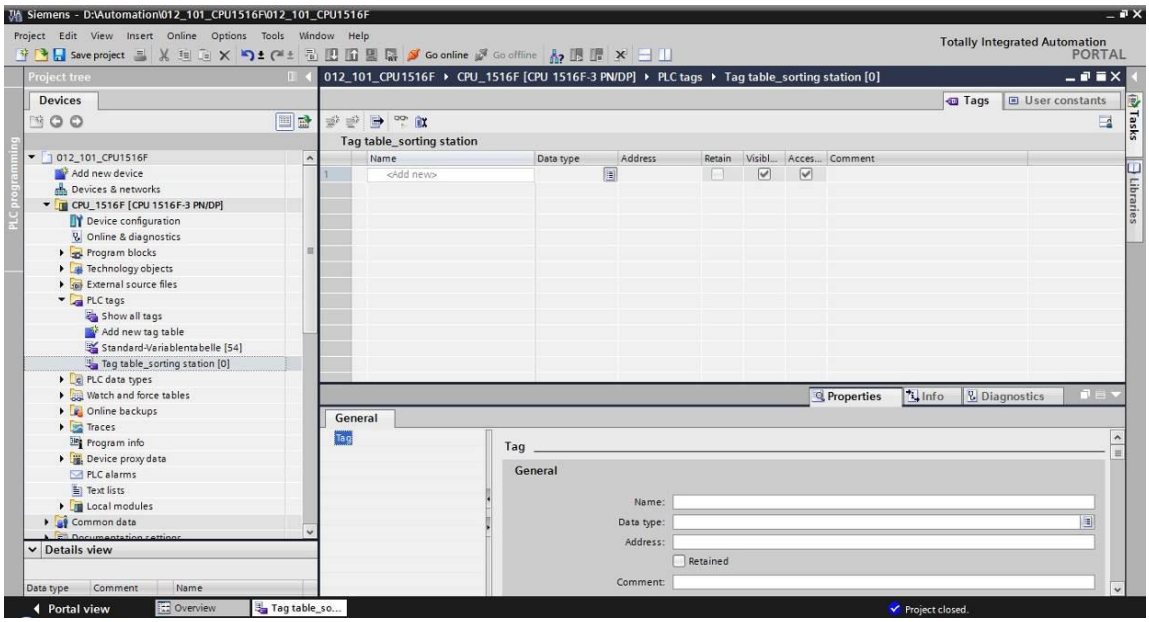
→ 프로젝트 뷰에서 컨트롤러의 PLC tags를 탐색하고 "Add new tag table"을 더블클릭하여 새 태그 테이블을 생성합니다.



→ 이렇게 생성된 태그 테이블의 이름을 "Tag_table_sorting_station"으로 변경합니다.
(→ "Tag_table_1"을 마우스 오른쪽 버튼으로 클릭 → "Rename" → Tag_table_sorting_station).

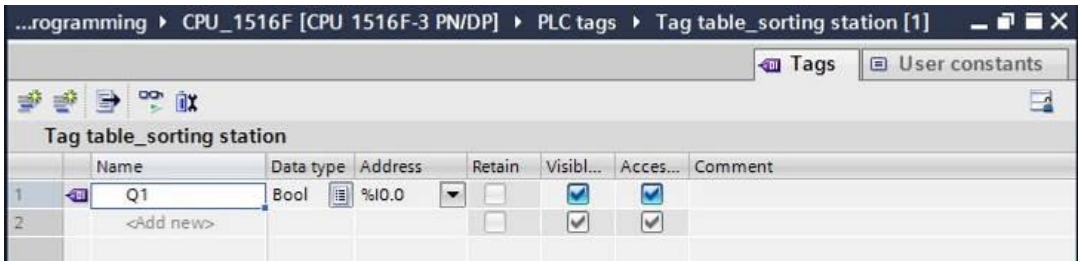


→ 더블클릭으로 태그 테이블을 엽니다. (→ Tag_table_sorting_station)

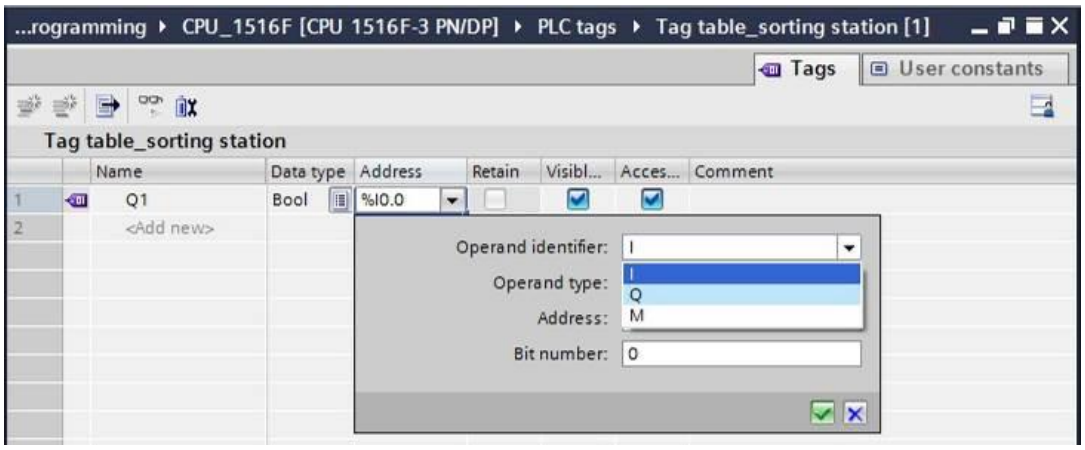


7.3 태그 테이블 내에서 새 태그 생성

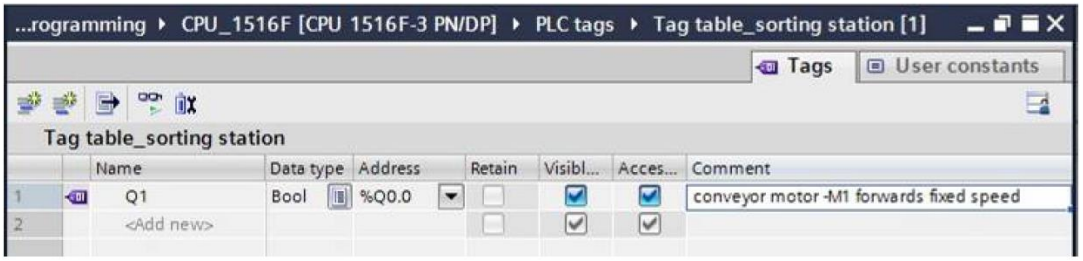
→ 이름 Q1을 추가하고 Enter 키를 눌러 입력을 확정합니다. 추가 태그를 아직 생성하지 않은 경우에는 TIA Portal이 데이터 타입 "Bool"과 주소 %I0.0을 자동으로 할당합니다. (→ <Add> → Q1 → Enter)



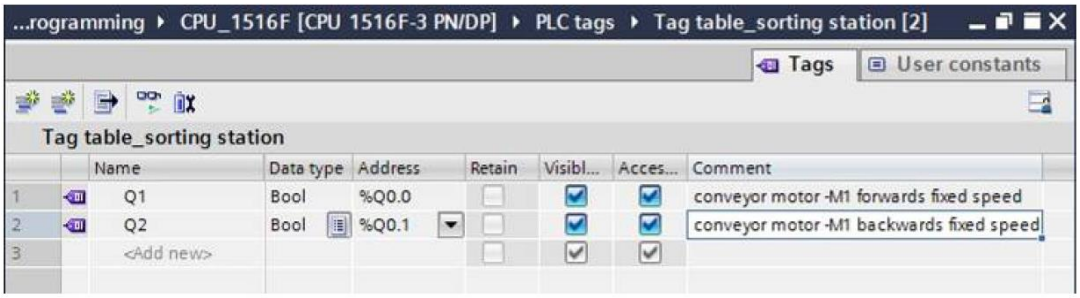
→ 이 값을 직접 입력하거나 드롭다운 화살표를 클릭해 "어드레싱" 메뉴를 열고 오퍼랜드 식별자를 Q로 변경한 다음 Enter를 누르거나 체크 표시를 클릭해 이를 확정함으로써 주소를 %Q0.0
(Q 0.0)으로 변경합니다. (→ %I0.0 → Operand identifier → Q →)



→ 태그에 "Conveyor motor -M1 forwards fixed speed" 코멘트를 입력합니다.

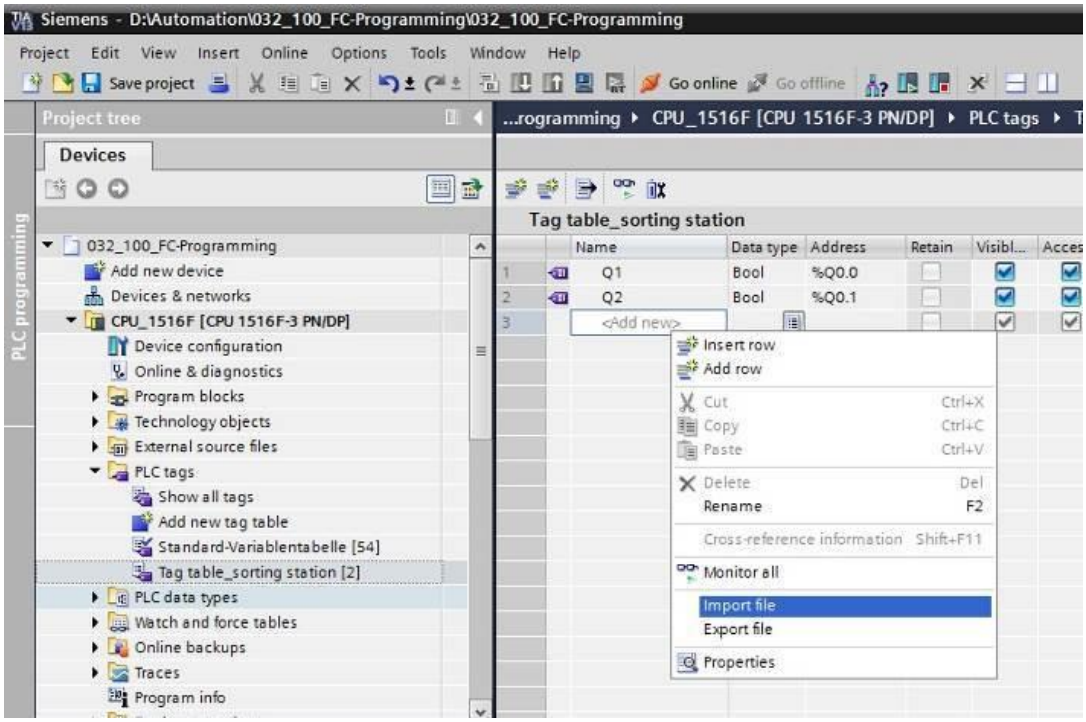


→ 라인 2에 새로운 Q2 태그를 추가합니다. TIA Portal은 라인 1과 동일한 데이터 타입을 자동으로 지정하고 %Q0.1 (Q0.1)의 주소를 "1" 증가시킵니다. "Conveyor motor -M1 backwards fixed speed" 코멘트를 입력합니다.
(→ <Add> → Q2 → Enter → Comment → Conveyor motor -M1 backwards fixed speed)



7.4 "Tag_table_sorting_station" 가져오기

→ 기존 심볼 테이블을 삽입하려면 생성된 "Tag_table_sorting_station"의 빈 필드를 마우스 오른쪽 버튼으로 클릭합니다. 바로 가기 메뉴에서 "Import file"을 선택합니다.
(→ 태그 테이블의 빈 필드에서 마우스 오른쪽 버튼을 클릭 → Import file)



→ 원하는 심볼 테이블 (예: .xlsx 형식)을 선택하고 “열기”를 클릭해 선택합니다.

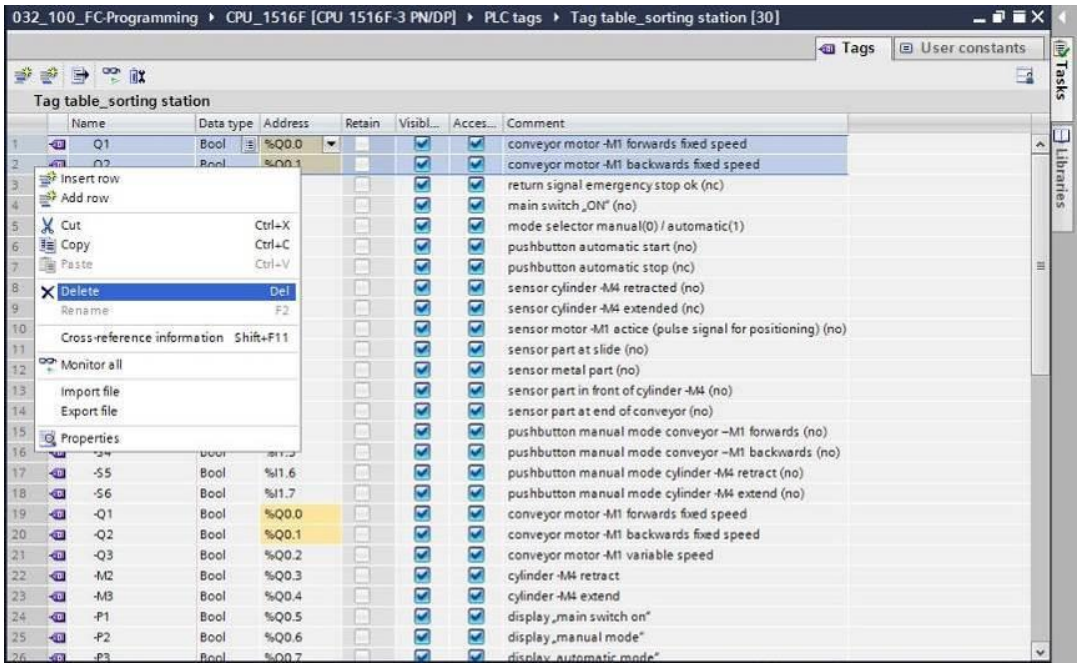
(→ SCE_EN_020-100_Tag_table_sorting_station… → Open)

→ 가져오기가 끝나면 선택 확인 창이 나타나는데, 여기에서 가져오기를 위한 로그 파일을 볼 수 있습니다. "OK"를 클릭합니다.

→ 일부 주소들이 주황색으로 강조 표시된 것을 볼 수 있습니다. 이들은 중복 주소로서, 혼란을 피하기 위해 연관된 태그의 이름에 자동으로 표시 됩니다.

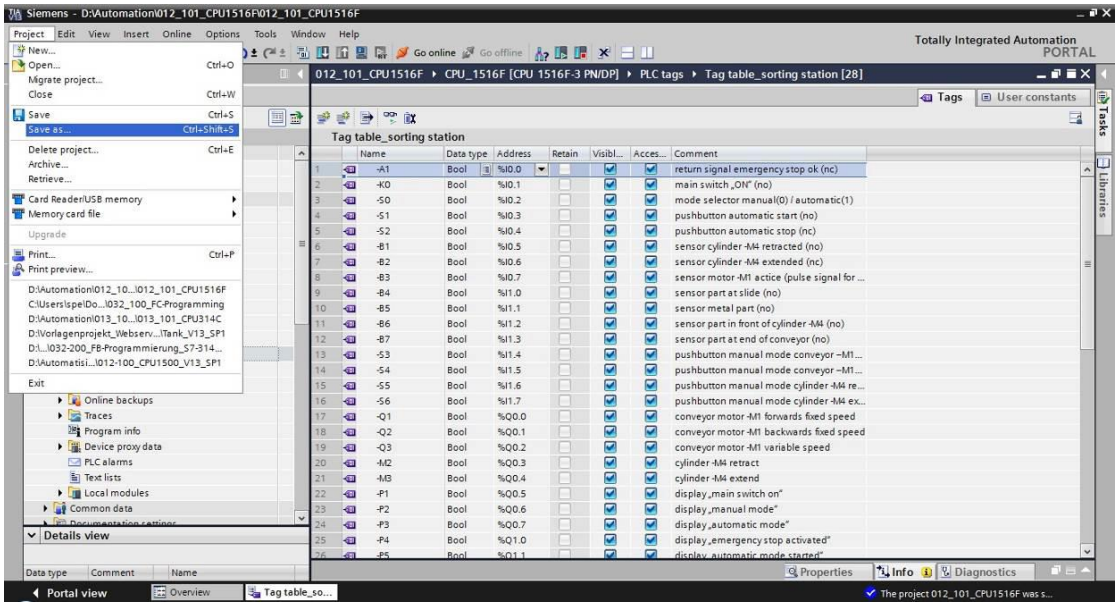
→ 라인을 선택하고 키보드에서 Del 키를 누르거나 단축 메뉴에서 "Delete"를 선택하여 중복 태그를 삭제합니다.

(→ 선택된 태그에서 마우스 오른쪽 버튼을 클릭 → Delete)



→ 디지털 입력 및 출력에 대한 완벽한 심볼 테이블이 화면에 나타납니다. 프로젝트를 032-200_FCProgramming라는 이름으로 저장을 합니다.

(→ Project → Save as … →032-200_FCProgramming → Save)

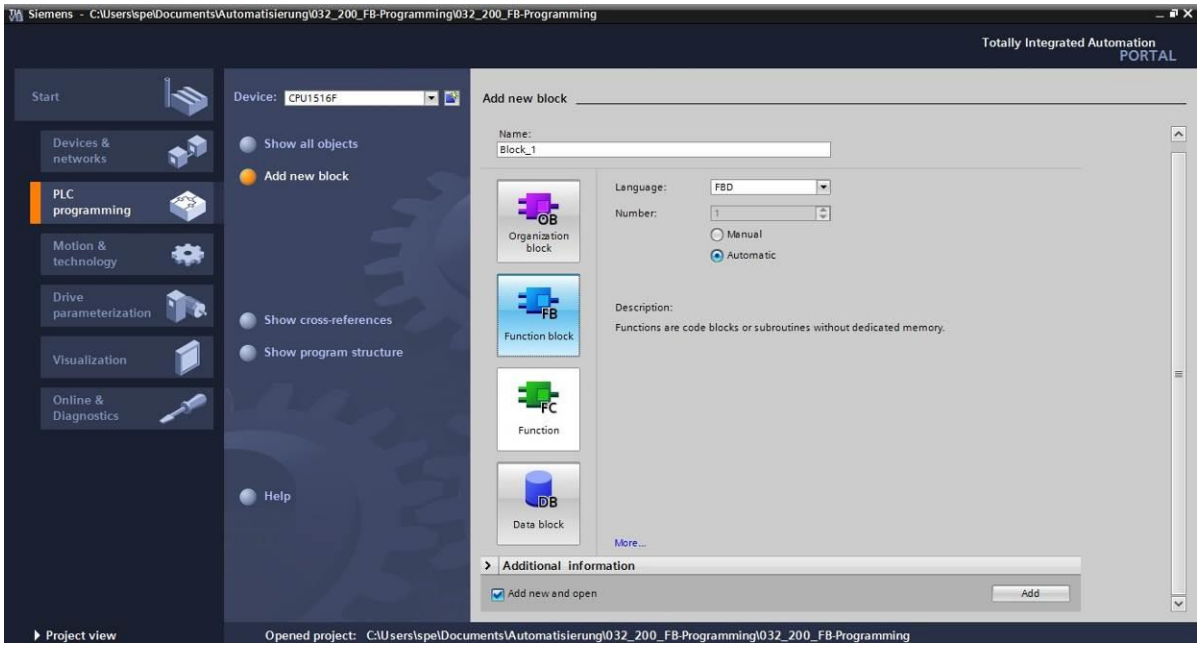


7.5 자동 모드의 컨베이어 모터를 위한 펑션 블록 FB1 "MOTOR_AUTO" 생성

→ 포털 뷰의 PLC 프로그래밍 섹션에서 "Add new block"를 클릭하여 새 펑션 블록을 생성합니다.

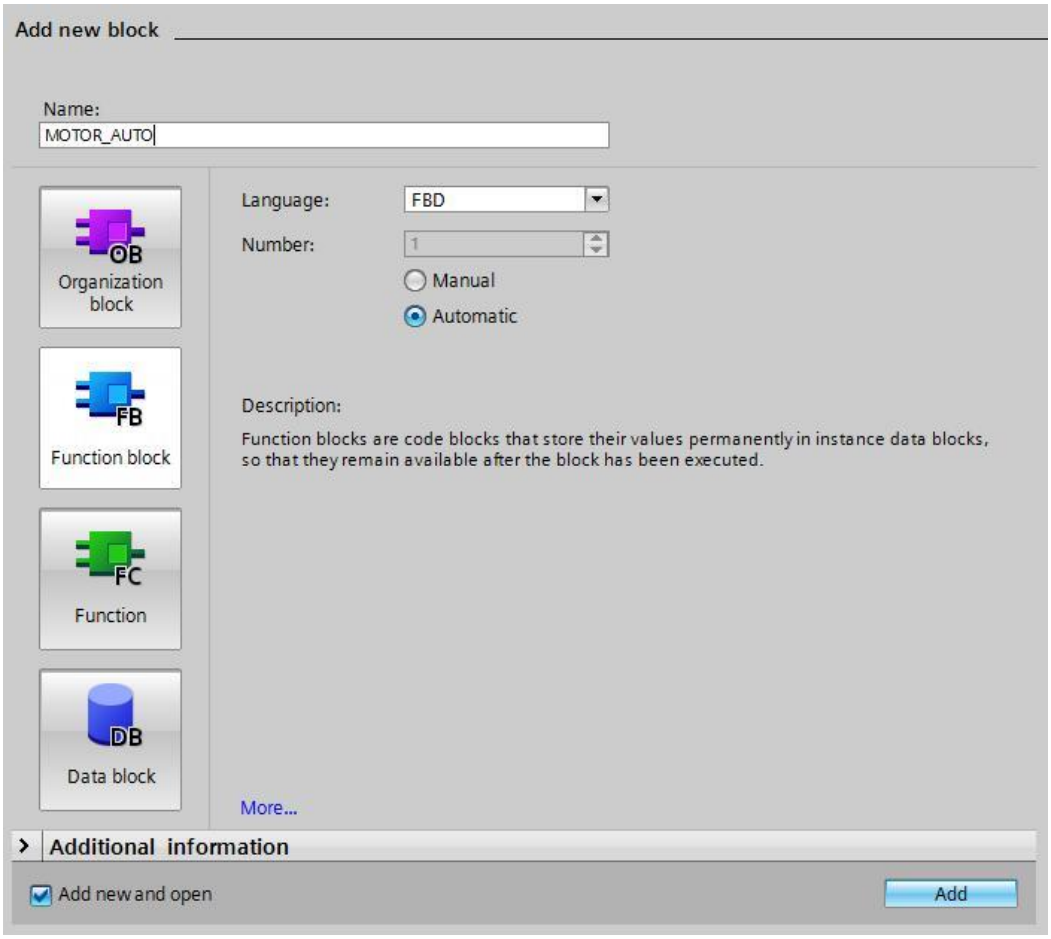
(→ PLC 프로그래밍 → Add new block → 

)



→ 새 블록의 이름을 "MOTOR_AUTO"로 변경하고 언어를 FBD로 설정한 다음, 자동 지정된 번호를 그대로 유지합니다. "Add new and open" 체크박스를 선택합니다. 프로젝트 뷰에서 생성된 펑션 블록으로 자동으로 이동됩니다. "Add"를 클릭합니다.

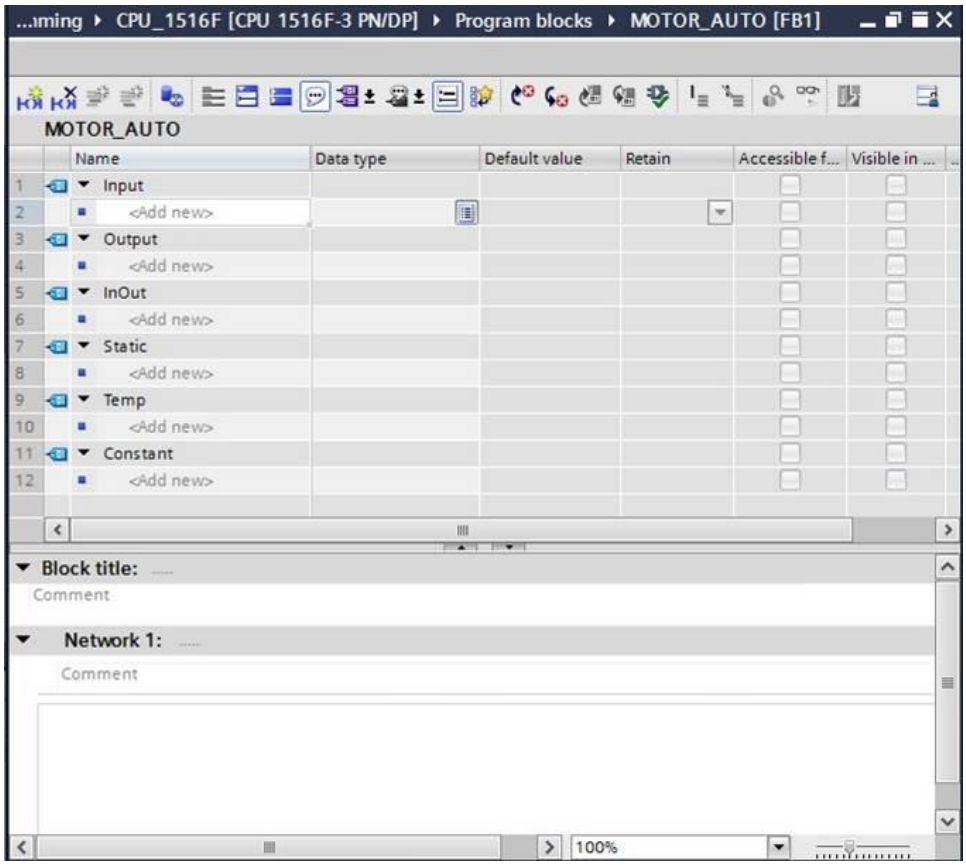
(→ Name: MOTOR_AUTO →Language: FBD → Number: Automatic → Add new and open → Add)



7.6 FB1 "MOTOR_AUTO"에 대한 인터페이스 정의

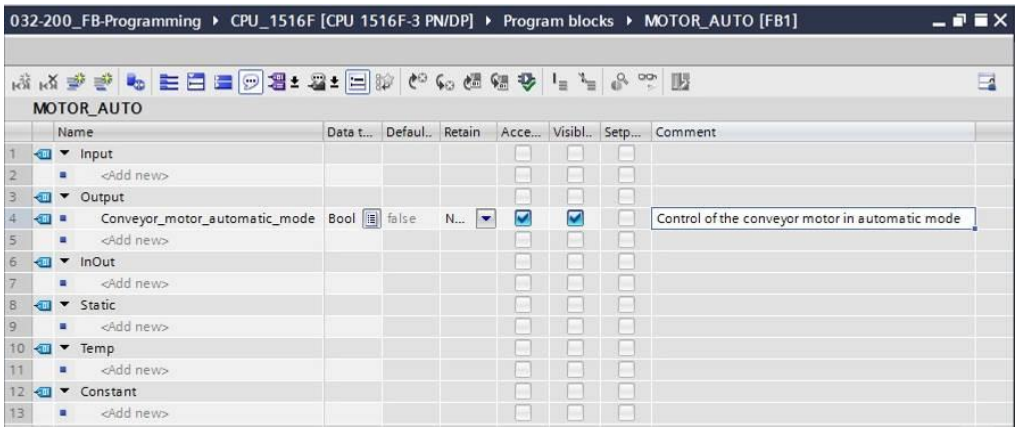
→ "Add new and open"를 선택하면 추가한 블록을 생성할 수 있도록 창에서 프로젝트 뷰가 열립니다.

→ 프로그래밍 뷰의 상단 부분에 펑션 블록에 대한 **인터페이스 목록(Interface description)**이 표시됩니다.



→ 컨베이어 모터를 제어하려면 바이너리 출력 신호가 필요합니다. 이러한 이유로 먼저 "Bool" 유형의 로컬 출력 태그 #Conveyor_motor_automatic_mode를 생성합니다. 파라미터에 대한 코멘트로 "Control of the conveyor motor in automatic mode"를 입력합니다.

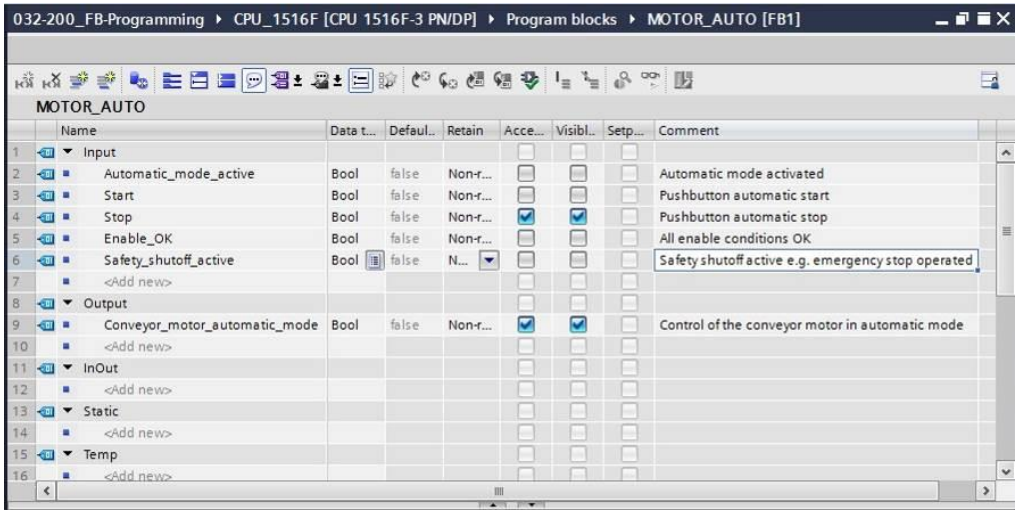
(→ Output: Conveyor_motor_automatic_mode → Bool → Control of the conveyor motor in automatic mode)



→ "Input"으로 가서 입력 인터페이스로서 파라미터 #Automatic_mode_active를 추가하고 Enter 키를 누르거나 입력 필드를 빠져나와서 입력을 확인합니다. 데이터 타입 "Bool"이 자동으로 지정이 되는데, 이를 그대로 유지합니다. 그 다음으로, 관련 코멘트 "Automatic mode activated"를 입력합니다.

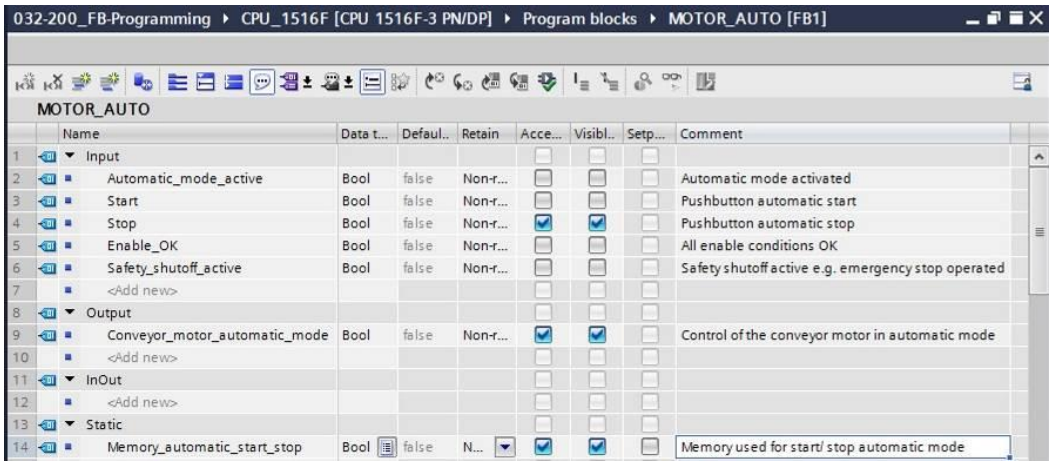
(→ Automatic_mode_active → Bool → Automatic mode activated)

→ "Input"으로 가서 추가 바이너리 입력 파라미터로서 #Start, #Stop, #Enable_OK 및 #Safety_shutoff_active를 추가하고 이들의 데이터 타입을 확인합니다. 설명 코멘트를 추가합니다.



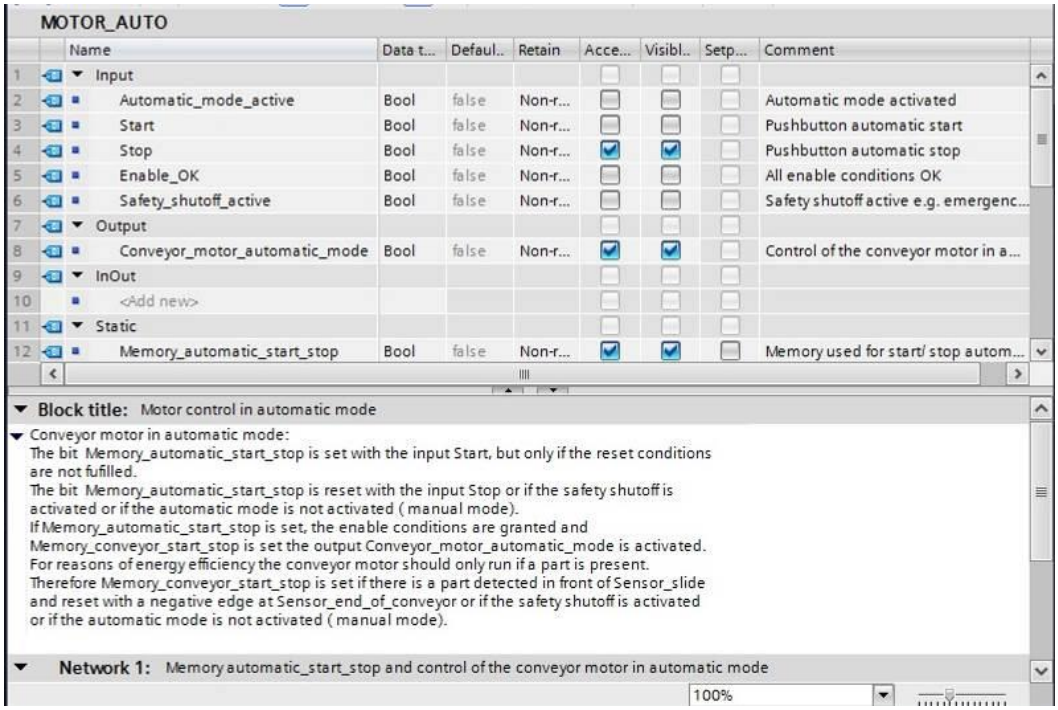
컨베이어는 푸시버튼을 통해 시작 및 정지됩니다. 따라서 메모리로서 "Static" 태그가 필요합니다. → "Static"으로 가서 태그 #Memory_automatic_start_stop을 추가하고 Enter 키를 누르거나 입력 필드를 빠져나와서 입력을 확정합니다. 데이터 타입 "Bool"이 자동으로 지정이 되는데, 이를 그대로 유지합니다. 코멘트 "Memory used for start/stop automatic mode"를 입력합니다.

(→ Memory_automatic_start_stop → Bool → Memory used for start/stop automatic mode)



→ 프로그램 문서화를 위해 블록 타이틀, 블록 코멘트, 네트워크 1를 표시할만한 네트워크 타이틀을 지정합니다.

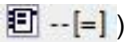
(→ Block title: Motor control in automatic mode → Network 1: Memory_automatic_start_stop and control of the conveyor motor in automatic mode)

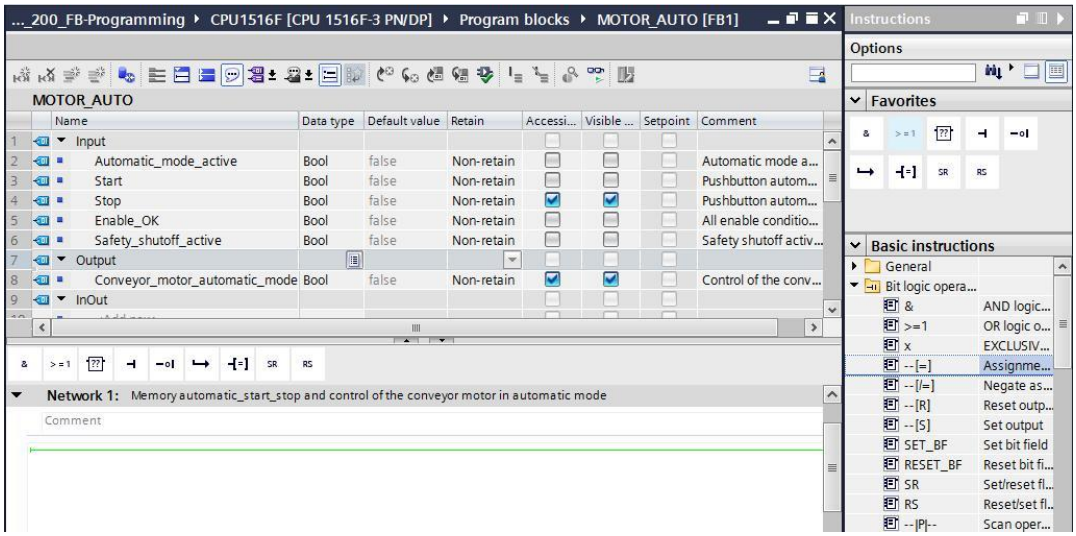


7.7 FB1: MOTOR_AUTO 프로그래밍

→ **인터페이스 목록**으로 가면 프로그래밍 창에 여러가지 논리 펑션을 포함한 도구 모음이 보이고, 그 아래 네트워크 영역이 있습니다. 블록 타이틀과 첫 번째 네트워크에 대한 타이틀은 앞서 이미 지정해 두었습니다. 개별 로직 블록을 이용해 네트워크 내에서 프로그래밍이 수행됩니다. 여러 네트워크로 나누어지기 때문에 프로그램의 명확성을 지킬 수 있습니다. 아래에서는 로직 블록을 삽입할 수 있는 몇가지 방법에 대해 알아보겠습니다.

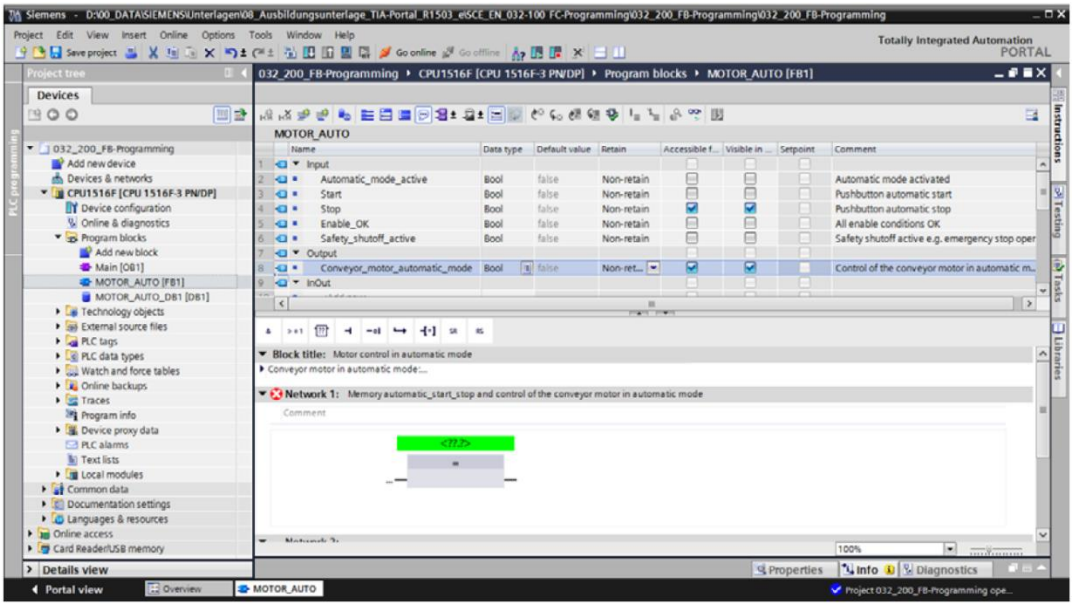
→ 프로그래밍 창의 오른쪽에 프로그램에서 사용할 수 있는 명령어 목록이 나타날 것입니다. "Basic instructions" 아래의 "Bit logic operations"로 가서 펑션 –[=] (지정)를 찾고 끌어다 놓기 기능을 이용해 네트워크 1로 이를 이동시킵니다 (녹색 선이 나타나고 마우스 포인터에 + 심볼이 표시됨).

(→ Instructions → Basic instructions → Bit logic operations → )

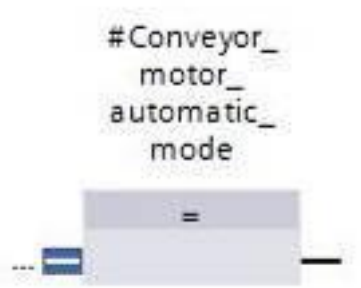


→ 이제 끌어다 놓기 기능을 이용해 출력 파라미터 #Conveyor_motor_automatic_mode를 방금 삽입한 블록 위의 <??.?>로 이동시킵니다. **인터페이스 목록**에서 파라미터를 선택하는 가장 좋은 방법은 파란색 심볼  을 클릭하여 드래그 하는 것입니다.

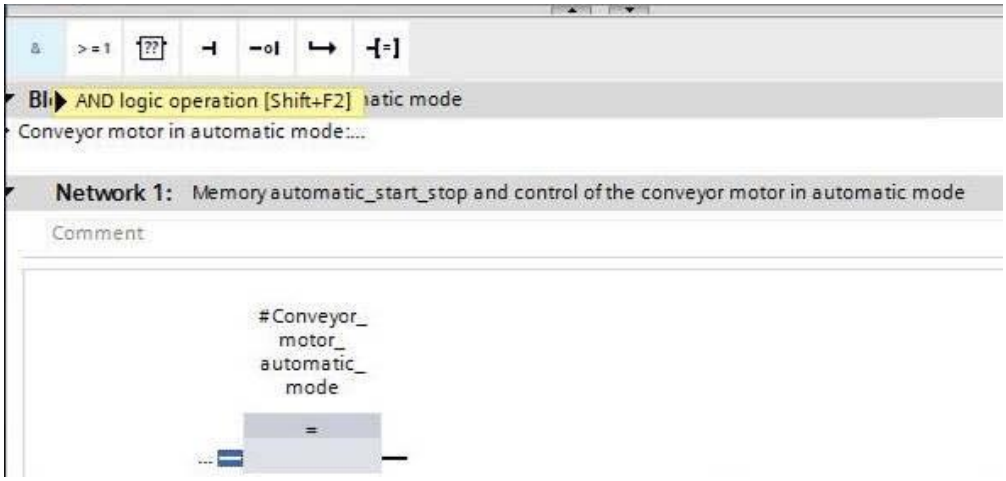
(→  Conveyor_motor_automatic_mode)



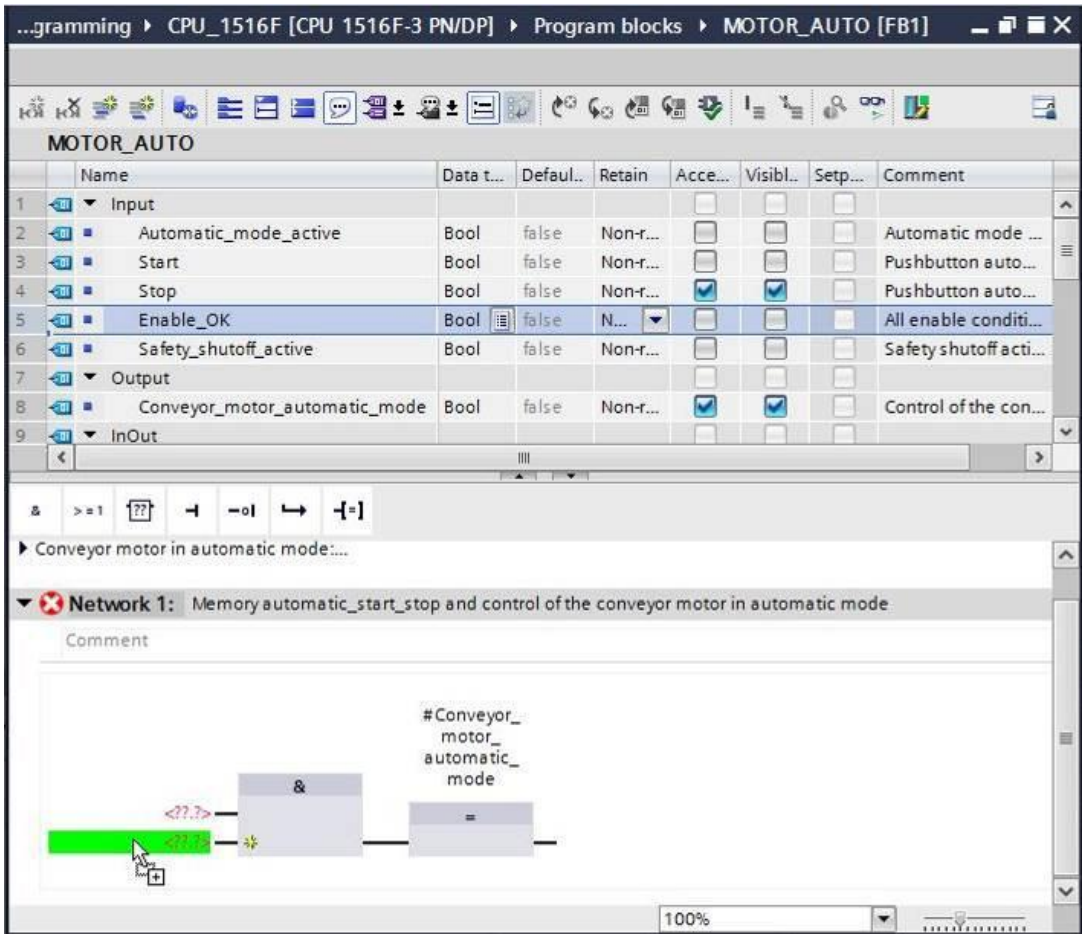
→ 이 블록에 의해 #Conveyor_motor_automatic_mode 파라미터에 값이 기록(write)됩니다. 그러나 입력 조건이 아직 정의되지 않았기 때문에 오류가 발생해 있습니다. 해당 블록의 입력에 AND 논리 연산을 사용하여 SR flip-flop과 #Enable_OK 파라미터를 논리적으로 결합해야 합니다. 이를 위해 먼저, 블록의 입력을 클릭해서 입력 라인이 파란색 백그라운드가 되도록 합니다.



→ 로직 도구 모음의 & 아이콘을 클릭해서 해당 블록 앞에 AND 논리 연산을 삽입합니다.

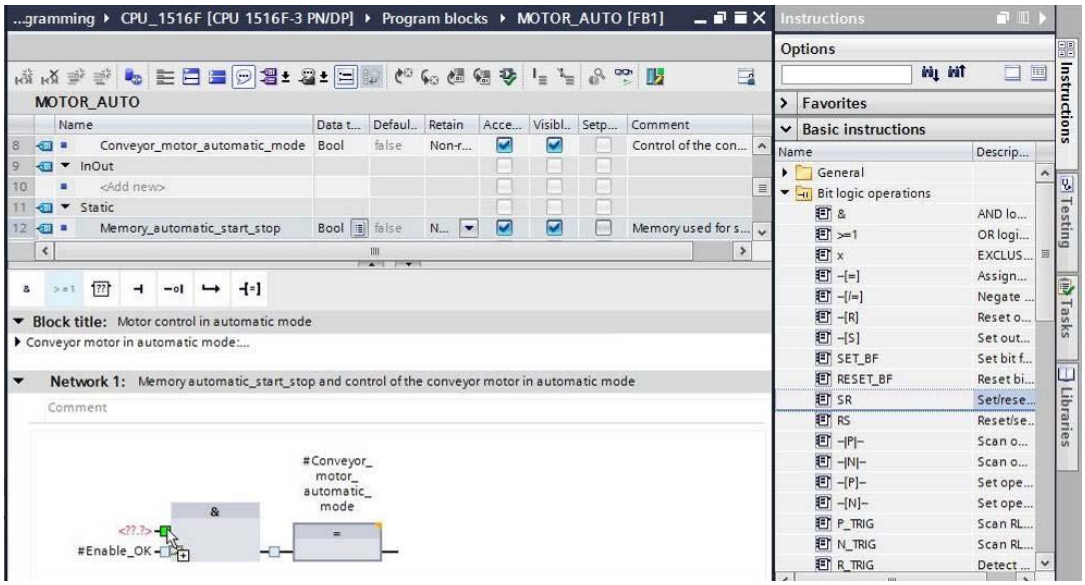


→ 끌어다 놓기 기능을 사용해 입력 파라미터 #Enable_OK를 AND 논리 연산의 두 번째 입력 <??.?>으로 이동시킵니다. (→ Enable_OK)

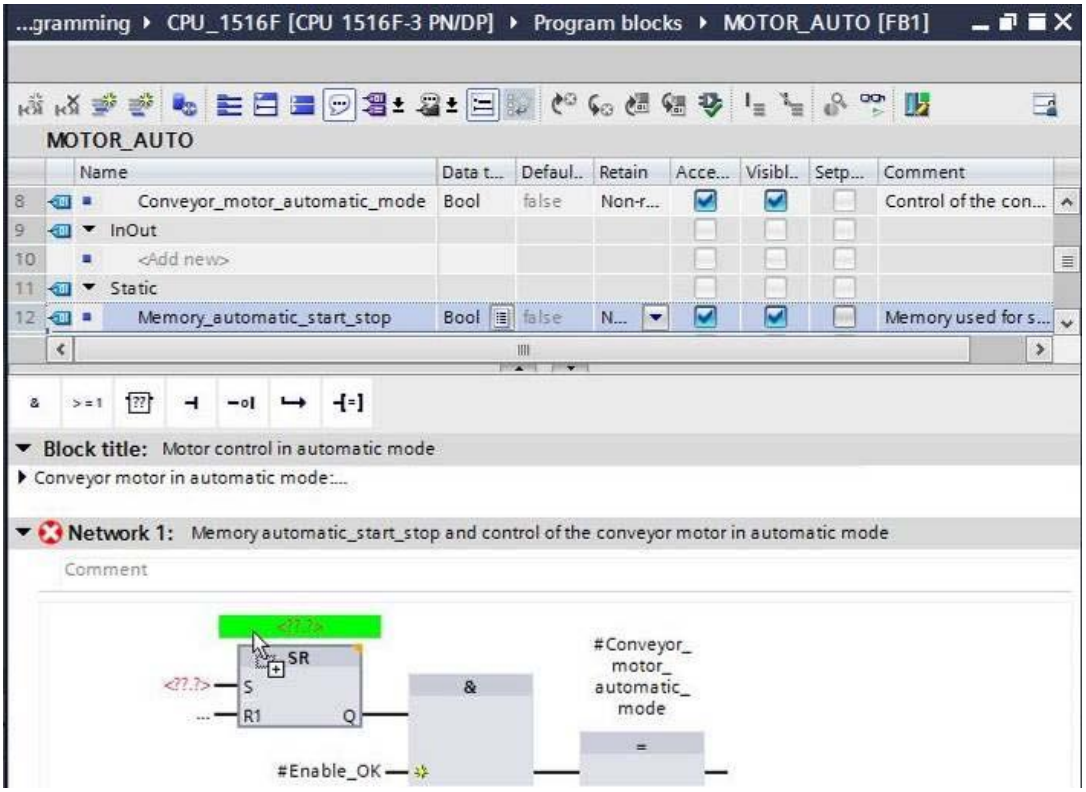


→ 끌어다 놓기 기능을 사용하여 "Basic instruction" 아래의 "Bit logic operations"에 있는 명령어 목록에서 Set/reset flip-flop 펑션 SR 를 선택해 AND 논리 연산의 첫 번째 입력 으로 이동시킵니다.

(→ Instructions → Basic instructions → Bit logic operations → SR →)

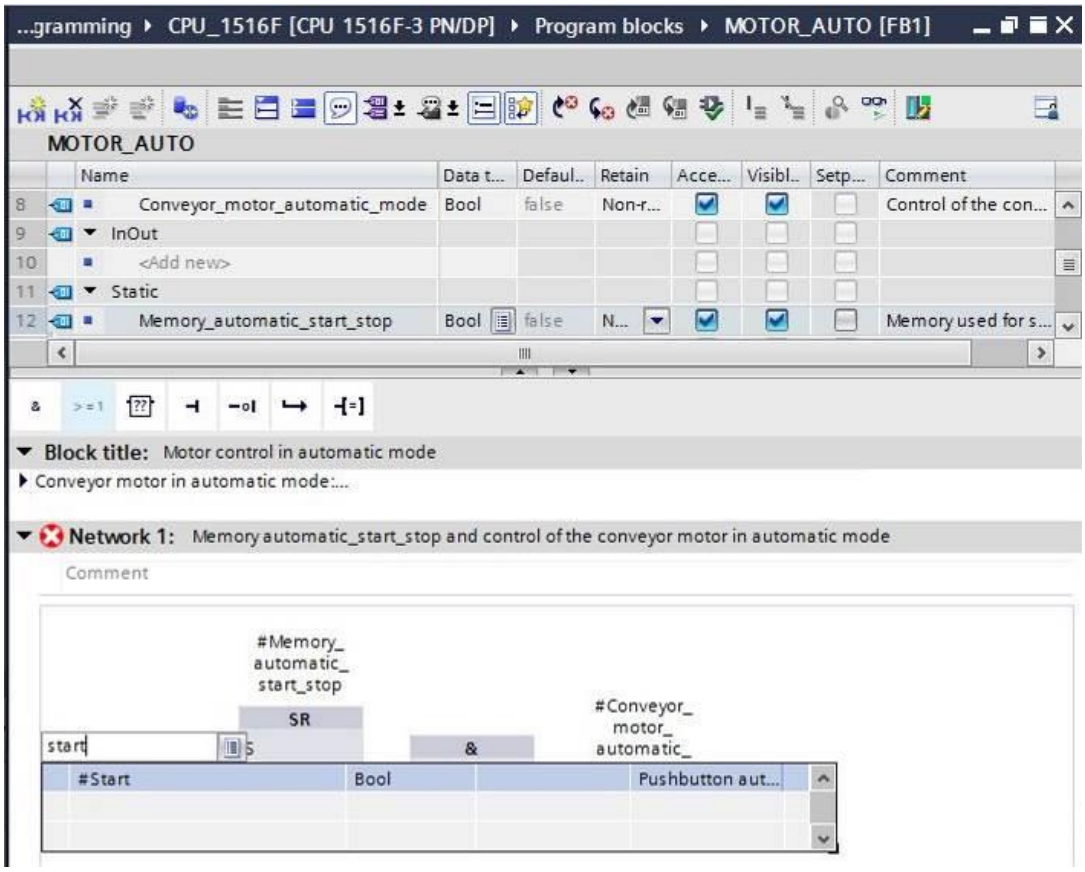


→ SR flip-flop을 사용하려면 메모리 태그가 필요합니다. 이를 위해 끌어다 놓기 기능을 이용해 정적 파라미터 #Memory_automatic_start_stop를 SR flip-flop 위의 <??.?>로 이동시킵니다. (→ Memory_automatic_start_stop)



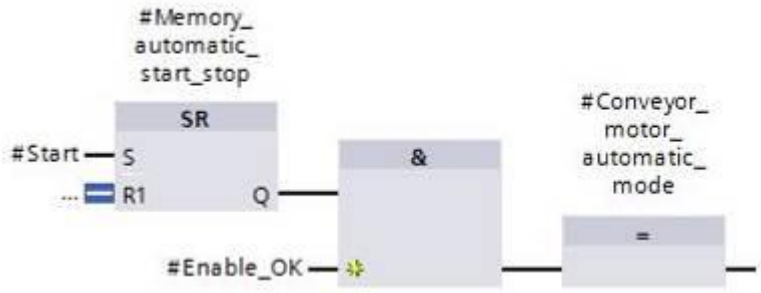
→ 입력 태그 #Start에 의해 #Memory_automatic_start_stop가 설정(set)됩니다. SR flip-flop의 S 입력 <??.?>을 두 번 클릭하고 이때 나타난 필드에 "Start"를 입력하면 "Start"로 시작되는 가용 태그 목록이 나타납니다. #Start 태그를 클릭하고 Enter 키를 눌러 이를 적용합니다.

(→ SR flip-flop → <??.?> → Start 입력 → #Start → Enter)

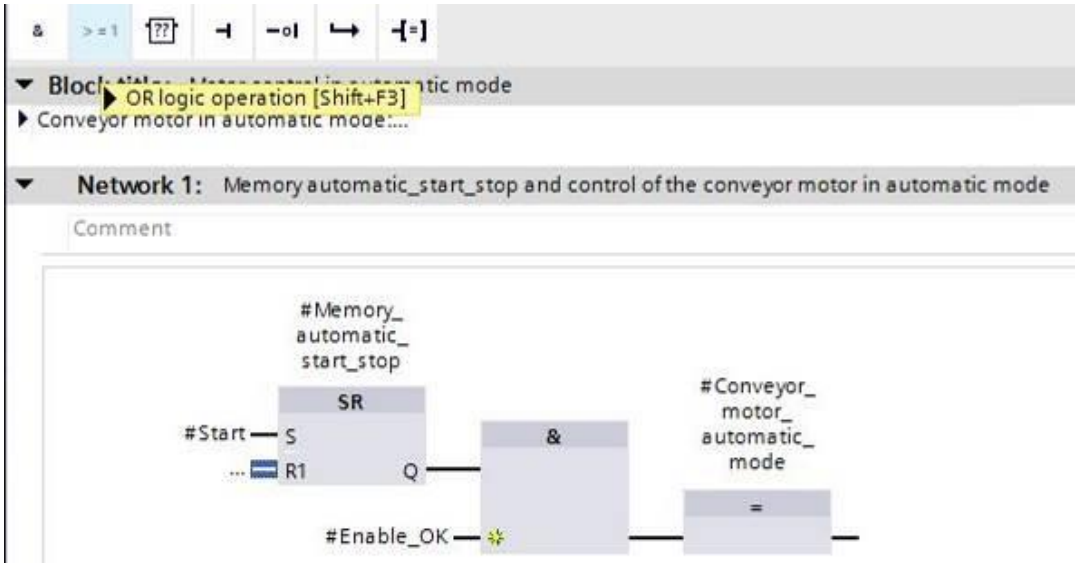


참고: 이런 방식으로 태그를 지정하면 태그 테이블에서 글로벌 태그를 혼동할 위험이 있습니다. 따라서 앞서 인터페이스 목록에서 끌어다 놓기 기능을 이용해 표현된 절차를 우선적으로 사용해야 합니다.

→ 여러 개의 조건을 통해 컨베이어 작동을 정지시킬 수 있습니다. 따라서 SR flip-flop의 R1 입력에서 OR 블록이 필요합니다. 먼저 SR flip-flop의 R1 입력을 클릭해서 입력 라인이 파란색 백그라운드가 되도록 합니다.

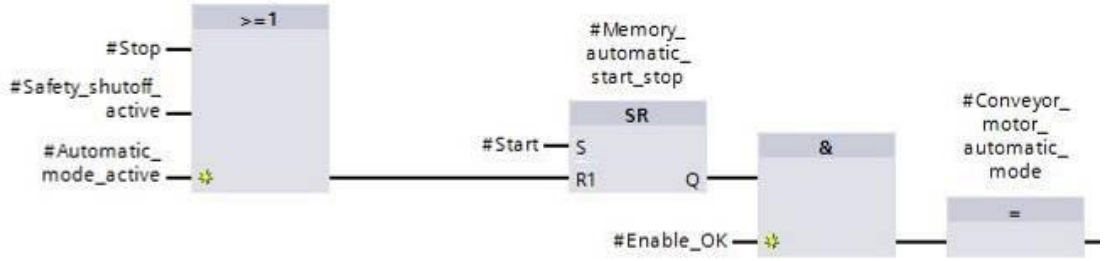


→ 로직 도구 모음의 >=1 아이콘을 클릭해서 OR 논리 연산을 삽입합니다.

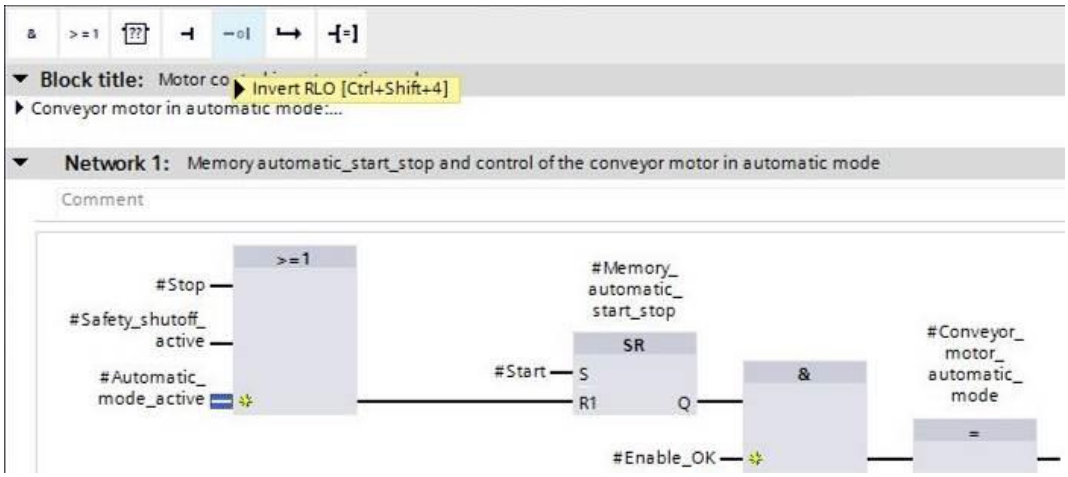


→ 처음에 OR 블록은 2개의 입력을 가지고 있습니다. 추가 입력 태그를 논리적으로 결합하려면 OR 블록의 노란색 별 표시 를 클릭합니다.

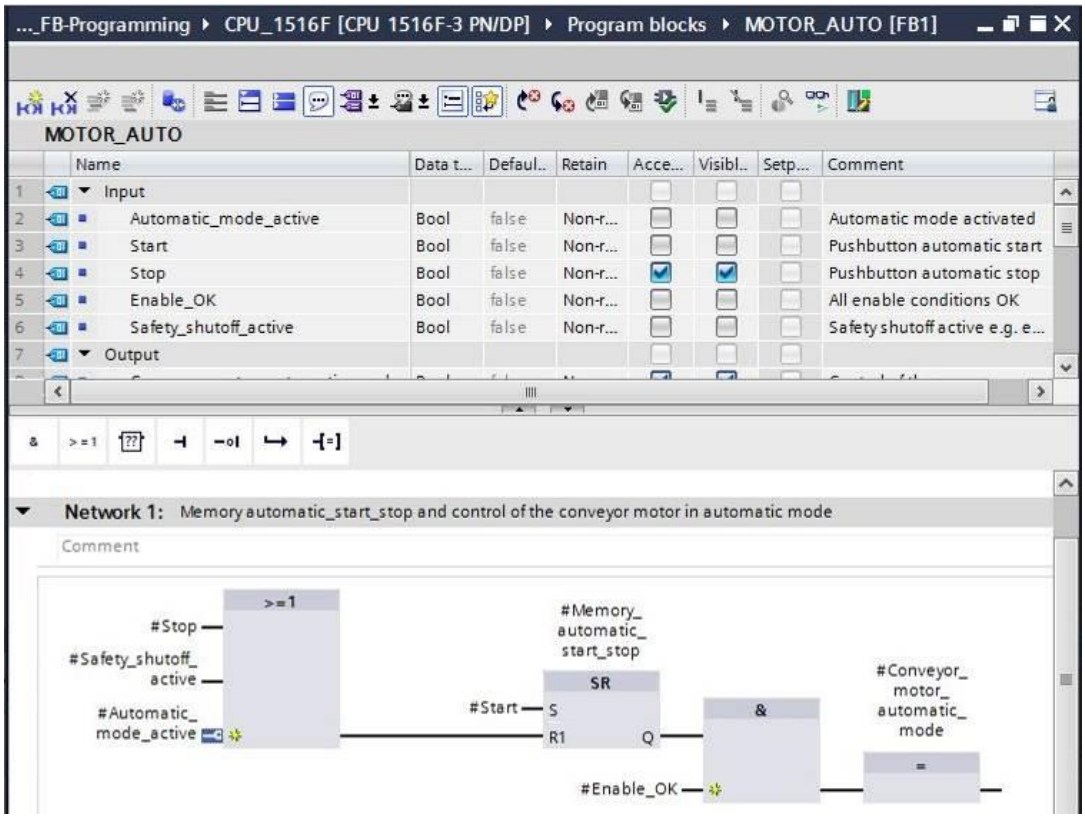
→ 입력 태그 #Stop, #Safety_shutoff_active 및 #Automatic_mode_active를 OR 블록의 3개 입력에 추가합니다.



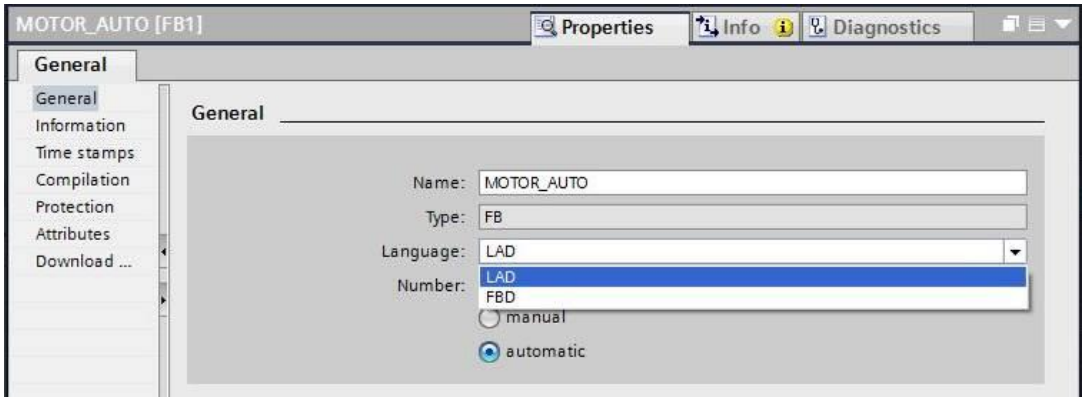
→ 파라미터 #Automatic_mode_active를 선택하고 -o| 를 클릭하면 이에 연결된 입력이 부정화(Not)됩니다.



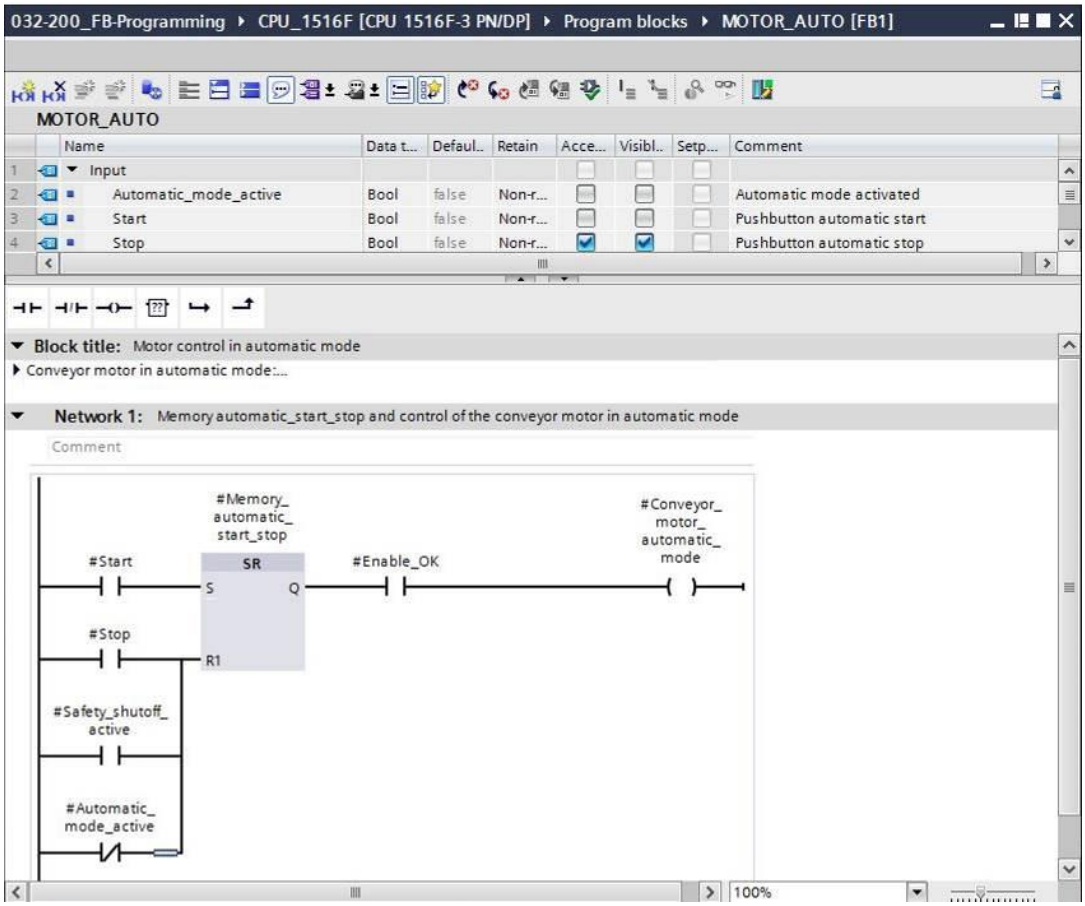
→ Save project 를 클릭하여 잊지 말고 프로젝트를 저장합니다. FBD에서 완료된 펑션 "MOTOR_AUTO" [FB1]가 아래와 같이 나타납니다.



→ 블록 속성의 "General"로 가서 언어를 래더 로직(LAD)으로 변경할 수 있습니다.
(→ Properties → General → Language: LAD)



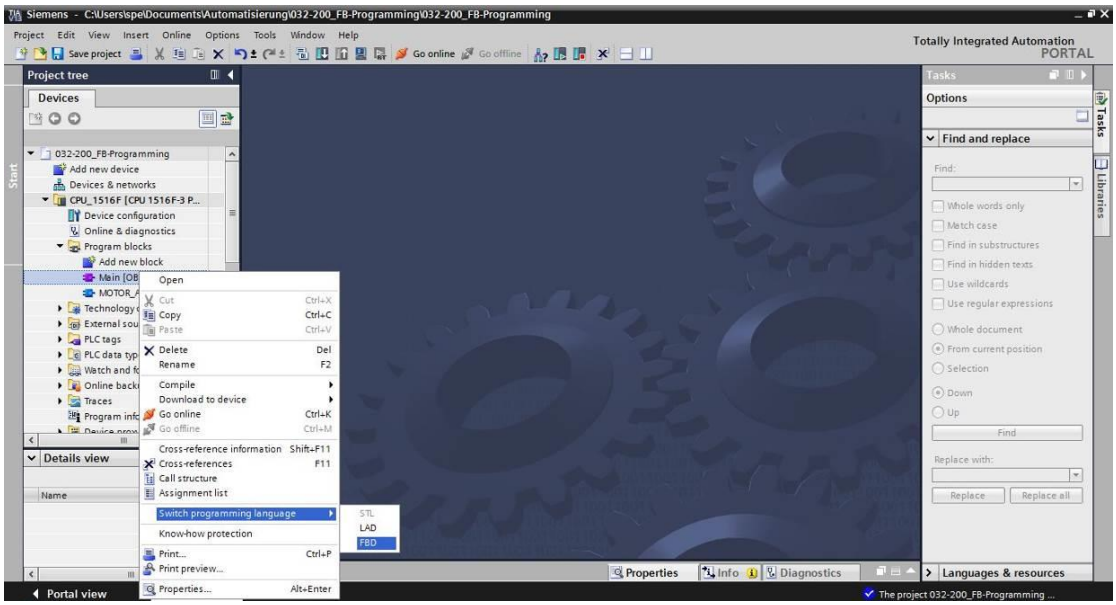
→ 이 프로그램은 LAD로 다음과 같이 나타납니다.



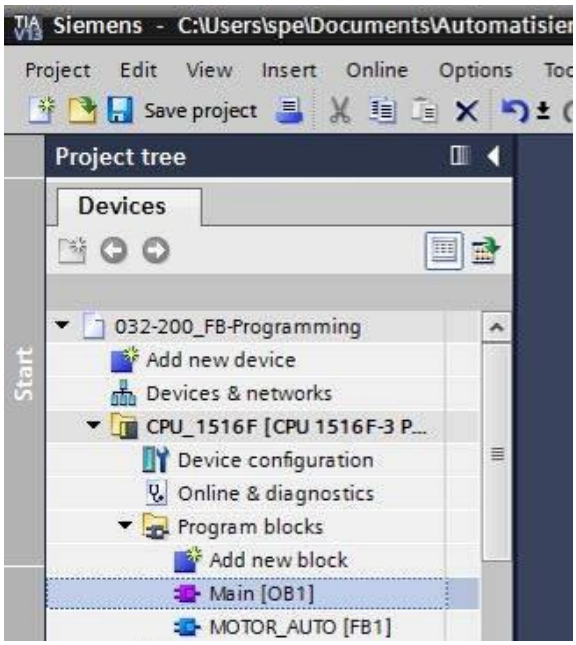
7.8 오거나이제이션 블록 OB1 프로그래밍 – 자동 모드에서 벨트 전진 트래킹 제어

→ 오거나이제이션 블록 "Main [OB1]"을 프로그래밍하기 전에 펑션 블록 다이어그램(FBD)으로 프로그래밍 언어를 전환해야 합니다. 이를 위해 먼저, "Program blocks" 폴더에서 "Main [OB1]"를 클릭합니다.

(→ CPU_1516F[CPU 1516F-3 PN/DP → Program blocks → Main [OB1] → Switch programming language → FBD)

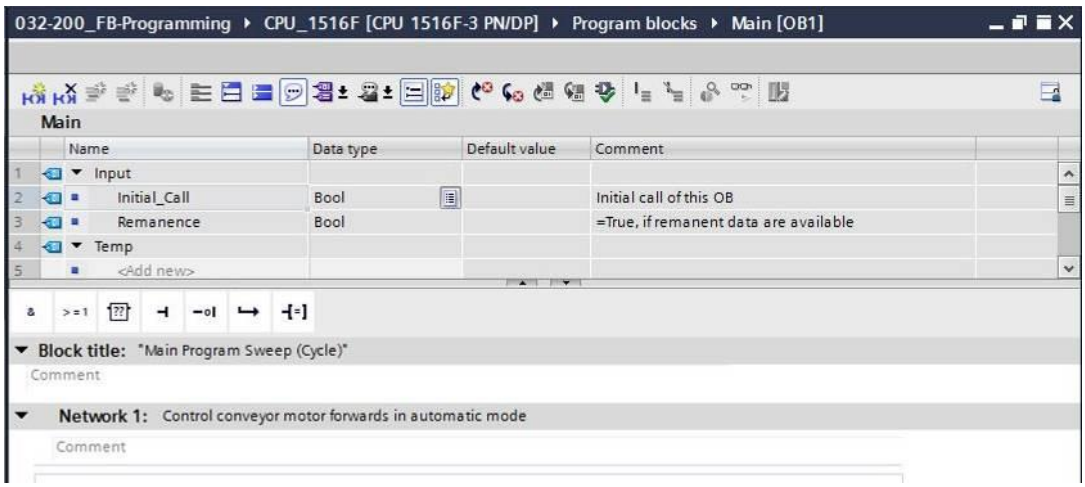


→ "Main [OB1]" 오거나이제이션 블록을 더블클릭해서 엽니다.

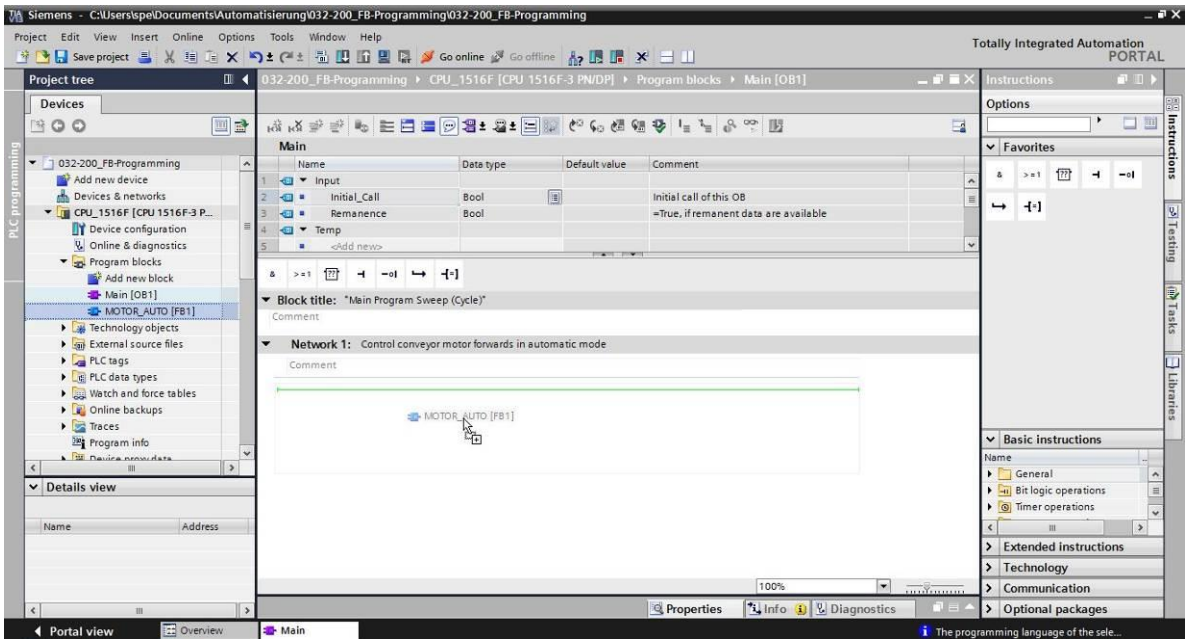


→ 네트워크 1의 이름을 "Control conveyor tracking forward in automatic mode"라고 입력합니다.

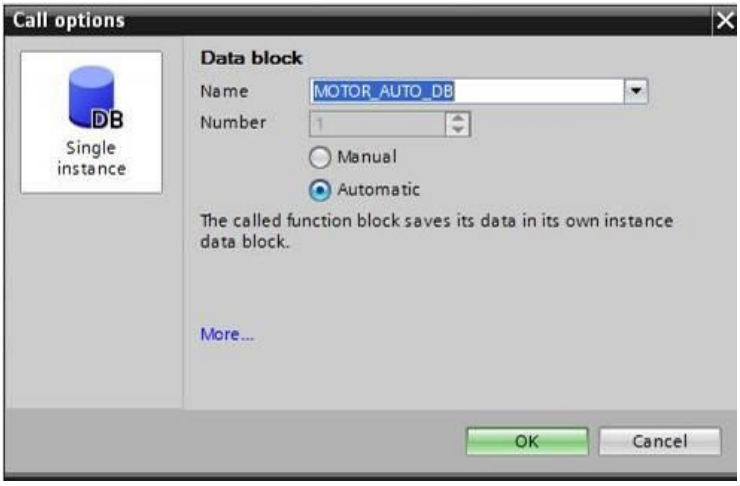
(→ Network 1:... → Control conveyor tracking forward in automatic mode)



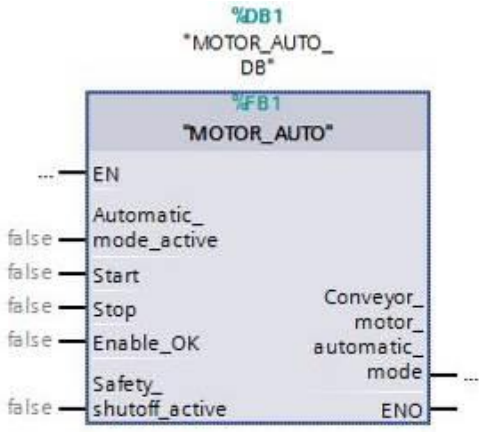
→ 끌어다 놓기 기능을 이용해 "MOTOR_AUTO [FB1]" 펑션 블록을 네트워크 1의 녹색 라인으로 이동시킵니다.



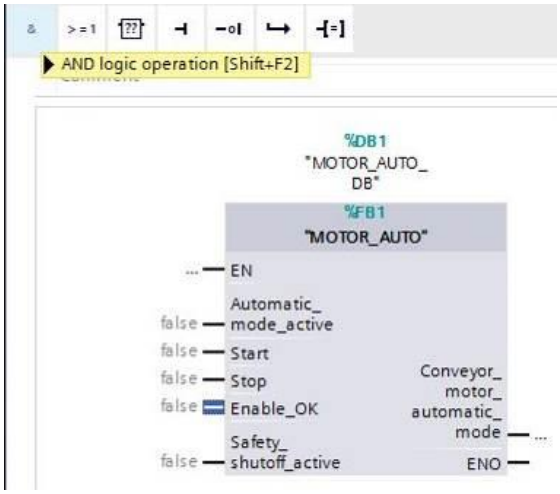
→ FB1 호출을 위한 인스턴스 데이터 블록이 자동으로 생성됩니다. 이름을 지정하고 OK를 클릭해 이를 적용합니다. (→ MOTOR_AUTO_DB → OK)



→ 앞서 정의한 인터페이스와 인스턴스 데이터 블록, EN 및 ENO를 포함한 연결 블록이 네트워크 1에 삽입됩니다.



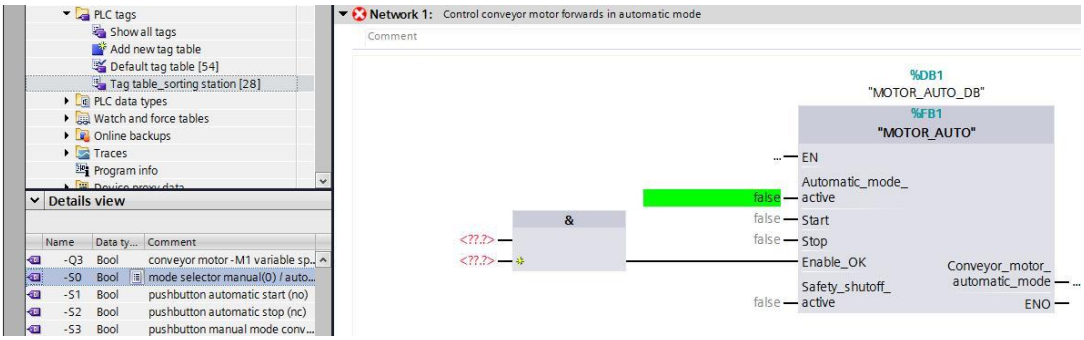
→ 입력 파라미터 "Enable_OK" 앞에 AND를 삽입하려면 먼저 입력 파라미터를 선택하고 로직 도구 모음의 아이콘 & 을 클릭해서 AND를 삽입합니다. (→ &).



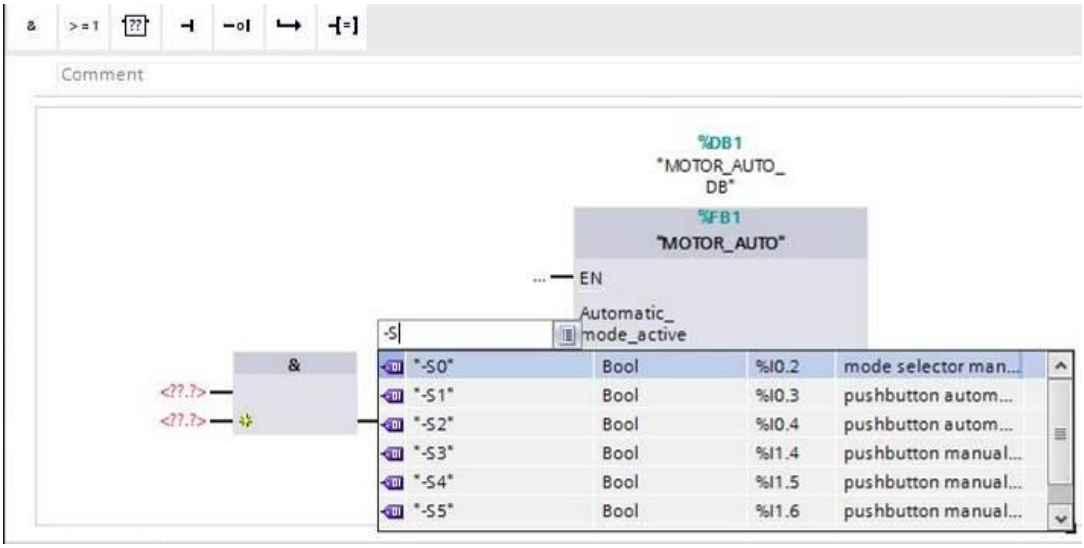
→ "Tag_table_sorting_station"에서 원하는 글로벌 태그에 블록을 연결하는 방법은 두 가지가 있습니다.

→ 프로젝트 트리에서 "Tag_table_sorting_station"을 선택하거나, 끌어다 놓기 기능을 이용해 Detail view에서 원하는 글로벌 태그를 FB1의 인터페이스로 이동시키는 방법 입니다.

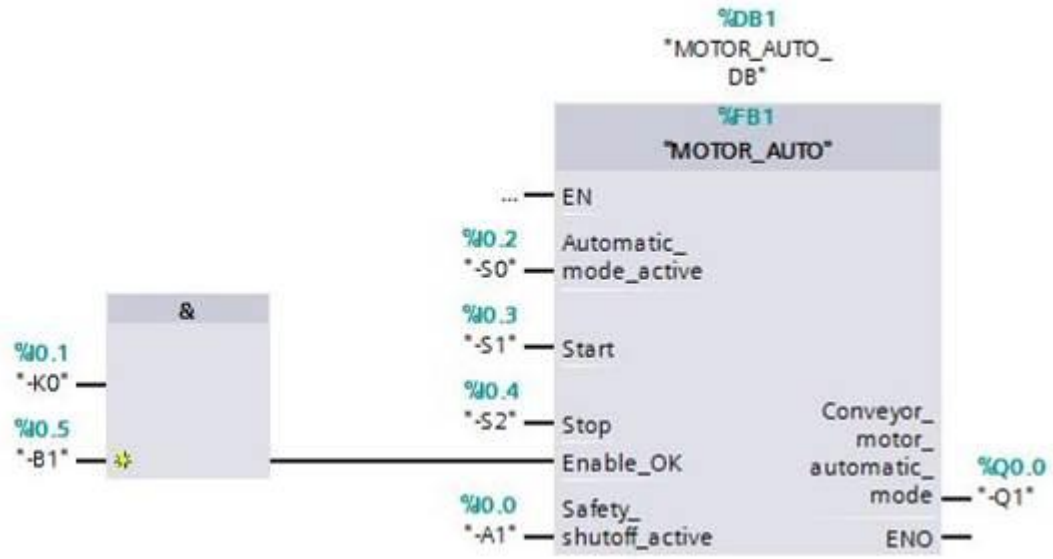
(→ Tag_table_sorting_station → Detail view → S0 → Automatic_mode_active)



→ 또는 <??.?>에 대해 원하는 글로벌 태그의 시작 문자 (예: "S")를 입력하고 화면에 나타난 목록에서 글로벌 입력 태그 "-S0" (%I0.2)를 선택합니다. (→ Automatic_mode_active → S 입력→ - S0 선택)

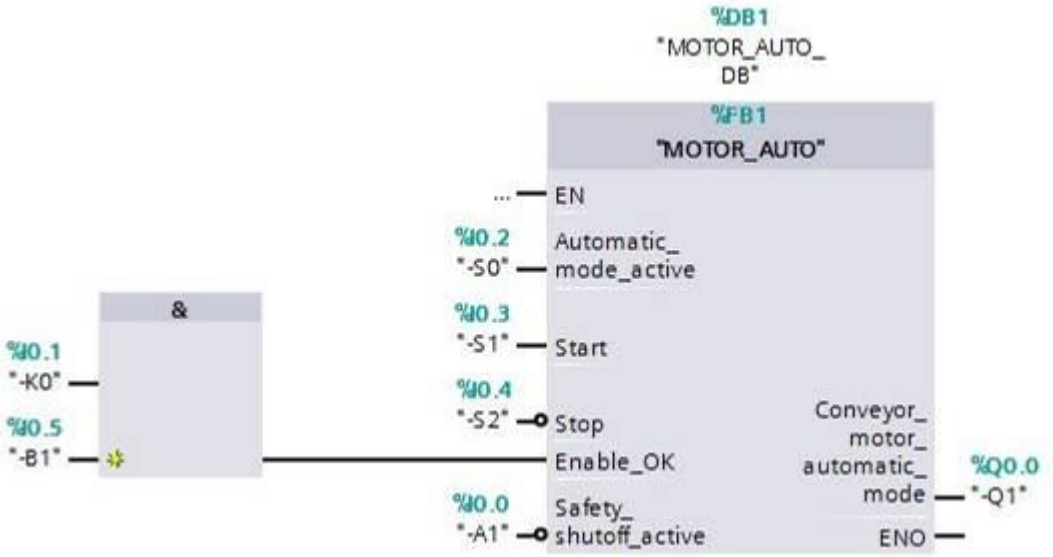


→ 나머지 입력 태그 "-S1", "-S2", "-K0", "-B1" 및 "-A1"을 삽입하고 출력 파라미터 "Conveyor_motor_automatic_mode"에 출력 태그 "-Q1" (%Q0.0)를 삽입합니다.

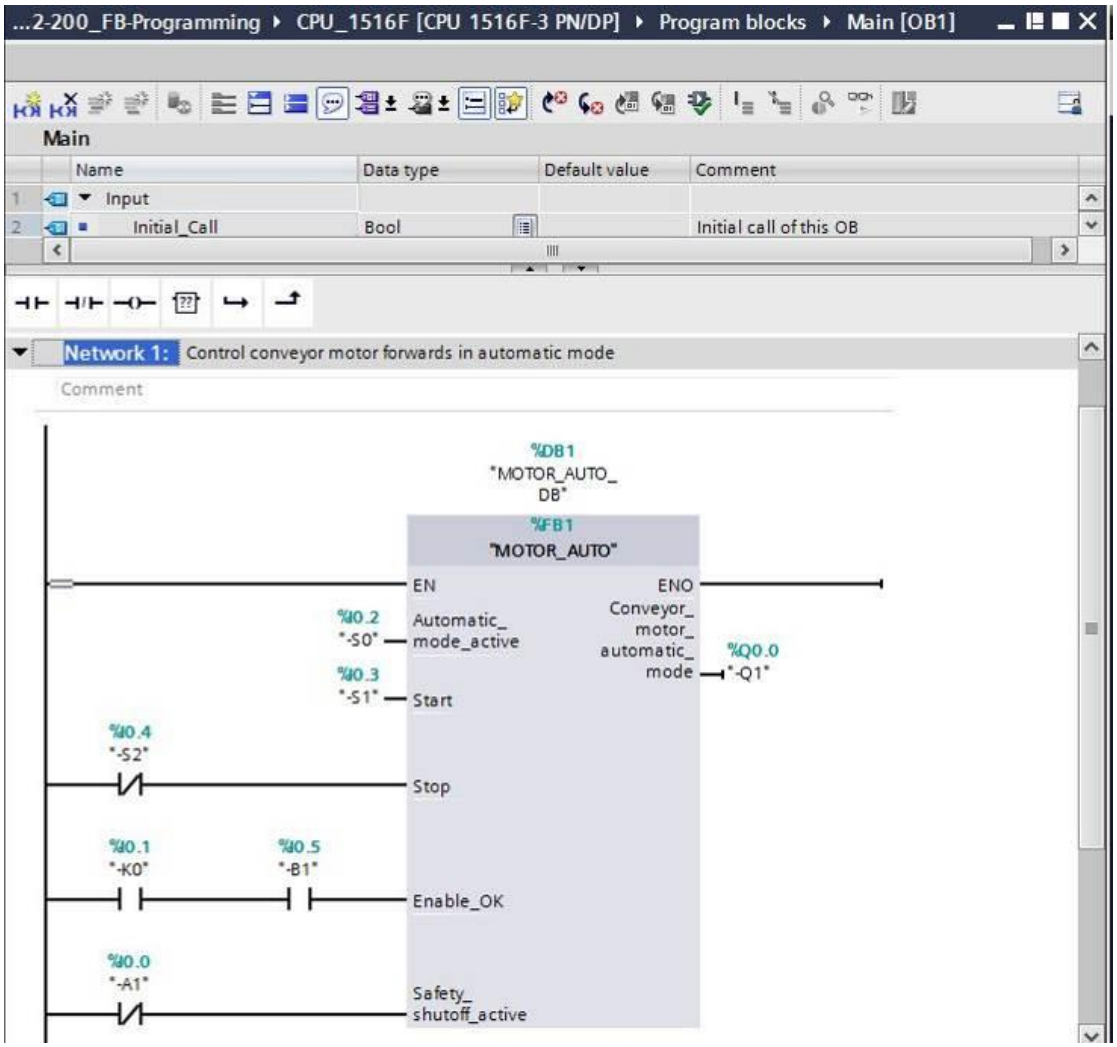


→ 입력 태그 "-S2" 및 "-A1"을 선택하고  를 클릭하면 이들에 대한 쿼리가 부정화됩니다.

(→ -S2 →  → -A1 → )



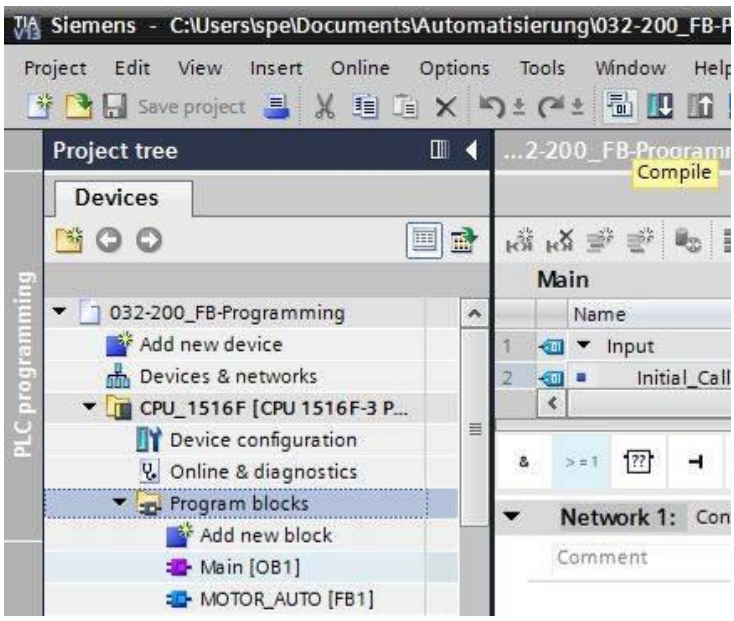
7.9 LAD (Ladder Logic) 프로그래밍 언어를 사용한 작성 결과물



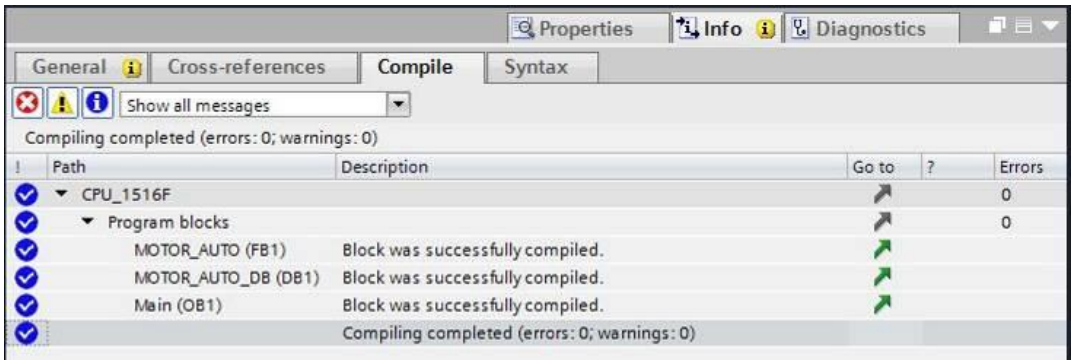
7.10 프로그램 저장 및 컴파일

→ 프로젝트를 저장하려면 메뉴에서 Save project 버튼을 선택합니다. 모든 블록을 컴파일하려면 "Program blocks 폴더를 클릭하고 메뉴에서 컴파일을 위한 아이콘 을 선택합니다.

(→ Save project → Program blocks →).



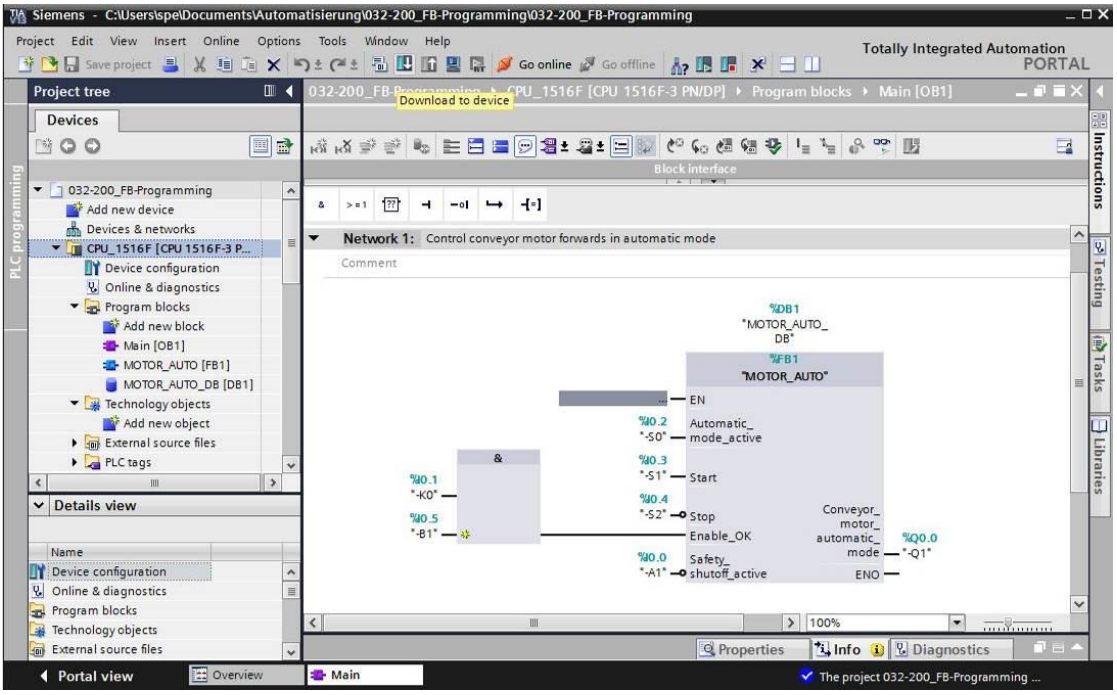
→ "info" 아래의 "Compile" 영역에 블록이 성공적으로 컴파일이 되었는지 나타납니다.



7.11 프로그램 다운로드

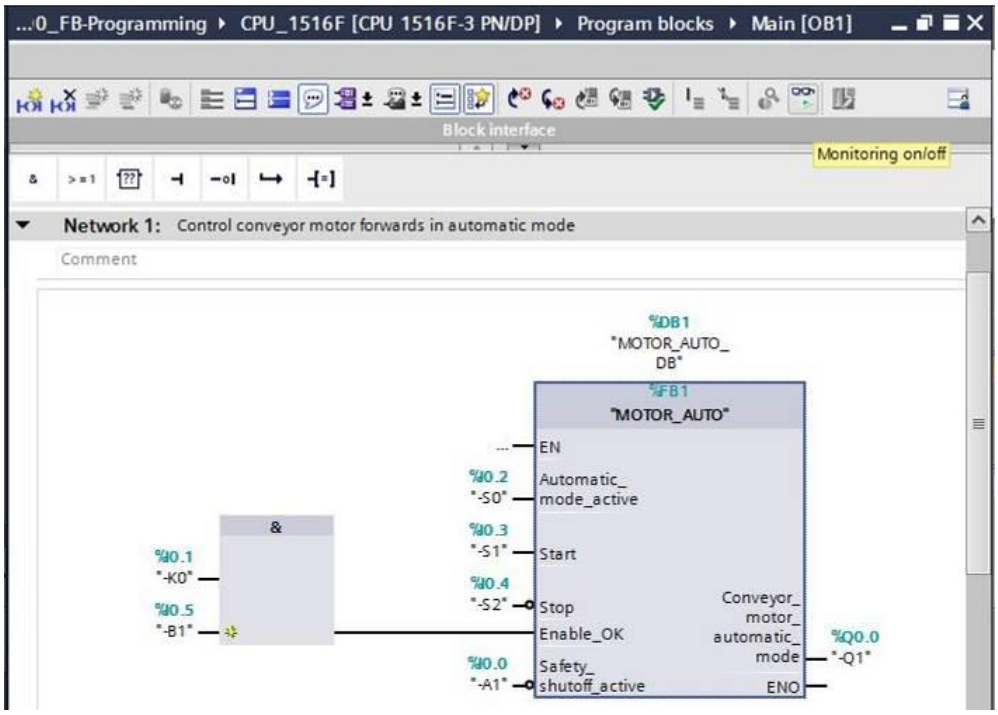
→ 컴파일이 성공적으로 완료되고 나면 앞서 설명한 하드웨어 구성을 위한 모듈에와 같이, 생성된 프로그램과 함께 전체 컨트롤러를 다운로드할 수 있습니다.

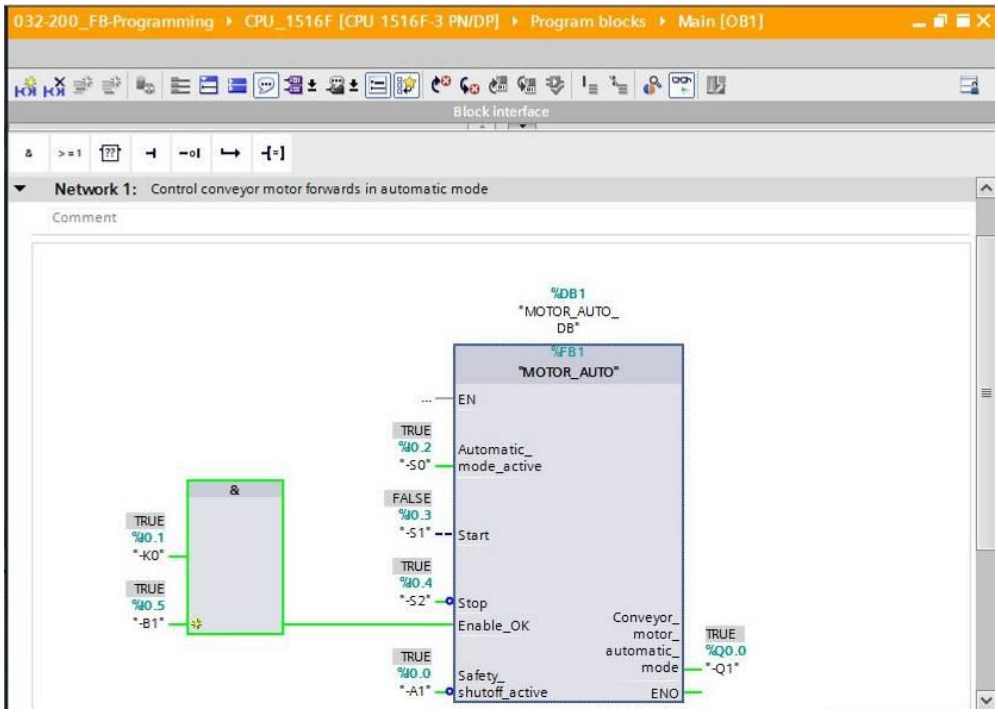
(→ ).



7.12 프로그램 블록 모니터링

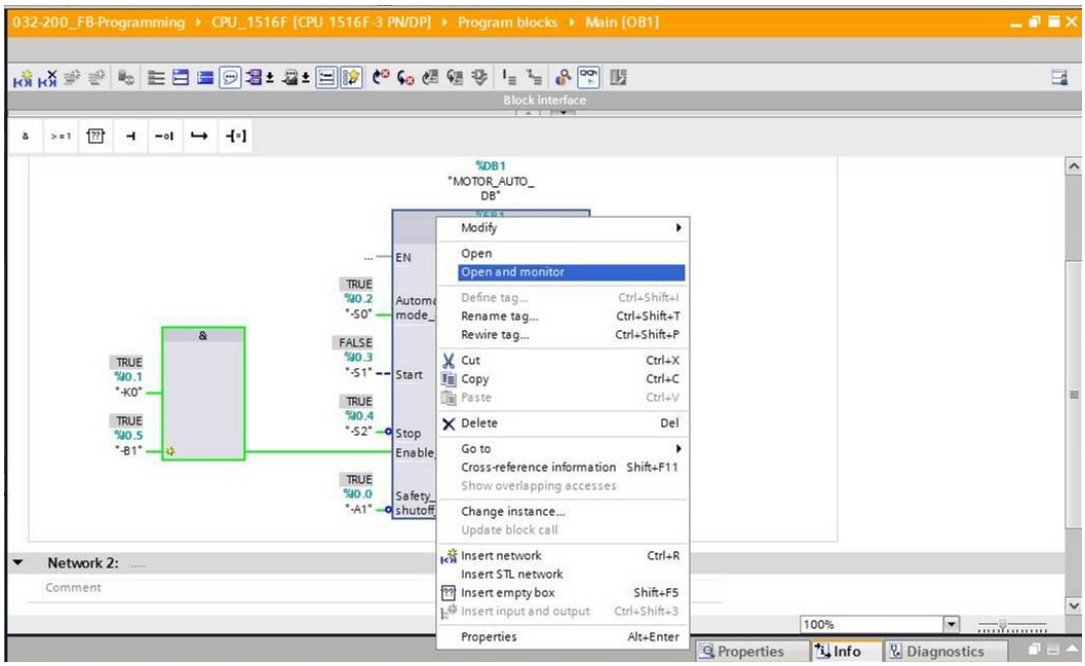
→ 다운로드된 프로그램을 모니터링하려면 해당 블록을 열어야 합니다. 아이콘을 클릭해서 모니터링을 활성화/비활성화할 수 있습니다. (→ Main [OB1] →)

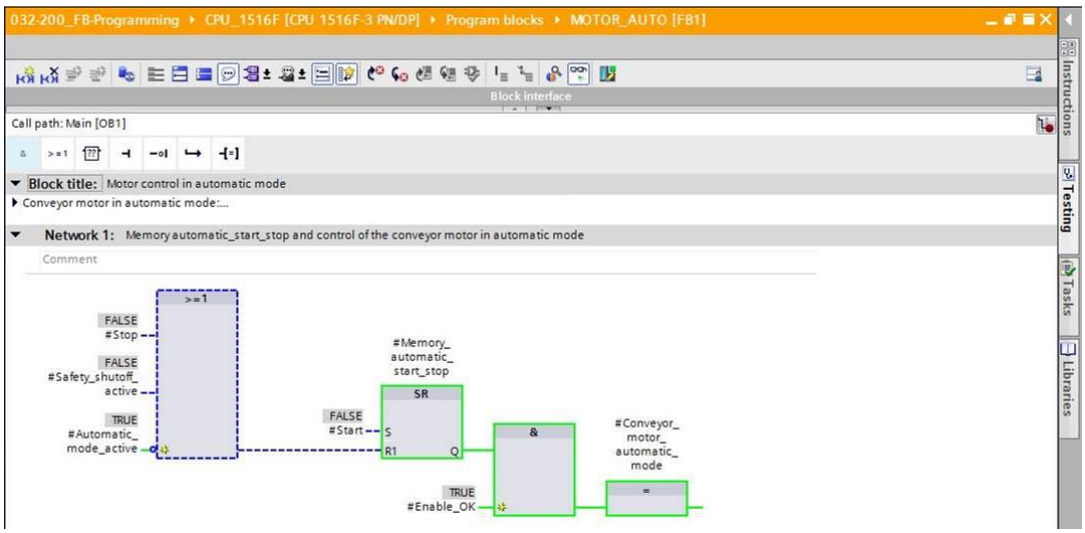




참고: 위의 모니터링 상태는 신호값과 컨트롤러에 다르게 보일 수 있습니다. 터미널에서의 신호 상태는 TRUE 또는 FALSE로 표시가 됩니다.

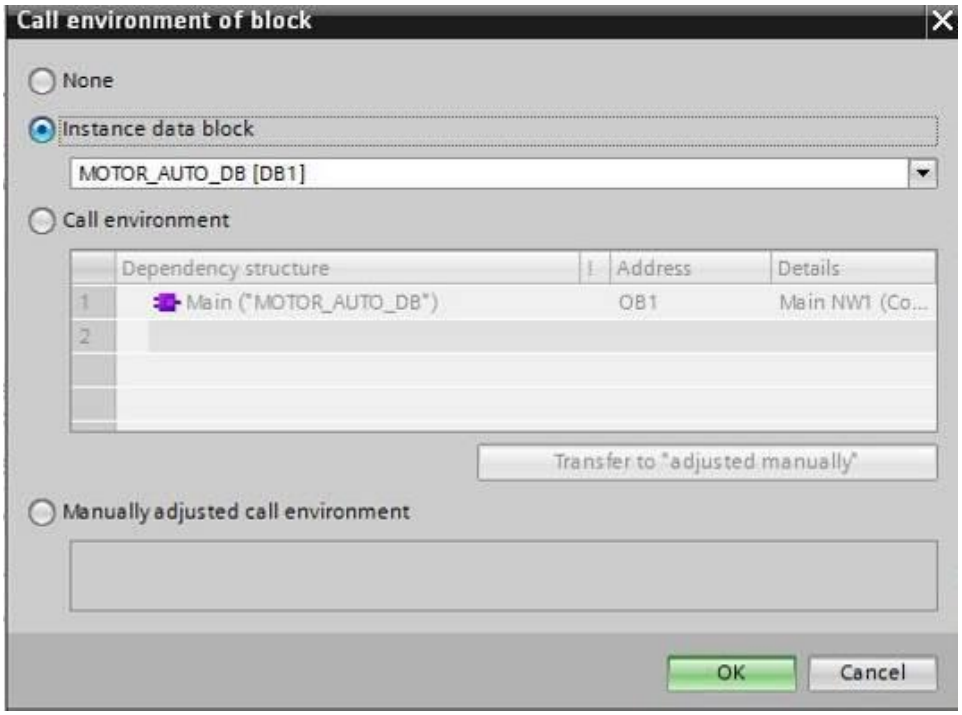
→ 마우스 오른쪽 버튼을 클릭한 후 "Open and monitoring"에서 "Main [OB1]" 오거나이제이션 블록이 호출한 "MOTOR_AUTO" [FB1] 펑션 블록을 직접 선택할 수 있습니다.
(→ "MOTOR_AUTO" [FB1] → Open and monitoring)

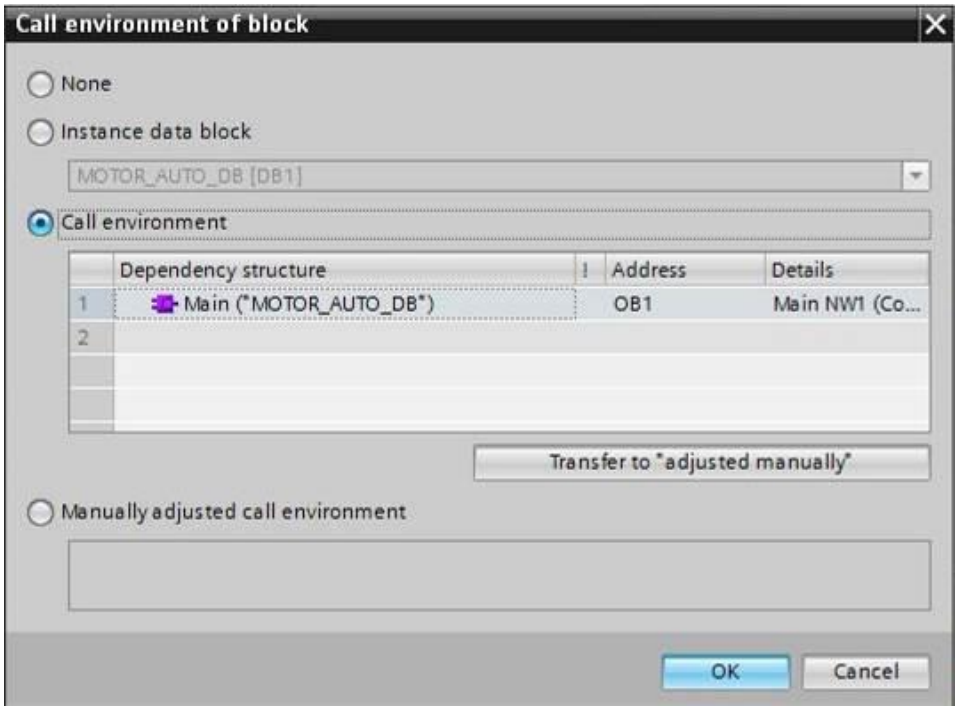




참고: 위의 모니터링 상태는 펑션과 컨트롤러에따라 다르게 보일 수 있습니다. 센서의 상태와 스테이션 상태가 여기에 TRUE 또는 FALSE로 표시가 됩니다.

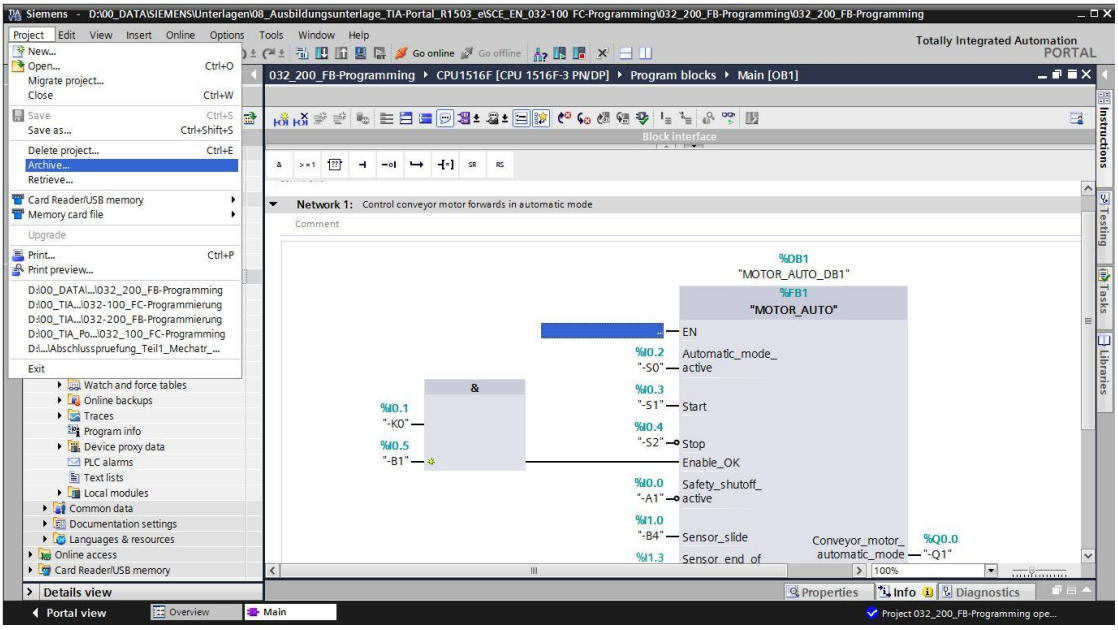
→ 여러 차례 호출이 되는 "MOTOR_AUTO" [FB1] 펑션 블록의 특정 사용 위치를 모니터링하려는 경우에는 아이콘을 이용해 모니터링을 수행할 수 있습니다. 두가지 방법 즉, 인스턴스 데이터 블록을 이용하는 방법과 Call environment를 이용하는 방법 호출 환경을 사용할 수 있습니다. (→ → 인스턴스 데이터 블록 → MOTOR_AUTO_DB1 [DB1] → Call environment → Address: OB1 → Details: Main NW1 → OK).





7.13 프로젝트 아카이브

→ 마지막 단계로 완전한 프로젝트를 아카이브하기 위해 "Project" 메뉴에서 "Archive ..." 항목을 선택합니다. 프로젝트를 아카이브하고자 하는 폴더를 선택하고 "TIA Portal project archive" 파일 유형으로 이를 저장합니다. (→ Project → Archive → TIA Portal project archive→ 032-200_FBProgramming…. → Save)



8 체크리스트

번호	설명	완료
1	오류 메시지 없이 성공적으로 컴파일	
2	오류 메시지 없이 성공적으로 다운로드	
3	스테이션 전원 켜기 (-K0 = 1) 실린더 복귀 / 피드백 활성화 (-B1 = 1) 비상 정지 오프 (-A1 = 1)가 활성화되지 않음 자동 모드 (-S0 = 1) 푸시버튼 자동 정지가 구동되지 않음 (-S2 =1) 자동 시작 푸시버튼을 짧게 누르고 (-S1 = 1), 컨베이어 모터 고정 속도로 전진 (-Q1 = 1) 스위치를 켠 상태로 유지	
4	자동 정지 푸시버튼을 짧게 누르기 (-S2 = 0) → Q1 = 0	
5	비상 정지 오프를 활성화 (-A1 = 0) → Q1 = 0	
6	수동 모드 (-S0 = 0) → Q1 = 0	
7	스테이션 전원 끄기 (-K0 = 0) → Q1 = 0	
8	실린더가 복귀되지 않음 (-B1 = 0) → Q1 = 0	
9	프로젝트가 성공적으로 아카이브 됨	

9 연습

9.1 과제 – 연습

이 연습에서는 에너지 절약 펑션을 MOTOR_AUTO [FB1] 펑션 블록에 추가해 보겠습니다. 그리고 확장된 펑션 블록을 계획, 프로그래밍 및 테스트해보겠습니다.

에너지 절약을 위해서는 부품이 있을 때만 컨베이어가 작동해야 합니다.

따라서 Memory_automatic_start_stop이 셋되어 있고 시작 조건이 충족되며 Memory_conveyor_start_stop이 셋되어 있을 때만 Conveyor_motor_automatic_mode 출력이 작동됩니다.

Memory_conveyor_start_stop은 Sensor_chute_occupied가 부품이 있음을 알릴 때 셋 되고, Sensor_end_of_conveyor에서 네거티브 엣지를 만들어 내거나 안전 전원 차단이 활성화되거나 자동 모드가 활성화되어 있지 않을 때(수동 모드) 리셋됩니다.

9.2 계획 수립

과제 수행에 대한 계획을 스스로으로 수립합니다.

참고: 온라인 도움말에서 SIMATIC S7-1500에서의 네거티브 엣지 사용에 대해 알아보십시오.

9.3 체크리스트 – 연습

번호	설명	완료
1	오류 메시지 없이 성공적으로 컴파일	
2	오류 메시지 없이 성공적으로 다운로드	
3	스테이션 전원 켜기 (-K0 = 1) 실린더 복귀 / 피드백 활성화 (-B1 = 1) 비상 정지 오프 (-A1 = 1)가 활성화되지 않음 자동 모드 (-S0 = 1) 푸시버튼 자동 정지가 구동되지 않음 (-S2 =1) 자동 시작 푸시버튼을 짧게 누르기 (-S2 = 1) 이송 장치의 센서가 활성화되고 (-B4 = 1) 컨베이어 모터 고정 속도로 전진 (-Q1 = 1) 스위치를 켠 상태로 유지	
4	컨베이어 끝의 센서 활성화 (-B7 = 1) → -Q1 = 0	
5	자동 정지 푸시버튼을 짧게 누르기 (-S2 = 0) → -Q1 = 0	
6	비상 정지 오프를 활성화 (-A1 = 0) → -Q1 = 0	
7	수동 모드 (-S0 = 0) → -Q1 = 0	
8	스테이션 전원 끄기 (-K0 = 0) → -Q1 = 0	
9	실린더가 복귀되지 않음 (-B1 = 0) → -Q1 = 0	
10	프로젝트가 성공적으로 아카이브 됨	

10 추가 정보

초기 및 심화 교육에 방향을 제시하는 도구의 차원에서 TIA Portal 모듈에 대한 추가 정보를 활용할 수 있습니다. 시작하기, 동영상, 교재, 앱, 매뉴얼, 프로그래밍 지침, 체험용 소프트웨어/펌웨어 등을 아래 링크에서 찾아보실 수 있습니다.

www.siemens.com/sce/s7-1500