최대의 이익을 위한 최대의 선택 !

LS산전에서는 저희 제품을 선택하시는 분들께 최대의 이익을 드리기 위하여
항상 최선의 노력을 다하고 있습니다.

프로그래머블 로직 컨트롤러

XGI / XGR/ XEC 명령어집

XGT Series 사용설명서

안전을 위한 주의사항

- 사용 전에 안전을 위한 주의사항을 반드시 읽고 정확하게 사용하여 주십시오.
- 사용설명서를 읽고 난 뒤에는 제품을 사용하는 사람이 항상 볼 수 있는 곳에 잘 보관하십시오.

LS산전
www.lsis.com

제품을 사용하기 전에…

제품을 안전하고 효율적으로 사용하기 위하여 본 사용설명서의 내용을 끝까지 잘 읽으신 후에 사용해 주십시오.

► 안전을 위한 주의 사항은 제품을 안전하고 올바르게 사용하여 사고나 위험을 미리 막기 위한 것이므로 반드시 지켜 주시기 바랍니다.

► 주의사항은 ‘경고’ 와 ‘주의’ 의 2가지로 구분되어 있으며, 각각의 의미는 다음과 같습니다.

⚠ **경고** 지시사항을 위반하였을 때, 심각한 상해나 사망이 발생할 가능성이 있는 경우

⚠ **주의** 지시사항을 위반하였을 때, 경미한 상해나 제품 손상이 발생할 가능성이 있는 경우

► 제품과 사용설명서에 표시된 그림 기호의 의미는 다음과 같습니다.

⚠ 는 위험이 발생할 우려가 있으므로 주의하라는 기호입니다.

⚠ 는 감전의 가능성이 있으므로 주의하라는 기호입니다.

► 사용설명서를 읽고 난 뒤에는 제품을 사용하는 사람이 항상 볼 수 있는 곳에 보관해 주십시오.

A급 기기 (업무용 방송통신기기)

‣ **이 기기는 업무용(A급)으로 전자파적합등록을 한 기기이오니 판매자 또는 사용자는 이 점을 주의하시기 바라며, 가정 외의 지역에서 사용하는 것을 목적으로 합니다.**

설계 시 주의 사항

⚠ 경 고

- **외부 전원, 또는 PLC모듈의 이상 발생시에 전체 제어 시스템을 보호하기 위해 PLC의 외부에 보호 회로를 설치하여 주십시오.**

  PLC의 오출력/오동작으로 인해 전체 시스템의 안전성에 심각한 문제를 초래할 수 있습니다.

  - PLC의 외부에 비상 정지 스위치, 보호 회로, 상/하한 리미트 스위치, 정/역방향 동작 인터록 회로 등 시스템을 물리적 손상으로부터 보호할 수 있는 장치를 설치하여 주십시오.
  - PLC의 CPU가 동작 중 워치독 타이머 에러, 모듈 착탈 에러 등 시스템의 고장을 감지하였을 때에는 시스템의 안전을 위해 전체 출력을 Off시킨 후, 동작을 멈추도록 설계되어 있습니다. 그러나 릴레이, TR등의 출력 소자 자체에 이상이 발생하여 CPU가 고장을 감지할 수 없는 경우에는 출력이 계속 On 상태로 유지될 수 있습니다. 따라서, 고장 발생시 심각한 문제를 유발할 수 있는 출력에는 출력 상태를 모니터링 할 수 있는 별도의 회로를 구축하여 주십시오.

- **출력 모듈에 정격 이상의 부하를 연결하거나 출력 회로가 단락되지 않도록 하여 주십시오.**

  화재의 위험이 있습니다.

- **출력 회로의 외부 전원이 PLC의 전원보다 먼저 On 되지 않도록 설계하여 주십시오.**

  오출력 또는 오동작의 원인이 될 수 있습니다.

- **컴퓨터 또는 기타 외부 기기가 통신을 통해 PLC와의 데이터 교환, 또는 PLC의 상태를 조작 (운전 모드 변경 등)하는 경우에는 통신 에러로 부터 시스템을 보호할 수 있도록 시퀀스 프로그램에 인터록을 설정하여 주십시오.**

  오출력 또는 오동작의 원인이 될 수 있습니다.

설계 시 주의 사항

⚠ 주 의

- 입출력 신호 또는 통신선은 고압선이나 동력선과는 최소 100mm 이상 떨어뜨려 배선하십시오.

  오출력 또는 오동작의 원인이 될 수 있습니다.

설치 시 주의 사항

⚠ 주 의

- PLC는 사용설명서 또는 데이터 시트의 일반 규격에 명기된 환경에서만 사용해 주십시오.

  감전/화재 또는 제품 오동작 및 열화의 원인이 됩니다.

- 모듈을 장착하기 전에 PLC의 전원이 꺼져 있는지 반드시 확인해 주십시오.

  감전, 또는 제품 손상의 원인이 됩니다.

- PLC의 각 모듈이 정확하게 고정되었는지 반드시 확인해 주십시오.

  제품이 느슨하거나 부정확하게 장착되면 오동작, 고장, 또는 낙하의 원인이 됩니다.

- I/O 또는 증설 커넥터가 정확하게 고정되었는지 확인해 주십시오.

  오입력 또는 오출력의 원인이 됩니다.

- 설치 환경에 진동이 많은 경우에는 PLC에 직접 진동이 인가되지 않도록 하여 주십시오.

  감전/화재 또는 오동작의 원인이 됩니다.

- 제품 안으로 금속성 이물질이 들어가지 않도록 하여 주십시오.

  감전/화재 또는 오동작의 원인이 됩니다.

배선 시 주의 사항

⚠ 경 고

- **배선 작업을 시작하기 전에 PLC의 전원 및 외부 전원이 꺼져 있는지 반드시 확인하여 주십시오.**

  감전 또는 제품 손상의 원인이 됩니다.

- **PLC 시스템의 전원을 투입하기 전에 모든 단자대의 커버가 정확하게 닫혀 있는지 확인하여 주십시오.**

  감전의 원인이 됩니다.

⚠ 주 의

- **각 제품의 정격 전압 및 단자 배열을 확인한 후 정확하게 배선하여 주십시오.**

  화재, 감전 사고 및 오동작의 원인이 됩니다.

- **배선시 단자의 나사는 규정 토크로 단단하게 조여 주십시오.**

  단자의 나사 조임이 느슨하면 단락, 화재, 또는 오동작의 원인이 됩니다.

- **FG 단자의 접지는 PLC전용 3종 접지를 반드시 사용해 주십시오.**

  접지가 되지 않은 경우, 오동작의 원인이 될 수 있습니다.

- **배선 작업 중 모듈 내로 배선 찌꺼기 등의 이물질이 들어가지 않도록 하여 주십시오.**

  화재, 제품 손상, 또는 오동작의 원인이 됩니다.

시운전, 보수 시 주의사항

⚠ 경 고

- **전원이 인가된 상태에서 단자대를 만지지 마십시오.**

  감전 또는 오동작의 원인이 됩니다..

- **청소를 하거나, 단자를 조일 때에는 PLC및 모든 외부 전원을 Off시킨 상태에서 실시하여 주십시오.**

  감전 또는 오동작의 원인이 됩니다.

- **배터리는 충전, 분해, 가열, Short, 납땜 등을 하지 마십시오.**

  발열, 파열, 발화에 의해 부상 또는 화재의 위험이 있습니다.

⚠ 주 의

- **모듈의 케이스로 부터 PCB를 분리하거나 제품을 개조하지 마십시오.**

  화재, 감전 사고 및 오동작의 원인이 됩니다.

- **모듈의 장착 또는 분리는 PLC 및 모든 외부 전원을 Off시킨 상태에서 실시하여 주십시오.**

  감전 또는 오동작의 원인이 됩니다.

- **무전기 또는 휴대전화는 PLC로 부터 30cm 이상 떨어뜨려 사용하여 주십시오.**

  오동작의 원인이 됩니다.

폐기 시 주의사항

⚠ 주 의

- **제품 및 배터리를 폐기할 경우, 산업 폐기물로 처리하여 주십시오.**

  유독 물질의 발생, 또는 폭발의 위험이 있습니다.

개 정 이 력

버전	일자	주요 변경 내용	관련 페이지
V1.0	'07. 3	초판 발행	-
V1.1	'07. 6	프로세스 제어 라이브러리 추가	제 13 장
V1.2	'07. 12	ST 언어 추가	제 14 장
V2.0	'08. 3	XGR CPU 기종 추가	전체
V2.1	'09. 4	1. XEC 기종 추가 2. XEC 용 함수 추가 (1) APM_SSSB (2) PIDAT (3) PIDHBD 3. 펑션 / 펑션블록 ST언어 프로그램 예 추가 4. ST언어 설명 수정	전체 11-31 13-4 13-8 제 7~11장 제 14 장
V2.2	'09. 9	1. 내용 추가 (1) XPM 신제품 관련 전용 명령어 추가	6-16, 17 11-73 ~ 116
V2.3	'09. 12	1. 내용 추가 (1) 위치 결정 명령어 4종 추가 (VRD, VWR)	6.4.10 6.4.11 11.4 11.5
V2.4	'10. 6	1. 내용추가 (1) 위치 결정 명령어 수정 / 추가	6.4.11 11.5
V2.5	'12. 7	1. 내용추가 (1) 위치 결정 명령어 추가	6.4.10~6.4.11 11.4~11.5
V2.6	'13. 5	1. 내용수정 및 추가 (1) PUTE/GETE 명령어 추가 (2) ARY_PUTM 명령어 사용 시리즈 수정	6.4.8 11.4 11.4

V2.7	'14. 4	1. 내용수정 및 추가	
		(1) CPT 명령어 설명 추가	8장
		(2) APM_SSSB 설명 수정	11장
		(3) UDATA 명령어 추가	11장
		(4) 토크동기 명령어(XPM_STC) 추가	11장
		(5) ST/SFC 내용 수정 및 추가	4장, 14장

※ 사용설명서의 번호는 사용설명서 뒷표지의 우측에 표기되어 있습니다.

LS산전 PLC를 구입하여 주셔서 감사 드립니다.
제품을 사용하시기 이전에 올바른 사용을 위하여 구입하신 제품의 기능과 성능, 설치, 프로그램 방법 등에 대해서 본 사용설명서의 내용을 숙지하여 주시고 최종 사용자와 유지 보수 책임자에게 본 사용설명서가 잘 전달될 수 있도록 하여 주시기 바랍니다.

다음의 사용설명서는 본 제품과 관련된 사용설명서입니다.
필요한 경우, 아래의 사용설명서의 내용을 보시고 주문하여 주시기 바랍니다.
또한, 당사 홈페이지 http://www.lsis.biz/ 에 접속하여 PDF파일로 다운로드 받으실 수 있습니다.

관련된 사용설명서 목록

사용설명서 명칭	사용설명서 내용	사용설명서 번호
XG5000 사용설명서	XGT 시리즈의 제품을 사용하여 프로그래밍, 인쇄, 모니터링, 디버깅과 같은 온라인 기능을 설명한 XG5000 소프트웨어 사용설명서입니다.	10310000746
XGK/XGB 명령어집	XGK/XGB CPU를 장착한 PLC시스템에서 사용하는 전체적인 명령어에 대해서 사용방법을 설명한 사용설명서입니다.	10310000509
XGI-CPUU 사용설명서	XGI CPU모듈, 전원 모듈, 베이스, 입출력 모듈, 증설 케이블의 각 규격 및 시스템 구성, EMC 규격 대응 등에 대해서 설명하고 있습니다.	10310000738
XGR-CPUH 사용설명서	XGR CPU모듈, 전원 모듈, 베이스, 입출력 모듈, 증설 케이블의 각 규격 및 시스템 구성, EMC 규격 대응 등에 대해서 설명하고 있습니다.	10310000919

제1장 개요

이 언어 설명서는 XGI/XGR/XEC PLC를 지원하는 언어에 대한 설명서입니다.
XGI/XGR/XEC PLC는 IEC (International Electrotechnical Commission - 국제전기표준회의)에서 국제 표준으로 발표한 언어를 기본으로 합니다.

1.1. IEC 61131-3 언어의 특징

IEC 언어에서 새로 도입한 가장 중요한 특징들은 다음과 같습니다.

▷ 다양하고 강력한 데이터 타입을 지원합니다.
▷ 펑션, 펑션 블록, 프로그램 같은 프로그램 구성 요소가 도입되어 상향식, 또는 하향식 설계가 가능하며, PLC 프로그램을 구조적으로 작성할 수 있습니다.
▷ 사용자가 작성한 프로그램을 다른 환경에서 사용할 수 있어 소프트웨어의 재사용을 가능하게 합니다.
▷ 다양한 언어를 지원하므로 사용자는 적용환경에 최적의 언어를 선택하여 사용할 수 있습니다.

1.2. 언어의 종류

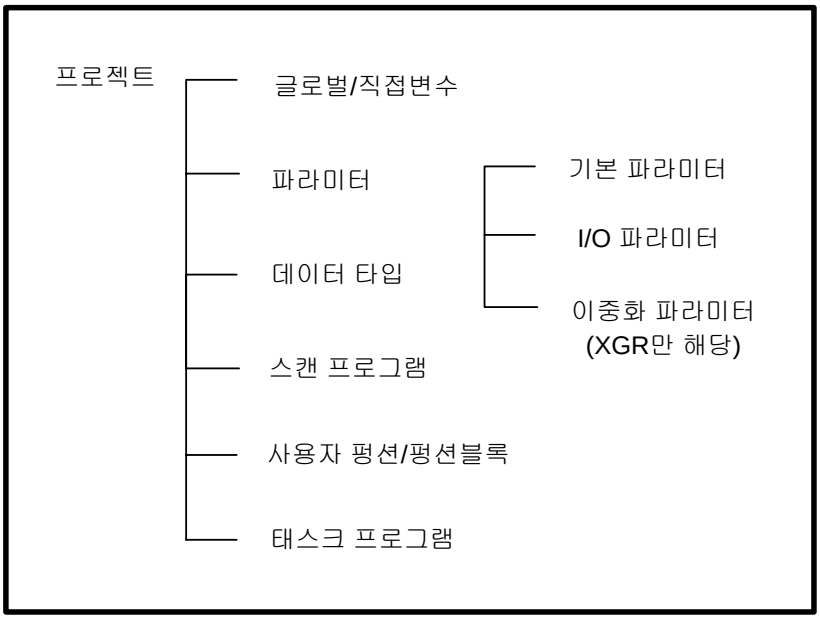
IEC에서 표준화한 PLC용 언어는 두 개의 도식 언어와 두 개의 문자식 언어, 그리고 SFC로 이루어져 있습니다.

▷ 도식 언어
a) LD (Ladder Diagram): 릴레이 로직 표현 방식의 언어
b) FBD (Function Block Diagram): 블록화한 기능을 서로 연결하여 프로그램을 표현하는 언어
▷ 문자식 언어
a) IL (Instruction List): 어셈블리 언어 형태의 언어
b) ST (Structured Text): 파스칼 형식의 고 수준 언어
▷ SFC (Sequential Function Chart)

제2장 소프트웨어 구조

2.1. 개요

PLC 응용 프로그램을 작성하기 전에 소프트웨어 측면에서 전체적인 PLC 시스템을 구성합니다. XGI/XGR/XEC PLC 에서는 PLC 시스템 전체를 하나의 프로젝트로 정의합니다. 프로젝트 안에는 하나의 PLC 시스템에 필요한 모든 것이 계층적으로 정의되어 있습니다.



2.2. 프로젝트(Project)

▷ XGI/XGR/XEC PLC 의 프로그램을 작성하기 위해서는 우선 프로젝트를 구성하여야 합니다. 하나의 프로젝트를 구성한다는 것은 하나의 PLC 시스템에 필요한 모든 구성 요소를 작성한다는 의미입니다. 즉, 가장 기본적인 스캔 프로그램(일반적인 PLC 프로그램)뿐만 아니라 기본 파라미터, I/O 파라미터 등을 작성합니다.

2.3. 글로벌/직접 변수

▷ 글로벌 변수 설정 부분, 직접 변수 설명문, 플래그 부분의 탭으로 보여주며 사용자가 필요한 정보를 작성 또는 사용하는 부분입니다.

2.4. 파라미터

▷ 파라미터 부분은 PLC 시스템의 기동 시 필요한 여러 가지 정보를 작성하는 부분입니다.
▷ 기본 파라미터: 기본 파라미터 정보 중 기본 운전, 시간, 출력 제어 설정을 위한 부분과 PLC 전원이 꺼져도 데이터를 보존하는 영역 설정 부분, PLC 에 에러가 발생했을 때 동작 방법의 설정을 위한 부분 그리고 MODBUS 정보 설정 부분으로 구성되어 있습니다.
▷ I/O 파라미터: PLC 슬롯에 사용할 I/O 종류를 설정하고, 해당 슬롯 별로 파라미터를 설정합니다.

2.5. 데이터 타입

▷ 데이터는 그 데이터의 고유 성질을 나타내는 데이터 타입을 가지고 있습니다. 예를 들어 ANY_NUM 으로 나타내면 LREAL,

REAL, LINT, DINT, INT, SINT, ULINT, UDINT, UINT, USINT 를 모두 포함합니다. 자세한 사항은 3.2 의 항목을 참고하시기 바랍니다.

2.6. 스캔 프로그램(Program)

▷ 입력 모듈에서 입력 데이터를 읽은 후 프로그램을 처음부터 끝까지 한번 수행하고, 그 수행 결과를 출력 모듈에 쓰는 일련의 동작을 반복하여 수행하는 응용 프로그램입니다.

2.7. 사용자 펑션/펑션 블록

▷ 펑션: 펑션은 내부에 상태를 보관하고 있는 데이터를 갖지 않습니다. 즉 입력이 일정하면 출력 값도 일정해야만 펑션이 됩니다.

▷ 펑션 블록: 펑션 블록은 내부에 데이터를 가질 수 있습니다. 펑션 블록은 사용하기 전에 변수를 선언하는 것처럼 인스턴스를 선언하여야 합니다. 인스턴스라는 것은 펑션 블록에서 사용하는 변수들의 집합입니다. 즉, 펑션 블록은 내부에서 사용하는 변수뿐 아니라 출력 값도 펑션 블록 자체에서 보관합니다. 따라서 인스턴스가 보관된 데이터 메모리를 기억하고 있습니다. 프로그램도 펑션 블록의 일종이라고 볼 수 있으며, 프로그램 역시 인스턴스를 선언하여야 합니다. 그러나 프로그램은 펑션 블록과 다르게 프로그램 안이나 펑션 블록 안에서 불러 사용할 수는 없습니다.

2.8. 태스크 프로그램

▷ 태스크 프로그램은 스캔 프로그램처럼 매 스캔 반복처리를 하지 않고, 실행 조건이 발생할 때만 실행을 합니다. 실행해야 할 태스크가 여러 개 대기하고 있는 경우는 우선 순위가 높은 태스크 프로그램부터 처리합니다. 우선 순위가 동일한 태스크가 대기 중일 때는 발생한 순서대로 처리합니다.

▷ 태스크 종류는 정주기 태스크와 내부 접점 태스크가 있습니다.

제3장 공통 요소

XGI/XGR/XEC PLC 의 프로그램 구성 요소(프로그램, 펑션, 펑션 블록)는 LD, SFC, ST 등 각기 다른 언어로 작성할 수 있습니다. 하지만 그 언어들도 공통적으로 사용하는 문법 요소들을 가지고 있습니다.

3.1. 표현

3.1.1. 식별자(Identifiers)

▷ 영문자나 밑줄 문자(_)로 시작하는 모든 문자, 숫자, 밑줄 문자의 조합이 식별자가 될 수 있습니다.
▷ 식별자는 변수의 이름으로 쓰입니다.
▷ 식별자는 빈 칸(Space)을 포함하지 않아야 합니다.
▷ 식별자는 보통 변수 또는 인스턴스 이름인 경우에는 한글, 영문, 한자 모두 제한이 없습니다.
▷ 영문자의 경우, 대 · 소문자를 구별하지 않고 모두 대문자로 인식합니다.

종 류	사 용 예
대문자와 숫자	IW210, IW215Z, QX75, IDENT
대문자와 숫자, 밑줄 글자	LIM_SW_2, LIMSW5, ABCD, AB_CD
밑줄 글자로 시작하는 대문자와 숫자	_MAIN, _12V7, _ABCD

3.1.2. 데이터의 표현

XGI/XGR/XEC PLC 에서 데이터로 사용하는 것은 숫자(Numeric Literals)와 문자열(Character String), 시간 문자(Time Literals) 등입니다.

종 류	사 용 예
정 수	-12, 0, 123_456, +986
실 수	-12.0, 0.0, 0.456, 3.14159_26
승수부를 갖는 실수	-1.34E-12, 1.0E+6, 1.234E6
2 진수	2#1111_1111, 2#11100000
8 진수	8#377(십진수 255) 8#340(십진수 224)
16 진수	16#FF(십진수 255) 16#E0(십진수 224)
BOOL 데이터	0, 1, TRUE, FALSE

1) 숫자(Numeric Literals)

▷ 숫자에는 정수(Integer Literals)와 실수(Real Literals)가 있습니다.
▷ 연속되지 않은 밑줄 글자(_)가 숫자 사이에 올 수 있으며 그 의미는 무시됩니다.
▷ 십진수는 일반적인 십진 표현법을 따르고 소수점이 있으면 실수로 구별됩니다.
▷ 승수(Exponent) 표현 시 +, -의 부호가 올 수 있습니다. 승수부를 구분하는 문자 'E'는 대소문자를 구분하지 않습니다.
▷ 승수부가 있는 실수의 사용시 다음은 가능하지 않습니다.
예) 12E-5 (×) 12.0E-5 (○)
▷ 정수에는 십진수 이외에 2,8,16 진수가 올 수 있으며, 숫자의 앞부분에 진수 #을 사용하여 구분합니다. 아무것도 붙

이지 않으면 십진수로 간주합니다.

▷ 16 진수 표현 시 0 - 9, A - F 를 쓰며 소문자 a - f 도 쓸 수 있습니다.
▷ 16 진수 표현 시에는 부호(+, -)가 올 수 없습니다.
▷ BOOL 데이터(Boolean Data)는 정수 0 과 1 로도 표현할 수 있습니다.

2) 문자열(Character String)

▷ 작은 따옴표(‘)로 둘러싸인 모든 문자가 문자열에 해당됩니다.
▷ 그 길이는 문자열 상수일 때에는 31 자 이내이며, 초기화에 사용할 때 역시 31 자로 제한합니다.

예) ‘CONVEYER’

3) 시간 문자(Time Literals)

▷ 시간 문자는 제어 사건(Control Event)의 경과 시간(Elapsed Time)을 재거나 조절하기 위한 경과 시간(Duration) 데이터와, 제어 사건의 시작점과 끝점의 시각을 표시하기 위한 날짜와 시각(Time Of Day And Date) 데이터로 구분됩니다.

(1) 경과 시간(Duration)

▷ 경과 시간 데이터는 예약어 'T#'또는 't#'으로 시작합니다.
▷ 일(d), 시(h), 분(m), 초(s), ms 의 순으로 써야 하고 어느 단위에서 시작되어도 상관없으며, 최소 단위인 ms 까지 꼭 쓰지 않아도 되나 중간 단위를 생략할 수는 없습니다.
▷ 밑줄 글자(_)는 사용하지 않습니다.
▷ 최대 단위에서의 오버플로(Overflow)는 허용되며, 최소 단위에서의 소수점 이하 표현도 ms 이외에는 가능합니다. 단 최대는 T#49d17h2m47s295ms 을 초과할 수 없습니다.
(즉 ms 단위로 32 비트)
▷ 소수점 이하 자릿수의 제한은 현재 초(s)단위에서의 3 자리까지입니다.
▷ ms 단위에서는 소수점이 올 수 없습니다.
▷ 단위를 나타내는 문자로는 대 · 소문자 어느 경우나 다 가능합니다.

내 용	사 용 예
경과 시간(Underline 없음)	T#14ms, T#14.7s, T#14.7m, T#14.7h t#14.7d, t#25h15m, t#5d14h12m18s356ms

(2) 날짜와 시각(Time Of Day And Date)

▷ 날짜와 시각의 표현 방법에는 날짜, 시각, 날짜와 시각의 3 가지가 있으며 다음과 같습니다.

내 용	접두 예약어
날짜 접두어	D#
시각 접두어	TOD#
날짜 시각 접두어	DT#

▷ 날짜의 시작점은 1984 년 1 월 1 일을 기점으로 합니다.
▷ 시각과 날짜 시각의 표현에는 엄격한 자릿수의 제한이 있으며, 초를 나타낼 경우 ms 단위는 소수점 이하 세 자리까지 가능합니다. (1ms 단위)
▷ 시각과 날짜 시각의 표현 시에는 모든 단위에서 오버플로(Overflow)가 허용되지 않습니다.

내 용	사 용 예
날짜	D#1984-06-25 d#1984-06-25
시각	T0D#15:36:55.36 tod#15:36:55.369
날짜 시각	DT#1984-06-25-15:36:55.36 dt#1984-06-25-15:36:55.369

3.2. 데이터 타입

데이터는 그 데이터의 고유 성질을 나타내는 데이터 타입을 가지고 있습니다.

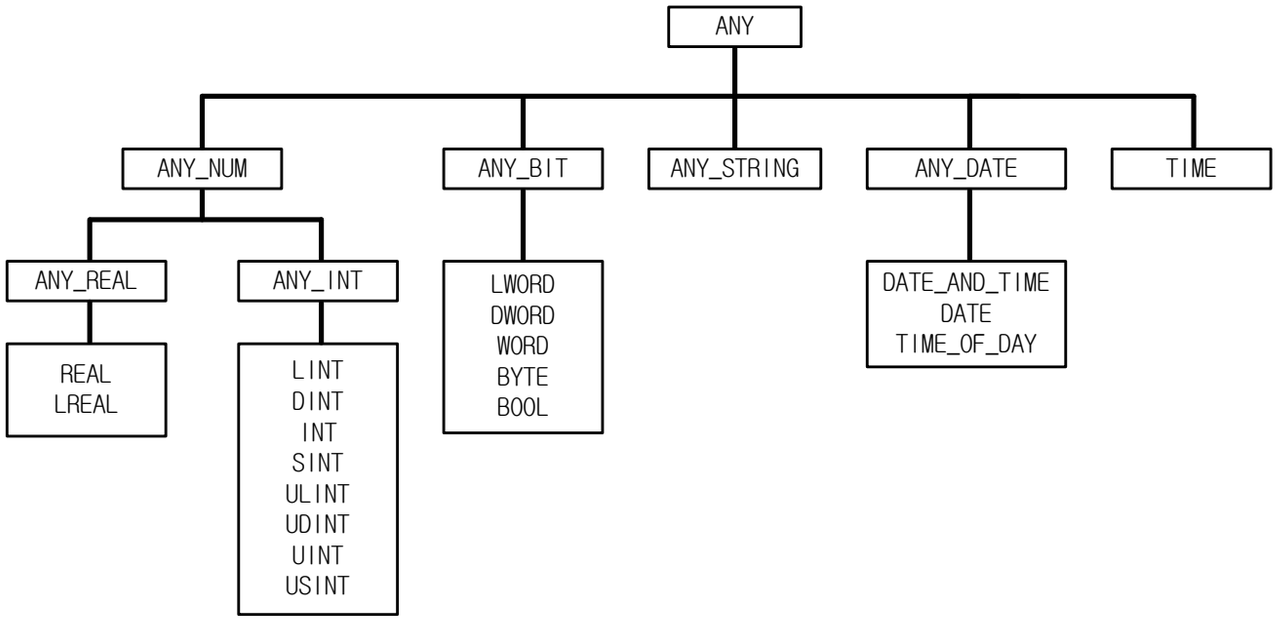
3.2.1. 기본 데이터 타입

XGI/XGR/XEC PLC 에서는 다음의 기본 데이터 타입을 지원합니다.

번호	예 약 어	데이터 타입	크기 (비트)	범 위
1	SINT	Short Integer	8	-128 ~ 127
2	INT	Integer	16	-32,768 ~ 32,767
3	DINT	Double Integer	32	-2,147,483,648 ~ 2,147,483,647
4	LINT	Long Integer	64	-2^{63} ~ $2^{63}-1$
5	USINT	Unsigned Short Integer	8	0 ~ 255
6	UINT	Unsigned Integer	16	0 ~ 65,535
7	UDINT	Unsigned Double Integer	32	0 ~ 4,294,967,295
8	ULINT	Unsigned Long Integer	64	0 ~ $2^{64}-1$
9	REAL	Real Numbers	32	-3.402823466e+038 ~ -1.175494351e-038 or 0 or 1.175494351e-038 ~ 3.402823466e+038
10	LREAL	Long Real Numbers	64	-1.7976931348623157e+308 ~ -2.2250738585072014e-308 or 0 or 2.2250738585072014e-308 ~ 1.7976931348623157e+308
11	TIME	Duration	32	T#0S ~ T#49D17H2M47S295MS
12	DATE	Date	16	D#1984-01-01 ~ D#2163-6-6
13	TIME_OF_DAY	Time Of Day	32	T0D#00:00:00 ~ T0D#23:59:59.999
14	DATE_AND_TIME	Date And Time Of Day	64	DT#1984-01-01-00:00:00 ~ DT#2163-06-06-23:59:59.999
15	STRING	Character String	32*8	-
16	BOOL	Boolean	1	0,1
17	BYTE	Bit String Of Length 8	8	16#0 ~ 16#FF
18	WORD	Bit String Of Length 16	16	16#0 ~ 16#FFFF
19	DWORD	Bit String Of Length 32	32	16#0 ~ 16#FFFFFFFF
20	LWORD	Bit String Of Length 64	64	16#0 ~ 16#FFFFFFFFFFFFFFFF

3.2.2. 데이터 타입 계층도

XGI/XGR/XEC PLC 에서 사용되는 데이터 타입은 다음과 같습니다.



▷ 앞으로 데이터 타입을 표현할 때, ANY_NUM 으로 나타내면 다음 계층도와 같이 LREAL, REAL, LINT, DINT, INT, SINT, ULINT, UDINT, UINT, USINT 를 모두 포함합니다.

▷ 예를 들어 타입이 ANY_BIT 로 표현되면 LWORD, DWORD, WORD, BYTE, BOOL 중 하나를 사용할 수 있습니다.

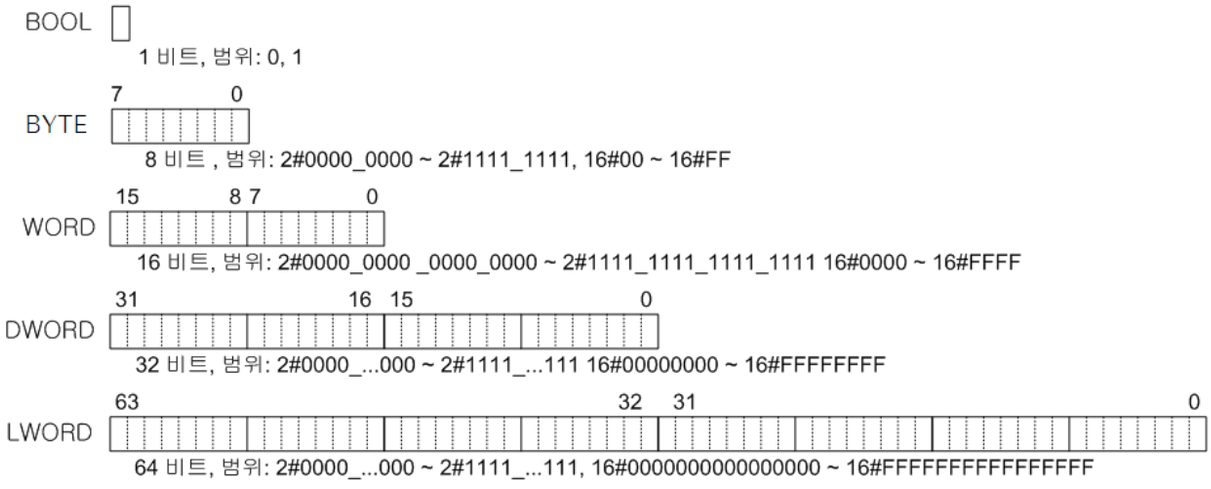
3.2.3. 초기값

데이터의 초기값을 지정하지 않으면 자동적으로 아래와 같이 지정됩니다.

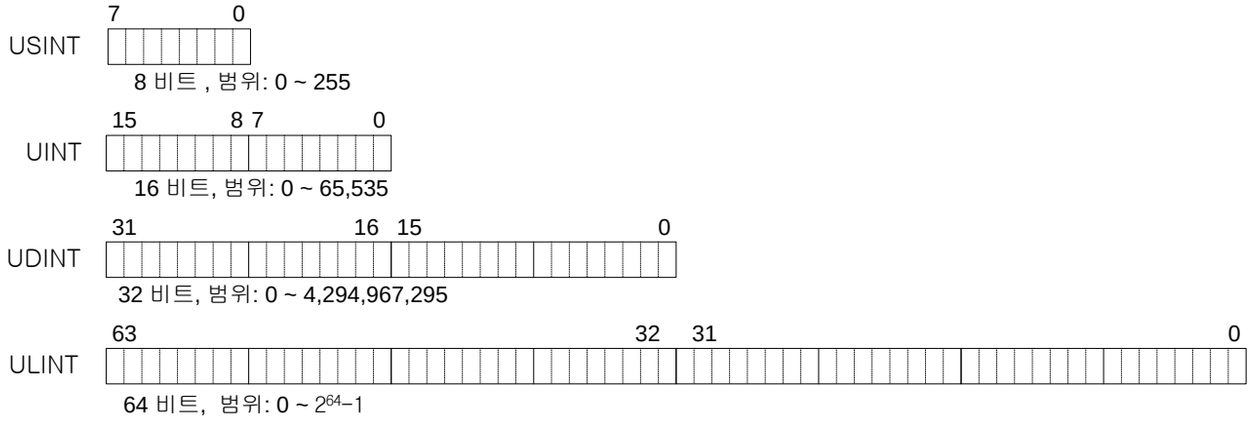
데이터 타입	초 기 값
SINT, INT, DINT, LINT	0
USINT, UINT, UDINT, ULINT	0
BOOL, BYTE, WORD, DWORD, LWORD	0
REAL, LREAL	0.0
TIME	T#0s
DATE	D#1984-01-01
TIME_OF_DAY	TOD#00:00:00
DATE_AND_TIME	DT#1984-01-01-00:00:00
STRING	'' (empty string)

3.2.4. 데이터 타입별 구조

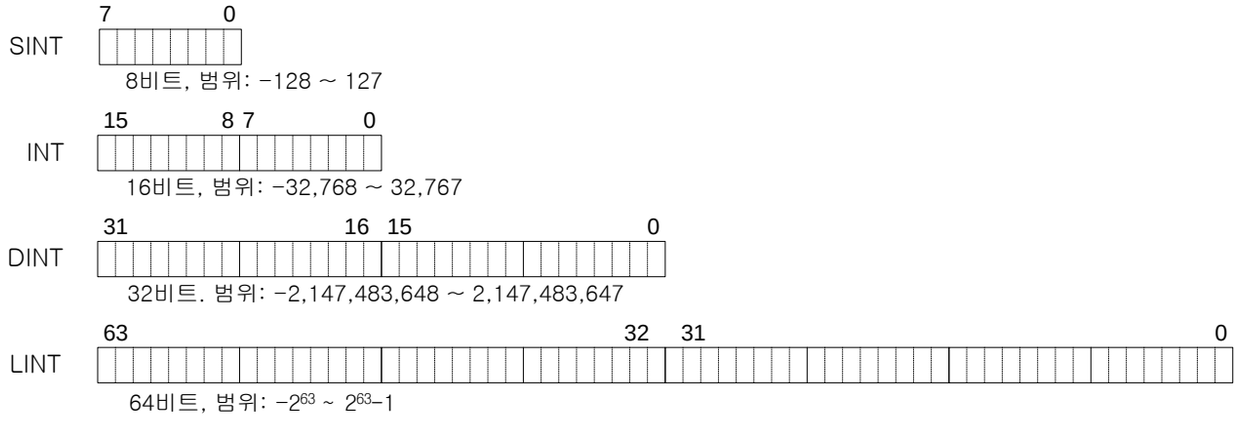
 # Bit String



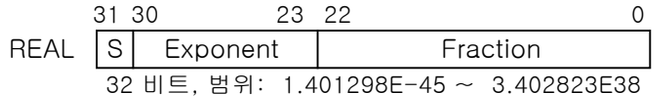
 # Unsigned Integer



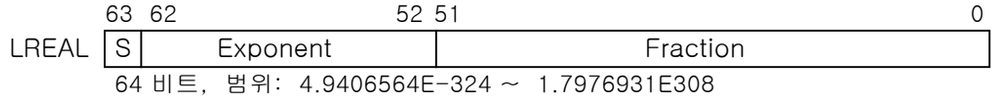
Integer (음수는 2' Complement 표현)



Real (IEEE Standard 754-1984 기준)



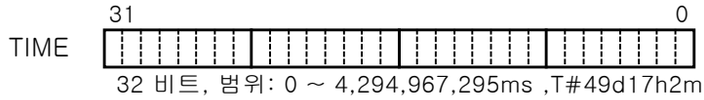
32 비트, 범위: 1.401298E-45 ~ 3.402823E38



64 비트, 범위: 4.9406564E-324 ~ 1.7976931E308

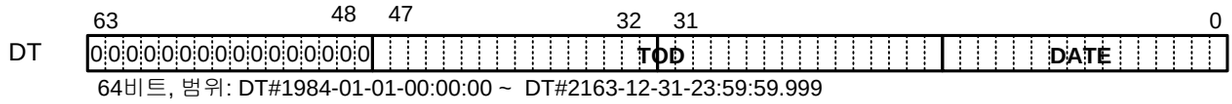
- S: 부호표시 (0일 때 양수, 1일 때 음수)
- Exponent: 2의 승수부 (2^{e-127}: $e=b_{30}b_{29}...b_{23}$, $e=b_{62}b_{61}...b_{52}$)
- Fraction: 소수점 이하 값 (Fraction: $f=b_{22}b_{21}...b_0$, $f=b_{51}b_{52}...b_0$)

Time

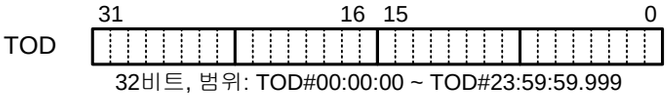


32 비트, 범위: 0 ~ 4,294,967,295ms ,T#49d17h2m47s295ms

Date



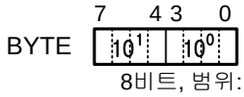
64비트, 범위: DT#1984-01-01-00:00:00 ~ DT#2163-12-31-23:59:59.999



32비트, 범위: TOD#00:00:00 ~ TOD#23:59:59.999

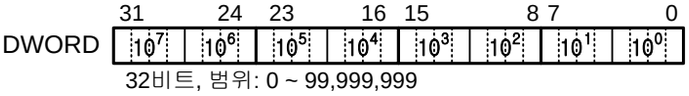
16비트, 범위: D#1984-01-01 ~ D#2163-6-6

#BCD

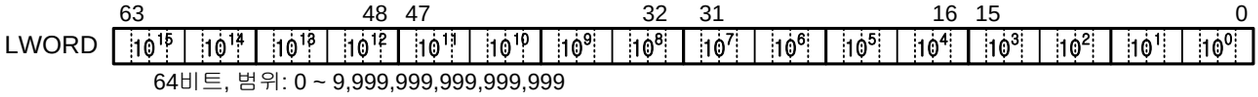


8비트, 범위: 0 ~ 99

16비트, 범위: 0 ~ 9999



32비트, 범위: 0 ~ 99,999,999



64비트, 범위: 0 ~ 9,999,999,999,999,999

3.3. 변수

변수란 프로그램 안에서 사용하는 데이터로서 값을 가지고 있습니다. 변수는 PLC 의 입력이나 출력, 내부 메모리 등과 같이 변할 수 있는 대상을 가리킵니다.

3.3.1. 변수의 표현

▷ 변수의 표현에는 2 가지가 있습니다.
식별자에 의한 변수: 식별자에 의해 변수에 이름을 부여하는 것
직접변수: PLC 의 입 · 출력 또는 기억 장소에 대하여 직접적으로 표현하는 것

▷ 식별자에 의한 변수는 다른 변수들과 구별하기 위하여 그 이름의 변수가 선언된 프로그램 안에서 유일해야 합니다.

▷ 직접 변수의 표현은 퍼센트 문자(%)를 시작으로 위치를 나타내는 접두어와 데이터의 크기를 나타내는 접두어 그리고 마침표로 분리되는 하나 이상의 부호 없는 정수의 순으로 표현할 수 있습니다. 그 접두어들은 다음에 나타나 있습니다.

위치 접두어

번 호	접두어	의 미
1	I	입력 위치(Input Location)
2	Q	출력 위치(Output Location)
3	M	내부 메모리 중 M 영역 위치(Memory Location)
4	R	내부 메모리 중 R 영역 위치(Memory Location)
5	W	내부 메모리 중 W 영역 위치(Memory Location)

크기 접두어

번 호	접두어	의 미
1	X	1 비트의 크기
2	None	1 비트의 크기
3	B	1 바이트(8 비트)의 크기
4	W	1 워드(16 비트)의 크기
5	D	1 더블 워드(32 비트)의 크기
6	L	1 롱 워드(64 비트)의 크기

표현 형식

%[위치 접두어][크기 접두어] n1.n2.n3

번 호	I, Q	M, R, W
n1	베이스 번호(0 부터 시작)	[크기 접두어]에 따른 n1 번째 데이터 (0 부터 시작)
n2	슬롯 번호(0 부터 시작)	n1 번째 데이터상의 n2 번째 비트 (0 부터 시작) : 생략 가능
n3	[크기 접두어]에 따른 n3 번째 데이터 (0 부터 시작)	사용하지 않음

예

%QX3.1.4 또는 %Q3.1.4	3 번 베이스의 1 번 슬롯의 4 번 출력(1 비트)
%IW2.4.1	2 번 베이스의 4 번 슬롯의 워드 단위로 1 번 입력(16 비트)
%MD48	48 의 위치에 있는 더블 워드 단위의 메모리
%MW40.3	40 의 위치에 있는 워드 단위의 메모리 중 3 번 비트
	(내부 메모리는 베이스, 슬롯 등의 개념이 없음)

▷ 접두어로는 소문자가 올 수 없습니다.
▷ 크기 접두어를 붙이지 않으면 그 변수는 1 비트로 처리합니다.
▷ 직접변수는 선언하지 않고 사용할 수 있습니다.

3.3.2. 변수의 선언

▷ 프로그램 구성 요소(즉 프로그램, 펑션, 펑션 블록)는 그 구성 요소에서 사용할 변수를 선언할 수 있는 선언 부분을 가지고 있습니다.
▷ 프로그램 구성 요소에서 변수를 사용하기 위해서는 우선 사용할 변수를 선언해야 합니다.
▷ 변수의 선언에서 설정해야 할 사항은 다음과 같습니다.

1) 변수 종류 : 변수를 어떻게 선언할 것인가를 설정합니다.

변수종류	내 용
VAR	읽고 쓸 수 있는 일반적인 변수
VAR_RETAIN	정전 유지 변수
VAR_CONSTANT	읽기만 할 수 있는 변수
VAR_EXTERNAL	VAR_GLOBAL 로 선언된 변수를 사용하기 위한 선언

2) 데이터 타입 : 변수의 데이터 타입을 지정합니다.
3) 메모리 할당 : 변수가 차지할 메모리를 할당합니다.
자동 —— 컴파일러가 변수의 위치를 자동으로 지정(자동 배치 변수).
사용자 정의(AT) —— 사용자가 직접표현 변수를 사용하여 강제로 위치를 지정(직접변수).

참고

자동 배치 변수는 그 실제 위치가 고정되어 있지 않습니다. 예를 들어 VAL1 이란 변수를 BOOL 데이터타입으로 선언하였다면 그 변수가 내부 데이터 영역의 어느 위치에 있는지 고정되어 있지 않다는 것입니다. 그 위치는 프로그램을 다 작성한 후 컴파일러와 링커에 의해 정해집니다. 만약 프로그램을 수정한 후에 다시 컴파일 하였다면 그 위치가 변할 수 있습니다. 자동 배치 변수의 장점은 사용자가 내부 변수로 사용하는 것들의 위치에 신경 쓰지 않아도 된다는 것입니다. 다른 이름으로 선언한 변수들은 결코 데이터 메모리에 중복되어 위치하지 않기 때문입니다. 직접변수는 변수의 위치가 정해지기 때문에 %I 와 %Q 를 제외하고는 될 수 있으면 사용하지 않는 것이 좋습니다. 직접변수는 자동 배치 변수가 아니므로 사용자가 잘못 사용할 경우, 중복될 수 있습니다.

▷ 초기값(Initial Value) 지정 : 변수의 초기값을 지정합니다. 지정하지 않으면 3.2.3. 항의 초기값으로 지정됩니다.

참고

VAR_EXTERNAL 의 선언 시에는 초기값을 줄 수 없습니다.
변수 선언 시 %I 와 %Q 로 강제 할당한 변수에는 초기값을 줄 수 없습니다.

▷ PLC 의 전원이 끊긴 후에도 데이터의 값을 유지할 필요가 있는 변수는 정전 유지(Retention)의 기능이 제공되는 VAR_RETAIN 을 써서 선언할 수 있으며 다음의 규칙을 따릅니다.
1) 정전 유지 변수는 시스템의 웜 리스타트시 그 값이 유지됩니다.
2) 시스템의 콜드 리스타트시에는 사용자가 정의한 초기값이나 기본 초기값으로 초기화됩니다.

▷ VAR_RETAIN 으로 선언되지 않은 변수는 콜드 리스타트나 웜 리스타트 어느 경우에도 사용자가 정의한 초기값이나 기본 초기값으로 초기화됩니다.

참고

변수 선언 시 %I 와 %Q 로 강제 할당한 변수는 변수종류를 VAR_RETAIN, VAR_CONSTANT 로 선언할 수 없습니다.

▷ 변수는 기본 데이터 타입을 인자로 갖는 어레이로 선언하여 사용할 수 있습니다. 어레이 변수로 선언할 때에는 인자로 사용할 데이터의 타입과 어레이의 크기를 설정하여야 합니다.
단, 기본 데이터 타입 중에 STRING 데이터 타입은 인자로 설정할 수 없습니다.

▷ 변수 선언의 유효 영역(Scope), 즉 변수를 사용할 수 있는 영역은 그 변수가 선언된 프로그램 구성 요소에 한합니다. 따라서 다른 프로그램 구성 요소에서 선언된 변수는 사용할 수 없습니다. 글로벌 변수로 선언된 변수는 이와 달리 모든 곳에서 VAR_EXTERNAL 선언에 의해 변수 접근이 가능합니다.

변수의 선언 예

변수 이름	변수형	데이터 타입	초기값	메모리 할당
I_VAL	VAR	INT	1234	자동
BIPOLAR	VAR_RETAIN	REAL	-	자동
LIMIT_SW	VAR	BOOL	-	%IX1.0.2
GLO_SW	VAR_EXTERNAL	DWORD	-	자동
READ_BUF	VAR	ARRAY OF INT[10]	-	자동

3.3.3. 예약 변수

▷ 예약 변수는 시스템에서 미리 선언한 변수들로서 플래그로 사용됩니다. 사용자가 이 변수 이름으로 변수 선언을 할 수는 없습니다.
▷ 이 예약 변수를 사용할 때에는 변수 선언 없이 사용합니다.
▷ 자세한 사항은 'XGI-CPUU 사용설명서'의 플래그 일람과 부록2 '플래그 일람'을 참조하시기 바랍니다.

3.3.4. 예약어

예약어는 시스템에서 사용하기 위해 미리 정의한 단어들입니다. 따라서 식별자로 이 예약어를 사용할 수는 없습니다.

예 약 어
ACTION ... END_ACTION
ARRAY ... OF
AT
CASE ... OF ... ELSE ... END_CASE CONFIGURATION ... END_CONFIGURATION
데이터 타입 이름
DATE#, D#DATE_AND_TIME#, DT#
EXIT
FOR ... TO ... BY ... DO ... END_FOR
FUNCTION ... END_FUNCTION
FUNCTION_BLOCK ... END_FUNCTION_BLOCK
펑션 블록의 이름들
IF ... THEN ... ELSIF ... ELSE ... END_IF
OK
연산자 (IL 언어) 연산자 (ST 언어)
PROGRAM
PROGRAM ... END_PROGRAM
REPEAT ... UNTIL ... END_REPEAT
RESOURCE ... END_RESOURCE
RETAIN
RETURN
STEP ... END_STEP
STRUCTURE ... END_STRUCTURE
T#
TASK ... WITH
TIME_OF_DAY#, TOD#
TRANSITION ... FROM... TO ... END_TRANSITION

예 약 어
TYPE ... END_TYPE
VAR ... END_VAR VAR_INPUT ... END_VAR VAR_OUTPUT ... END_VAR VAR_IN_OUT ... END_VAR VAR_EXTERNAL ... END_VAR
VAR_ACCESS ... END_VAR
VAR_GLOBAL ... END_VAR
WHILE ... DO ... END_WHILE
WITH

3.4. 프로그램 종류

▷ 프로그램 종류로는 사용자 펑션, 사용자 펑션 블록, 프로그램이 있습니다.
▷ 프로그램에서 자기 자신의 프로그램을 호출할 수는 없습니다. (재귀 호출 금지)

3.4.1. 사용자 펑션

▷ 사용자 펑션은 내부에 상태를 보관하고 있는 데이터를 갖지 않습니다. 즉 입력이 일정하면 출력 값도 일정해야만 사용자 펑션이 됩니다.
▷ 사용자 펑션의 내부 변수는 초기값을 가질 수 없습니다.
▷ 사용자 펑션은 변수를 VAR_EXTERNAL 로 선언하여 사용할 수 없습니다.
▷ 사용자 펑션 안에서는 직접변수들을 사용할 수 없습니다.
▷ 사용자 펑션은 프로그램 구성 요소에서 호출하여 사용합니다.
▷ 사용자 펑션을 호출하는 프로그램 구성 요소에서 사용자 펑션으로의 데이터 전달은 입력을 통하여 실행합니다.
▷ 사용자 펑션 안에서는 사용자 펑션 블록이나 프로그램을 호출할 수 없습니다.
▷ 사용자 펑션은 그 사용자 펑션 이름과 하나의 출력 데이터 타입 변수 이름이 같습니다. 이 변수는 하나의 사용자 펑션을 만들 때 자동적으로 생성되며 사용자 펑션에서 결과값을 이 변수에 넣어서 출력시킵니다.

3.4.2. 사용자 펑션 블록

▷ 사용자 펑션 블록은 출력이 여러 개가 될 수 있습니다.
▷ 사용자 펑션 블록은 내부에 데이터를 가질 수 있습니다. 사용자 펑션 블록은 사용하기 전에 변수를 선언하는 것처럼 인스턴스를 선언하여야 합니다. 인스턴스라는 것은 사용자 펑션 블록에서 사용하는 변수들의 집합입니다. 사용자 펑션 블록은 내부에서 사용하는 변수뿐 아니라 출력 값도 자체에서 보관하여야 하므로 데이터 메모리를 가지고 있어야 하는데 바로 그것이 인스턴스입니다. 프로그램도 사용자 펑션 블록의 일종이라고 볼 수 있으며, 프로그램 역시 인스턴스를 선언하여야 합니다.
▷ 사용자 펑션 블록 안에서는 직접 변수를 사용할 수 있습니다. 또한 글로벌 변수로 선언되고 사용자 정의(AT)로 강제 배치된 직접변수는 VAR_EXTERNAL 로 선언하여 사용할 수 있습니다.
▷ 사용자 펑션 블록 안에서는 프로그램을 호출할 수 없습니다.

3.4.3. 프로그램

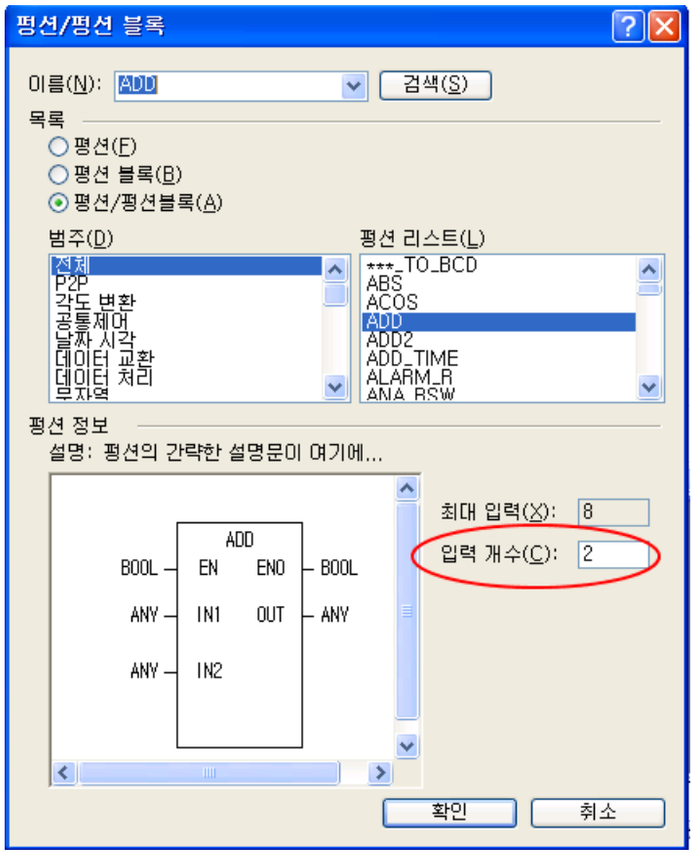
▷ 프로그램은 사용자 펑션 블록과 같이 인스턴스를 선언하여 사용합니다.
▷ 프로그램 안에서는 직접 변수를 사용할 수 있습니다.
▷ 프로그램에는 입/출력변수가 없습니다.

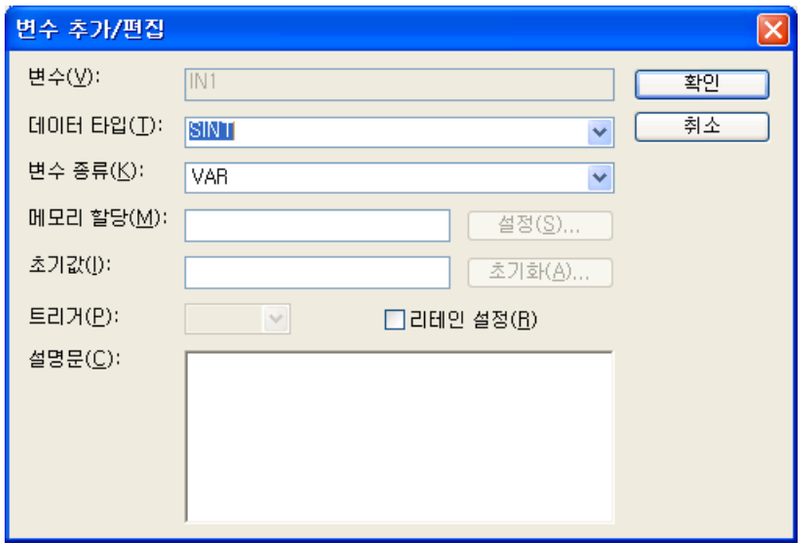
▷ 프로그램에는 사용자 펑션 및 사용자 펑션 블록을 호출할 수 있습니다.

3.5. 명령어 선정

3.5.1. 내부적으로 결정되는 명령어

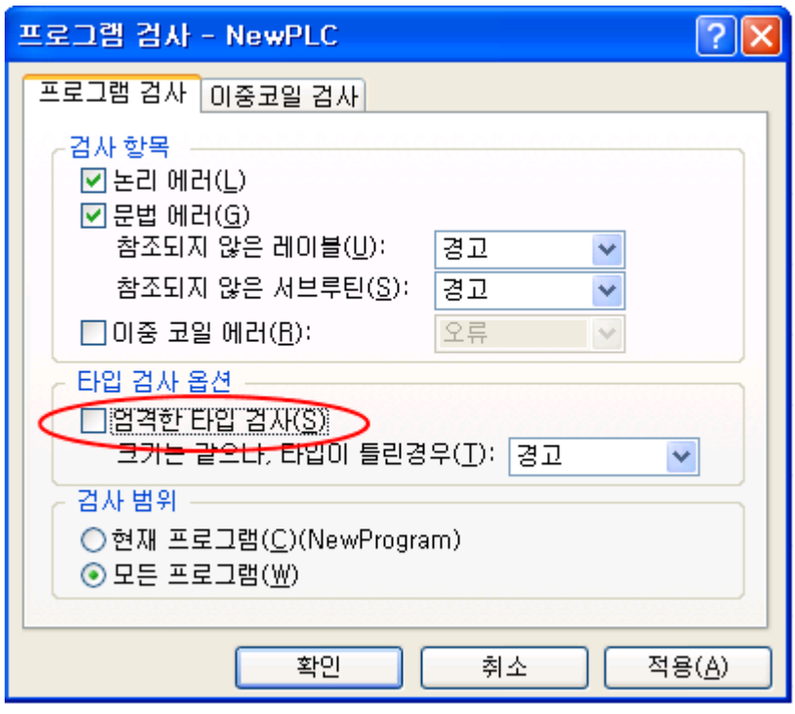
▷ 펑션의 경우 하나의 이름을 갖지만 여러 종류의 변수타입을 입력할 수 있는 명령어는 사용되는 변수에 따라서 내부적으로 여러 명령어로 구분됩니다. 예를 들어 ADD 의 경우, 설정한 입력개수 및 입력/출력 변수 타입에 따라서 여러 종류로 구분되어 처리됩니다. 아래와 같이 선택한 경우에는 래더 프로그램 상에서 보이는 명령어는 ADD 이지만, 내부적으로 ADD2_SINT 명령어를 수행하게 되는 것입니다.





▷ 내부적으로 사용되는 명령어의 선정은 사용자가 선택한 변수 타입에 따라 XG5000 에서 자동 선정하게 됩니다. 예를 들어 ADD 명령 중 입력개수 2 개를 선택하고, 입력과 출력변수를 모두 DINT 로 선택하면 위에서 설명한 것처럼 ADD2_DINT 선정됩니다.

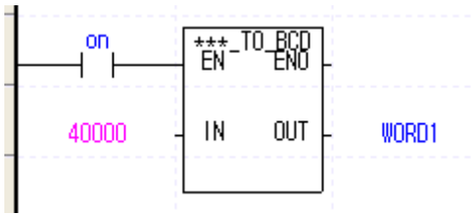
▷ IEC 에서는 같은 타입끼리의 연산만을 허용하고 있으나, XG5000 에서는 검사옵션에 엄격한 검사 옵션을 두어 변수의 크기(BYTE, WORD, DWORD, LWORD)만 같으면 연산을 허용하는 옵션이 있습니다.



3.5.2. 명령어 선정 규칙

▷ 사용된 변수가 한가지 타입이 아니고, 크기가 같은 여러 타입일 경우, 입력 변수의 타입이 우선 반영됩니다. 만약, 입력변수에 모두 상수만 사용되었을 경우, 출력변수에 선언된 타입에 따라 내부적으로 사용되는 명령어가 결정됩니다.

▷ 입력변수의 타입은 여러 가지가 올 수 있고 출력변수의 타입은 한가지만 올 수 있는 명령어에서, 입력에 상수를 사용하게 되면 XG5000 에서는 상수의 값에 따라 명령어를 결정합니다.
***_TO_BCD 로 예를 들어 보면,



위와 같이 사용했을 경우, 입력변수가 상수이므로 출력변수의 타입을 보고 상세 명령어를 결정하게 되는데 이 명령어는 출력이 WORD 인 명령어가 INT_TO_BCD_WORD/UINT_TO_BCD_WORD 두 개 존재합니다. 이 경우에는 상수의 타입에 따라 UINT_TO_BCD_WORD 가 선정됩니다. 상수 사용시 양의 값이면 무조건 부호 없는 수(Unsigned)로 판단하고 음수일 경우에는 부호 있는 수(Signed)로 판단을 합니다.

제4장 SFC(Sequential Function Chart)

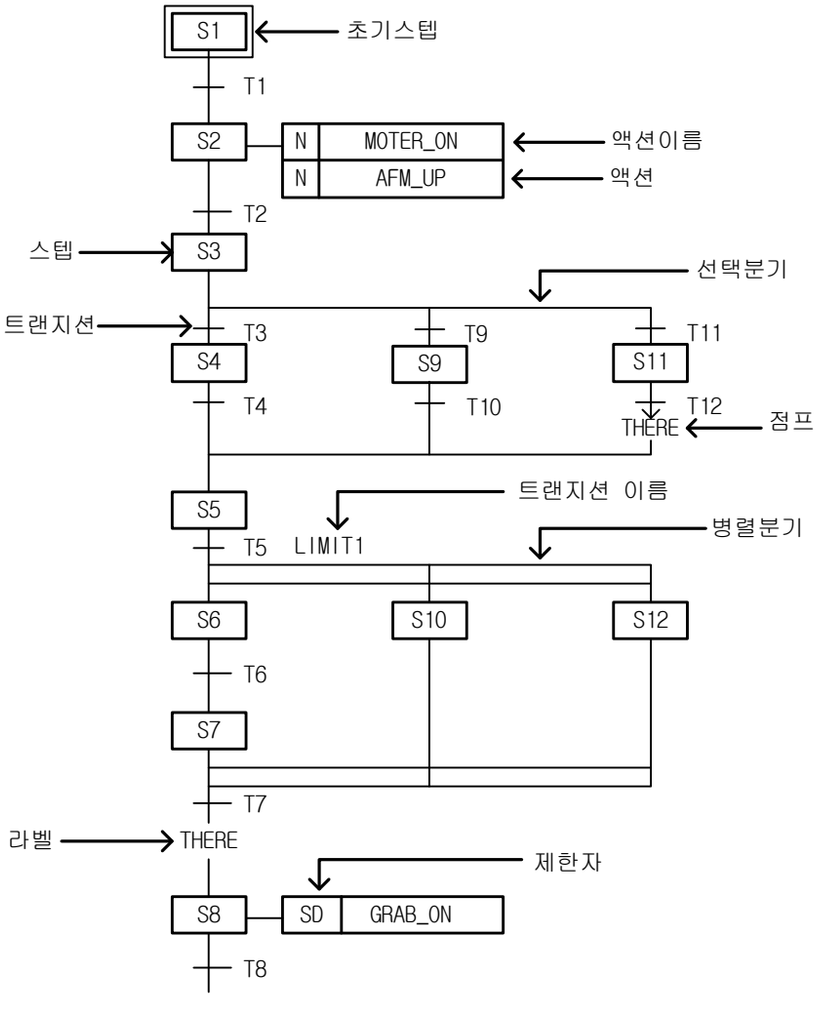
4.1. 개요

4.1.1. SFC

▷ SFC 는 종래의 PLC 언어를 이용하여 응용 프로그램을 실행 처리 순서에 따라 나누어 플로차트 형식으로 전개하는 구조화 표현 방식 언어입니다.

▷ SFC 는 응용 프로그램을 스텝과 트랜지션으로 분할하여 서로 연결하는 방법을 제공하며, 각 스텝은 액션으로, 각 트랜지션은 트랜지션 조건과 연관됩니다.

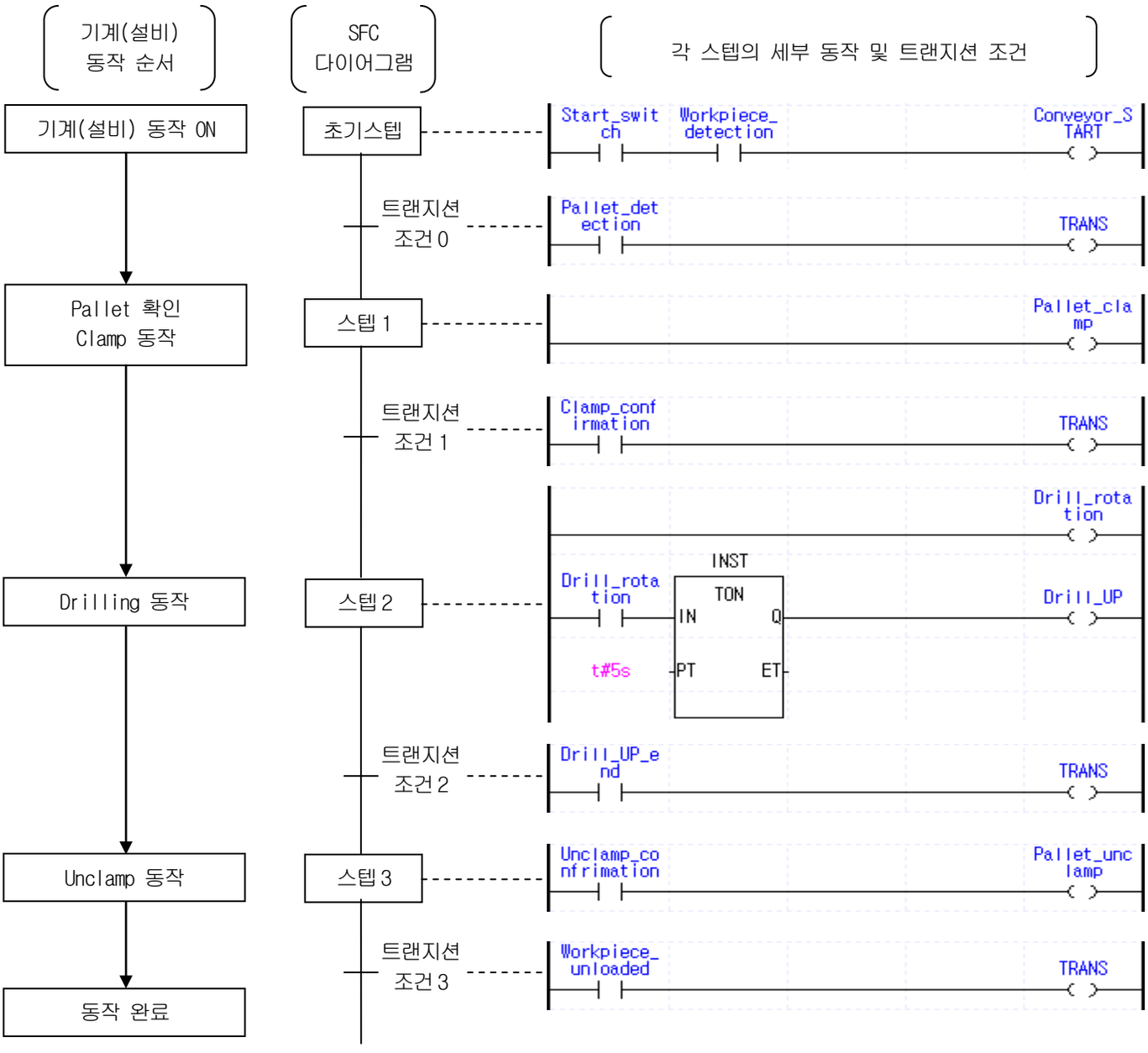
▷ 형태



4.1.2. SFC 프로그램 설명

▷ SFC 프로그램은 기계(설비)의 동작 단위를 나타내는 스텝으로 구성되어 있습니다.

▷ 각 스텝에서 LD/SFC/ST 를 사용하여 세부 동작을 프로그래밍 할 수 있습니다.



▷ SFC 프로그램은 초기 스텝부터 시작하여 마지막 스텝까지 트랜지션 조건에 따라 각각의 스텝들을 실행시킵니다.

▷ SFC 프로그램이 시작되면 '초기' 스텝이 제일 먼저 실행됩니다.

▷ 트랜지션 조건 1 이 만족될 때까지 초기 스텝이 계속 실행됩니다. 트랜지션 조건 1 이 만족되면, 초기 스텝의 실행이 중지되고 다음 스텝이 실행됩니다.

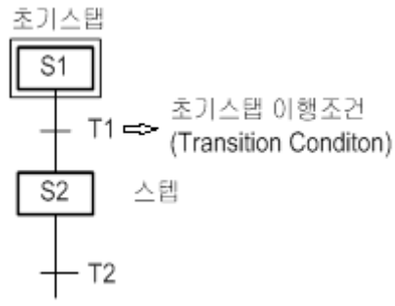
4.2. SFC 구조

4.2.1. 스텝

▷ 스텝은 액션이 연결됨으로써 시퀀스 제어의 단위를 나타냅니다.

▷ 스텝이 활성화 상태이면 부착되어 있는 액션의 내용이 실행됩니다.

▷ 초기 스텝은 최초로 활성화되는 스텝입니다.



▷ 최초의 활성화 상태인 초기 스텝(S1)의 다음 이행 조건(Transition Condition)이 성립되면, 현재 활성화 상태인 스텝(S1)은 비활성화 상태로 되고 다음에 연결된 스텝(S2)이 활성화 상태로 됩니다.

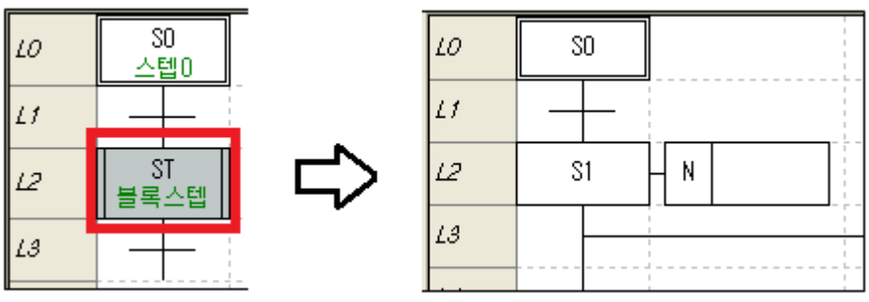
4.2.2. 블록 스텝

▷ 블록 스텝이 활성화 되면 블록 내에 지정된 SFC 프로그램이 실행됩니다.

▷ SFC 프로그램만 작성 가능합니다. (LD/ST는 작성이 불가)

▷ 블록 스텝에는 액션을 연결 할 수 없습니다.

▷ 블록 스텝이 비활성화 되면 블록 내부에서 실행중인 SFC 프로그램이 비활성화 됩니다.



<블록 스텝 내부의 SFC 프로그램>

4.2.3. 트랜지션

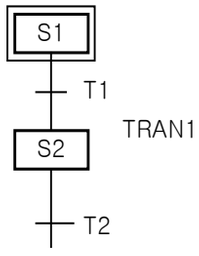
▷ 트랜지션은 스텝간의 실행 처리 이행 조건을 나타냅니다.

▷ 이행 조건은 PLC 언어인 LD/ST(Stuctured Text)로 표현되어야 합니다.

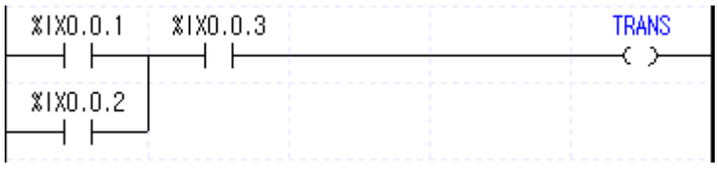
▷ 이행 조건의 결과는 항상 BOOL 로 되어야 하며, 그 변수의 이름은 어느 트랜지션이나 TRANS 가 됩니다.

▷ 이행 조건의 결과가 1 일 경우, 현재 스텝은 비활성화되고 다음 스텝이 활성화됩니다.

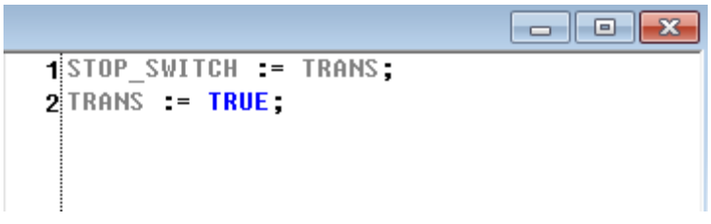
▷ 스텝과 스텝 사이에는 반드시 트랜지션이 있어야 합니다.



(1) TRAN1 의 내용(LD 작성 예)



(2) TRAN1 의 내용(ST 작성 예)



TRANS 가 On 되면 S1 이 비활성화되고 S2 가 활성화 상태가 됩니다.

TRANS 변수는 내부적으로 선언된 변수입니다.

모든 트랜지션에서 이행 조건을 TRANS 변수로 출력시켜야 합니다.

4.2.4. 액션

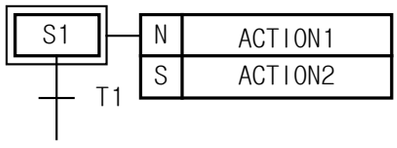
▷ 각 스텝에는 액션을 2 개까지 연결할 수 있습니다.

▷ 액션이 없는 스텝은 대기 액션으로 여겨지며, 다음의 이행 조건이 1 이 될 때까지 대기상태가 됩니다.

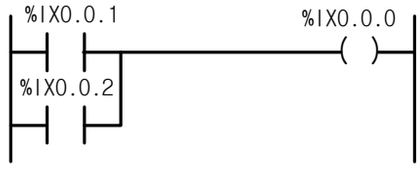
▷ 액션은 PLC 언어인 LD/SFC/ST 로 구성되고, 스텝이 활성화될 동안 액션의 내용이 실행됩니다.

▷ 액션 제한자가 액션을 제어하는 데 사용됩니다.

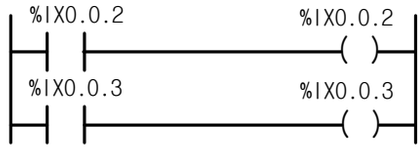
▷ 액션이 활성화되었다가 비활성화 상태로 될 때 액션에서 실행된 접점 출력은 0 으로 됩니다.
단, S, R, 펑션, 펑션 블록 출력은 비활성화되기 전의 상태를 유지합니다.



ACTION1 의 내용(LD 작성 예)

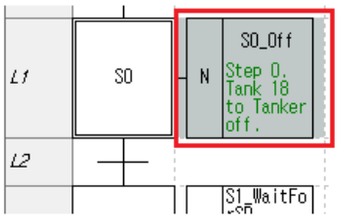


ACTION2 의 내용(LD 작성 예)



- ACTION1 은 S1 이 활성화된 경우에만 실행됩니다.
- ACTION2 은 S1 이 활성화된 후 R 제한자를 만날 때까지 실행됩니다.
  S1 이 비활성화되어도 계속 실행합니다.
- 액션이 비활성화되는 순간, 이 액션을 포스트 스캔(Post Scan)한 후 다음 스텝으로 넘어갑니다.

ACTION1 의 내용(ST 언어 작성 예)




```
//Step Transition Hold Messages

T19ToTankerMessages_A1 := 12001;

IF NOT T19ToTankerStepTimeDone THEN
    T19ToTankerMessages_A1 := 414;
ELSIF NOT T19toTankerAllow THEN
    T19ToTankerMessages_A1 := 400;
END_IF;
```

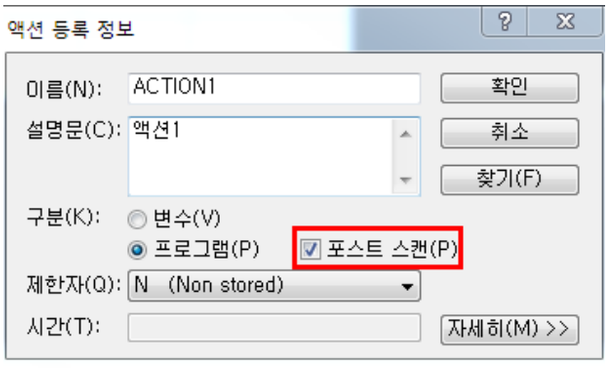
참고

포스트 스캔

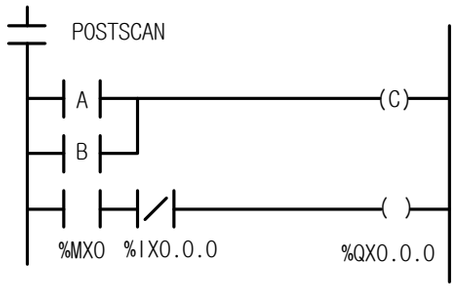
액션이 비활성화되는 순간 이 액션을 다시 한 번 스캔 합니다.

이때 액션 프로그램의 처음에 임의의 접점(값이 0 인 접점)이 있는 것으로 간주하고 스캔 하기 때문에 접점으로 이루어진 프로그램의 출력은 0 이 됩니다.

펑션, 펑션 블록, S, R 출력 등은 해당되지 않습니다.



<액션 등록 정보창>



그림에서 포스트스캔 접점이 0 이므로 C 와 %QX0.0.0 은 0 으로 됩니다.

4.2.5. 액션 제한자(Action Qualifier)

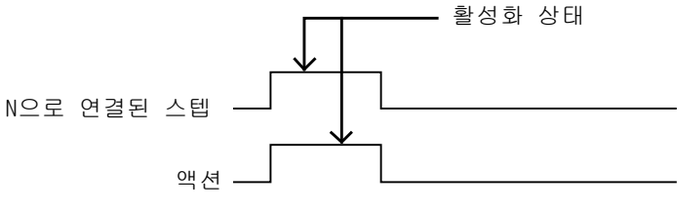
▷ 액션이 사용될 때마다 액션 제한자가 사용됩니다.

▷ 스텝에 연관된 액션은 지정된 제한자에 따라 실행 시점과 시간이 정의됩니다.

▷ 액션 제한자의 종류는 다음과 같습니다.

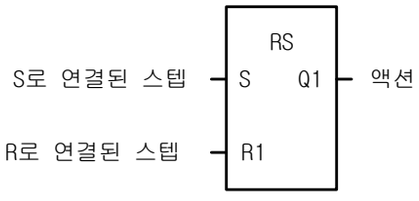
1) N(Non-Stored)

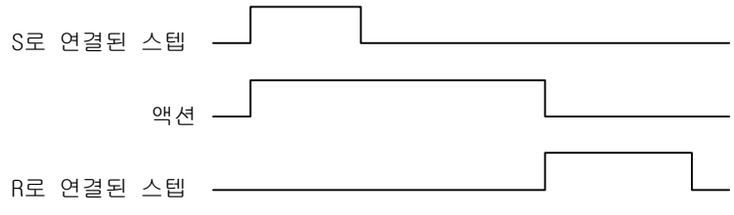
스텝이 활성화된 동안만 액션이 실행됩니다.



2) S(Set)

스텝이 활성화되면 R 제한자가 실행될 때까지 액션이 실행됩니다.



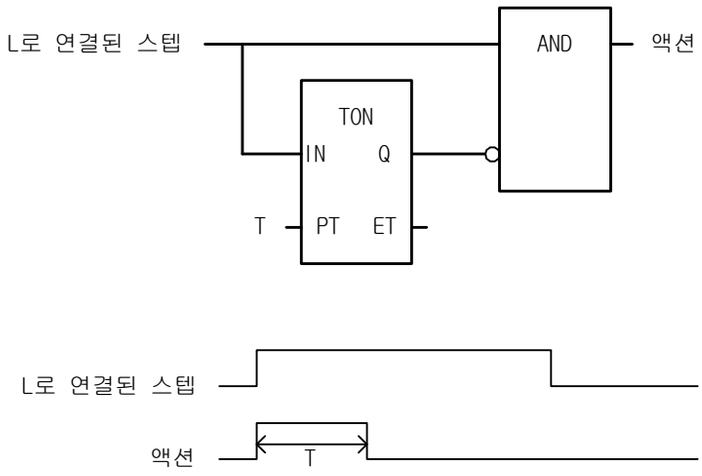


3) R(Overriding Reset)

이전에 S, SD, DS, SL 제한자로 실행된 액션의 실행을 중지시킵니다.

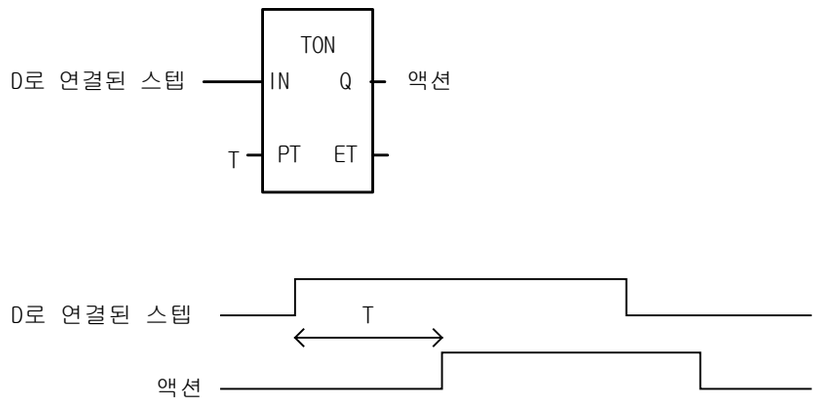
4) L(Time Limited)

스텝이 활성화된 후 지정된 시간까지, 또는 스텝이 비활성화될 때까지 액션이 실행됩니다.



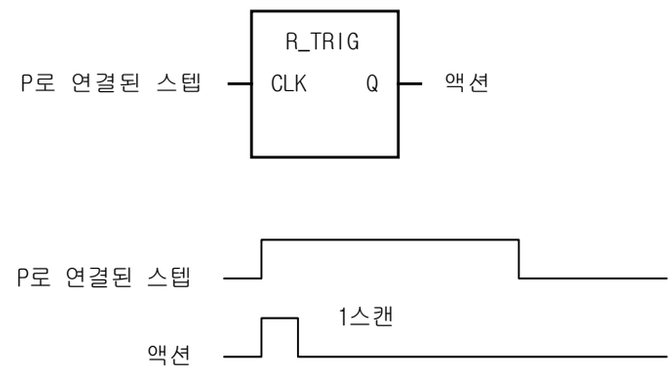
5) D(Time Delayed)

스텝이 활성화된 후 지정된 시간이 경과한 후부터 비활성화될 때까지 액션이 실행됩니다.



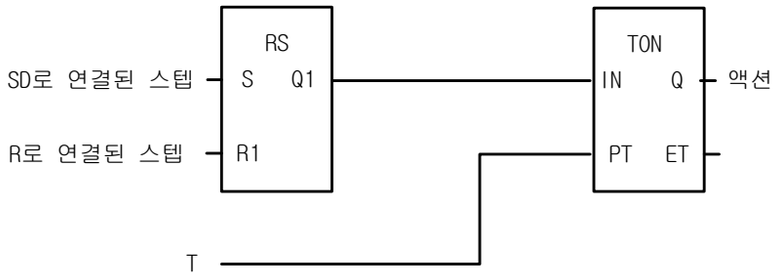
6) P(Pulse)

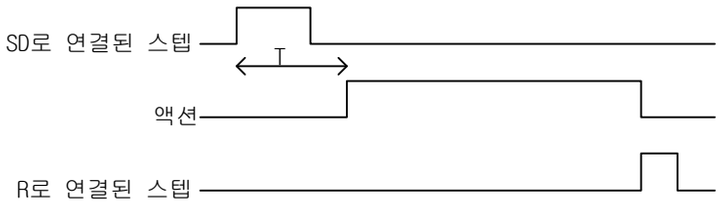
스텝이 활성화된 순간에만 액션이 실행됩니다.



7) SD(Stored & Time Delayed)

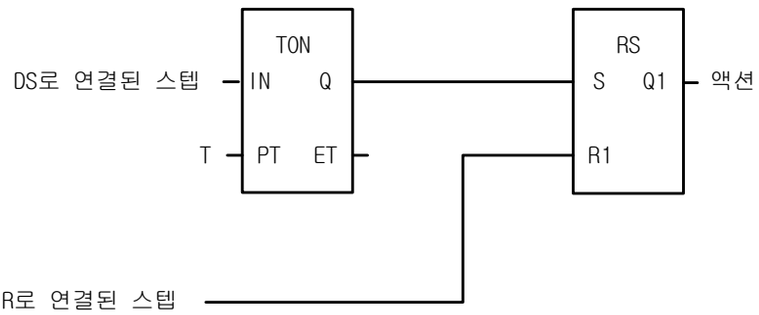
스텝이 활성화 된 후 지정된 시간이 경과한 후부터 R 제한자가 실행될 때까지 액션이 실행됩니다. 단, 시간이 경과하기 전에 R 제한자가 실행되면 액션은 실행되지 않습니다.

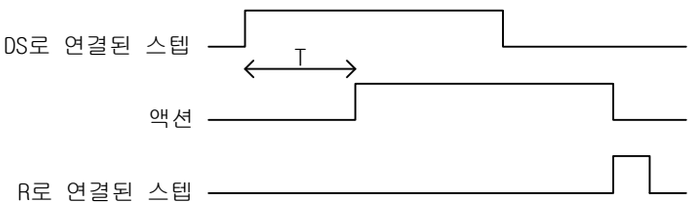




8) DS(Delayed & Stored)

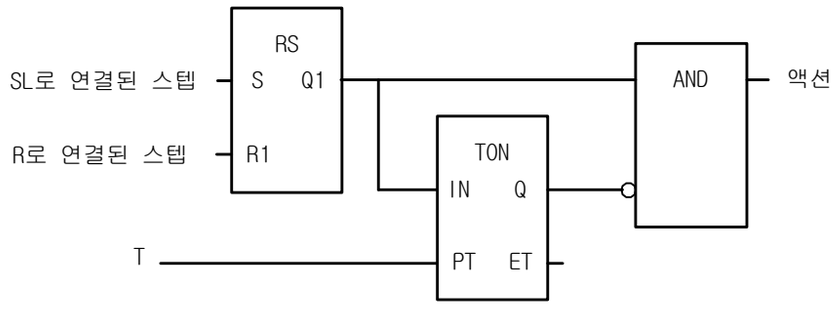
스텝이 활성화 된 후 지정된 시간이 경과한 후부터 R 제한자가 실행될 때까지 액션이 실행됩니다. 단, 시간이 경과하기 전에 스텝이 비활성화되거나 R 제한자가 실행되면 액션은 실행되지 않습니다.

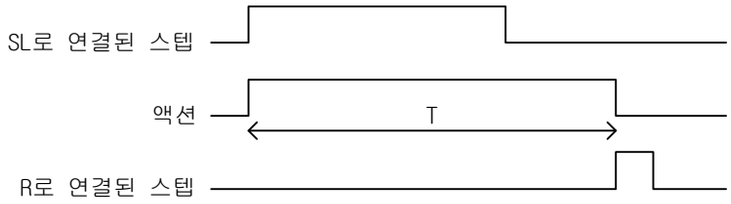




9) SL(Stored & Timed Limited)

스텝이 활성화된 후 지정된 시간까지, 또는 R 제한자가 실행될 때까지 액션이 실행됩니다.



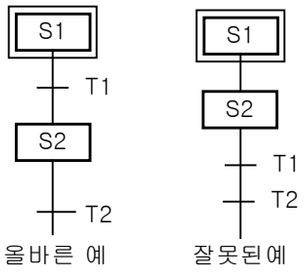


4.3. 전개 규칙

4.3.1. 직렬 연결

▷ 2 개의 스텝은 직접 연결되지 않고 항상 트랜지션에 의해 분리됩니다.

▷ 2 개의 트랜지션은 직접 연결되지 않고 항상 스텝에 의해 분리됩니다.

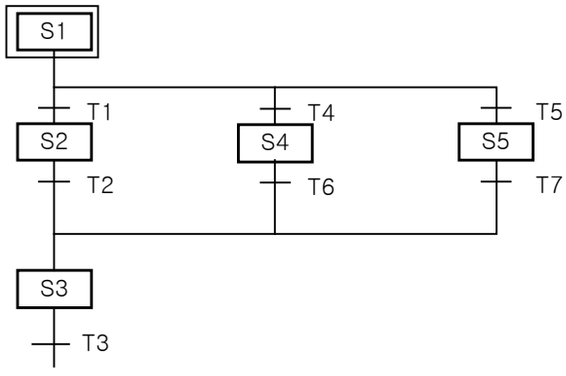


▷ 직렬로 연결되어 있는 스텝간의 이행은 상위 스텝이 활성화된 상태에서 다음에 연결된 트랜지션의 이행 조건이 1 로 되면 하위 스텝이 활성화 상태가 됩니다.

4.3.2. 선택 분기

▷ 선택 분기로 연결되어 있으면 상위 스텝이 활성화된 상태에서 다음에 연결된 2 개 이상의 트랜지션 중 이행 조건이 1 로 된 곳의 다음 스텝이 활성화됩니다. 그 다음은 직렬 연결과 동일합니다.

예



* T1 의 이행 조건이 1 이 되었을 경우
  S1 -> S2 -> S3 순으로 활성화 상태가 됩니다.
* T4 의 이행 조건이 1 이 되었을 경우
  S1 -> S4 -> S3 순으로 활성화 상태가 됩니다.
* T5 의 이행 조건이 1 이 되었을 경우
  S1 -> S5 -> S3 순으로 활성화 상태가 됩니다.

이행 조건이 동시에 1 이 되었을 경우에는 가장 왼쪽에 있는 트랜지션쪽으로 전개됩니다.

* T1, T4 의 이행 조건이 동시에 1 이 되었을 경우 S1 -> S2 -> S3 순으로 활성화 상태가 됩니다.

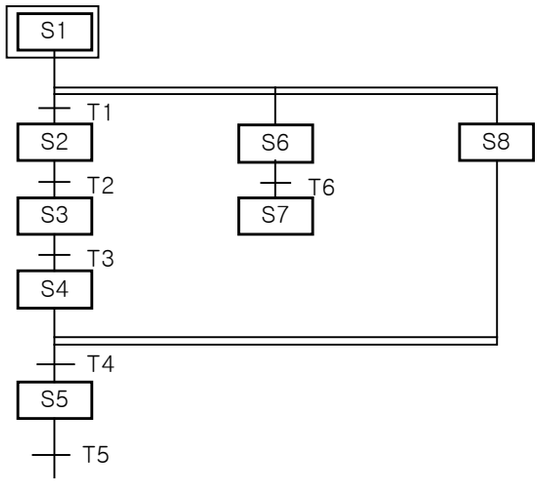
* T4, T5 의 이행 조건이 동시에 1 이 되었을 경우 S1 -> S4 -> S3 순으로 활성화 상태가 됩니다.

4.3.3. 병렬 분기

▷ 병렬 분기로 연결되어 있으면 상위 스텝이 활성화 상태에서 다음에 연결되어있는 트랜지션의 이행 조건이 1 로 되면 이 트랜지션 밑에 연결된 모든 스텝이 활성화 상태로 됩니다. 각 분기의 전개는 직렬연결과 동일합니다. 이때 활성화 상태인 스텝은 분기의 수만큼 존재하게 됩니다.

▷ 병렬 분기에서 합쳐질 경우, 각 분기의 마지막 스텝이 모두 활성화일 경우에 트랜지션의 이행 조건이 1 로 되면 다음에 연결되어 있는 스텝이 활성화 상태로 됩니다.

예



- S1 이 활성화된 상태에서 이행 조건 T1 이 1 이면 S2, S6, S8 이 활성화 상태로 되고, S1 이 비활성화 상태로 됩니다.
- S4, S7, S8 이 활성화된 상태에서 이행 조건 T4 가 1 이면 S5 가 활성화 상태로 되고, S4, S7, S8 은 비활성화 상태로 됩니다.

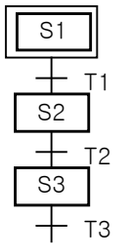
* Active 순서

S1-+->S2---->S3---->S4-+->S5
+->S6---->S7--------+
+->S8--------------+

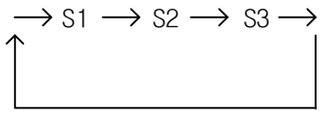
4.3.4. 점프

▷ SFC 마지막 스텝이 활성화 상태로 된 후 다음에 연결되어 있는 트랜지션의 이행 조건이 1 로 되면 SFC 초기 스텝 (Initial Step)이 활성화 상태로 됩니다.

예



- 활성 순서

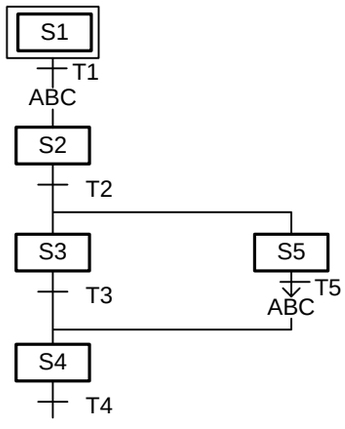


▷ 점프를 사용하면 원하는 곳으로 전개를 이어 나갈 수 있습니다.

▷ 점프는 SFC 프로그램 끝 또는 선택 분기 끝에만 올 수 있습니다.
병렬 분기 안으로 또는 밖으로는 점프할 수 없습니다. 병렬 분기 안에서의 점프는 가능합니다.

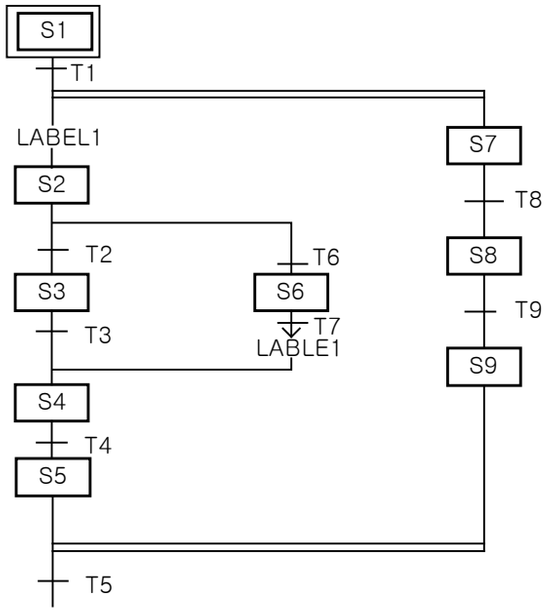
예

1) 선택분기 끝에서의 점프

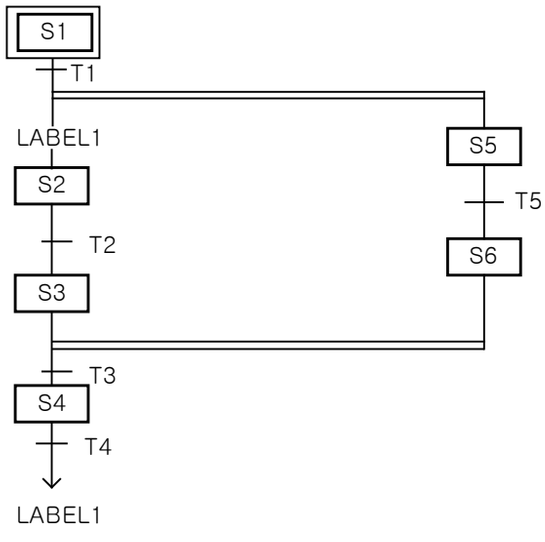


- S5 다음에는 S2 이 활성화됩니다.

2) 병렬 분기 안에서의 점프



3) 병렬 분기 안으로는 점프할 수 없습니다.



4.4. SFC 편집

4.4.1. 프로그램 편집

4.4.1.1. 편집 도구

SFC 편집 요소의 입력은 SFC 도구 모음에서 입력할 요소를 선택한 후 지정한 위치에서 마우스를 클릭하거나 단축키를 눌러 시작합니다.

기호	단축키	설명
	Esc	선택 모드로 변경
	-	스텝+트랜지션 또는 트랜지션+스텝
	-	액션
	-	블록+트랜지션 또는 트랜지션+블록
	-	레이블
	-	점프
	-	왼쪽 분기
	-	오른쪽 분기

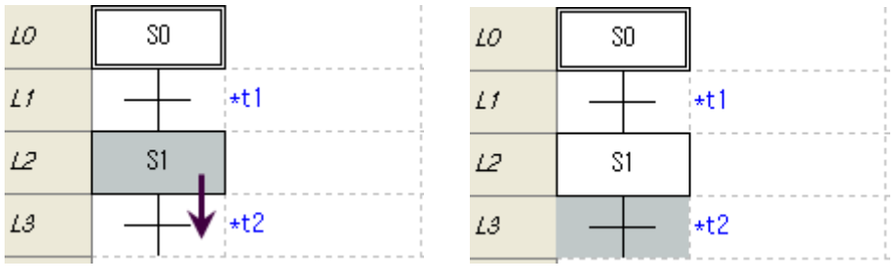
알아두기

- 편집 도구모음의 단축키 표현에서 s는 Shift 키를, c는 Ctrl 키를 a는 Alt 키를 표시합니다.
- 편집 도구에서 설명한 단축키는 XG5000에서 기본으로 제공하는 단축키를 기준으로 설명합니다.
- 사용자 정의 단축키 설정은 제2장 기본사용법의 2.4 단축키 설정하기를 참고하시기 바랍니다.

4.4.1.2. 스텝/트랜지션 입력

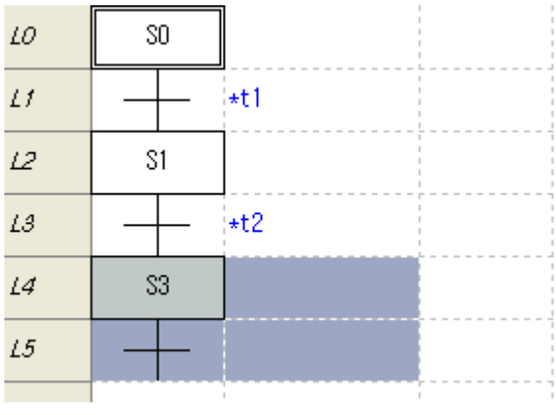
스텝/트랜지션을 입력합니다.

[순서]
1. 입력하고자 하는 위치로 커서를 이동시킵니다.



2. 도구 모음에서 스텝을 선택하고 편집 영역을 클릭합니다. 또는 입력하고자 하는 접점에 해당하는 단축키를 누릅니다.

3. 새로운 스텝/트랜지션이 입력됩니다.



알아두기

- 스텝의 이름은 기본적으로 자동으로 주어집니다. 필요에 따라 바꿀 수 있습니다.
- 입력 위치에 따라 스텝 + 트랜지션 또는 트랜지션 + 스텝으로 입력됩니다.

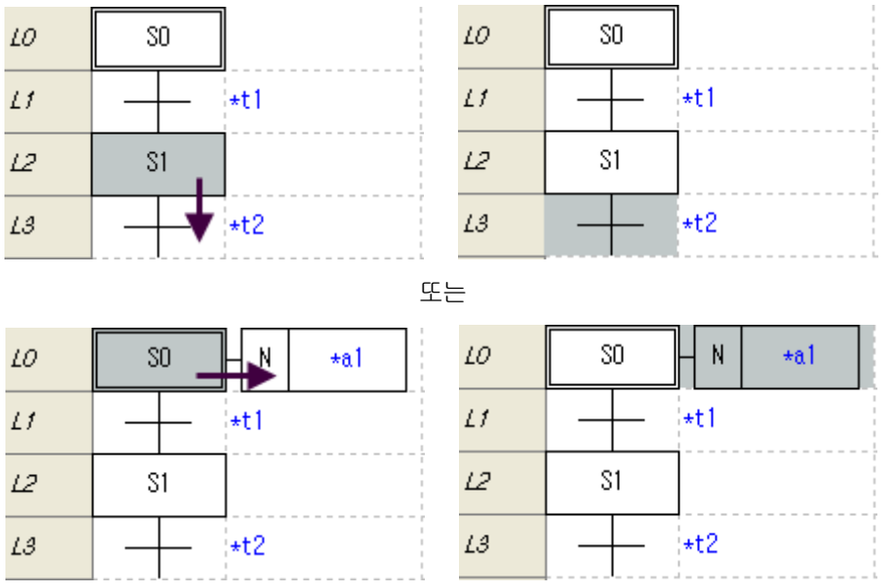
	선택된 위치의 항목
스텝 + 트랜지션	트랜지션, 병렬분기 시작 라인, 레이블, 선택분기 종료 라인
트랜지션 + 스텝	스텝, 블록, 선택분기의 시작 라인, 병렬분기 종료 라인

-

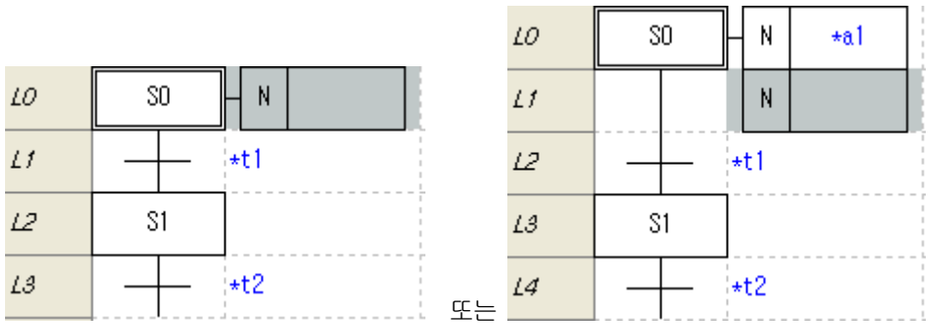
4.4.1.3. 액션 입력

액션은 연결된 스텝이 활성화 되었을 때 수행합니다. 액션에 연결된 프로그램을 수행합니다.

[순서]
1. 입력하고자 하는 위치로 커서를 이동 시킵니다.



2. 액션 입력 단축키를 선택합니다. 또는 도구 모음에서 액션을 선택하고 입력할 편집 영역을 선택합니다.



알아두기

- 액션은 선택된 항목에 따라 입력되는 위치가 다릅니다.

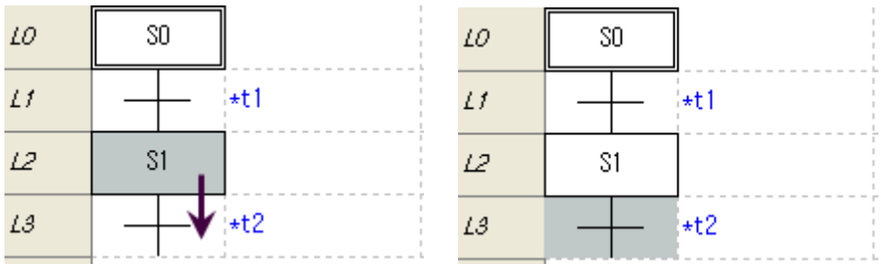
위치	선택된 위치의 항목
오른쪽	스텝 선택 시 스텝의 오른쪽
아래쪽	액션 선택 시 액션의 아래쪽

- 하나의 스텝에 연결된 액션의 개수 제한이 없습니다.
- 액션은 블록에는 연결될 수 없습니다.

4.4.1.4. 블록/트랜지션 입력

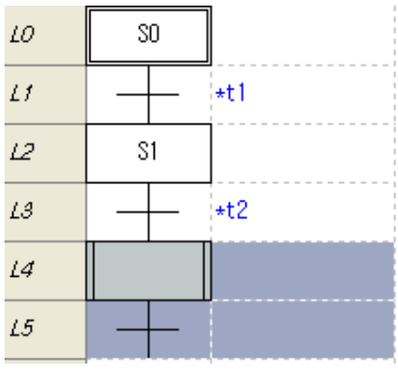
블록/트랜지션을 입력합니다.

[순서]
1. 입력하고자 하는 위치로 커서를 이동시킵니다.



2. 도구 모음에서 블록을 선택하고 편집 영역을 클릭합니다. 또는 해당하는 단축키를 누릅니다.

3. 새로운 블록/트랜지션이 입력됩니다.



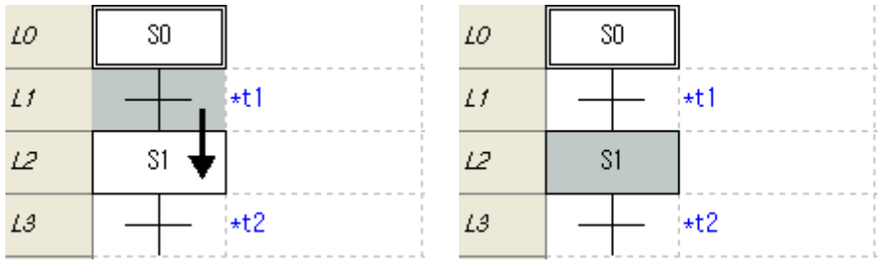
알아두기

- 블록 입력 위치 기준은 스텝 입력과 동일합니다.

4.4.1.5. 레이블 입력

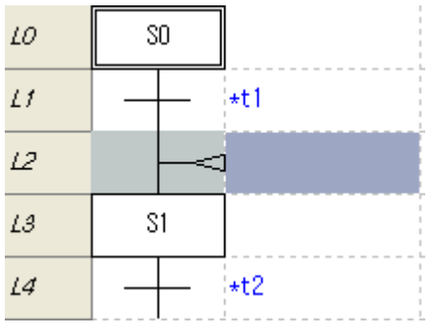
레이블을 입력합니다.

[순서]
1. 입력하고자 하는 위치로 커서를 이동시킵니다.



2. 도구 모음에서 레이블을 선택하고 편집 영역을 클릭합니다. 또는 입력 단축키를 누릅니다.

3. 새로운 레이블이 입력됩니다.



알아두기

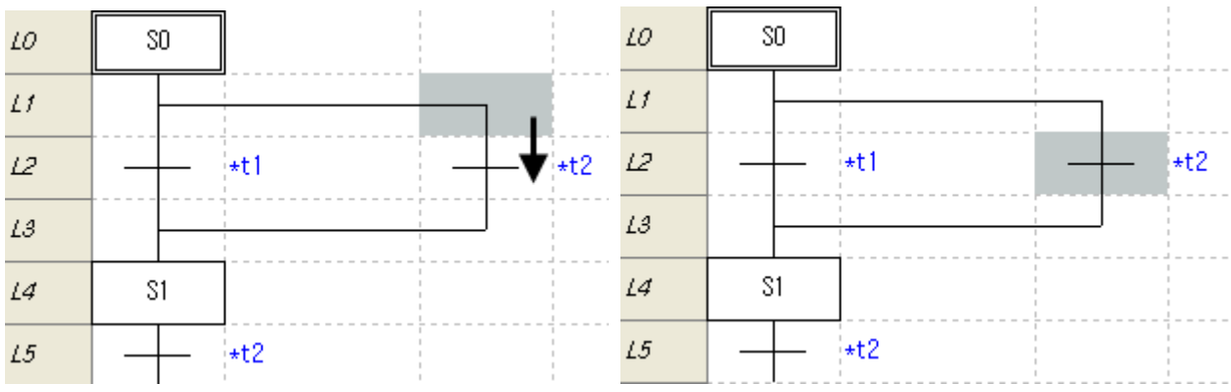
- 레이블은 스텝 또는 블록을 선택해야만 입력할 수 있습니다.
- 레이블은 선택된 스텝 또는 블록 이전에 입력됩니다

4.4.1.6. 점프 입력

점프를 입력합니다.

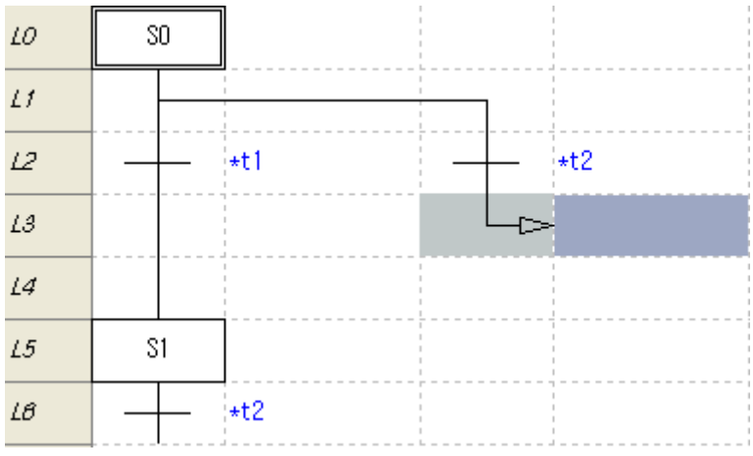
[순서]

1. 입력하고자 하는 위치로 커서를 이동시킵니다.



2. 도구 모음에서 점프를 선택하고 편집 영역을 클릭합니다. 또는 입력 단축키를 누릅니다.

3. 새로운 점프가 입력됩니다.



알아두기

- 점프는 선택분기 내의 마지막 트랜지션을 선택하거나, 프로그램 마지막 열의 트랜지션을 선택해야만 입력할 수 있습니다.
- 선택된 트랜지션 다음에 점프가 입력됩니다.
- 같은 프로그램 내로만 점프할 수 있습니다.

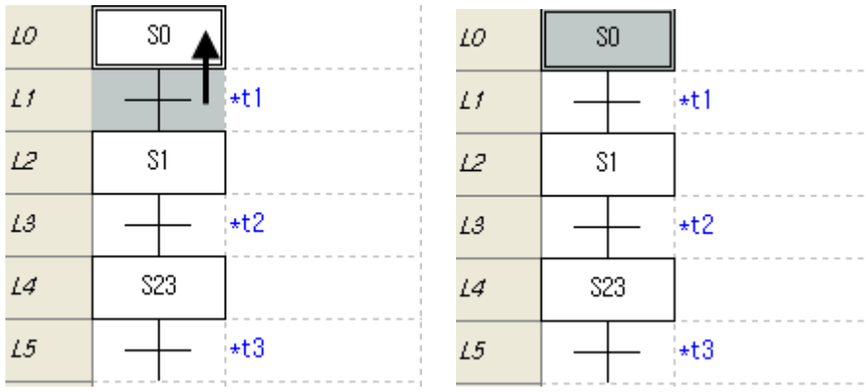
4.4.1.7. 왼쪽 분기 입력

왼쪽 분기를 입력합니다. 선택분기 만들기를 예로 듭니다.

1) 분기 만들기

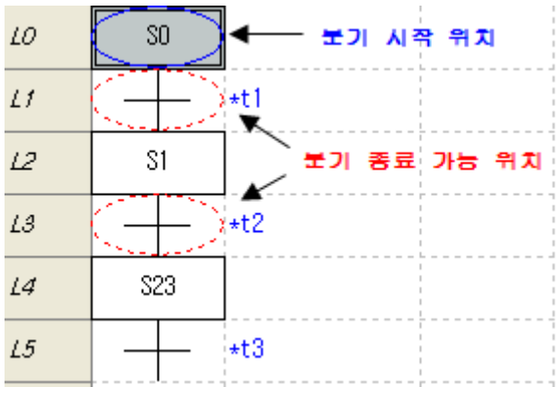
[순서]

1. 분기를 시작고자 하는 위치로 커서를 이동시킵니다.

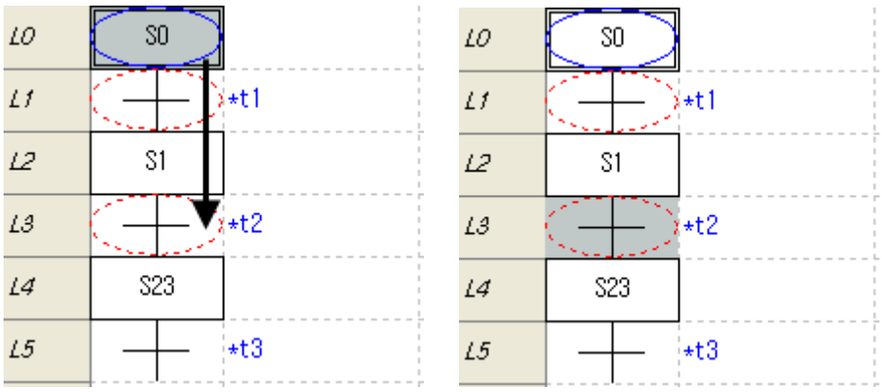


2. 도구 모음에서 왼쪽 분기를 선택하고 편집 영역을 클릭합니다. 또는 입력 단축키를 누릅니다.

3. 분기 시작 위치와 분기 연결 가능 위치를 표시합니다.

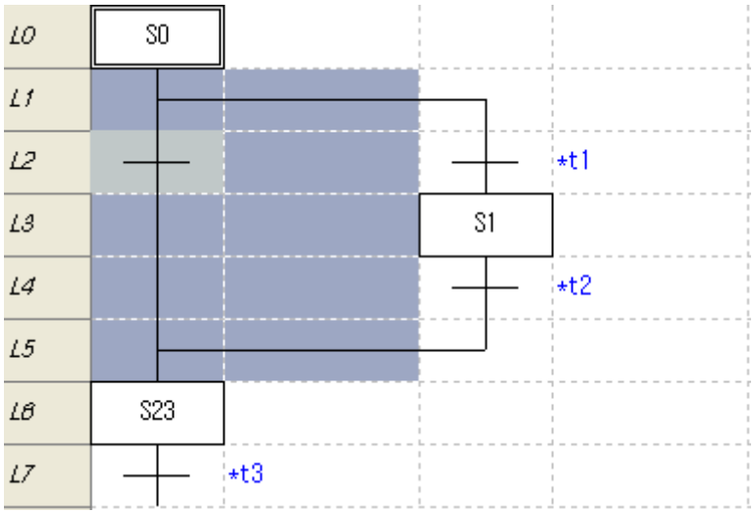


4. 분기 종료 위치로 커서를 이동 시킵니다.



5. 분기 종료 위치를 선택합니다.

6. 분기가 만들어 집니다.



알아두기

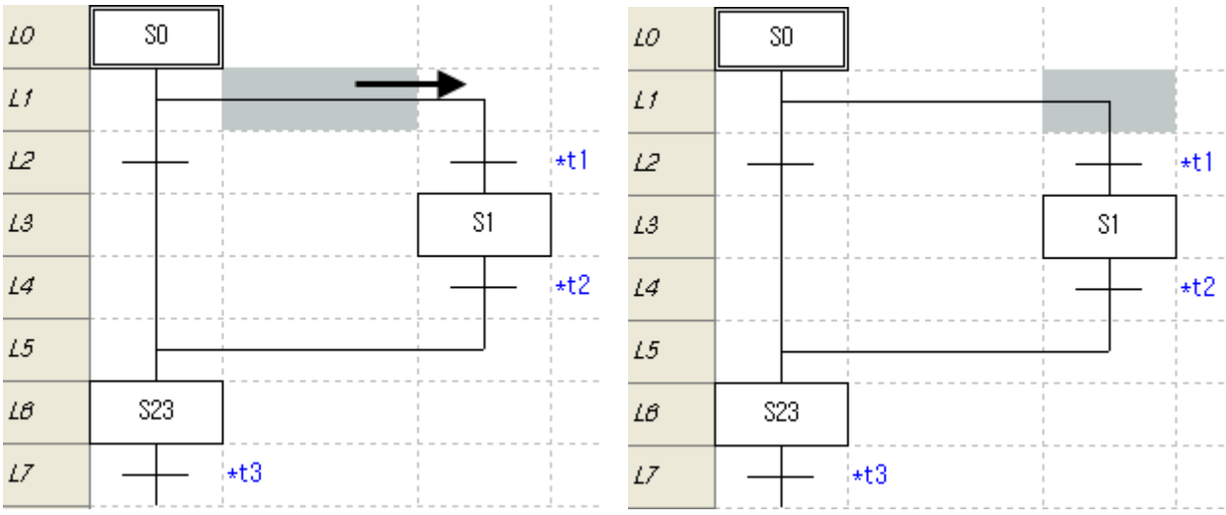
- 분기 시작 선택 위치에 따라 선택분기 또는 병렬분기가 생성됩니다.

	분기 시작 위치 위치의 항목	선 모양
선택분기	스텝, 블록	한 줄 가로선
병렬분기	트랜지션	두 줄 가로선

2) 분기 증가 시키기

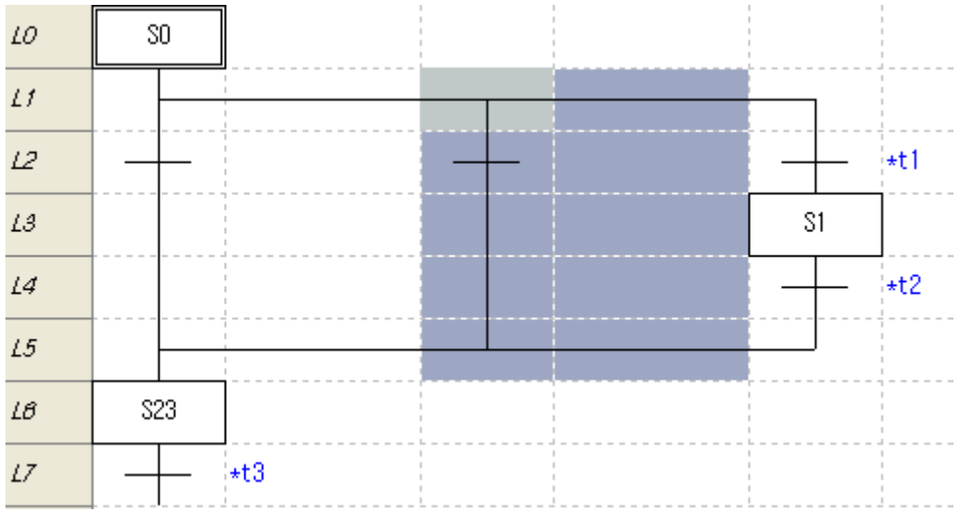
[순서]

1. 분기 증가 시킬 위치로 커서를 이동합니다.



2. 도구 모음에서 왼쪽 분기를 선택하고 편집 영역을 클릭합니다. 또는 입력 단축키를 누릅니다.

3. 새로운 분기가 입력됩니다.



알아두기

- 분기의 증가 개수는 제한이 없지만 가로열의 수 제한에 걸려 더 이상 분기 증가를 할 수 없을 수 있습니다.
- 선택분기에 병렬분기로 또는 병렬분기에 선택분기로 분기가 증가될 수는 없습니다.

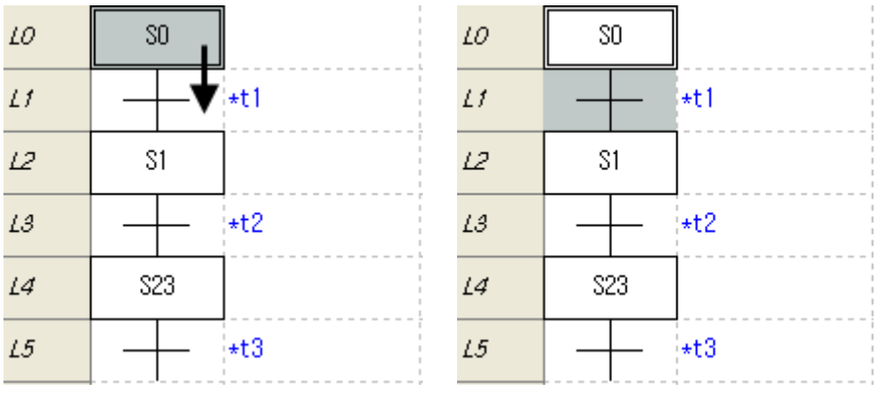
4.4.1.8. 오른쪽 분기 입력

오른쪽 분기를 입력합니다. 병렬분기를 예로 듭니다.

1) 분기 만들기

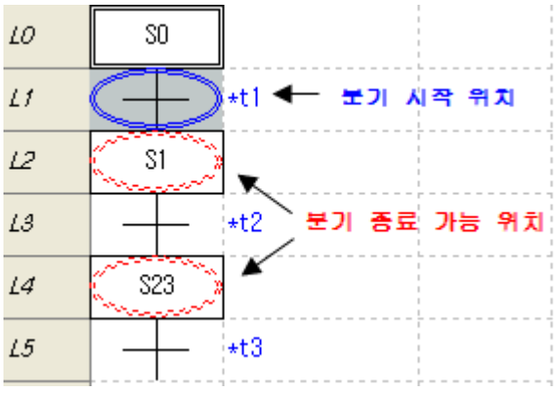
[순서]

1. 분기를 시작고자 하는 위치로 커서를 이동시킵니다.

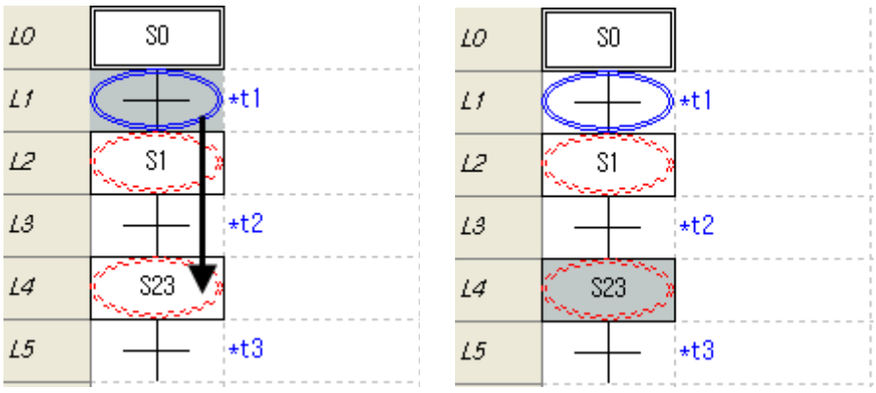


2. 도구 모음에서 오른쪽 분기를 선택하고 편집 영역을 클릭합니다. 또는 입력 단축키를 누릅니다.

3. 분기 시작 위치와 분기 연결 가능 위치를 표시합니다.

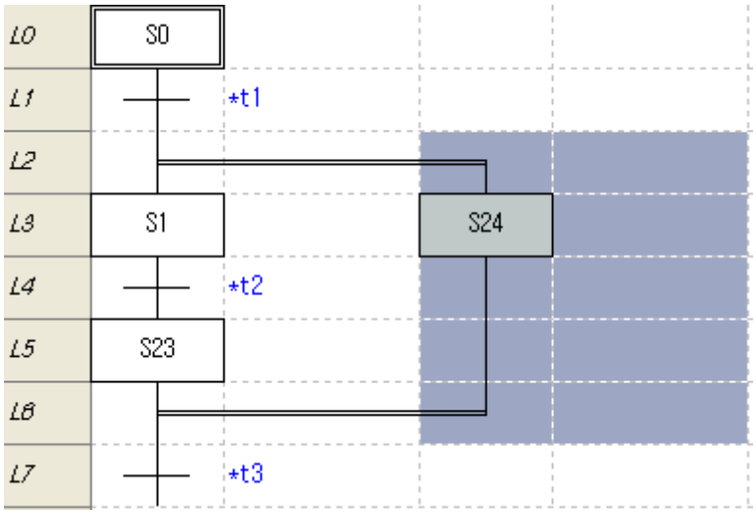


4. 분기 종료 위치로 커서를 이동 시킵니다.



5. 분기 종료 위치를 선택합니다.

6. 분기가 만들어 집니다.



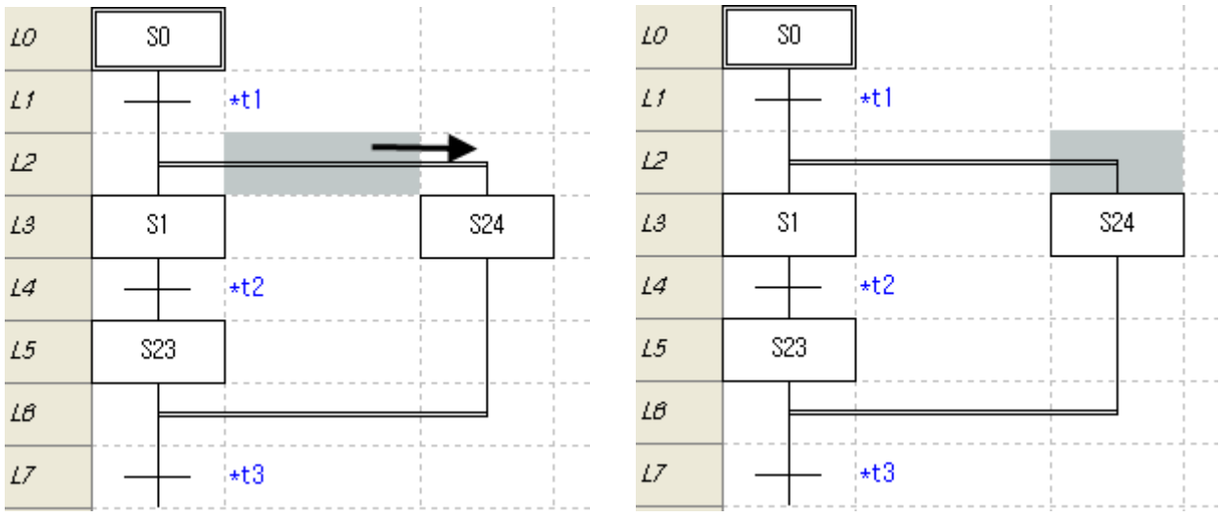
알아두기

- 선택분기 또는 병렬분기 생성 원칙은 왼쪽 분기와 동일하므로 6장 2.7 절 “왼쪽분기 입력” 부분을 참조하시기 바랍니다.

2) 분기 증가 시키기

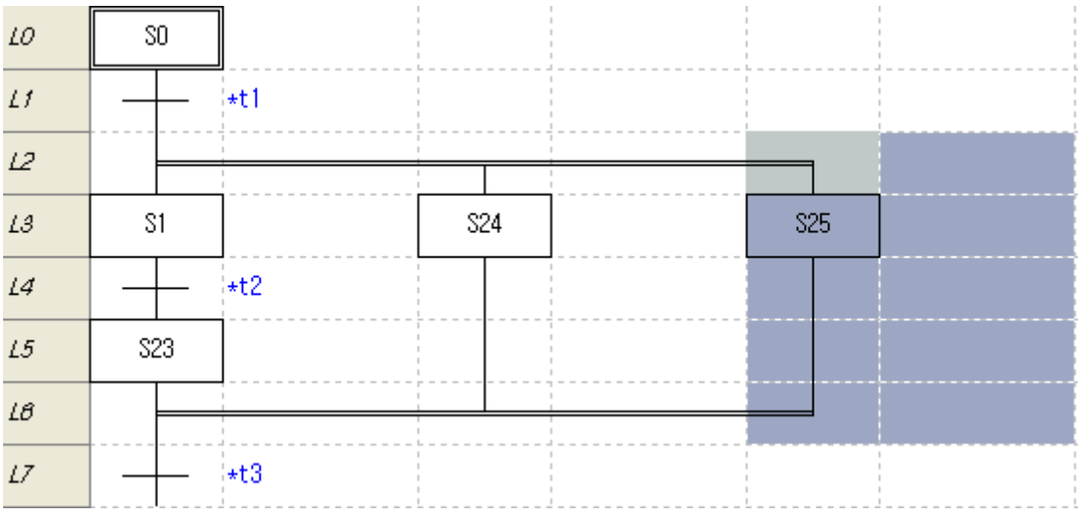
[순서]

1. 분기 증가 시킬 위치로 커서를 이동합니다.



2. 도구 모음에서 오른쪽 분기를 선택하고 편집 영역을 클릭합니다. 또는 입력 단축키를 누릅니다.

3. 새로운 분기가 입력됩니다.

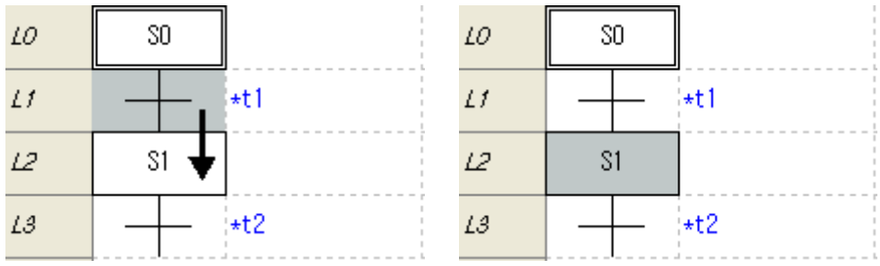


4.4.1.9. 스텝 등록 정보 편집

스텝의 이름, 설명문 등의 내용을 변경합니다.

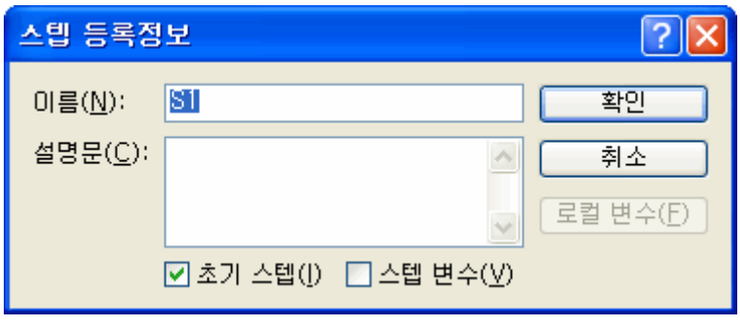
[순서]

1. 편집하고자 하는 위치로 커서를 이동합니다.



2. Enter 키 또는 마우스 왼쪽 더블 클릭합니다.

[대화 상자]



[대화 상자 설명]

a. 이름: 스텝의 이름을 입력합니다.
b. 설명문: 스텝의 설명문을 입력합니다.
c. 초기 스텝: 초기 스텝으로 지정합니다.
d. 스텝 변수: 스텝 이름을 로컬 변수 목록에서 참조합니다.
e. 로컬 변수: 로컬 변수 목록에서 변수를 지정할 수 있습니다.
f. 확인: 입력한 내용을 저장하고 대화상자를 닫습니다.
g. 취소: 대화상자를 닫습니다.

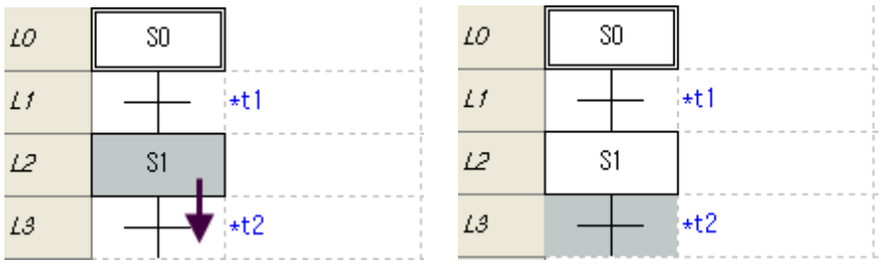
알아두기

- 초기 스텝은 프로그램의 수행 시작 스텝입니다. 프로그램 내에서는 하나만 지정할 수 있습니다. 프로그램 검사 시 오류를 검사합니다.
- 스텝 변수를 사용시 다른 프로그램에서 스텝 이름의 변수를 참조할 수 있습니다.

4.4.1.10. 트랜지션 등록 정보 편집

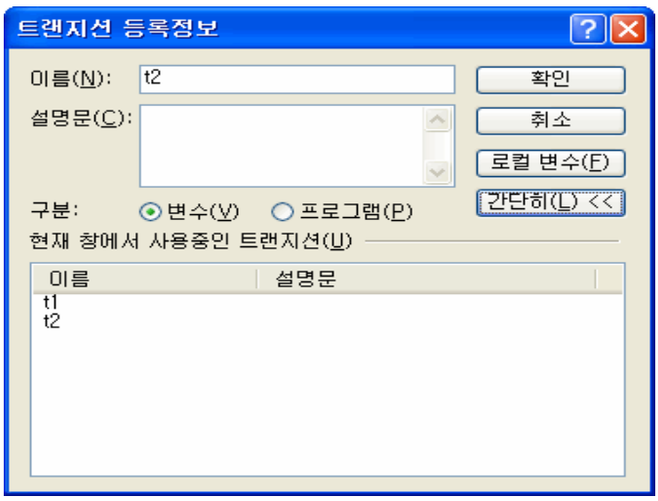
트랜지션의 이름, 설명문 등의 내용을 변경합니다.

[순서]
1. 편집하고자 하는 위치로 커서를 이동합니다.



2. Enter 키 또는 마우스 왼쪽 더블 클릭합니다.

[대화 상자]



[대화 상자 설명]
a. 이름: 트랜지션의 이름을 입력합니다.
b. 설명문: 트랜지션의 설명문을 입력합니다.
c. 구분: 트랜지션을 변수 또는 프로그램으로 지정할 수 있습니다.
d. 목록: 현재 프로그램 창에서 사용중인 다른 트랜지션을 보여줍니다.
e. 로컬 변수: 로컬 변수 목록에서 변수를 지정할 수 있습니다.
f. 간단히: 목록을 숨깁니다.
g. 확인: 입력한 내용을 저장하고 대화상자를 닫습니다.
h. 최소: 대화상자를 닫습니다.

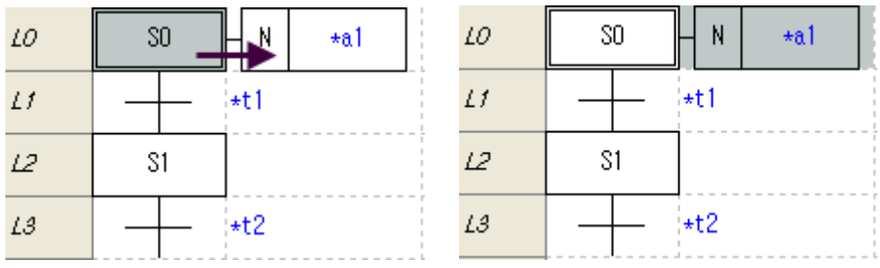
알아두기

- 트랜지션 프로그램은 LD로만 작성할 수 있습니다.

4.4.1.11. 액션 등록 정보 편집

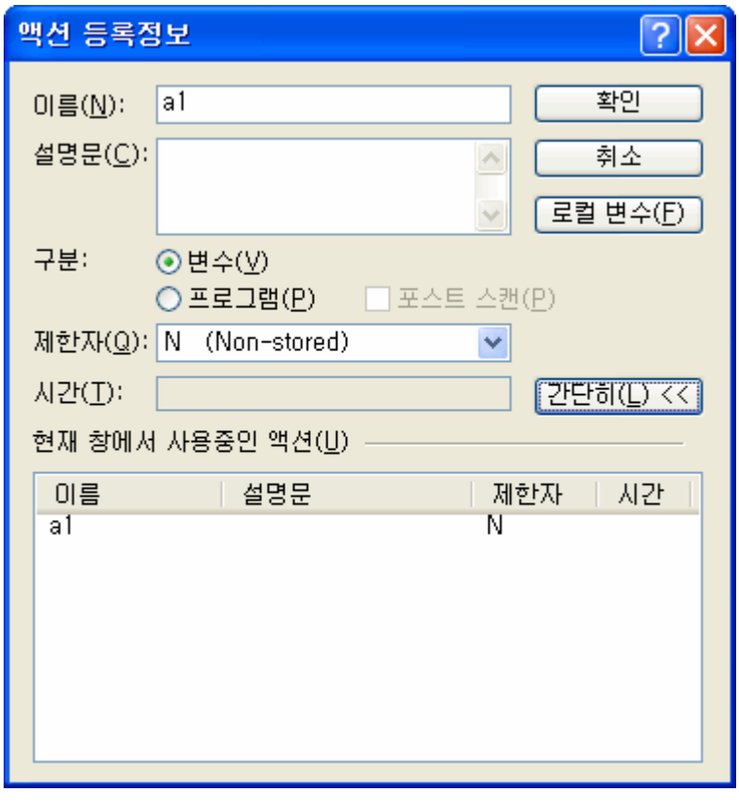
액션의 이름, 설명문, 제한자 등의 내용을 변경합니다.

[순서]
1. 편집하고자 하는 위치로 커서를 이동합니다.



2. Enter 키 또는 마우스 왼쪽 더블 클릭합니다.

[대화 상자]



[대화 상자 설명]

a. 이름: 액션의 이름을 입력합니다.

b. 설명문: 액션의 설명문을 입력합니다.

c. 구분: 액션을 변수 또는 프로그램으로 지정할 수 있습니다.

d. 포스트 스캔: 액션 수행한 후 액션 내 코일을 비 활성화 합니다.

e. 목록: 현재 프로그램 창에서 사용중인 다른 트랜지션을 보여줍니다.

f. 제한자: 액션 제한자에 따라 다른 동작을 할 수 있습니다.

g. 시간: 제한자에 따라 시간 값을 입력해야 합니다.

h. 로컬 변수: 로컬 변수 목록에서 변수를 지정할 수 있습니다.

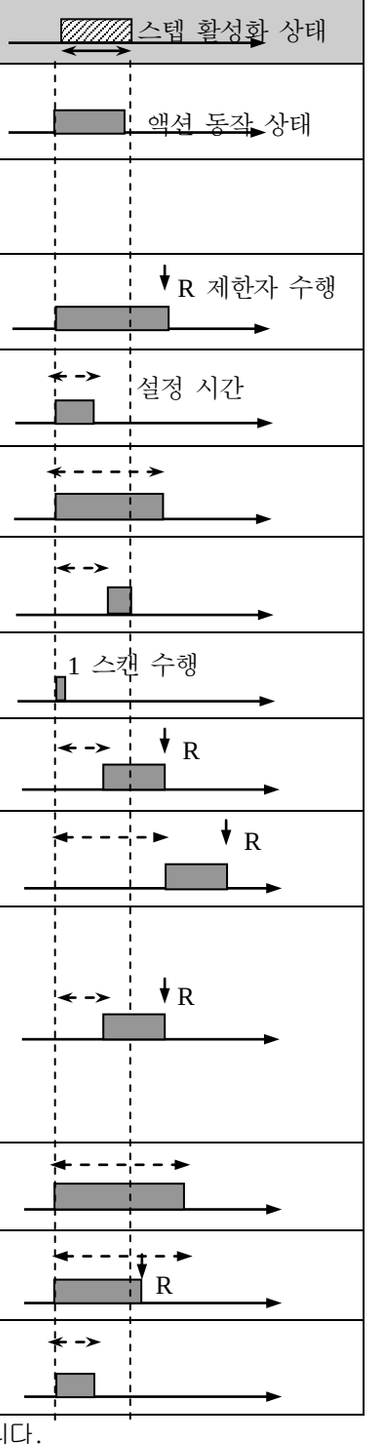
i. 간단히: 목록을 숨깁니다.

j. 확인: 입력한 내용을 저장하고 대화상자를 닫습니다.

k. 최소: 대화상자를 닫습니다.

알아두기

- 액션은 제한자에 따라 동작 특성이 달라집니다.

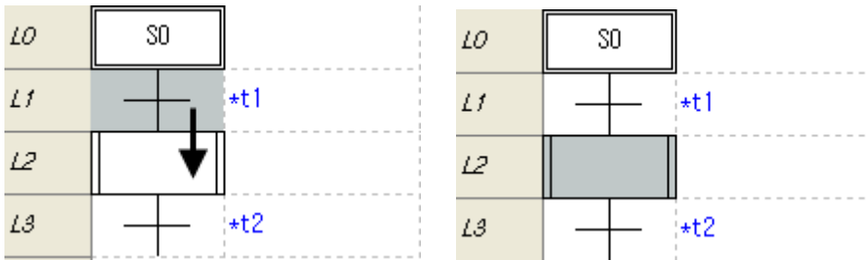
제한자	기능	동작 특성
N (Non-stored)	스텝이 활성화 되어 있을 때만 액션이 수행됩니다.	
R (overriding Reset)	스텝이 활성화 되면 S, SD, DS, SL 제한자로 실행된 같은 액션의 수행을 중지 합니다.	
S (Set)	스텝이 활성화 되면 R 제한자가 실행될 때까지 수행됩니다.	
L (time Limited)	스텝이 활성화 된 시점부터 설정된 시간만큼 액션이 수행됩니다.	
D (time Delayed)	스텝이 활성화 된 시점부터 설정된 시간 경과 후에 액션이 수행됩니다.	
P (Pulse)	스텝이 활성화된 순간만 액션이 수행됩니다.	
SD (Stored & time Delay)	스텝이 활성화 된 후 시점부터 설정된 시간 경과 후부터 R 제한자가 실행될 때까지 액션이 수행됩니다. 단 설정된 시간 이전에 R 제한자가 수행되면 액션은 수행되지 않습니다.	
DS (time Delayed & Stored)	스텝이 활성화 된 후 시점부터 설정된 시간 경과 후부터 R 제한자가 실행될 때까지 액션이 수행됩니다. 단 설정된 시간 이전에 스텝이 비 활성화 되거나 R 제한자가 수행되면 액션은 수행되지 않습니다.	
SL (Stored & time Limited)	스텝이 활성화 된 시점부터 설정된 시간만큼 액션이 수행되고 설정시간 종료 또는 R 제한자가 실행될 때 액션이 종료합니다.	

- 시간 값은 "T#1h2m3s" "T#2m" "T#15s"와 같은 표현식을 사용할 수 있습니다.
- 액션 프로그램은 LD 와 SFC 로 작성할 수 있습니다.

4.4.1.12. 블록 등록 정보 편집

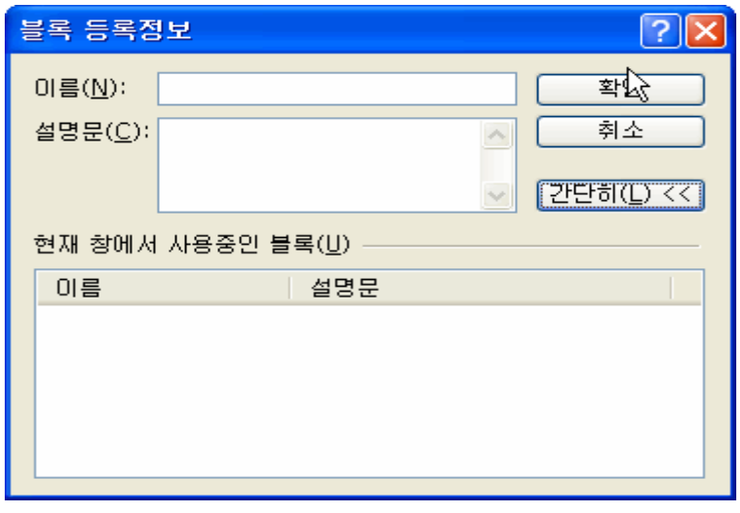
블록의 이름, 설명문 등의 내용을 변경합니다.

[순서]
1. 편집하고자 하는 위치로 커서를 이동합니다.



2. Enter 키 또는 마우스 왼쪽 더블 클릭합니다.

[대화 상자]



[대화 상자 설명]
a. 이름: 블록의 이름을 입력합니다.
b. 설명문: 블록의 설명문을 입력합니다.
c. 설명문: 블록의 설명문을 입력합니다.
d. 목록: 현재 프로그램 창에서 사용중인 다른 블록을 보여줍니다.
e. 간단히: 목록을 숨깁니다.
f. 확인: 입력한 내용을 저장하고 대화상자를 닫습니다.
g. 최소: 대화상자를 닫습니다.

알아두기

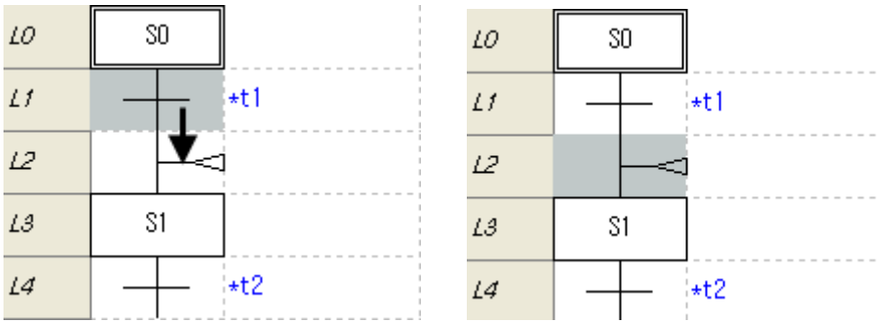
- 블록은 SFC 프로그램으로만 내용을 작성할 수 있습니다.
- 블록에는 액션이 연결되지 않습니다.

4.4.1.13. 레이블 등록 정보 편집

레이블의 이름을 변경합니다.

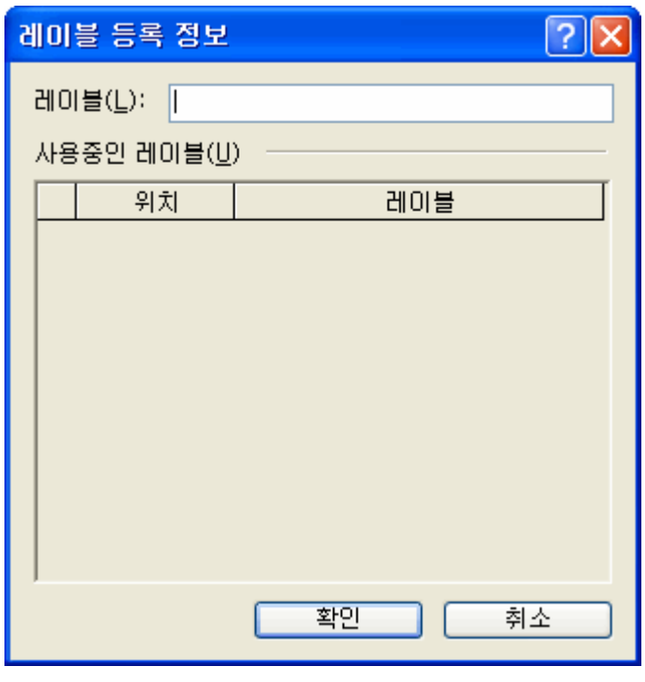
[순서]

1. 편집하고자 하는 위치로 커서를 이동합니다.



2. Enter 키 또는 마우스 왼쪽 더블 클릭합니다.

[대화 상자]



[대화 상자 설명]

a. 레이블: 레이블의 이름을 입력합니다.

b. 목록: 현재 프로그램 창에서 사용중인 다른 레이블을 보여줍니다.

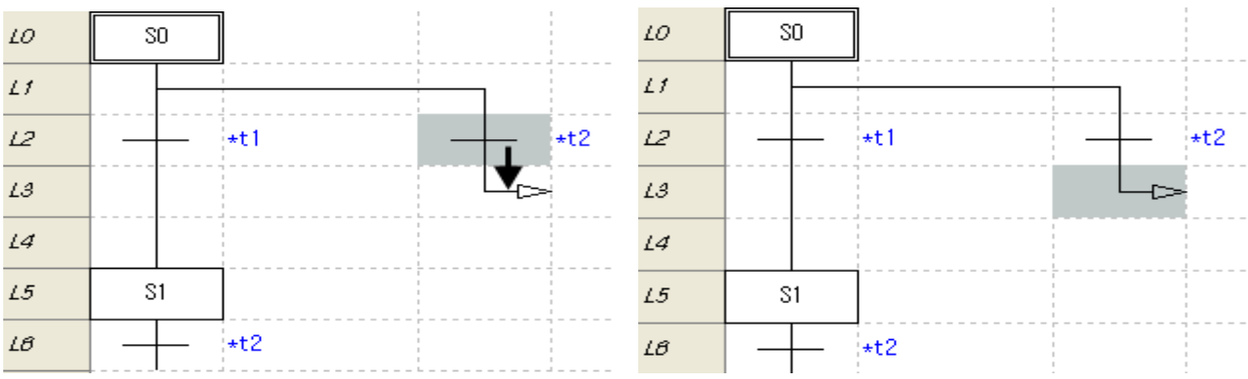
c. 확인: 입력한 내용을 저장하고 대화상자를 닫습니다.

d. 최소: 대화상자를 닫습니다.

4.4.1.14. 점프 등록 정보

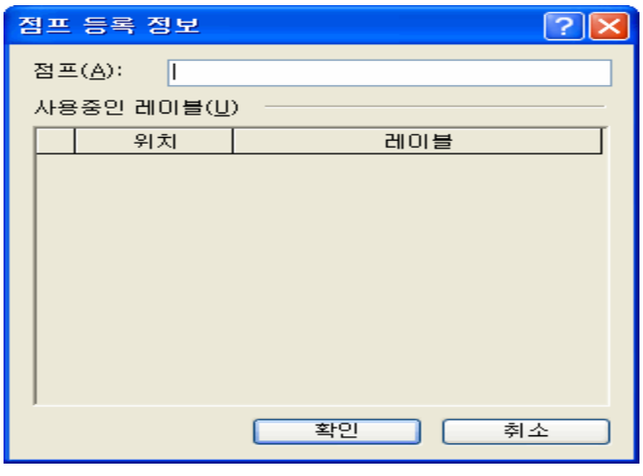
점프의 이름을 변경합니다.

[순서]
1. 편집하고자 하는 위치로 커서를 이동합니다.



2. Enter 키 또는 마우스 왼쪽 더블 클릭합니다.

[대화 상자]



[대화 상자 설명]

a. 점프: 점프의 이름을 입력합니다.

b. 목록: 현재 프로그램 창에서 사용중인 다른 레이블을 보여줍니다.

c. 확인: 입력한 내용을 저장하고 대화상자를 닫습니다.

d. 최소: 대화상자를 닫습니다.

알아두기

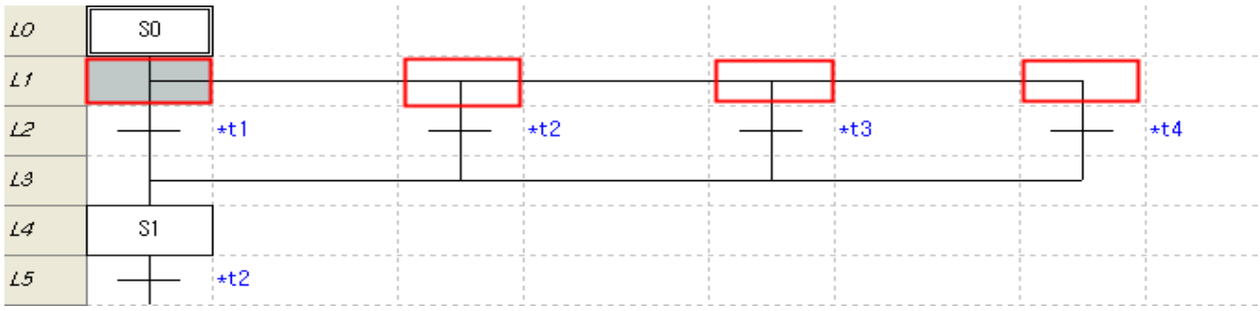
- 점프 이름은 레이블 이름과 동일 해야 합니다.
- 다른 SFC 프로그램으로는 점프 할 수 없습니다.

4.4.1.15. 선택 분기 우선 순위 설정

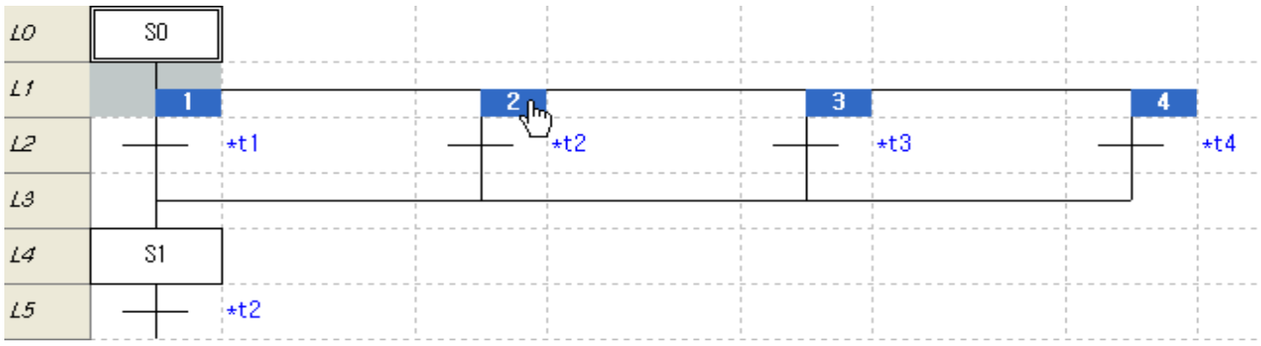
입력된 선택 분기의 프로그램 수행 우선 순위를 설정할 수 있습니다.

[순서]

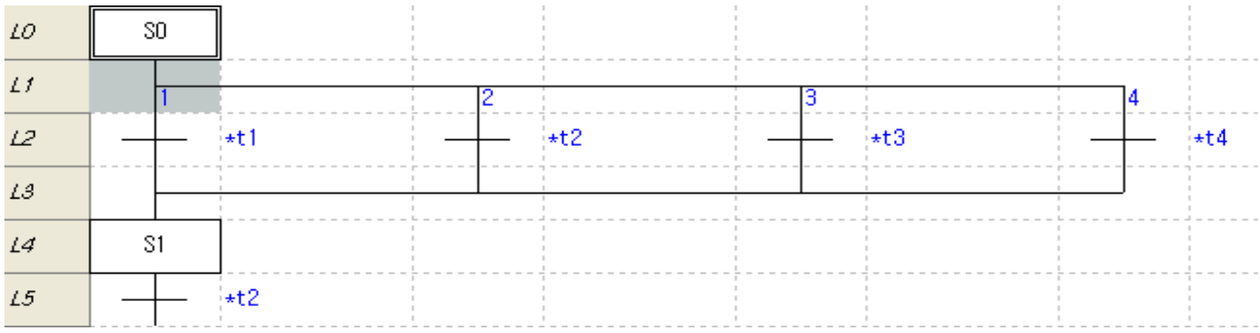
1. 선택 분기의 시작 위치로 커서를 이동합니다.



2. 메뉴 [편집] - [선택분기 우선순위 설정]을 선택합니다.
3. 선택 분기의 시작 위치를 마우스로 차례로 왼쪽 클릭하여 원하는 우선순위로 지정합니다.



4. 선택 분기의 시작 위치 이외의 영역을 클릭하면 편집이 끝납니다.



알아두기

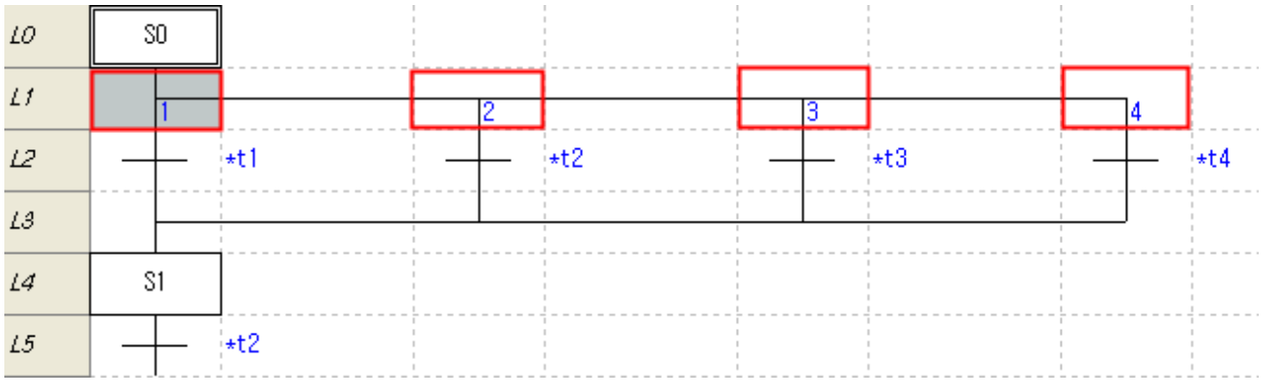
- 선택 분기의 우선 순위는 분기의 수만큼 지정할 수 있습니다.
- 선택 분기 우선 순위가 지정되지 않은 경우는 왼쪽에서 오른쪽 방향으로 프로그램 컴파일 됩니다. 즉 왼쪽부터 프로그램 수행됩니다.

4.4.1.16. 선택 분기 우선 순위 해제

입력된 선택 분기 우선 순위를 제거합니다.

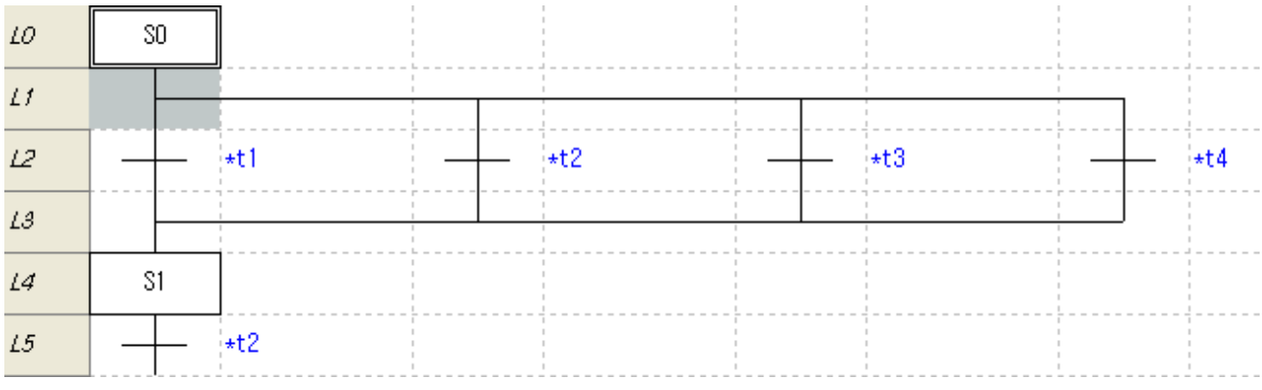
[순서]

1. 선택 분기 우선 순위가 지정된 선택 분기의 시작 위치를 선택합니다.



2. 메뉴 [편집] - [선택분기 우선순위 해제]을 선택합니다.

3. 선택 분기 우선 순위가 해제됩니다.



4.4.1.17. 요소 삭제

입력한 스텝, 트랜지션, 액션, 블록, 레이블, 점프, 분기를 삭제합니다.

[순서]
1. 삭제하고자 하는 요소 위치로 커서를 이동시킵니다.
2. 메뉴 [편집]-[삭제]를 선택합니다.

알아두기

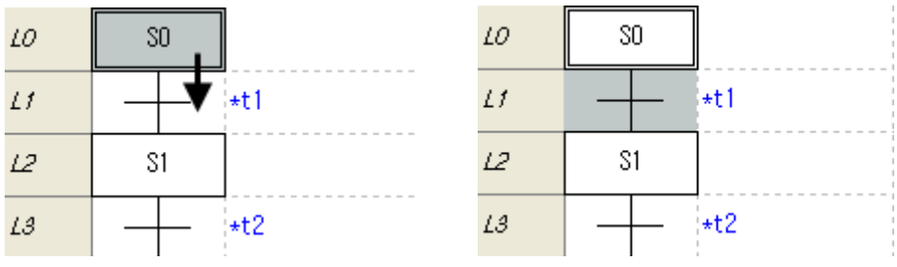
- 스텝, 트랜지션, 블록은 선택된 위치에 따라 아래 위치의 요소가 같이 삭제됩니다.
- 삭제되지 않는 경우도 있습니다. 이 경우 경고 메시지 후 자동 편집 취소됩니다.
- 분기의 시작을 선택하고 삭제 시 분기 아래 모든 내용이 삭제됩니다.

4.4.1.18. 복사/잘라내기/붙여넣기

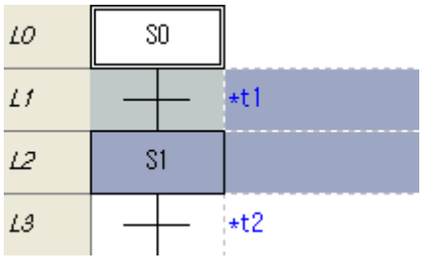
선택된 영역의 데이터를 복사하거나, 잘라내어 지정한 위치로 붙여 넣기 할 수 있습니다. 복사와 다르게 잘라내기는 현재 선택된 영역의 데이터를 삭제합니다.

1) 복사/붙여넣기

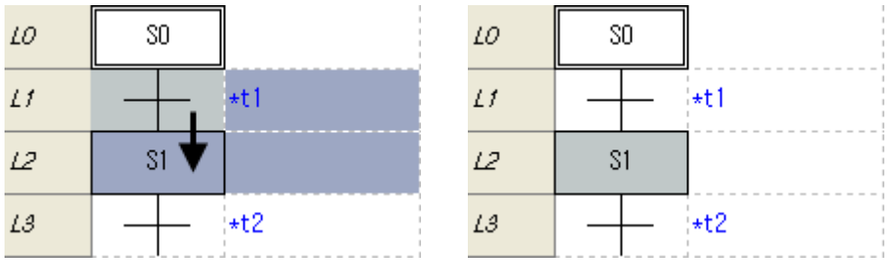
[순서]
1. 복사하고자 하는 영역을 선택합니다.



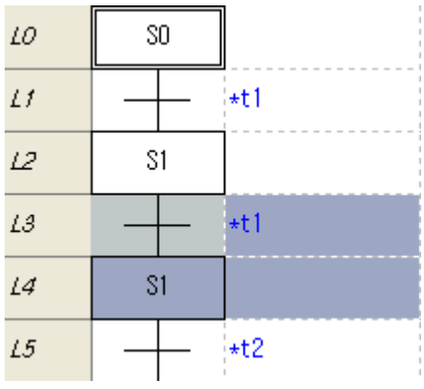
2. 메뉴 [편집]-[복사]를 선택합니다
.
3. 복사된 영역이 표시됩니다.



4. 붙여넣고자 하는 위치로 커서를 이동시킵니다.



5. 메뉴 [편집]-[붙여넣기]를 선택합니다.



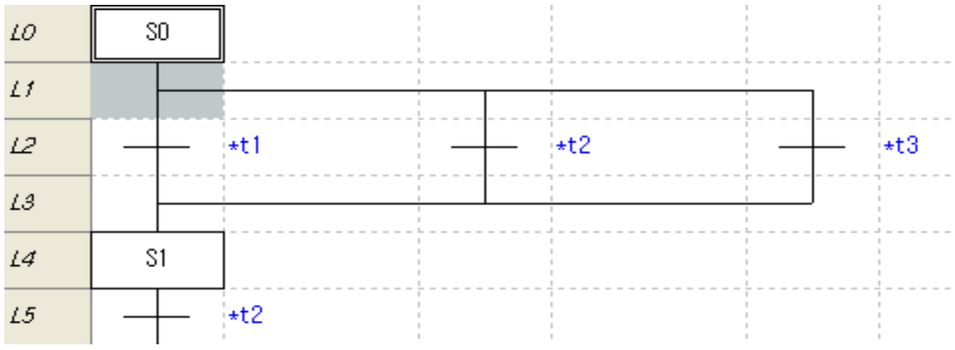
알아두기

- 스텝, 트랜지션, 블록은 선택된 위치에 따라 아래 위치의 요소가 같이 복사됩니다.
- 붙여넣기 위치에 따라 붙여넣기 안 되는 경우도 있습니다. 이때는 자동 편집 취소됩니다.
- 분기의 시작을 선택하고 복사 시 분기 아래 모든 내용이 같이 복사됩니다.
- 분기의 붙여넣기는 분기의 시작 위치를 선택하고 붙여넣기 해야 가능합니다.
- 선택분기는 선택분기에만 병렬분기는 병렬분기에만 붙여넣기 할 수 있습니다.
- 액션은 스텝을 선택하고 붙여넣기 할 수 있습니다.
- 붙여넣기의 원칙은 요소 입력과 같은 동작을 하는 것입니다.

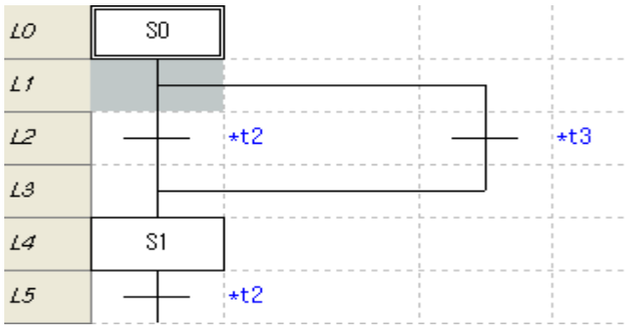
2) 잘라내기/붙여넣기

[순서]

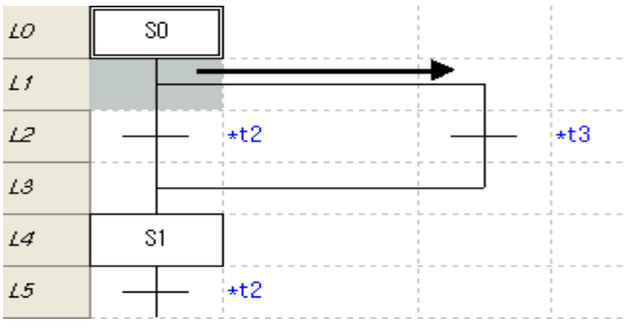
1. 잘라내고자 하는 영역을 선택합니다.

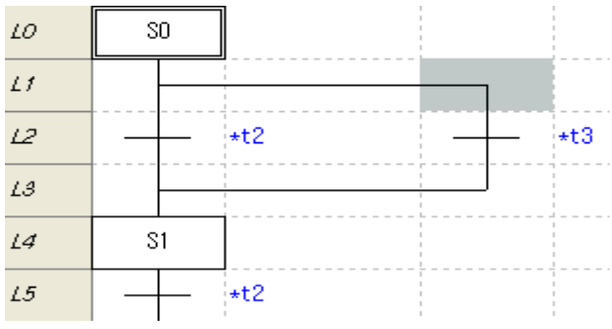


2. 메뉴 [편집]-[잘라내기]를 선택합니다.

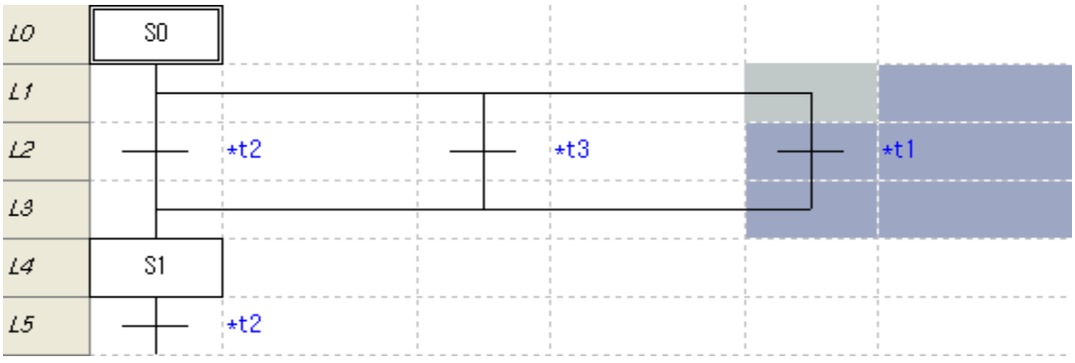


3. 붙여 넣고자 하는 영역으로 커서를 이동시킵니다.





4. 메뉴 [편집]-[붙여넣기]를 선택합니다.

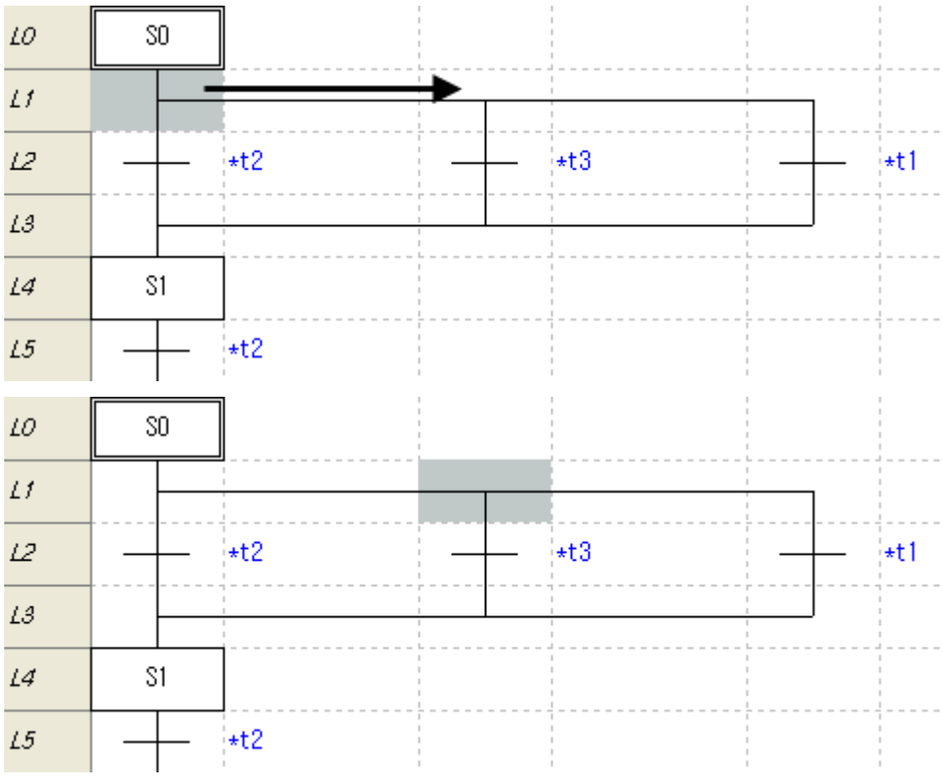


4.4.1.19. 편집 취소 및 재 실행

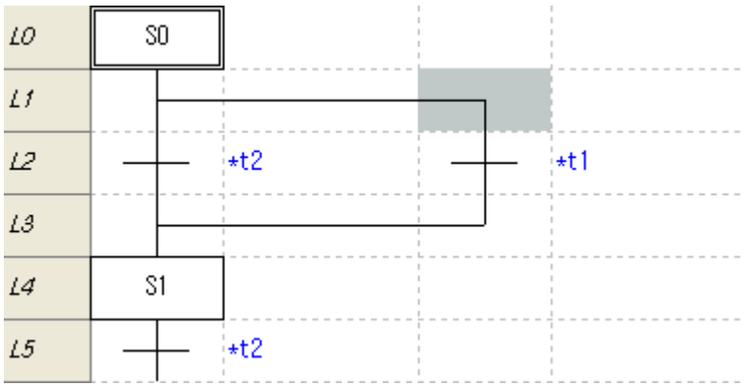
프로그램 편집 시 편집한 내용을 이전 상태로 취소 시키거나, 취소한 내용을 재 실행할 수 있습니다.

1) 편집 취소(삭제 예)

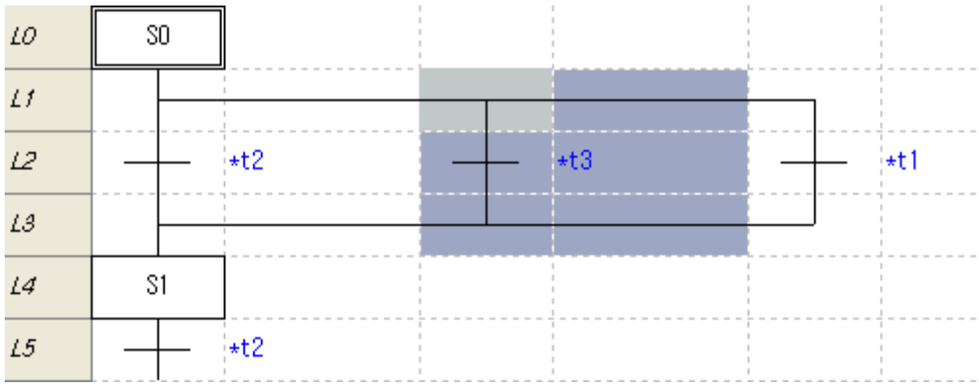
[순서]
1. 삭제하고자 하는 위치로 커서의 위치를 이동시킵니다.



2. 메뉴 [편집]-[삭제]를 선택합니다.



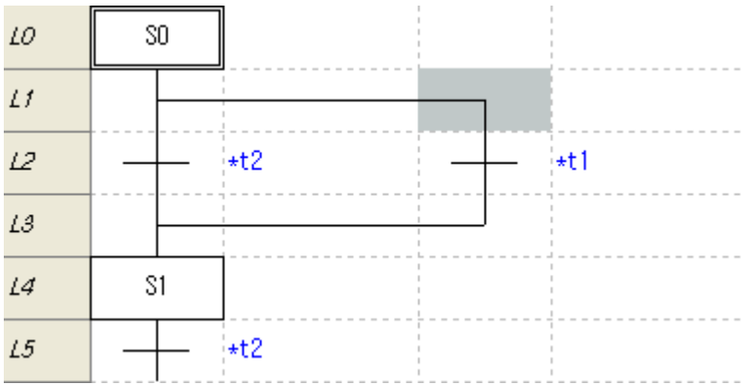
3. 메뉴 [편집]-[편집 취소]를 선택합니다.



4. 편집 된 영역을 표시하고, 편집 취소됩니다.

2) 재 실행(삭제 예)

[순서]
1. 메뉴 [편집]-[재 실행]을 선택합니다.



알아두기

- 편집한 모든 내용에 대해 실행 취소 및 재 실행이 가능합니다.
- 실행 취소의 횟수에는 제한이 없습니다.

4.4.1.20. 프로그램 편집 모드

SFC 프로그램은 삽입 모드만 지원합니다.

삽입 모드: 

4.4.2. 프로그램 보기

SFC 프로그램 보기 옵션 및 확대 축소에 대해 설명합니다.

4.4.2.1. 프로그램 확대/축소 배율 변경

SFC 프로그램이 화면에 표시되는 배율을 변경합니다.

1) 확대

[순서]
1. 메뉴 [보기]-[화면 확대]를 선택합니다.

2) 축소

[순서]
1. 메뉴 [보기]-[화면 축소]를 선택합니다.

알아두기

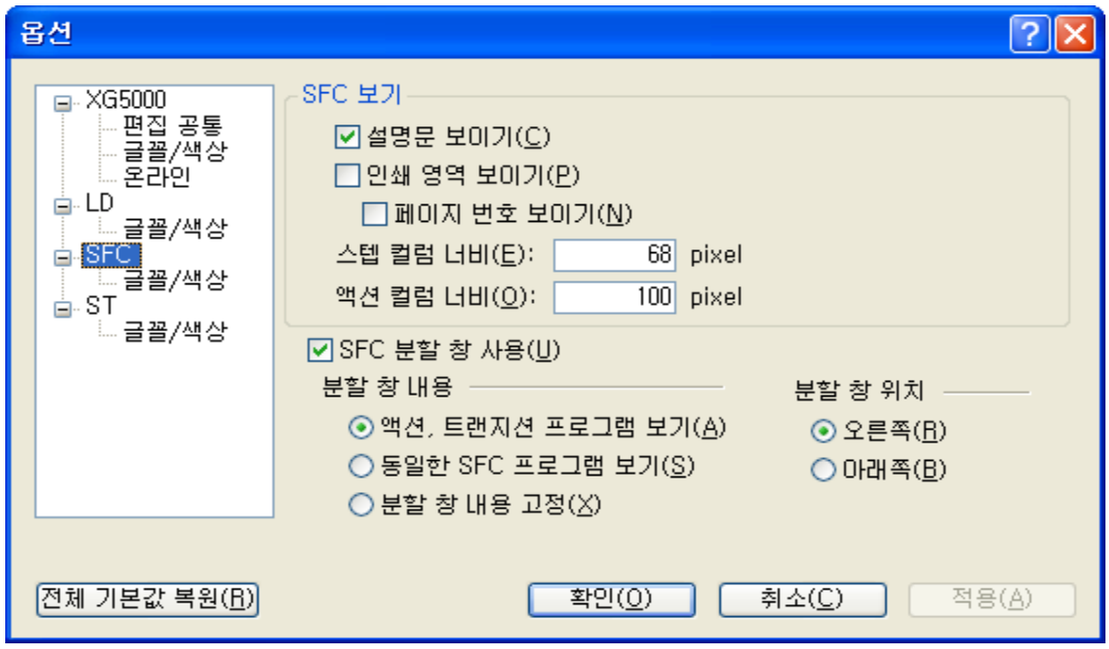
- 휠이 있는 마우스에서 Ctrl+위쪽 휠은 한 단계씩 축소합니다.
- 휠이 있는 마우스에서 Ctrl+아래쪽 휠은 한 단계씩 확대합니다.
- 보기 도구 모음의 선택 상자에서 배율을 선택하거나, 직접 입력할 수 있습니다. 자세한 사항은 제2장 기본 사용법의 2.2절 도구 모음을 참고 바랍니다.

4.4.2.2. SFC 화면 속성

SFC 프로그램 화면에 보일 내용 및 위치 등의 정보를 변경할 수 있습니다.

[순서]
1. 메뉴 [도구]-[옵션]-[SFC] 항목을 선택합니다.

[대화 상자]

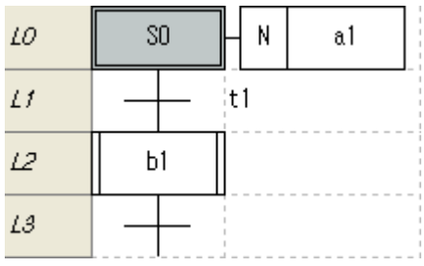


[대화 상자 설명]
a. 설명문 보이기: 스텝, 트랜지션, 액션, 블록의 설명문을 화면에 보이게 합니다.
b. 인쇄 영역 보이기: 인쇄 가능 영역을 화면에 굵은 점선으로 표시합니다.
c. 페이지 번호 보이기: 인쇄 가능 영역 내에 인쇄될 페이지 번호를 표시합니다.
d. 스텝 세로열 너비: 스텝, 트랜지션 위치의 세로열의 너비를 설정할 수 있습니다.
e. 액션 세로열 너비: 액션 위치의 세로열의 너비를 설정할 수 있습니다.
f. SFC 분할 창 사용: SFC 분할 창을 사용할 수 있습니다.
g. 분할 창 내용: 분할된 창에 어떤 프로그램의 내용을 보여줄지 결정할 수 있습니다.
h. 분할 창 위치: SFC 창을 어떤 방향으로 분할할지 결정 할 수 있습니다.
I. 확인: 입력한 내용을 저장하고 대화상자를 닫습니다.
j. 최소: 대화상자를 닫습니다.
k. 적용: 설정 사항을 현재의 SFC 창에 적용합니다.
l. 전체 기본값 복원: 설정 내용을 XG5000 설치될 때의 값으로 적용합니다.

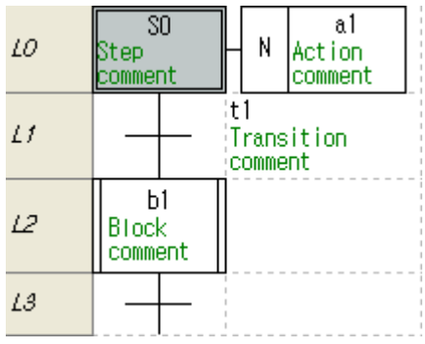
알아두기

- 스텝 세로열 너비의 범위는 20~200 입니다.
- 액션 세로열 너비의 범위는 70~400 입니다.

1) 설명문 보이기

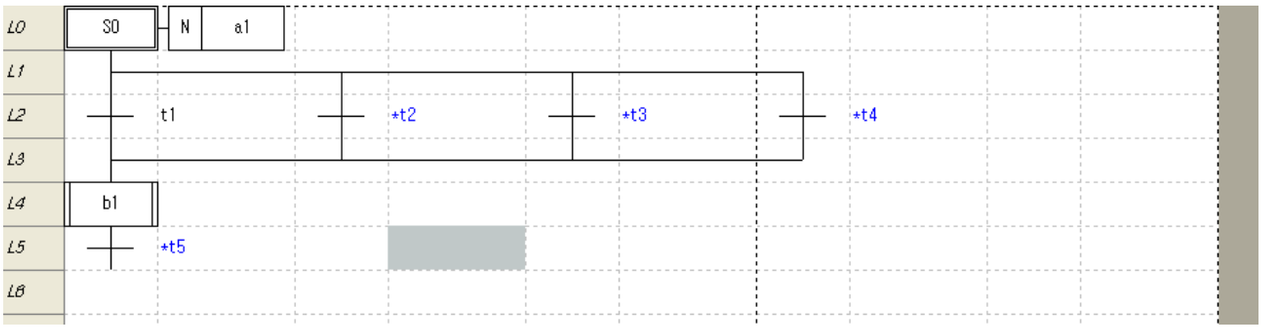


<설명문 안 보이기 상태>

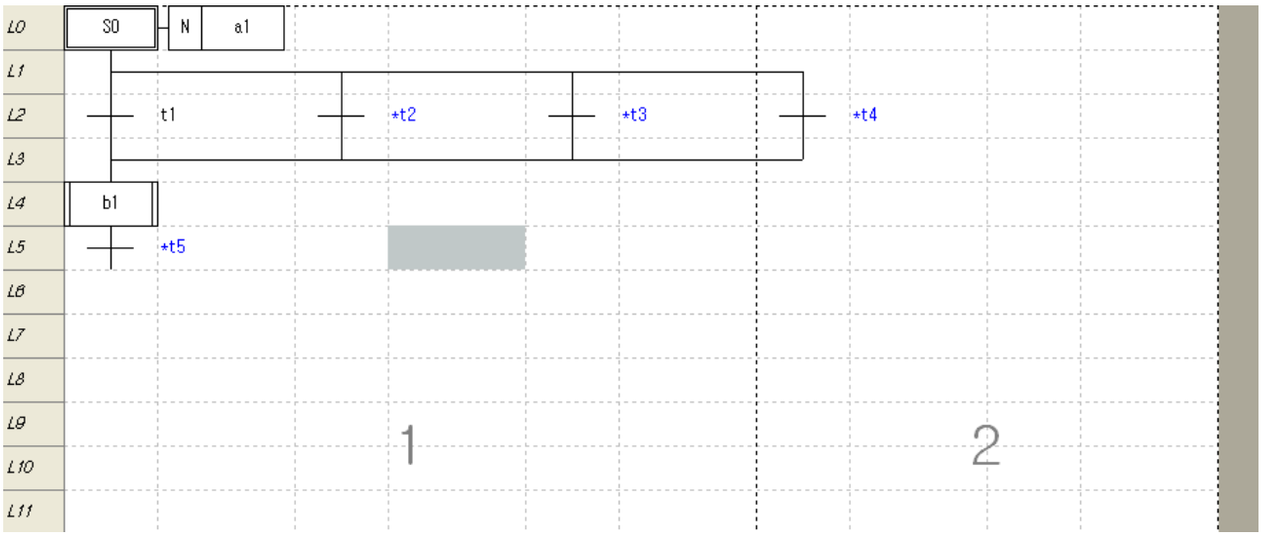


<설명문 보이기 상태>

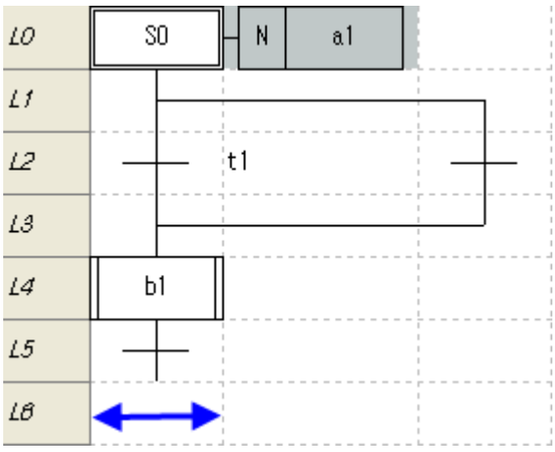
2) 인쇄 영역 보이기



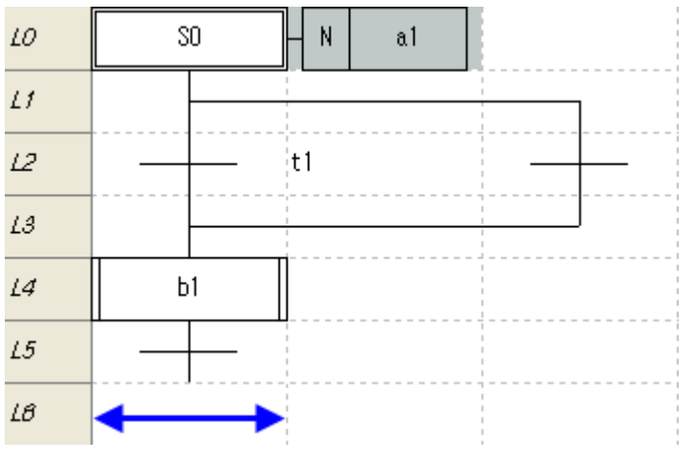
3) 페이지 번호 보이기



4) 스텝 세로열 너비

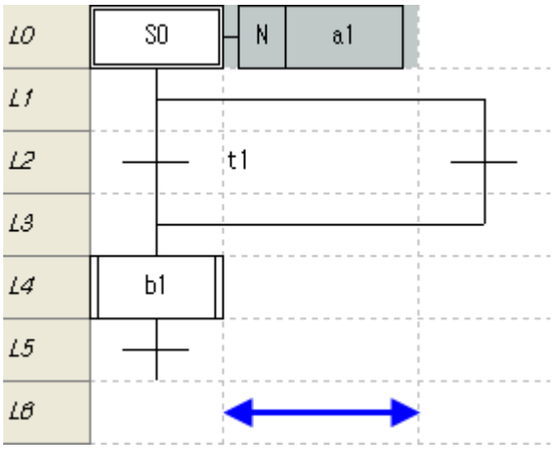


<스텝 세로열 68 pixel(기본) 화면>

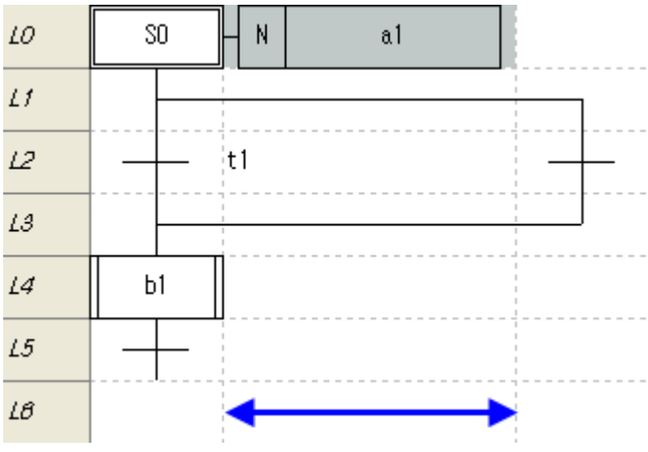


<스텝 세로열 100 pixel 화면>

5) 액션 세로열 너비

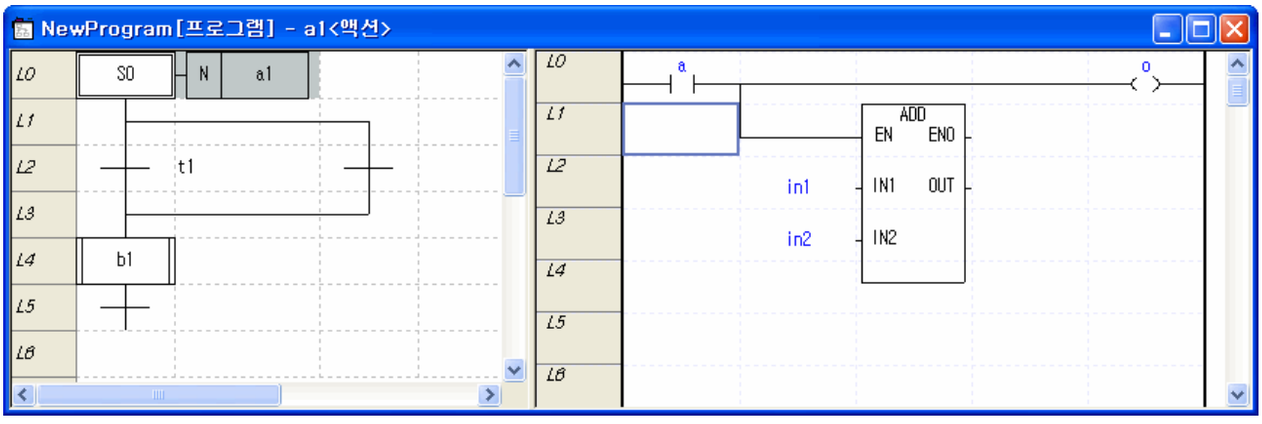


<액션 세로열 100 pixel(기본) 화면>

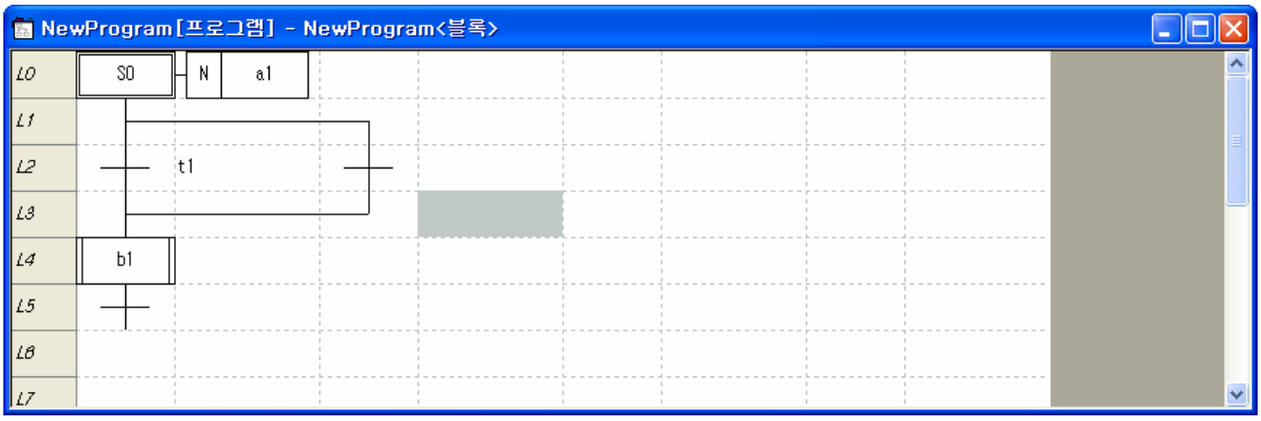


<액션 세로열 150 pixel 화면>

6) 분할 창 사용

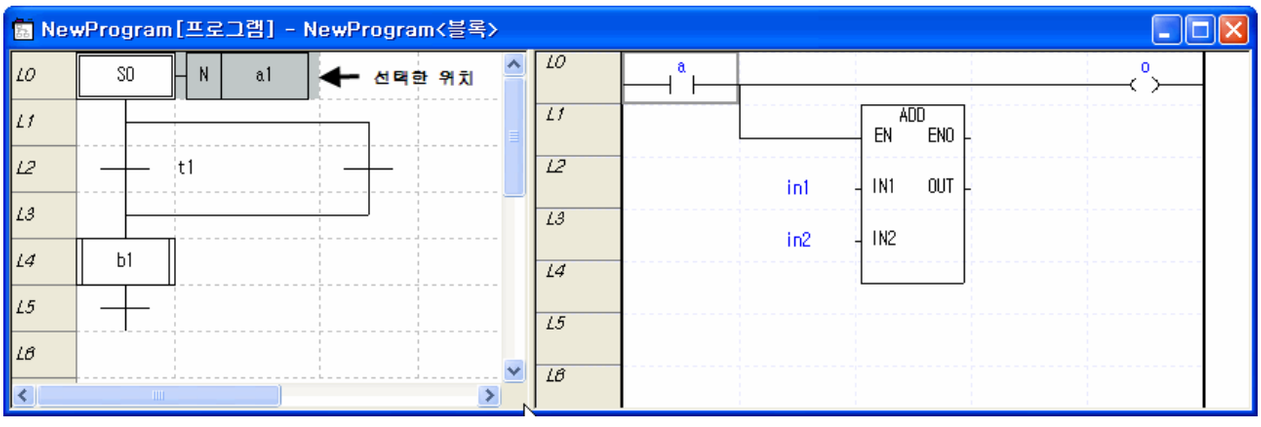


<분할 창 사용 중인 화면>

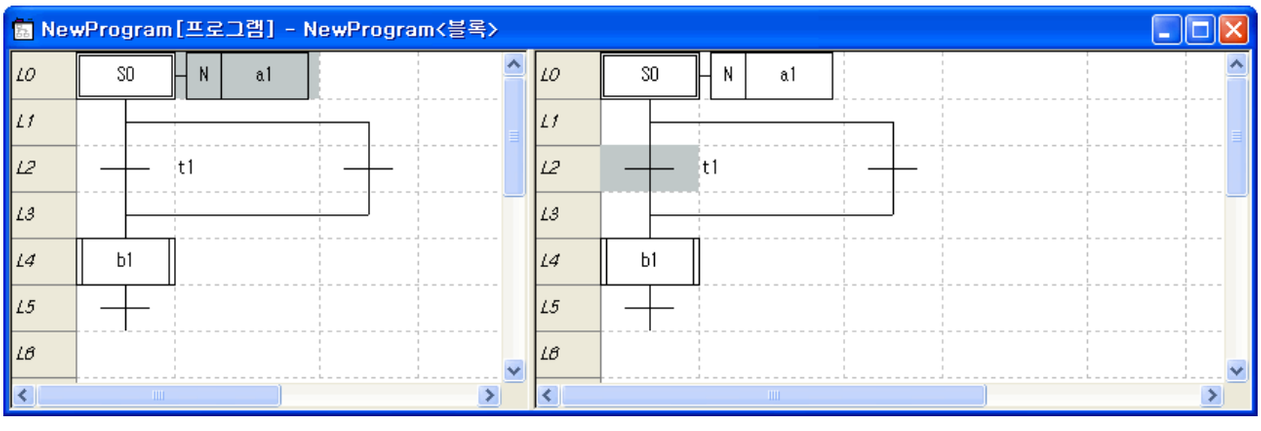


<분할 창 사용하지 않는 화면>

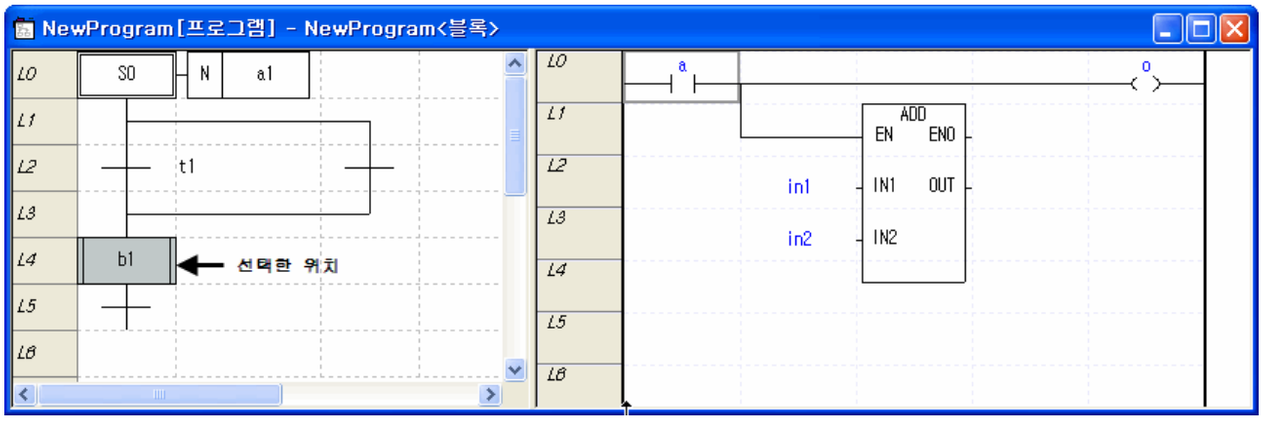
7) 분할 창 내용



<선택된 액션, 트랜지션 보기: 액션 선택함>

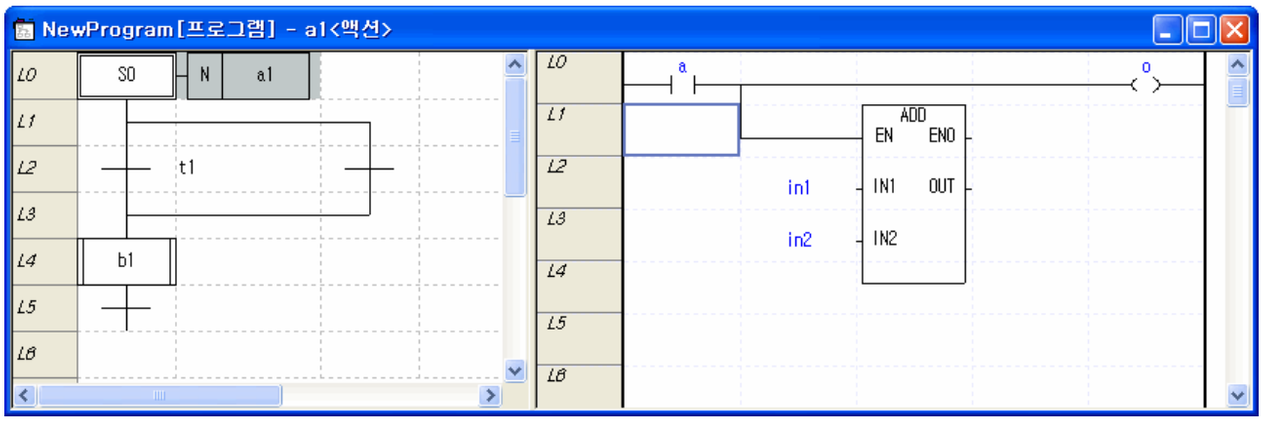


<동일한 SFC 프로그램 보기>

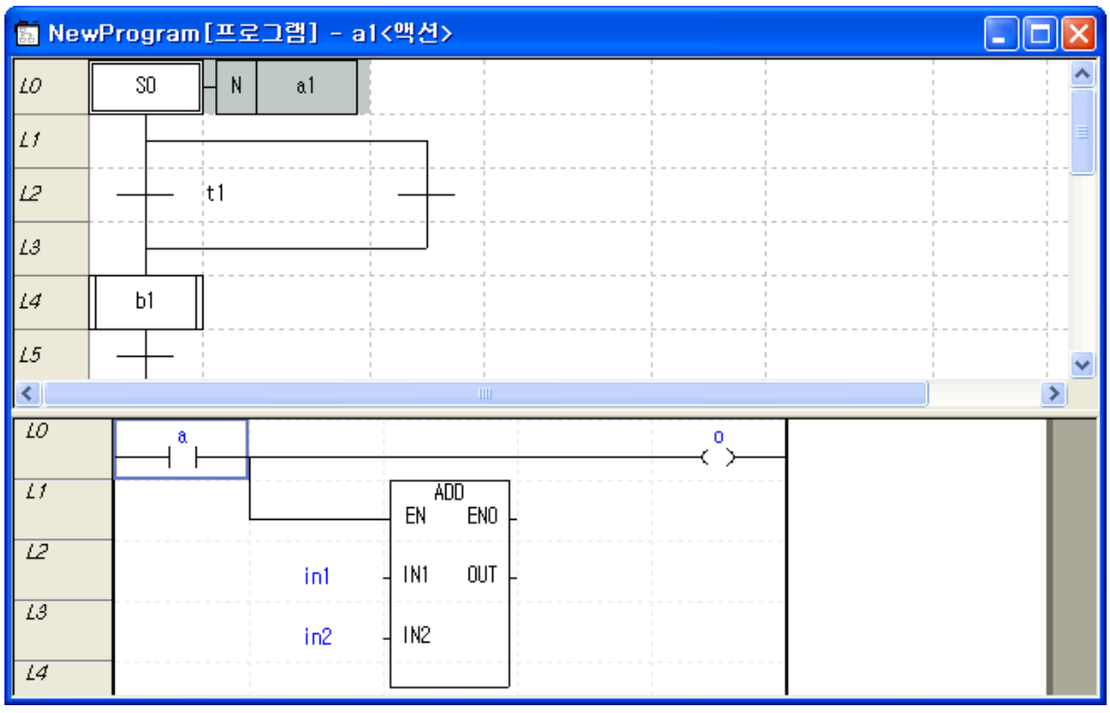


<분할 창 내용 고정 보기: 선택한 위치와 다르게 이전에 고정한 내용 보임>

8) 분할 창 위치



<분할 창 위치 오른쪽 화면>



<분할 창 위치 아래쪽 화면>

4.4.2.3. 블록/액션/트랜지션 목록 보기

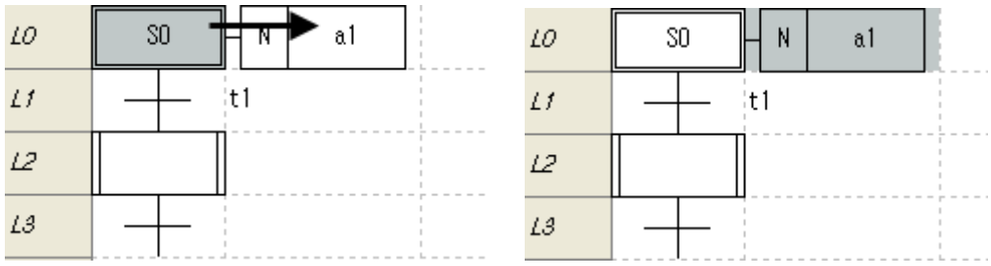
SFC 프로그램 내에서 사용중인 다른 프로그램 목록을 확인, 이름 및 설명문 등을 수정할 수 있습니다.

[순서]
1. 메뉴 [보기]-[블록/액션/트랜지션 목록] 항목을 선택합니다.
2. 목록 화면이 나옵니다.

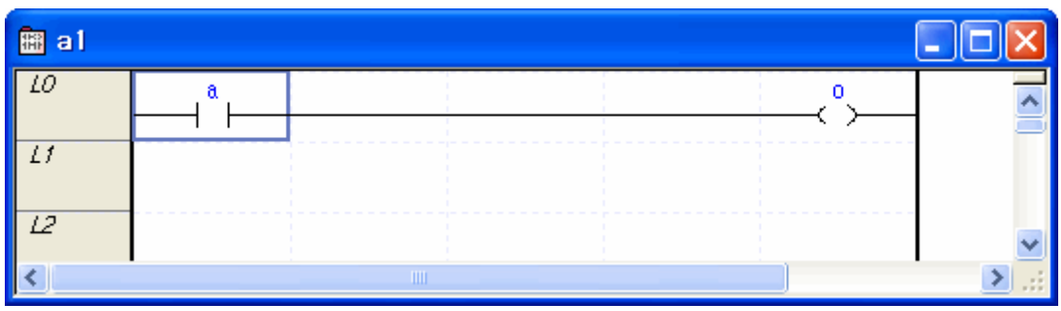
4.4.2.4. 프로그램 열기

SFC 프로그램에서 선택한 위치의 블록, 액션, 트랜지션의 프로그램을 새 창으로 열 수 있습니다.

[순서]
1. 프로그램 열기 하고자 하는 위치를 선택합니다.



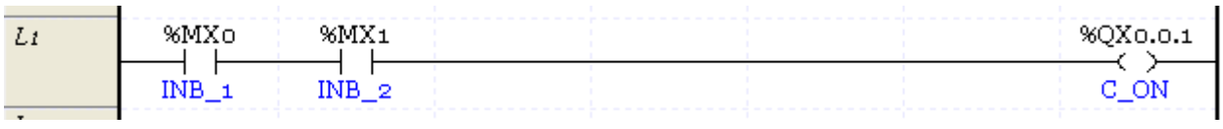
2. 메뉴 [보기]-[프로그램 열기]를 선택합니다.
3. 선택된 프로그램을 새 창에서 엽니다.



4.4.2.5. 디바이스/변수 보기

접점, 코일 및 펑션(블록)에 사용된 변수 또는 디바이스에 대하여 디바이스/변수 명으로 표시합니다. 만일, 변수에 디바이스가 없는 경우에는 변수 명으로 표시합니다.

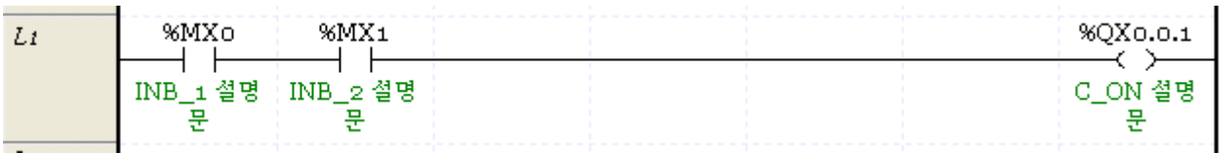
[순서]
1. 메뉴 [보기]-[디바이스/변수 보기] 항목을 선택합니다.



4.4.2.6. 디바이스/설명문 보기

접점, 코일 및 펑션(블록)에 사용된 변수 또는 디바이스에 대하여 디바이스/설명문으로 표시합니다. 만일, 변수에 디바이스가 없는 경우에는 변수 명으로 표시합니다.

[순서]
1. 메뉴 [보기]-[디바이스/설명문 보기] 항목을 선택합니다.



알아두기

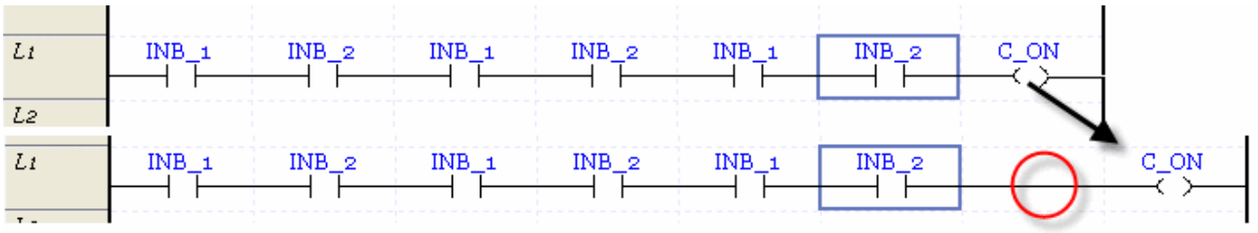
- 보기 옵션 변경 시, 편집된 프로그램 양에 따라 다소 시간이 걸릴 수 있습니다.
- 인쇄 시에는 디바이스/변수/설명문 인쇄 기능을 지원합니다. 인쇄 옵션에 대한 상세 내용은 14장 인쇄 항목을 참고하시기 바랍니다.

4.4.2.7. 접점 수 조절

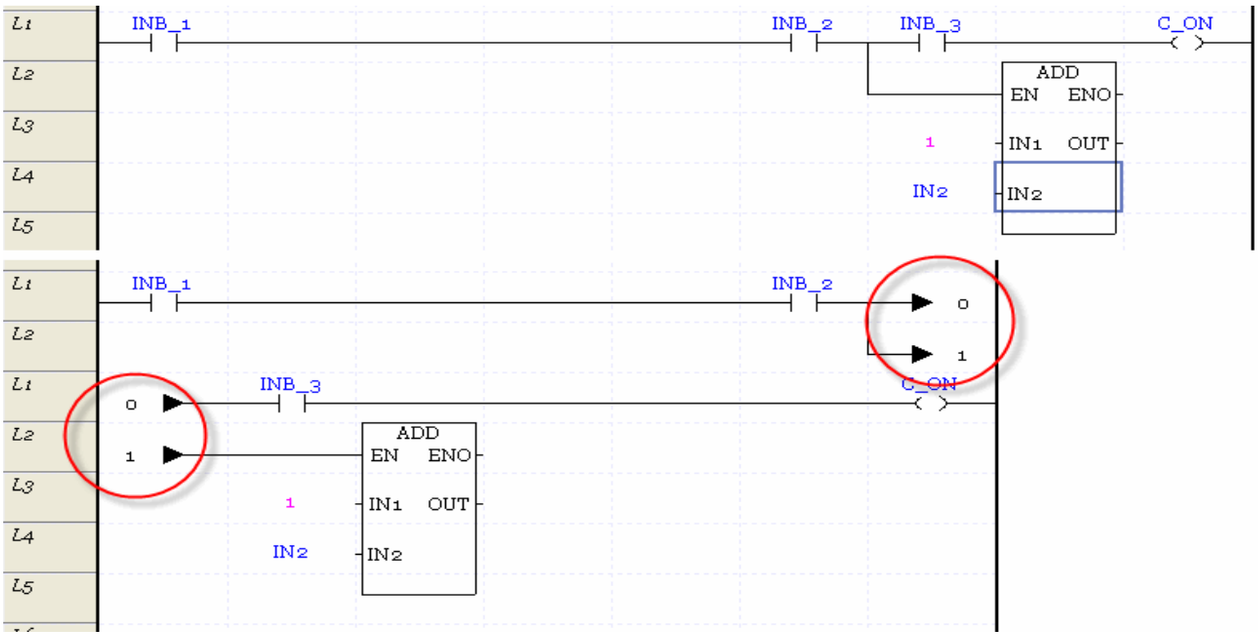
화면에 표시되는 접점 수를 조절합니다. 여기서 접점 수는 총 (가로 셀 - 1)로 출력 위치는 제외합니다.

[순서]

1. 메뉴 [보기]-[접점 수 변경]-[*접점 수*]를 선택합니다.

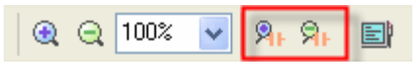


만일, 만일 현재 화면에 표시되는 가장 오른쪽에 있는 데이터가 표시할 접점 수 보다 더 큰 경우, 화살표를 포함한 렁으로 표시될 수 있습니다



알아두기

- 보기 도구 모음에서 한 칸씩 증가 또는 감소 시킬 수 있습니다.

  4.4.2.7.1.1.1.1. 

- 설정 가능한 접점 수는 9 ~ 31입니다.
- SFC의 액션 및 트랜지션에 사용된 LD 프로그램의 최소 설정 접점 수는 5 입니다.

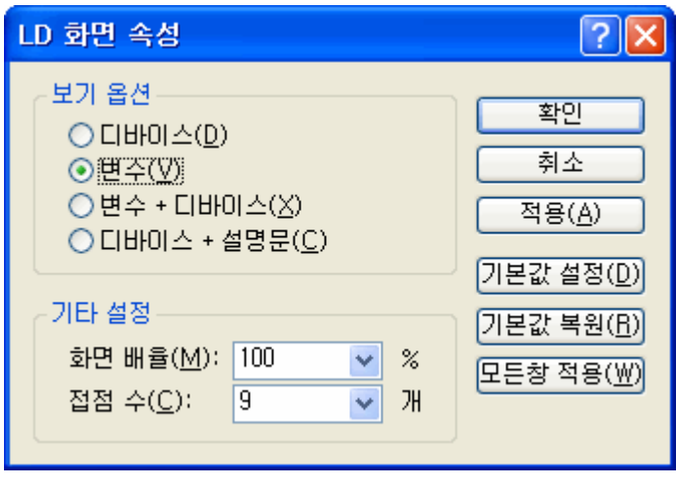
4.4.2.8. LD 화면 속성

LD 화면의 보기 속성을 지정합니다. 화면 속성에서는 디바이스, 변수, 설명문 보기 옵션에 대한 설정 및 배율, 접점 수를 한번에 설정할 수 있습니다. 또한 LD 화면 전체에 대해서 동일한 속성을 지정할 수 있습니다.

[순서]

1. 메뉴 [보기]-[LD 화면 속성]을 선택합니다.
2. LD 화면 속성을 변경한 후 확인을 누릅니다.

[대화 상자]



[대화 상자 설명]

a. 보기 옵션: 변수 및 디바이스에 대한 보기 옵션을 지정합니다.
b. 화면 배율: 화면에 표시할 배율을 지정합니다. 배율은 40~200%까지 설정 가능합니다.
c. 접점 수: 화면에 표시되는 접점 수를 지정합니다.
d. 확인: 설정한 사항을 적용하고 대화 상자를 닫습니다.
e. 취소: 대화 상자를 닫습니다.
f. 적용: 설정 사항을 현재의 LD 창에 적용합니다.
g. 기본값 설정: 현재 설정을 LD 창에 대한 기본 값으로 설정합니다. 새 LD 프로그램을 작성하면 현재 설정한 보기 모드로 표시됩니다.
h. 기본값 복원: 현재 설정을 기본 설정으로 복원합니다.
I. 모든 창 적용: 현재 설정을 모든 창에 적용합니다.

알아두기

- LD 화면 속성에서는 디바이스/변수/설명문 보기 옵션을 지정할 수 없습니다.
- 모든 창 적용 시, LD 프로그램뿐 아니라, LD 로 작성한 SFC 프로그램 액션 및 트랜지션, LD 로 작성한 사용자 펑션(블록)에 모두 적용됩니다.

4.4.3. 편집 부가 기능

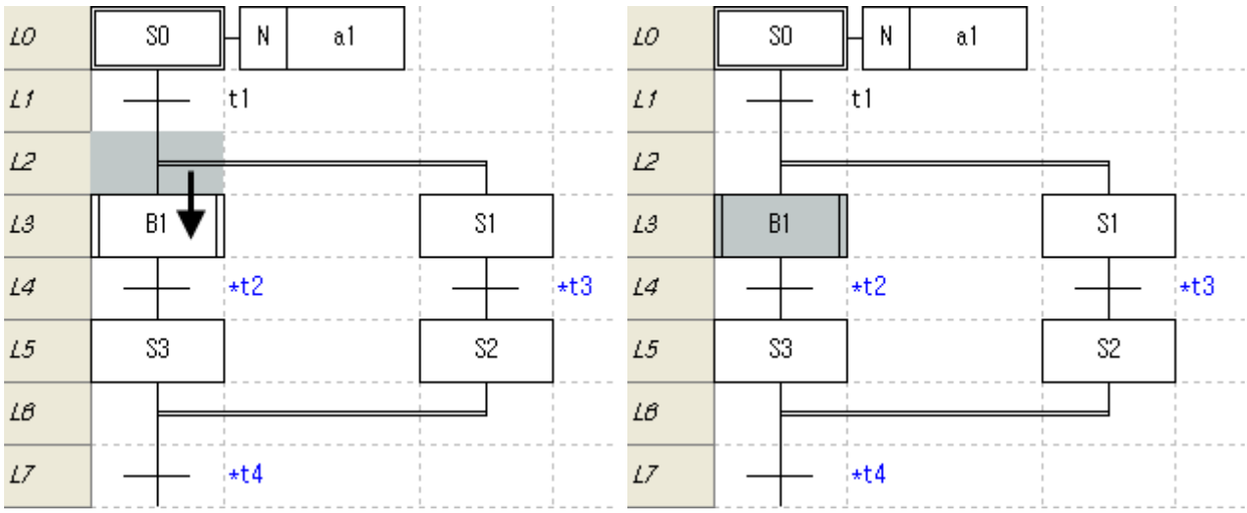
편집의 편리성을 위한 부가 기능을 설명합니다.

4.4.3.1. 북 마크

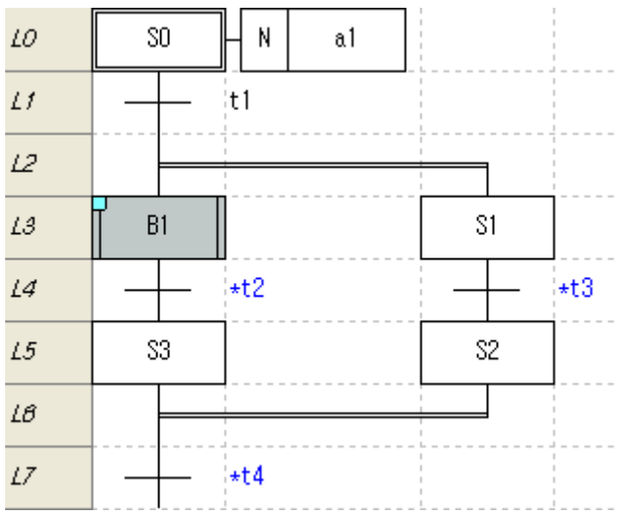
북 마크를 설정하여, 관심 있는 부분으로 쉽게 이동할 수 있습니다.

1) 북 마크 설정

[순서]
1. 북 마크를 설정하고자 하는 위치로 커서를 이동시킵니다.



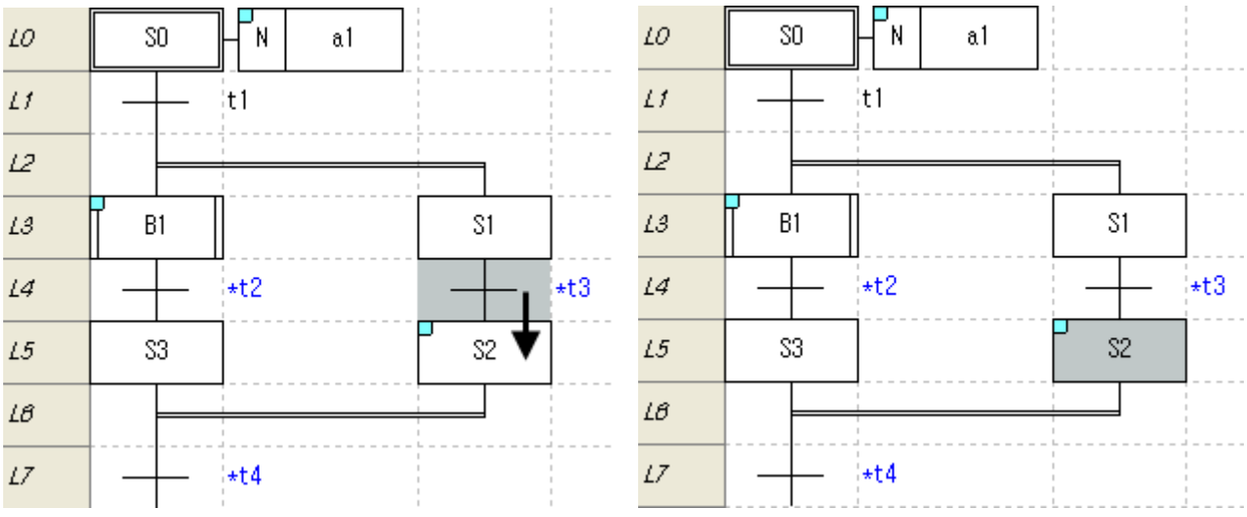
2. 메뉴 [편집]-[북 마크]-[설정/해제]를 선택합니다.



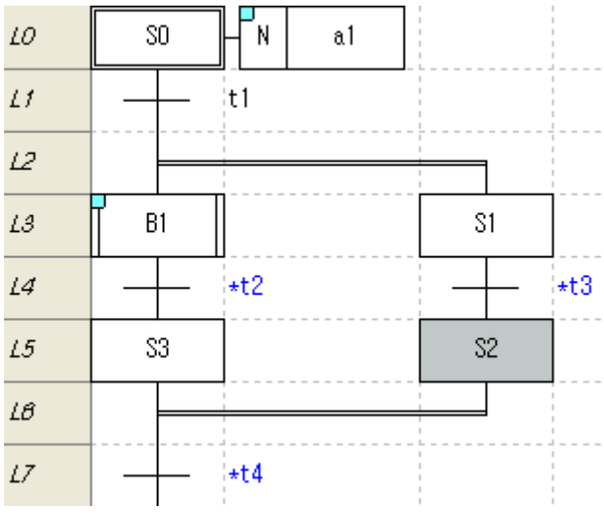
2) 북 마크 해제

[순서]

1. 북 마크를 해제하고자 하는 위치로 커서를 이동시킵니다.



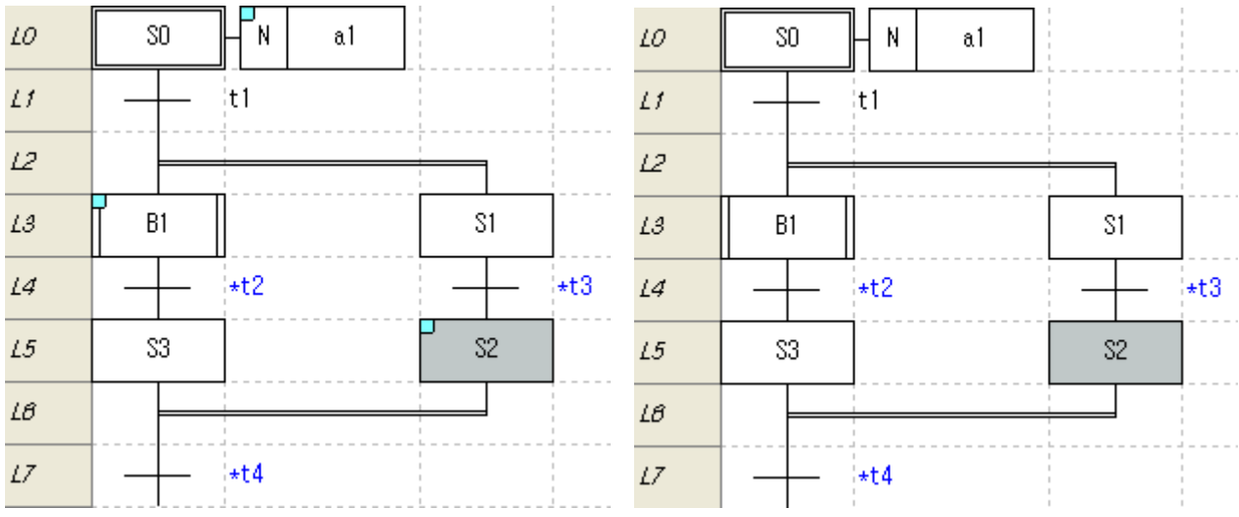
2. 메뉴 [편집]-[북 마크]-[설정/해제]를 선택합니다.



3) 모든 북 마크 해제

[순서]

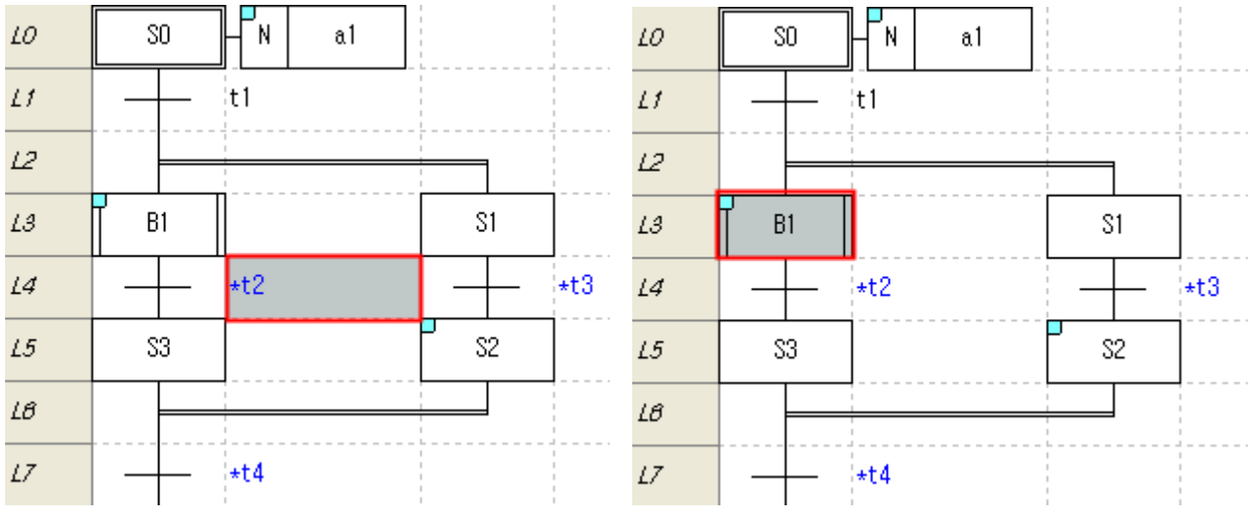
1. 메뉴 [편집]-[북 마크]-[모두 해제]를 선택합니다.



4) 이전 북마크 이동

[순서]

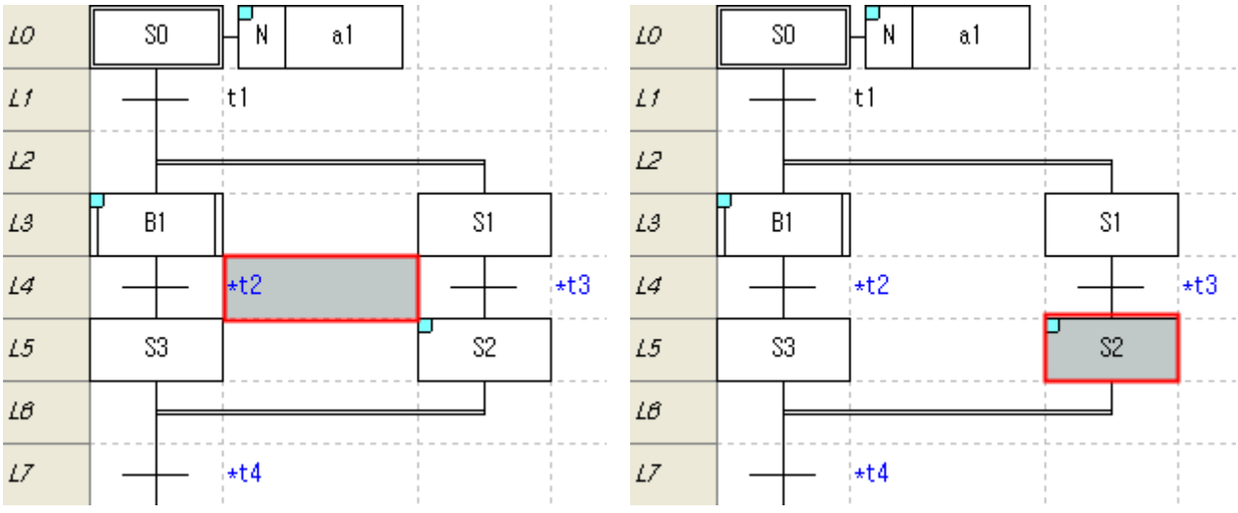
1. 메뉴 [편집]-[북 마크]-[이전 북마크]를 선택합니다.



5) 다음 북마크 이동

[순서]

1. 메뉴 [편집]-[북 마크]-[다음 북마크]를 선택합니다.



알아두기

- 북 마크는 셀 단위로 설정됩니다.
- 북 마크는 편집 사항이 아니므로, 설정/해제에 관한 사항은 편집 취소 및 재 실행에 포함되지 않습니다.

4.4.3.2. 찾아가기

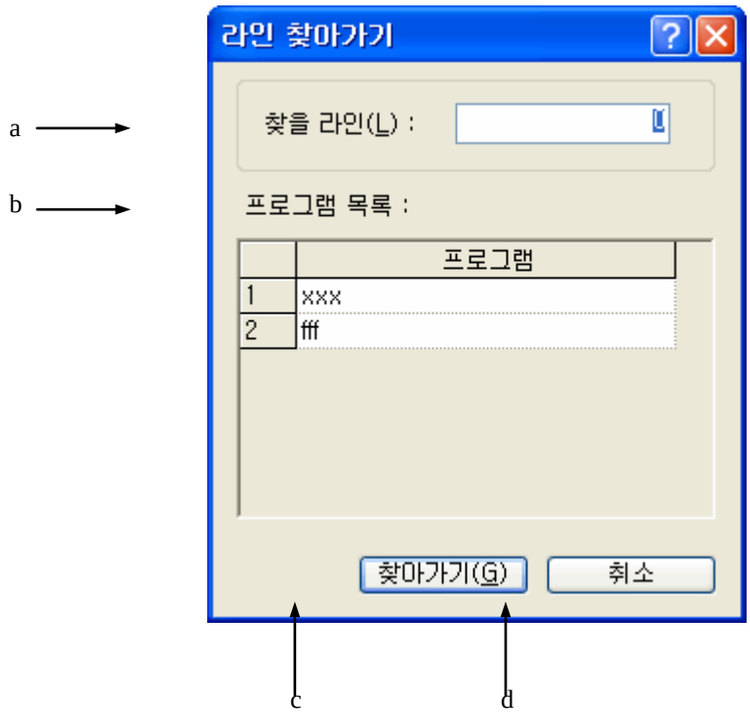
지정한 라인의 위치로 이동하거나, 편집한 레이블, 렁 설명문 위치로 찾아갈 수 있습니다.

1) 라인 찾아가기

[순서]
1. 메뉴 [찾기/바꾸기]-[찾아가기]-[스텝/라인]을 선택합니다.

2. 대화 상자에서 이동할 스텝을 입력합니다.

[대화 상자]



[대화 상자 설명]
a. 찾을 라인: 이동하고자 하는 라인을 입력합니다.
b. 프로그램 목록: 현재 PLC의 프로그램 목록을 표시합니다.
c. 찾아가기: 대화 상자를 닫고 선택한 프로그램의 찾을 스텝으로 이동합니다.
d. 취소: 대화 상자를 닫습니다.

알아두기

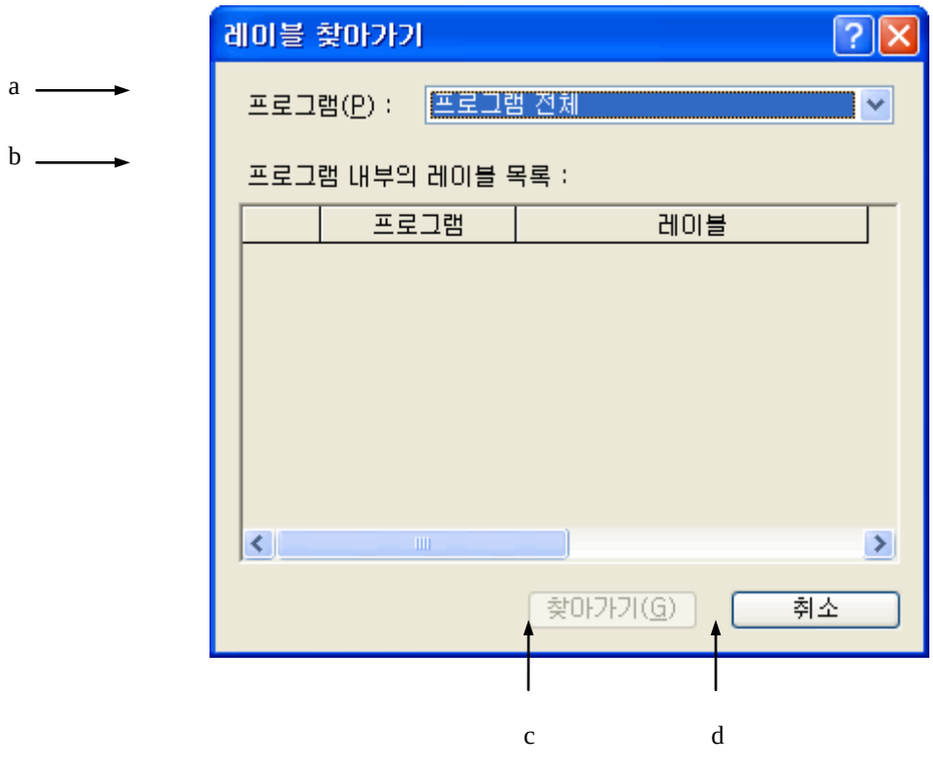
- 라인 찾아가기는 LD 프로그램에서만 사용 가능합니다.
- SFC의 액션/트랜지션으로 사용한 LD는 찾아가기에서 동작하지 않습니다.

2) 레이블 찾아가기

[순서]
1. 메뉴 [찾기/바꾸기]-[찾아가기]-[레이블]을 선택합니다.

2. 대화 상자에서 찾아갈 레이블을 선택합니다.

[대화 상자]



[대화 상자 설명]
a. 프로그램: 현재 PLC의 프로그램 목록을 표시합니다. '프로그램 전체' 를 선택한 경우 모든 레이블에 대한 리스트가 표시됩니다.
b. 프로그램 내부의 레이블 목록: 선택한 프로그램에서 사용 중인 레이블에 대한 목록을 표시합니다.
c. 찾아가기: 대화 상자를 닫고 선택한 레이블로 이동합니다.
d. 취소: 대화 상자를 닫습니다.

알아두기

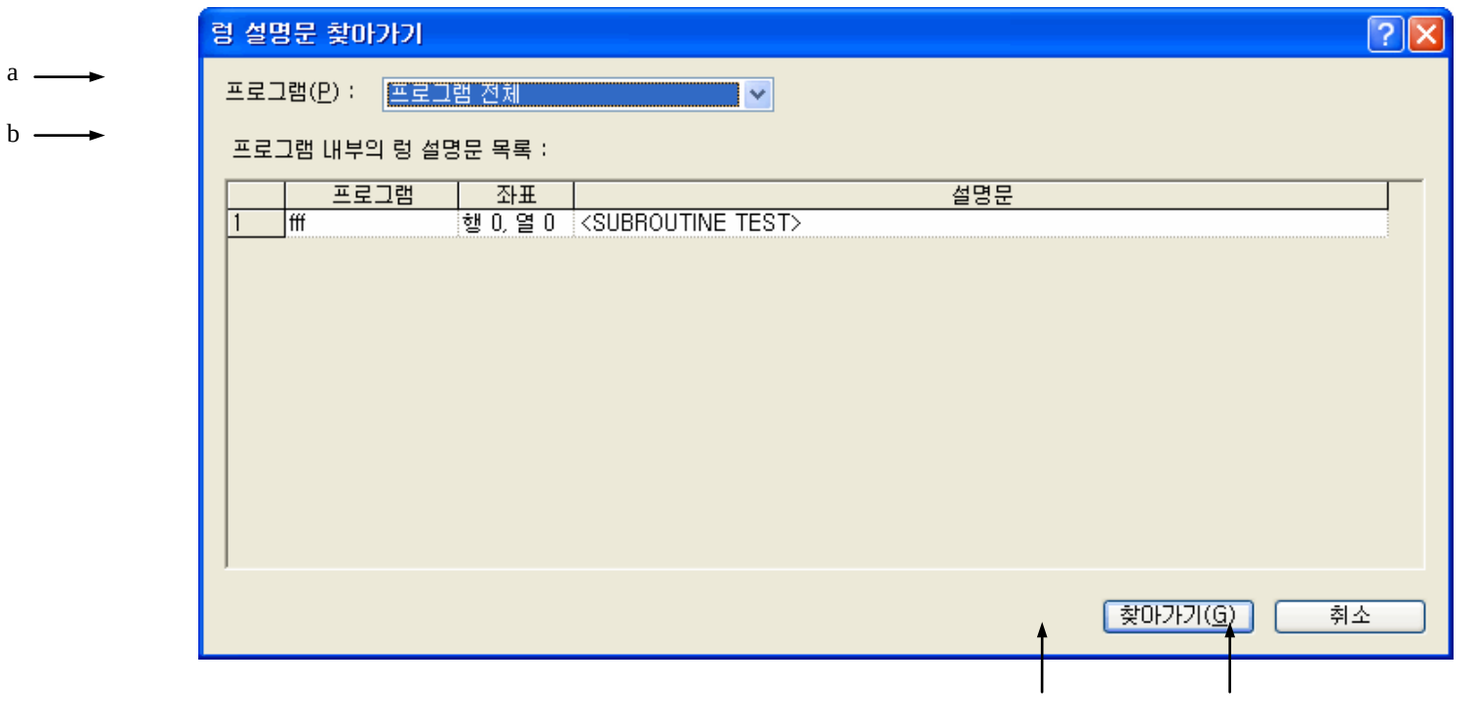
- 레이블 찾아가기는 LD 프로그램에서만 사용 가능합니다.
- SFC의 액션/트랜지션으로 사용한 LD는 찾아가기에서 동작하지 않습니다.

3) 렁 설명문 찾아가기

[순서]
1. 메뉴 [찾기/바꾸기]-[찾아가기]-[렁 설명문]을 선택합니다.

2. 대화 상자에서 찾아갈 렁 설명문을 선택합니다.

[대화 상자]



[대화 상자 설명]
a. 프로그램: 현재 PLC의 프로그램 목록을 표시합니다. '프로그램 전체' 를 선택한 경우 모든 렁 설명문에 대한 리스트가 표시됩니다.
b. 프로그램 내부의 렁 설명문 목록: 선택한 프로그램에 있는 렁 설명문을 표시합니다.
c. 찾아가기: 대화 상자를 닫고 선택한 렁 설명문으로 이동합니다.
d. 취소: 대화 상자를 닫습니다.

알아두기

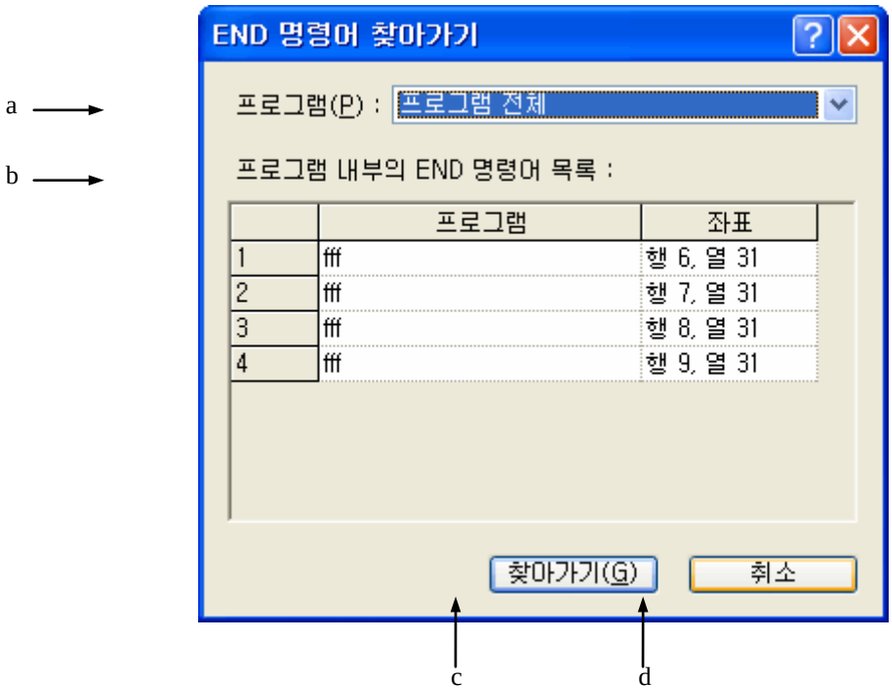
- 렁 설명문 찾아가기는 LD 프로그램에서만 사용 가능합니다.
- SFC의 액션/트랜지션으로 사용한 LD는 찾아가기에서 동작하지 않습니다.

4) END 명령어 찾아가기

[순서]
1. 메뉴 [찾기/바꾸기]-[찾아가기]-[END 명령어]를 선택합니다.

2. 대화 상자에서 찾아갈 END 명령어를 선택합니다.

[대화 상자]



[대화 상자 설명]
a. 프로그램: 현재 PLC의 프로그램 목록을 표시합니다. '프로그램 전체' 를 선택한 경우 모든 END 명령어가 표시됩니다.
b. 프로그램 내부의 END 목록: 선택한 프로그램에 있는 END 명령어를 표시합니다.
c. 찾아가기: 대화 상자를 닫고 선택한 END 명령어로 이동합니다.
d. 취소: 대화 상자를 닫습니다.

알아두기

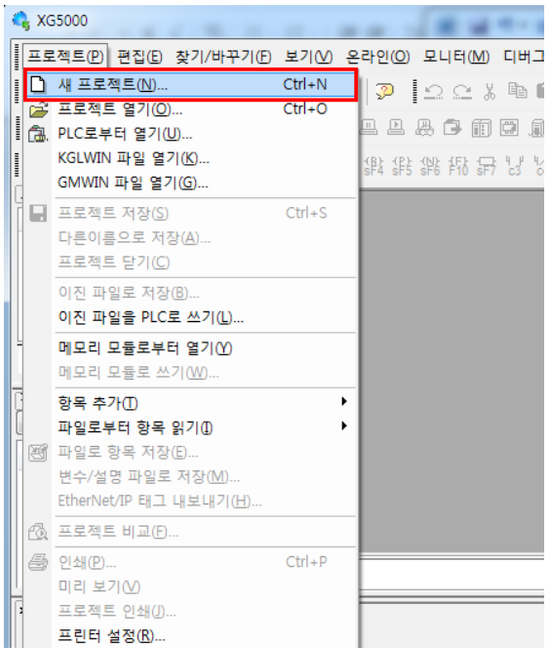
- END 명령어 찾아가기는 LD 프로그램에서만 사용 가능합니다.
- SFC의 액션/트랜지션으로 사용한 LD는 찾아가기에서 동작하지 않습니다.

4.5. 프로그래밍 가이드

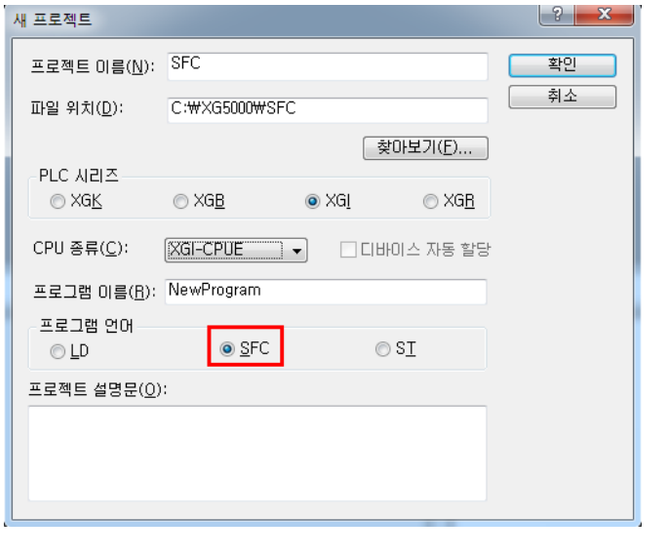
SFC 프로그램의 입력에서 쓰기 및 모니터링까지의 일련의 기본 조작을 설명
예제 프로그램을 이용한 프로그램 작성 과정 설명

<예제 프로그램>
적색등 5 초, 녹색등 3 초동안 번갈아 가면서 무한 반복 점등되는 신호등 프로그램 작성(처음엔 녹색등 점등 상태)

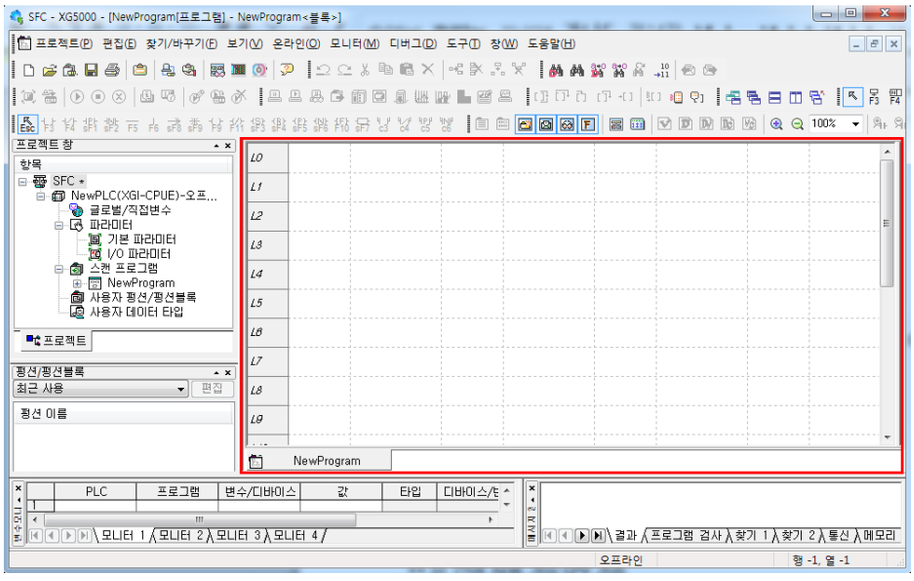
4.5.1. SFC 용 프로그램 신규 생성



① 메뉴[프로젝트] -> [새 프로젝트]를 클릭



② 프로그램 언어 SFC 설정 후 [확인]



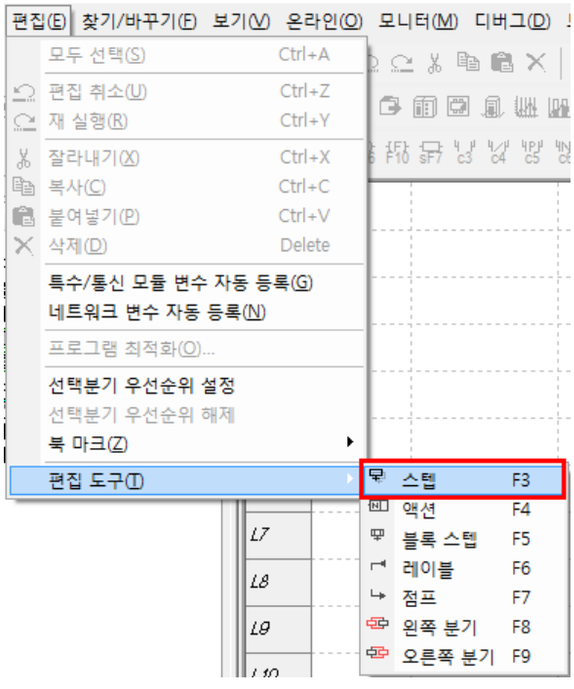
③ SFC 편집화면이 열립니다.

** SFC 언어 사용 가능 CPU 기종

XGK (디바이스 자동 할당 체크시 사용가능)	XGB	XGI	XGR
XGK-CPUA	XGB-XECH	XGI-CPUE	XGR-CPUH
XGK-CPUE	XGB-XECS	XGI-CPUS	XGR-INC
XGK-CPUS	XGB-XECE	XGI-CPUH	
XGK-CPUH		XGI-CPUU	
XGK-CPUU		XGI-CPUU/D	

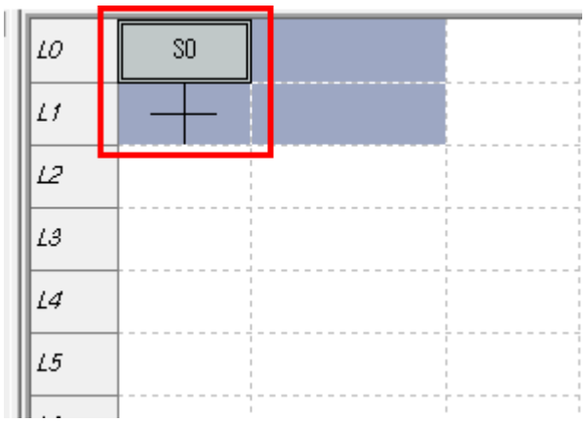
4.5.2. 프로그램 작성

4.5.2.1. 스텝 입력



① [편집] -> [편집도구] -> [스텝]을 클릭
또는 단축키(F3) 입력

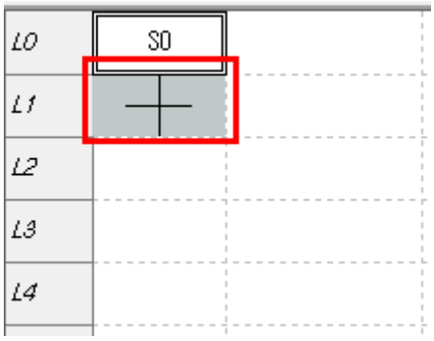
② 프로그램 편집 창에 마우스 커서를 올리고
마우스 [왼쪽 버튼] 클릭



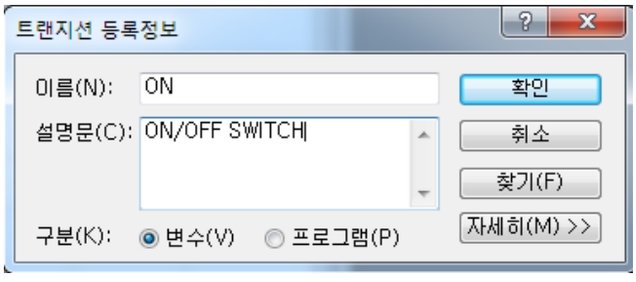
③ 초기 스텝 생성 완료

4.5.2.2. 트랜지션 입력

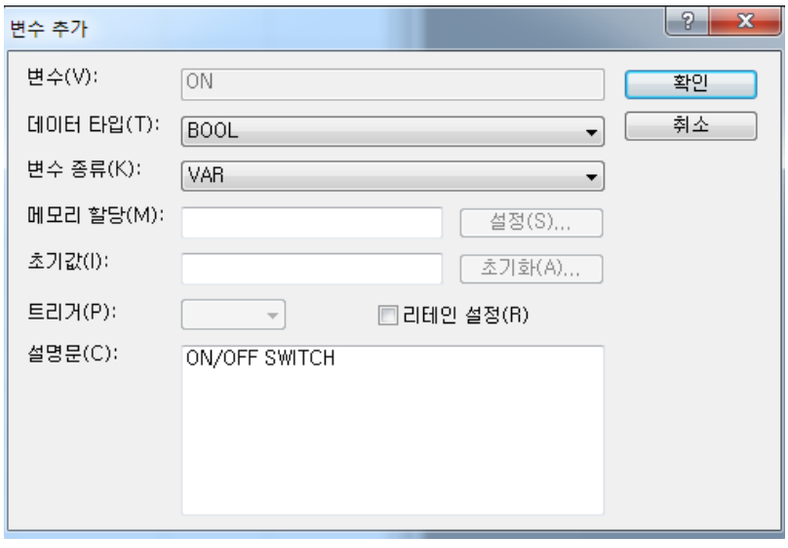
위에서 생성된 초기스텝의 다음 스텝으로 넘어가기 위한 트랜지션 조건을 설정합니다.
ON/OFF 스위치를 변수로 지정하여 트랜지션 조건으로 지정합니다.



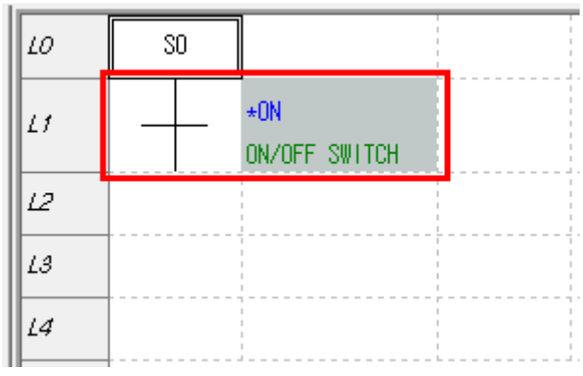
① 프로그램 편집 창의 트랜지션을 더블 클릭



② 트랜지션 조건을 변수 'ON'으로 지정



③ 변수 추가 등록

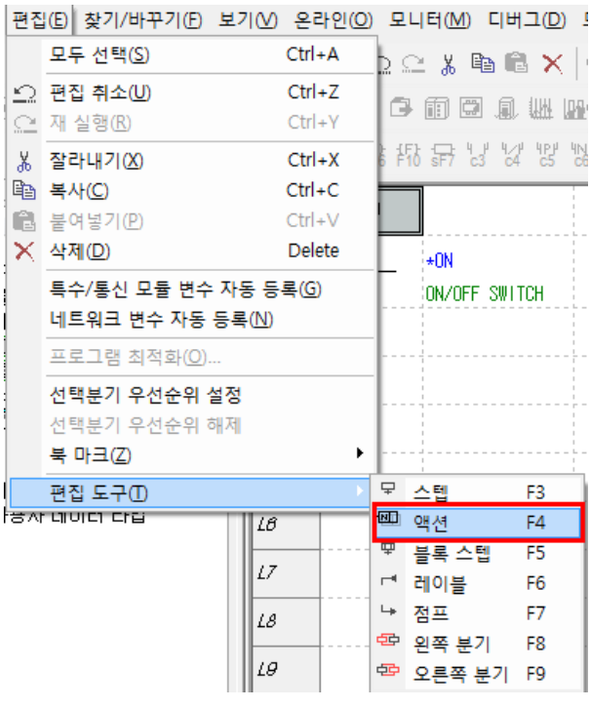


④ 트랜지션 조건이 입력되었습니다.

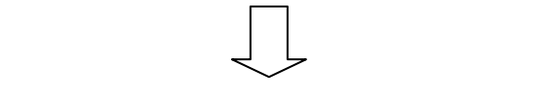
4.5.2.3. 액션 입력

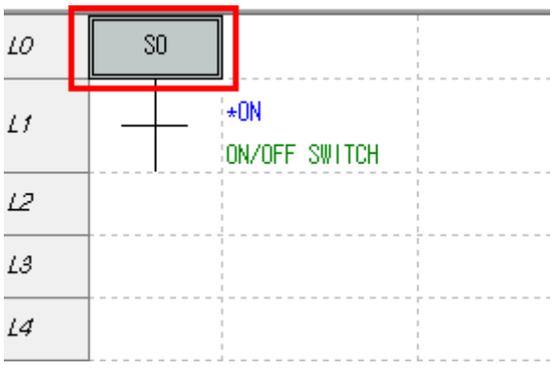
초기 상태(녹색등 점등)를 설정하기 위해 초기 스텝에 액션을 등록합니다.

액션 제한자는 SET 을 사용함으로서 초기에 해당 변수를 항상 ON 합니다.

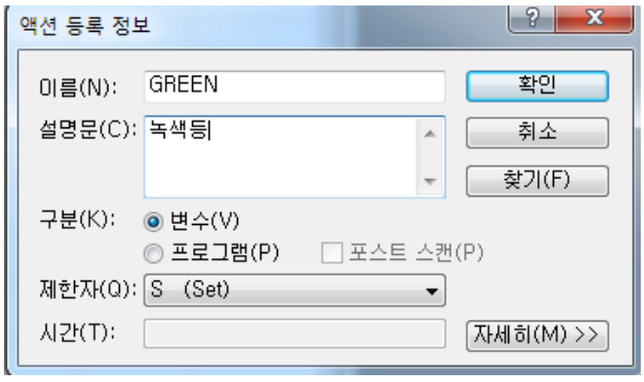


① [편집] -> [편집도구] -> [액션] 클릭
또는 단축키(F4) 입력

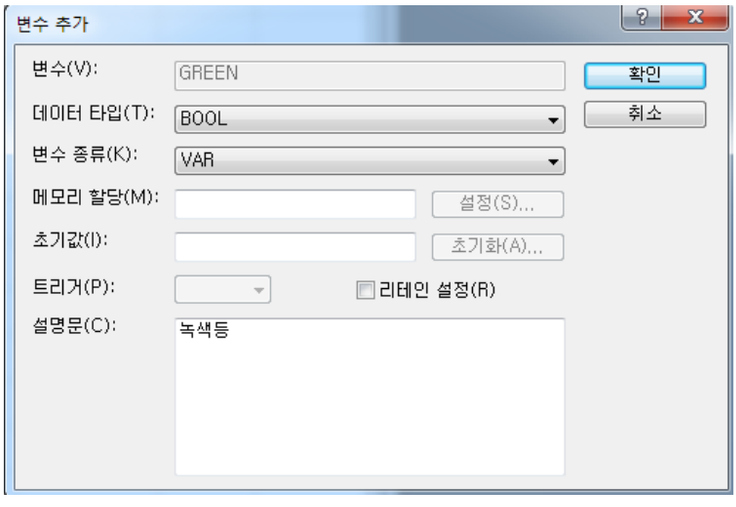




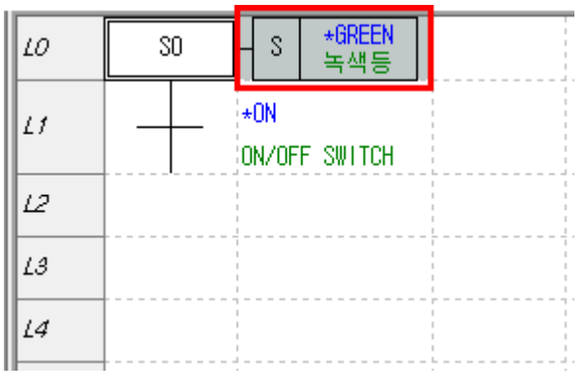
② 초기 스텝인 'S0'에 마우스 커서를 올리고
[왼쪽 버튼] 클릭



③ 변수 정보(녹색등)를 입력합니다.
액션 제한자는 SET 을 사용합니다.

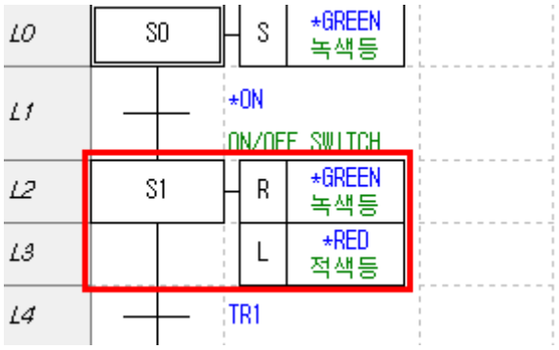


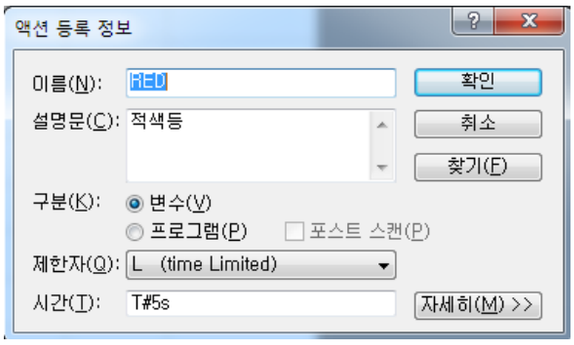
④ 변수 추가 등록



⑤ 액션이 입력되었습니다.

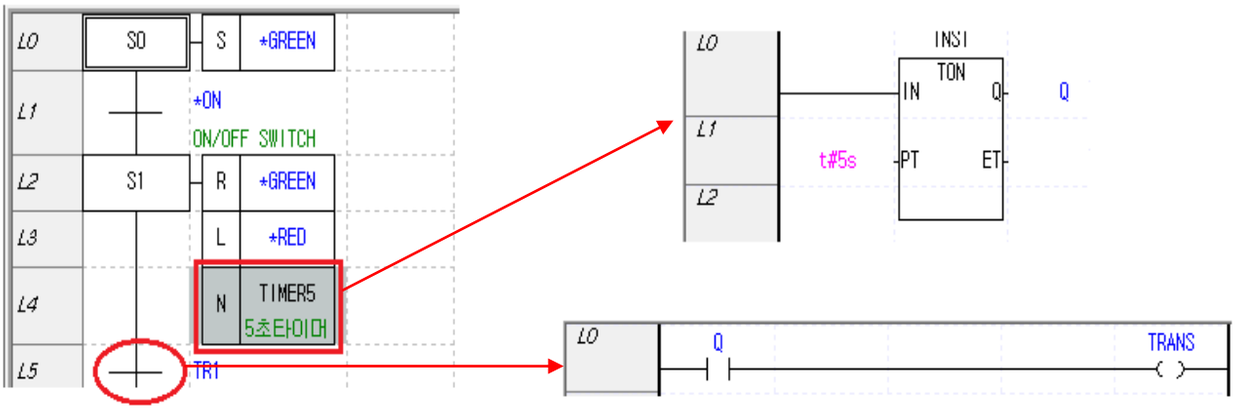
4.5.2.4. 프로그램 입력

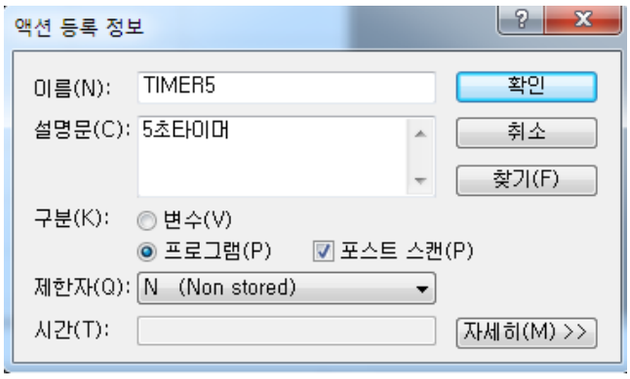




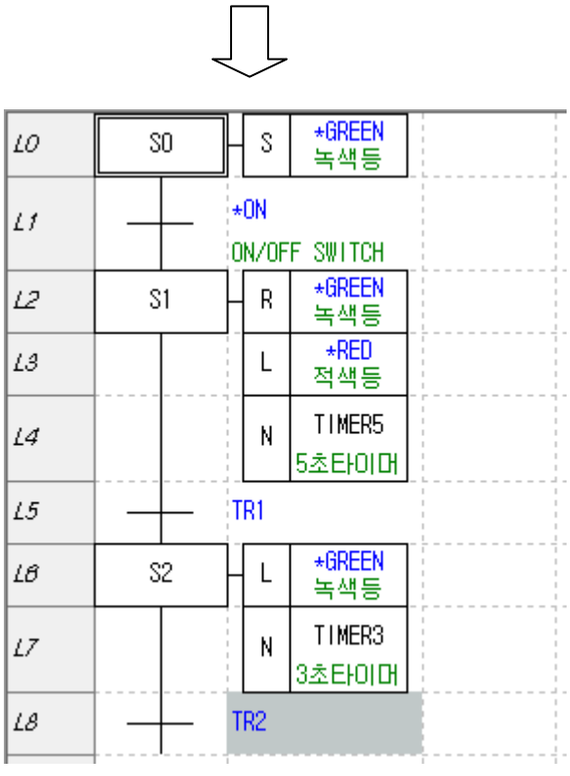
① 좌측과 같이 스텝과 액션을 입력합니다.

- 녹색등 소등 (RESET)
- 적색등 점등 (Time limited)





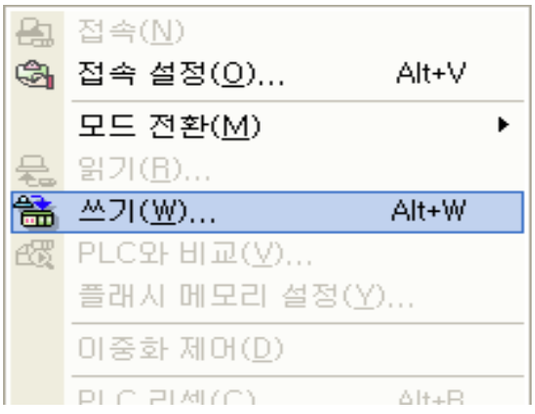
② 위와 같이 타이머와 트랜지션 조건을 추가 합니다. 액션 등록 정보는 아래와 같이 포스트 스캔을 체크합니다.



③ 위와 동일한 방법으로 좌측과 같이 프로그램을 입력합니다.

4.5.3. 프로그램 쓰기 및 모니터링

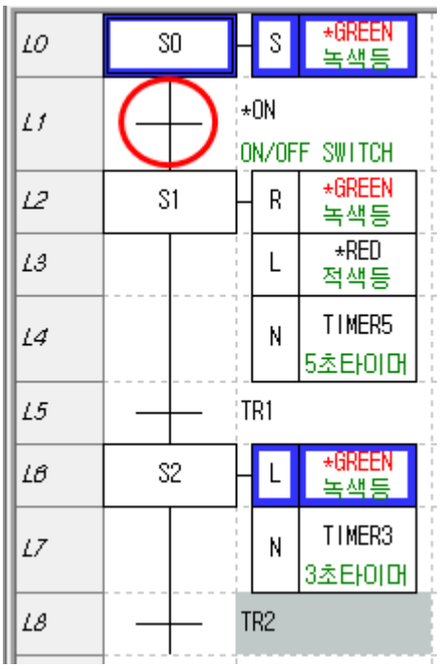
작성한 프로그램을 PLC 에 쓰는 과정과 모니터링 방법에 대해 설명합니다.



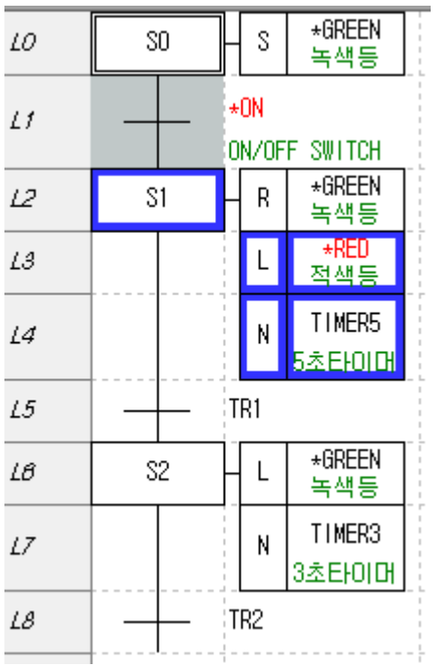
①[온라인] -> [쓰기] 를 클릭합니다.

② 프로그램 검사가 완료되었습니다.

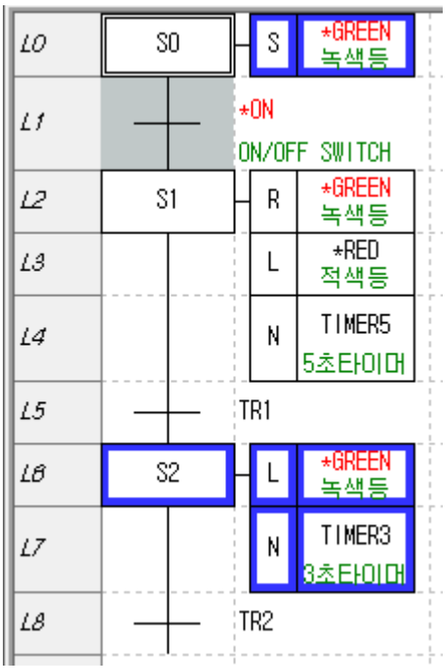
```
------------------------------ PLC: "NewPLC" ------------------------------
글로벌 변수 검사 중...

NewProgram[로컬 변수] 검사 중...

NewProgram[프로그램] 검사 중...

        블록 프로그램 "NewProgram" 검사 중...

        액션 프로그램 "TIMER5" 검사 중...

        액션 프로그램 "TIMER3" 검사 중...

        트랜지션 프로그램 "TR1" 검사 중...

        트랜지션 프로그램 "TR2" 검사 중...


링크 검사 중...

이중 코일 검사 중...
발견된 이중 코일이 없습니다.

사용안된 변수 검사 중...
NewProgram[로컬 변수] 검사 중...

글로벌 변수 검사 중...


실행 프로그램이 2516 Bytes 사용 되었습니다. (3.8%, 최대 64 KB)
업로드 프로그램이 3209 Bytes 사용 되었습니다. (2.4%, 최대 128 KB)
자동 할당 변수가 431 Bytes 사용 되었습니다. (0.7%, 최대 64 KB)

발견된 오류가 없습니다.
```



③ RUN 진입 후 초기 상태입니다.
-녹색등 점등 상태



④ ON 변수 ON 으로 설정 시
-적색등 5 초간 점등 상태



⑤ 5 초 뒤 다시 녹색등 점등 상태
-3 초 뒤 적색등 점등 이후 계속 반복

4.6. 부록

4.6.1. 단축키

다음의 단축키는 커서 이동에 관한 단축키입니다. 해당 단축키는 XG5000 에서 재정의 할 수 없습니다.

단축키	설명
F3	스텝을 입력합니다.
F4	액션을 입력합니다.
F5	블록스텝을 입력합니다.
F6	레이블을 입력합니다.
F7	점프를 입력합니다.
F8	왼쪽 분기를 입력합니다.
F9	오른쪽 분기를 입력합니다.
Home	열의 시작으로 이동합니다.
Ctrl+Home	프로그램의 시작으로 이동합니다.
→	현재 커서를 오른쪽으로 한 칸 이동합니다.
←	현재 커서를 왼쪽으로 한 칸 이동합니다.
↑	현재 커서를 위쪽으로 한 칸 이동합니다.
↓	현재 커서를 아래쪽으로 한 칸 이동합니다.
End	열의 끝으로 이동합니다.
Ctrl+End	편집된 가장 마지막 줄로 이동합니다.

4.6.2. 제한 사항

SFC 프로그램 편집 시 다음과 같은 기능 제한이 있습니다.

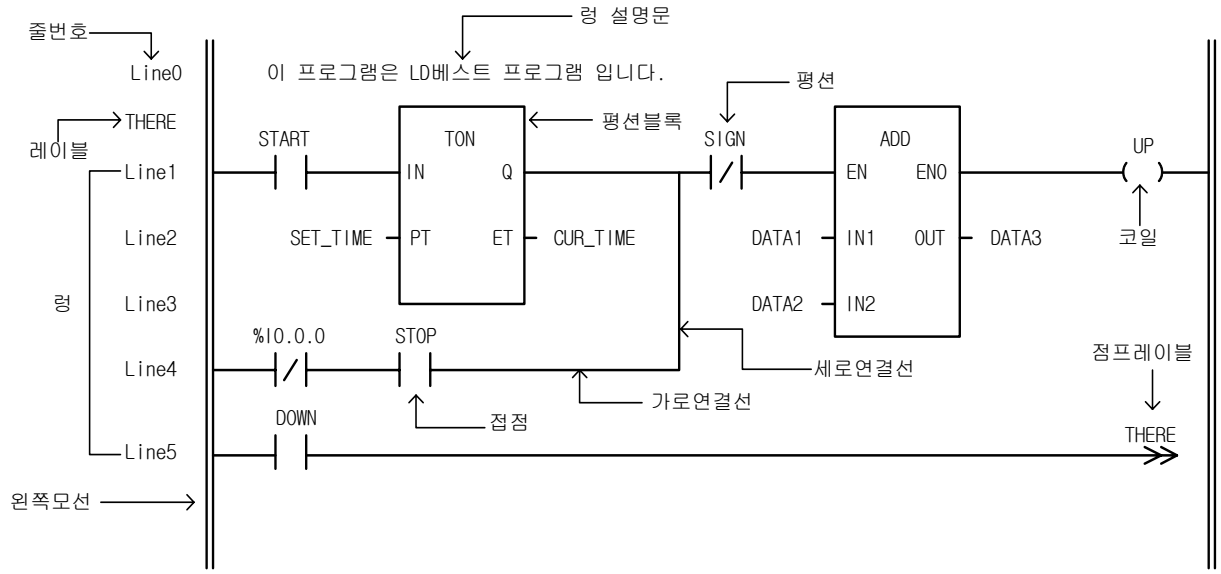
항목	내용	제한사항
최대 스텝 수	스텝 변수 사용한 스텝을 제외하고, 프로그램 내에서 사용 가능한 최대 스텝의 수를 의미합니다.	2,048 개
최대 라인(row) 수	편집 가능한 최대 라인의 수를 의미합니다.	65,535 라인
최대 열(column) 수	편집 가능한 최대 열의 수를 의미합니다.	65,535 열

제5장 LD(Ladder Diagram)

5.1. 개요

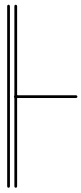
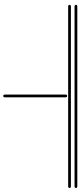
▷ LD 프로그램은 릴레이 로직 다이어그램에서 많이 사용하는 코일이나 접점 등의 그래픽 기호를 통하여 PLC 의 프로그램을 표현하는 것입니다.

▷ 형태



5.2. 모선

▷ LD 그래픽 구성도의 왼쪽 끝과 오른쪽 끝에는 전원선 개념의 모선이 세로로 양쪽에 놓여 있게 됩니다.

No	기호	이름	설명
1		왼쪽 모선	언제나 BOOL 1 의 값을 가지고 있습니다.
2		오른쪽 모선	값은 정해지지 않습니다

5.3. 연결선

▷ 왼쪽 모선의 BOOL 1 값은 작성한 도면에 따라 오른쪽으로 전달됩니다. 그 전달되는 값을 가진 선을 전원 흐름선 또는 연결선이라고 하며, 접점이나 코일에 연결되어 있는 선입니다. 전원 흐름선은 언제나 BOOL 값을 가지고 있으며, 한 렁(Rung)에서 하나만 존재합니다. 여기서 렁이란 LD 의 처음부터 밑으로 내려가는 선이 없는 줄까지를 말합니다.

▷ LD 의 각 요소를 연결하는 연결선에는 가로 연결선과 세로 연결선이 있습니다.

No.	기호	이름	설명
1	———	가로연결선	왼쪽의 값을 오른쪽으로 전달
2		세로연결선	왼쪽에 있는 가로 연결선들의 논리합

5.4. 접점

▷ 접점은 왼쪽에 있는 가로연결선의 상태와 현 접점과 연관된 BOOL 입력, 출력, 또는 메모리 변수 간의 논리곱(Boolean AND)을 한 값을 오른쪽에 위치한 가로 연결선에 전달합니다. 접점과 관련된 변수 값 자체는 변화시키지 않습니다. 표준 접점 기호는 다음 표와 같습니다.

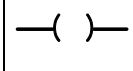
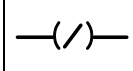
정적 접점			
No	기호	이름	설명
1	*** 	평상시 열린 접점 (Normally Open Contact)	BOOL 변수("***" 로 표시된 것)의 상태가 On 일 때에는 왼쪽의 연결선 상태는 오른쪽의 연결선으로 복사됩니다. 그렇지 않을 경우에는 오른쪽의 연결선 상 태가 Off 입니다.
2	***	평상시 닫힌 접점 (Normally Closed Contact)	BOOL 변수("***" 로 표시된 것)의 상태가 Off 일 때에는 왼쪽의 연결선 상태는 오른쪽의 연결선으로 복사됩니다. 그렇지 않을 경우에는 오른쪽의 연결선 상태가 Off 입니다.

상태 변환 검출 접점			
No	기호	이름	설명
3	*** 	양 변환 검출 접점 (Positive Transition-Sensing Contact)	BOOL 변수("***" 로 표시된 것)의 값이 전 스캔에서 Off 였던 것이 현재 스캔에서 On 으로 되고, 왼쪽 연결선 상태가 On 되어 있는 경우에 한해서 오른쪽의 연결선 상태는 현재 스캔 동안에 On 이 됩니다.
4	*** 	음 변환 검출 접점 (Negative Transition-Sensing Contact)	BOOL 변수("***" 로 표시된 것)의 값이 전 스캔에서 On 이었던 것이 현재 스캔에서 Off 되고 왼쪽 연결선 상태가 On 되어 있는 경우에 한해서 오른쪽의 연결선 상태는 현재 스캔 동안에 On 이 됩니다.

5.5. 코일

▷ 코일은 왼쪽의 연결선의 상태 또는 상태 변환에 대한 처리 결과를 연관된 BOOL 변수에 저장시킵니다. 표준 코일 기호는 다음 표와 같습니다.

▷ 코일은 LD 의 가장 오른쪽에만 올 수 있습니다. 즉 코일의 우측에는 언제나 오른쪽 모선만 있습니다.

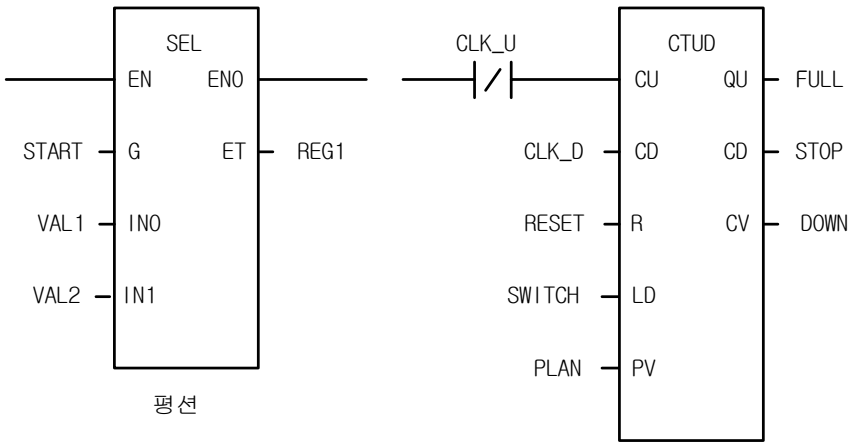
임시 코일(Momentary Coils)			
No.	기호	이름	설명
1	*** 	코일(Coil)	왼쪽에 있는 연결선의 상태를 관련된 BOOL 변수("***" 로 표시된 것)에 넣습니다.
2	*** 	역 코일(Negated Coil)	왼쪽에 있는 연결선 상태의 역(Negated)값을 관련된 BOOL 변수("***" 로 표시된 것)에 넣습니다. 즉, 왼쪽 연결선 상태 Off 이면 관련된 변수를 On 시키고, 왼쪽 연결선 상태가 On 이면 관련된 변수를 Off 시킵니다.
래치 코일(Latched Coils)			
No.	기호	이름	설명
3	*** 	Set(Latch) Coil	왼쪽의 연결선 상태가 On 이 되었을 때에는 관련된 BOOL 변수("***" 로 표시 된 것)는 On 이 되고 Reset 코일에 의해 Off 되기 전까지는 On 되어 있는 상태로 유지됩니다.
4	*** 	Reset(Unlatch) Coil	왼쪽의 연결선 상태가 On 이 되었을 때에는 관련된 BOOL 변수("***" 로 표시 된 것)는 Off 되고 Set 코일에 의해 On 되기 전까지는 Off 되어 있는 상태로 유지됩니다.

상태 변환 검출 코일(Transition-Sensing Coils)			
No.	기호	이름	설명
5	*** —(P)—	양 변환 검출 코일 (Positive Transition-Sensing Coil)	왼쪽 연결선 상태가 바로 전 스캔에서 Off 였던 것이 현재 스캔에서 On 이 되어 있는 경우에 관련된 BOOL 변수("***" 로 표시된 것)의 값은 현재 스캔 동안만 On 이 됩니다.
6	*** —(N)—	음 변환 검출 코일 (Negative Transition-Sensing Coil)	왼쪽 연결선 상태가 바로 전 스캔에서 On 이었던 것이 현재 스캔에서 Off 되어 있는 경우에 관련된 BOOL 변수("***" 로 표시된 것)의 값은 현재 스캔 동안만 On 이 됩니다.

5.6. 펑션과 펑션 블록의 호출

▷ 펑션과 펑션 블록에 대한 실제적인 입출력 연결은 입출력 표시가 있는 블록 외부에 적절한 데이터 또는 변수를 기입함으로써 이루어집니다.

예

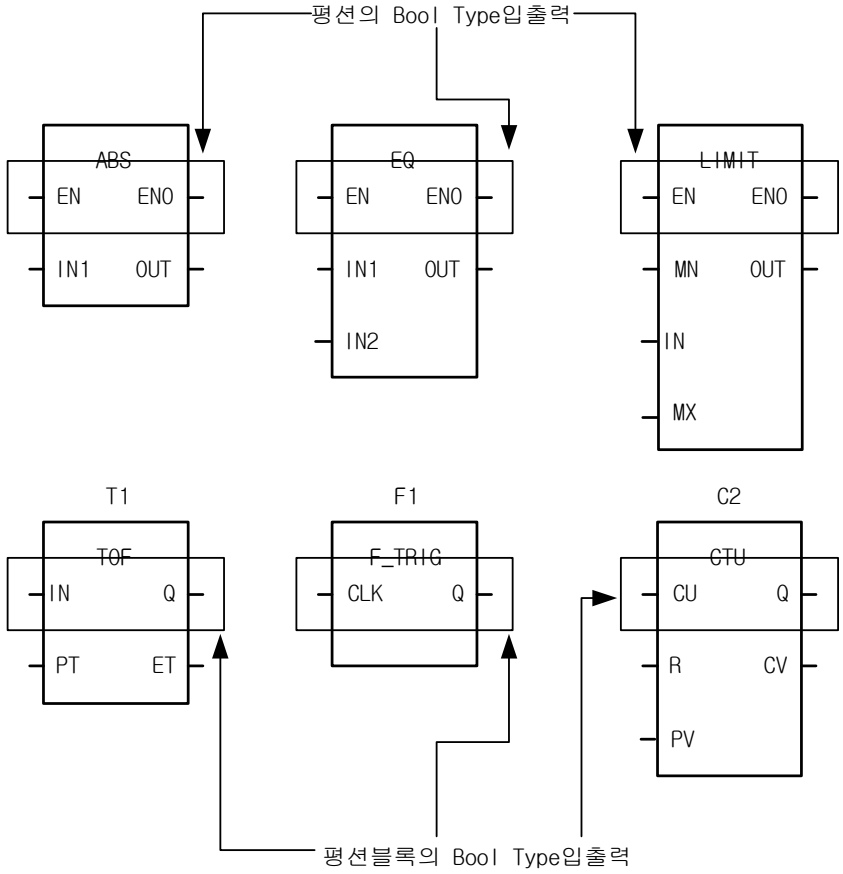


펑션

펑션블록

▷ 펑션이나 펑션 블록 내부로의 전원 흐름(Power Flow)을 허용하기 위해서는 적어도 한 개의 BOOL 타입 입력과 BOOL 타입 출력이 각 펑션이나 펑션 블록마다 존재해야 합니다. 펑션에서는 EN 과 ENO 가 BOOL 타입 입출력이며, 펑션 블록에서는 첫 번째 입력과 첫 번째 출력의 데이터 타입이 BOOL 입니다.

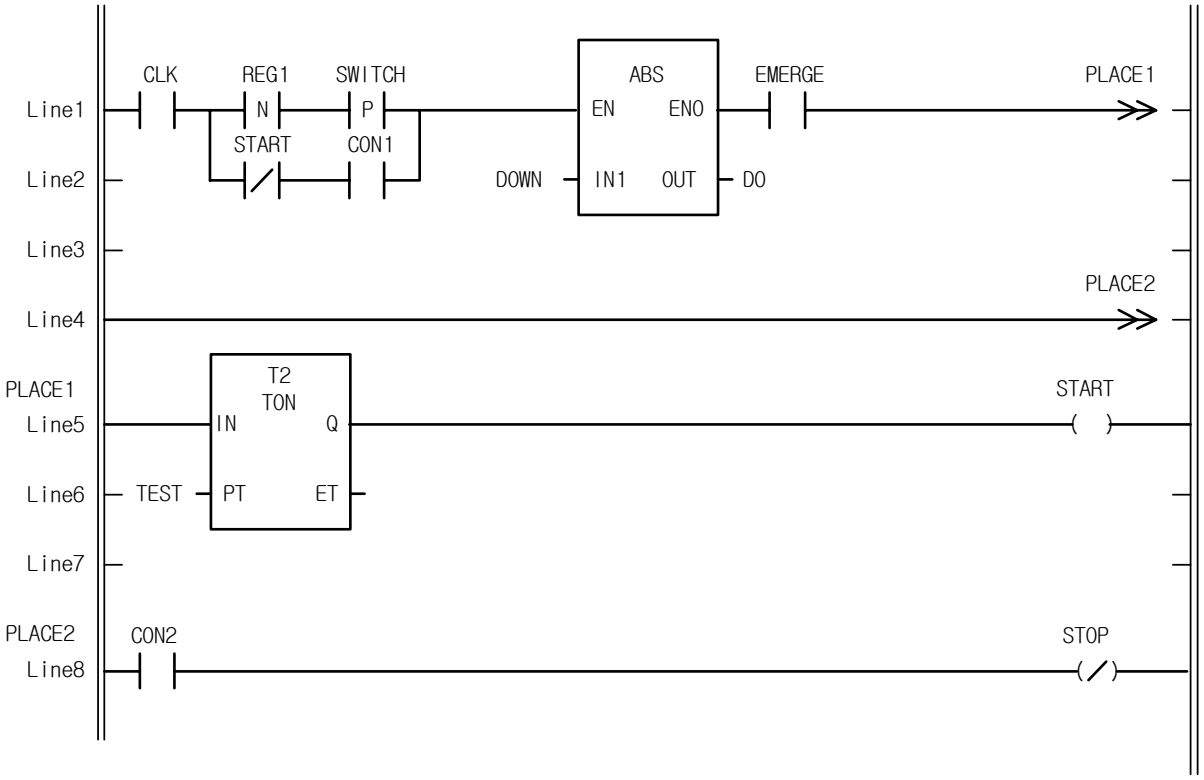
예



▷ LD 에서의 펑션은 EN 이라는 입력과 ENO 라는 출력이 붙습니다. 두 개 다 BOOL 데이터 타입을 가지고 있으며, EN 입력 값이 BOOL 1 이면 그 펑션을 수행하고, BOOL 0 이면 그 펑션을 수행하지 않습니다. ENO 출력은 보통 EN 값이 그대로 나오지만 그 펑션의 수행 시 에러가 발생하면 EN 값이 BOOL 1 이라도 ENO 값은 BOOL 0 이 나옵니다. 펑션의 EN 은 언제나 전원 흐름선이 되어야 하지만 ENO 는 꼭 전원 흐름선이 될 필요는 없습니다. 하지만 ENO 가 아닌 펑션 출력에 전원 흐름선을 연결할 때에는 그 출력의 데이터 타입이 반드시 BOOL 이어야 합니다. 또한 ENO 가 아닌 펑션 출력에 전원 흐름선을 연결할 때에는 ENO 출력에는 아무것도 연결하면 안됩니다. 펑션의 모든 입력은 펑션의 왼쪽에 그 값을 기입함으로써 지정되는데 빠짐없이 지정하여야 합니다. 펑션의 출력 값은 펑션의 오른쪽에 지정한 변수에 보관됩니다.

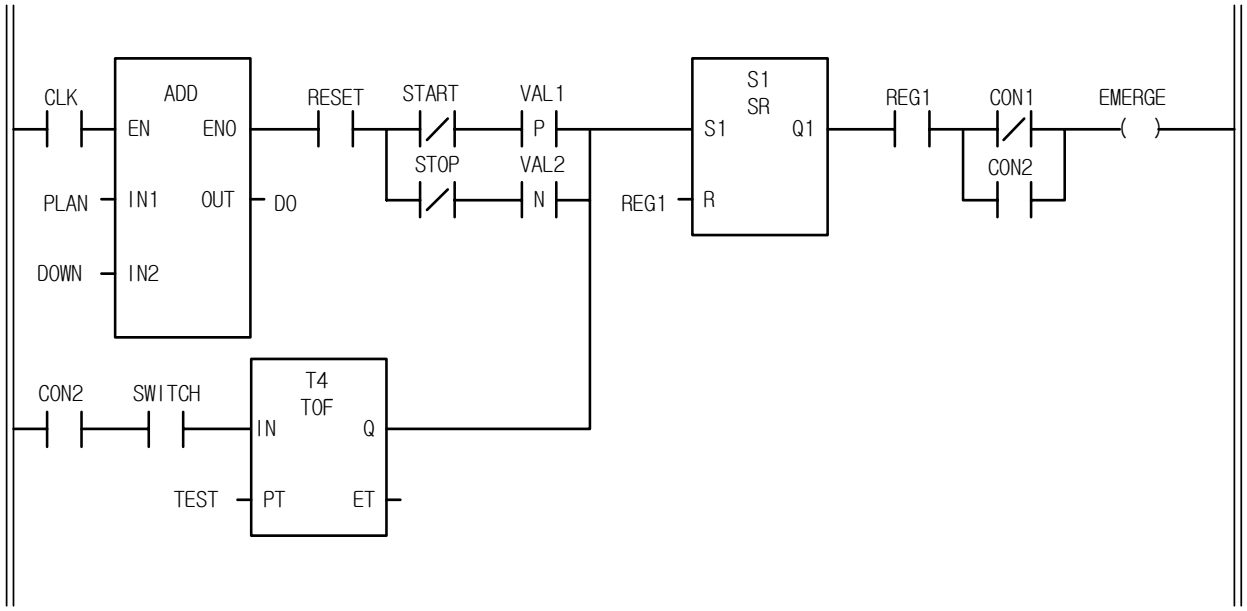
▷ LD 펑션 블록의 입력도 펑션과 같은 방법으로 지정합니다. 펑션 블록의 출력은 그 인스턴스 안에 저장되어 있으므로 변수를 지정하지 않더라도 상관없습니다. 펑션 블록에는 EN, ENO 입출력이 없으므로 펑션 블록을 만나면 항시 수행합니다. 따라서 어떤 로직 결과에 따라 펑션 블록의 수행 여부를 결정하기 위해서는 점프(—>>)를 사용하여야 합니다. 펑션 블록에 전원 흐름선을 연결할 때는 역시 데이터 타입이 BOOL 인 입출력에 연결하여야 합니다.

예



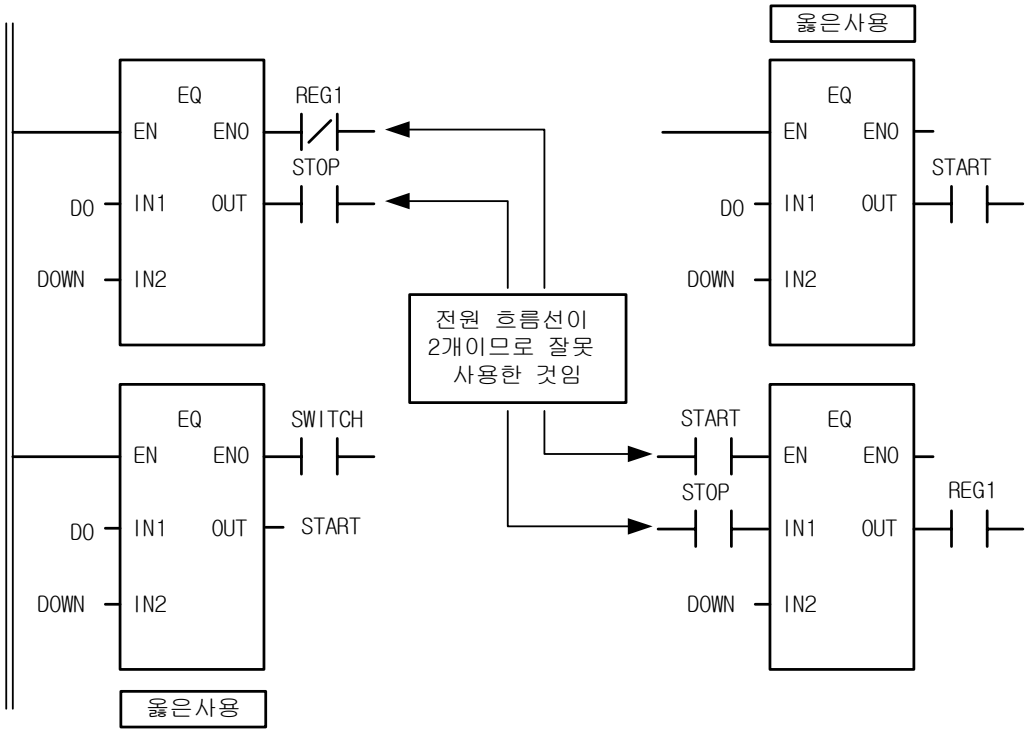
▷ LD 에서 펑션과 펑션 블록은 어느 곳에라도 올 수 있습니다. 펑션과 펑션 블록의 출력에 전원 흐름선을 연결하고 거기에 접점 등을 연결하여 로직 연산을 계속할 수도 있습니다.

예



▷ 하나의 펑션이나 펑션 블록에 연결할 수 있는 전원 흐름선은 단 하나입니다.

예



제6장 펑션과 펑션 블록

펑션과 펑션 블록에 대한 목록 요약입니다. 각각의 펑션과 펑션 블록에 대해서는 7장, 8장 기본/응용 펑션과 9장, 10장 기본/응용 펑션 블록을 참고 하십시오.

6.1. 기본 펑션

6.1.1. 타입 변환 펑션

각각의 입력 데이터 타입을 출력 데이터 타입으로 변환합니다.

펑션 그룹	펑션 이름	입력 데이터 타입	출력 데이터 타입	비 고
ARY_ASC_TO_***	ARY_ASC_TO_BYTE	WORD(ASCII)	BYTE	-
	ARY_ASC_TO_BCD	WORD(ASCII)	BYTE(BCD)	-
ARY_BYTE_TO_***	ARY_BYTE_TO_ASC	BYTE	WORD(ASCII)	-
ARY_BCD_TO_***	ARY_BCD_TO_ ASC	BYTE(BCD)	WORD(ASCII)	-
ASC_TO_***	ASC_TO_BCD	BYTE(BCD)	USINT	-
	ASC_TO_BYTE	WORD(BCD)	UINT	-
BCD_TO_***	BYTE_BCD_TO_SINT	BYTE(BCD)	SINT	-
	WORD_BCD_TO_INT	WORD(BCD)	INT	-
	DWORD_BCD_TO_DINT	DWORD(BCD)	DINT	-
	LWORD_BCD_TO_LINT	LWORD(BCD)	LINT	-
	BYTE_BCD_TO_USINT	BYTE(BCD)	USINT	-
	WORD_BCD_TO_UINT	WORD(BCD)	UINT	-
	DWORD_BCD_TO_UDINT	DWORD(BCD)	UDINT	-
	LWORD_BCD_TO_ULINT	LWORD(BCD)	ULINT	-
BCD_TO_ASC	BCD_TO_ASC	BYTE(BCD)	WORD	-
BYTE_TO_ASC	BYTE_TO_ASC	BYTE	ASC(BYTE)	-
TRUNC	TRUNC_REAL	REAL	DINT	-
	TRUNC_LREAL	LREAL	LINT	-
REAL_TO_***	REAL_TO_SINT	REAL	SINT	-
	REAL_TO_INT	REAL	INT	-
	REAL_TO_DINT	REAL	DINT	-
	REAL_TO_LINT	REAL	LINT	-
	REAL_TO_USINT	REAL	USINT	-
	REAL_TO_UINT	REAL	UINT	-
	REAL_TO_UDINT	REAL	UDINT	-
	REAL_TO_ULINT	REAL	ULINT	-
	REAL_TO_DWORD	REAL	DWORD	-
	REAL_TO_LREAL	REAL	LREAL	-
	REAL_TO_STRING	REAL	STRING	-
LREAL_TO_***	LREAL_TO_SINT	LREAL	SINT	-
	LREAL_TO_INT	LREAL	INT	-
	LREAL_TO_DINT	LREAL	DINT	-
	LREAL_TO_LINT	LREAL	LINT	-
	LREAL_TO_USINT	LREAL	USINT	-

펑션 그룹	펑션 이름	입력 데이터 타입	출력 데이터 타입	비 고
LREAL_TO_***	LREAL_TO_UINT	LREAL	UINT	-
	LREAL_TO_UDINT	LREAL	UDINT	-
	LREAL_TO_ULINT	LREAL	ULINT	-
	LREAL_TO_LWORD	LREAL	LWORD	-
	LREAL_TO_REAL	LREAL	REAL	-
	LREAL_TO_STRING	LREAL	STRING	-
SINT_TO_***	SINT_TO_INT	SINT	INT	-
	SINT_TO_DINT	SINT	DINT	-
	SINT_TO_LINT	SINT	LINT	-
	SINT_TO_USINT	SINT	USINT	-
	SINT_TO_UINT	SINT	UINT	-
	SINT_TO_UDINT	SINT	UDINT	-
	SINT_TO_ULINT	SINT	ULINT	-
	SINT_TO_BOOL	SINT	BOOL	-
	SINT_TO_BYTE	SINT	BYTE	-
	SINT_TO_WORD	SINT	WORD	-
	SINT_TO_DWORD	SINT	DWORD	-
	SINT_TO_LWORD	SINT	LWORD	-
	SINT_TO_REAL	SINT	REAL	-
	SINT_TO_LREAL	SINT	LREAL	-
	SINT_TO_STRING	SINT	STRING	-
INT_TO_***	INT_TO_SINT	INT	SINT	-
	INT_TO_DINT	INT	DINT	-
	INT_TO_LINT	INT	LINT	-
	INT_TO_USINT	INT	USINT	-
	INT_TO_UINT	INT	UINT	-
	INT_TO_UDINT	INT	UDINT	-
	INT_TO_ULINT	INT	ULINT	-
	INT_TO_BOOL	INT	BOOL	-
	INT_TO_BYTE	INT	BYTE	-
	INT_TO_WORD	INT	WORD	-
	INT_TO_DWORD	INT	DWORD	-
	INT_TO_LWORD	INT	LWORD	-
	INT_TO_REAL	INT	REAL	-
	INT_TO_LREAL	INT	LREAL	-
	INT_TO_STRING	INT	STRING	-
DINT_TO_***	DINT_TO_SINT	DINT	SINT	-
	DINT_TO_INT	DINT	INT	-
	DINT_TO_LINT	DINT	LINT	-
	DINT_TO_USINT	DINT	USINT	-
	DINT_TO_UINT	DINT	UINT	-
	DINT_TO_UDINT	DINT	UDINT	-
	DINT_TO_ULINT	DINT	ULINT	-
	DINT_TO_BOOL	DINT	BOOL	-
	DINT_TO_BYTE	DINT	BYTE	-
	DINT_TO_WORD	DINT	WORD	-

펑션 그룹	펑션 이름	입력 데이터 타입	출력 데이터 타입	비 고
DINT_TO_***	DINT_TO_DWORD	DINT	DWORD	-
	DINT_TO_LWORD	DINT	LWORD	-
	DINT_TO_REAL	DINT	REAL	-
	DINT_TO_LREAL	DINT	LREAL	-
	DINT_TO_STRING	DINT	STRING	-
LINT_TO_***	LINT_TO_SINT	LINT	SINT	-
	LINT_TO_INT	LINT	INT	-
	LINT_TO_DINT	LINT	DINT	-
	LINT_TO_USINT	LINT	USINT	-
	LINT_TO_UINT	LINT	UINT	-
	LINT_TO_UDINT	LINT	UDINT	-
	LINT_TO_ULINT	LINT	ULINT	-
	LINT_TO_BOOL	LINT	BOOL	-
	LINT_TO_BYTE	LINT	BYTE	-
	LINT_TO_WORD	LINT	WORD	-
	LINT_TO_DWORD	LINT	DWORD	-
	LINT_TO_LWORD	LINT	LWORD	-
	LINT_TO_REAL	LINT	REAL	-
	LINT_TO_LREAL	LINT	LREAL	-
	LINT_TO_STRING	LINT	STRING	-
USINT_TO_***	USINT_TO_SINT	USINT	SINT	-
	USINT_TO_INT	USINT	INT	-
	USINT_TO_DINT	USINT	DINT	-
	USINT_TO_LINT	USINT	LINT	-
	USINT_TO_UINT	USINT	UINT	-
	USINT_TO_UDINT	USINT	UDINT	-
	USINT_TO_ULINT	USINT	ULINT	-
	USINT_TO_BOOL	USINT	BOOL	-
	USINT_TO_BYTE	USINT	BYTE	-
	USINT_TO_WORD	USINT	WORD	-
	USINT_TO_DWORD	USINT	DWORD	-
	USINT_TO_LWORD	USINT	LWORD	-
	USINT_TO_REAL	USINT	REAL	-
	USINT_TO_LREAL	USINT	LREAL	-
	USINT_TO_STRING	USINT	STRING	-
UINT_TO_***	UINT_TO_SINT	UINT	SINT	-
	UINT_TO_INT	UINT	INT	-
	UINT_TO_DINT	UINT	DINT	-
	UINT_TO_LINT	UINT	LINT	-
	UINT_TO_USINT	UINT	USINT	-
	UINT_TO_UDINT	UINT	UDINT	-
	UINT_TO_ULINT	UINT	ULINT	-
	UINT_TO_BOOL	UINT	BOOL	-
	UINT_TO_BYTE	UINT	BYTE	-
	UINT_TO_WORD	UINT	WORD	-
	UINT_TO_DWORD	UINT	DWORD	-

펑션 그룹	펑션 이름	입력 데이터 타입	출력 데이터 타입	비 고
UINT_TO_***	UINT_TO_LWORD	UINT	LWORD	-
	UINT_TO_REAL	UINT	REAL	-
	UINT_TO_STRING	UINT	STRING	-
	UINT_TO_LREAL	UINT	LREAL	-
	UINT_TO_DATE	UINT	DATE	-
UDINT_TO_***	UDINT_TO_SINT	UDINT	SINT	-
	UDINT_TO_INT	UDINT	INT	-
	UDINT_TO_DINT	UDINT	DINT	-
	UDINT_TO_LINT	UDINT	LINT	-
	UDINT_TO_USINT	UDINT	USINT	-
	UDINT_TO_UINT	UDINT	UINT	-
	UDINT_TO_ULINT	UDINT	ULINT	-
	UDINT_TO_BOOL	UDINT	BOOL	-
	UDINT_TO_BYTE	UDINT	BYTE	-
	UDINT_TO_WORD	UDINT	WORD	-
	UDINT_TO_DWORD	UDINT	DWORD	-
	UDINT_TO_LWORD	UDINT	LWORD	-
	UDINT_TO_REAL	UDINT	REAL	-
	UDINT_TO_LREAL	UDINT	LREAL	-
	UDINT_TO_TOD	UDINT	TOD	-
	UDINT_TO_TIME	UDINT	TIME	-
	UDINT_TO_STRING	UDINT	STRING	-
ULINT_TO_***	ULINT_TO_SINT	ULINT	SINT	-
	ULINT_TO_INT	ULINT	INT	-
	ULINT_TO_DINT	ULINT	DINT	-
	ULINT_TO_LINT	ULINT	LINT	-
	ULINT_TO_USINT	ULINT	USINT	-
	ULINT_TO_UINT	ULINT	UINT	-
	ULINT_TO_UDINT	ULINT	UDINT	-
	ULINT_TO_BOOL	ULINT	BOOL	-
	ULINT_TO_BYTE	ULINT	BYTE	-
	ULINT_TO_WORD	ULINT	WORD	-
	ULINT_TO_DWORD	ULINT	DWORD	-
	ULINT_TO_LWORD	ULINT	LWORD	-
	ULINT_TO_REAL	ULINT	REAL	-
	ULINT_TO_LREAL	ULINT	LREAL	-
	ULINT_TO_STRING	ULINT	STRING	-
BOOL_TO_***	BOOL_TO_SINT	BOOL	SINT	-
	BOOL_TO_INT	BOOL	INT	-
	BOOL_TO_DINT	BOOL	DINT	-
	BOOL_TO_LINT	BOOL	LINT	-
	BOOL_TO_USINT	BOOL	USINT	-
	BOOL_TO_UINT	BOOL	UINT	-
	BOOL_TO_UDINT	BOOL	UDINT	-
	BOOL_TO_ULINT	BOOL	ULINT	-
	BOOL_TO_BYTE	BOOL	BYTE	-

펑션 그룹	펑션 이름	입력 데이터 타입	출력 데이터 타입	비 고
BOOL_TO_***	BOOL_TO_WORD	BOOL	WORD	-
	BOOL_TO_DWORD	BOOL	DWORD	-
	BOOL_TO_LWORD	BOOL	LWORD	-
	BOOL_TO_STRING	BOOL	STRING	-
BYTE_TO_***	BYTE_TO_SINT	BYTE	SINT	-
	BYTE_TO_INT	BYTE	INT	-
	BYTE_TO_DINT	BYTE	DINT	-
	BYTE_TO_LINT	BYTE	LINT	-
	BYTE_TO_USINT	BYTE	USINT	-
	BYTE_TO_UINT	BYTE	UINT	-
	BYTE_TO_UDINT	BYTE	UDINT	-
	BYTE_TO_ULINT	BYTE	ULINT	-
	BYTE_TO_BOOL	BYTE	BOOL	-
	BYTE_TO_WORD	BYTE	WORD	-
	BYTE_TO_DWORD	BYTE	DWORD	-
	BYTE_TO_LWORD	BYTE	LWORD	-
	BYTE_TO_STRING	BYTE	STRING	-
WORD_TO_***	WORD_TO_SINT	WORD	SINT	-
	WORD_TO_INT	WORD	INT	-
	WORD_TO_DINT	WORD	DINT	-
	WORD_TO_LINT	WORD	LINT	-
	WORD_TO_USINT	WORD	USINT	-
	WORD_TO_UINT	WORD	UINT	-
	WORD_TO_UDINT	WORD	UDINT	-
	WORD_TO_ULINT	WORD	ULINT	-
	WORD_TO_BOOL	WORD	BOOL	-
	WORD_TO_BYTE	WORD	BYTE	-
	WORD_TO_DWORD	WORD	DWORD	-
	WORD_TO_LWORD	WORD	LWORD	-
	WORD_TO_DATE	WORD	DATE	-
	WORD_TO_STRING	WORD	STRING	-
DWORD_TO_***	DWORD_TO_SINT	DWORD	SINT	-
	DWORD_TO_INT	DWORD	INT	-
	DWORD_TO_DINT	DWORD	DINT	-
	DWORD_TO_LINT	DWORD	LINT	-
	DWORD_TO_USINT	DWORD	USINT	-
	DWORD_TO_UINT	DWORD	UINT	-
	DWORD_TO_UDINT	DWORD	UDINT	-
	DWORD_TO_ULINT	DWORD	ULINT	-
	DWORD_TO_BOOL	DWORD	BOOL	-
	DWORD_TO_BYTE	DWORD	BYTE	-
	DWORD_TO_WORD	DWORD	WORD	-
	DWORD_TO_LWORD	DWORD	LWORD	-
	DWORD_TO_REAL	DWORD	REAL	-
	DWORD_TO_TIME	DWORD	TIME	-
	DWORD_TO_TOD	DWORD	TOD	-

펑션 그룹	펑션 이름	입력 데이터 타입	출력 데이터 타입	비 고
DWORD_TO_***	DWORD_TO_STRING	DWORD	STRING	-
LWORD_TO_***	LWORD_TO_SINT	LWORD	SINT	-
	LWORD_TO_INT	LWORD	INT	-
	LWORD_TO_DINT	LWORD	DINT	-
	LWORD_TO_LINT	LWORD	LINT	-
	LWORD_TO_USINT	LWORD	USINT	-
	LWORD_TO_UINT	LWORD	UINT	-
	LWORD_TO_UDINT	LWORD	UDINT	-
	LWORD_TO_ULINT	LWORD	ULINT	-
	LWORD_TO_BOOL	LWORD	BOOL	-
	LWORD_TO_BYTE	LWORD	BYTE	-
	LWORD_TO_WORD	LWORD	WORD	-
	LWORD_TO_DWORD	LWORD	DWORD	-
	LWORD_TO_LREAL	LWORD	LREAL	-
	LWORD_TO_DT	LWORD	DT	-
	LWORD_TO_STRING	LWORD	STRING	-
STRING_TO_***	STRING _TO_SINT	STRING	SINT	-
	STRING _TO_INT	STRING	INT	-
	STRING _TO_DINT	STRING	DINT	-
	STRING _TO_LINT	STRING	LINT	-
	STRING _TO_USINT	STRING	USINT	-
	STRING _TO_UINT	STRING	UINT	-
	STRING _TO_UDINT	STRING	UDINT	-
	STRING _TO_ULINT	STRING	ULINT	-
	STRING _TO_BOOL	STRING	BOOL	-
	STRING _TO_BYTE	STRING	BYTE	-
	STRING _TO_WORD	STRING	WORD	-
	STRING _TO_DWORD	STRING	DWORD	-
	STRING _TO_LWORD	STRING	LWORD	-
	STRING _TO_REAL	STRING	REAL	-
	STRING _TO_LREAL	STRING	LREAL	-
	STRING _TO_DT	STRING	DT	-
	STRING _TO_DATE	STRING	DATE	-
	STRING _TO_TOD	STRING	TOD	-
	STRING _TO_TIME	STRING	TIME	-
TIME_TO_***	TIME_TO_UDINT	TIME	UDINT	-
	TIME_TO_DWORD	TIME	DWORD	-
	TIME_TO_STRING	TIME	STRING	-
DATE_TO_***	DATE_TO_UINT	DATE	UINT	-
	DATE_TO_WORD	DATE	WORD	-
	DATE_TO_STRING	DATE	STRING	-
TOD_TO_***	TOD_TO_UDINT	TOD	UDINT	-
	TOD_TO_DWORD	TOD	DWORD	-
	TOD_TO_STRING	TOD	STRING	-

펑션 그룹	펑션 이름	입력 데이터 타입	출력 데이터 타입	비 고
DT_TO_***	DT_TO_LWORD	DT	LWORD	-
	DT_TO_DATE	DT	DATE	-
	DT_TO_TOD	DT	TOD	-
	DT_TO_STRING	DT	STRING	-
***_TO_BCD	SINT_TO_BCD_BYTE	SINT	BYTE(BCD)	-
	INT_TO_BCD_WORD	INT	WORD(BCD)	-
	DINT_TO_BCD_DWORD	DINT	DWORD(BCD)	-
	LINT_TO_BCD_LWORD	LINT	LWORD(BCD)	-
	USINT_TO_BCD_BYTE	USINT	BYTE(BCD)	-
	UINT_TO_BCD_WORD	UINT	WORD(BCD)	-
	UDINT_TO_BCD_DWORD	UDINT	DWORD(BCD)	-
	ULINT_TO_BCD_LWORD	ULINT	LWORD(BCD)	-

6.1.2. 수치 연산 펑션

1) 하나의 입력을 갖는 수치 연산 펑션

No.	펑션 이름	기 능	비 고
일반 펑션			
1	ABS	절대값 연산(Absolute Value)	-
2	SQRT	제곱근 연산(Square Root)	-
로그 펑션			
3	LN	자연 대수 연산(Natural Logarithm)	-
4	LOG	상용 대수 연산(Common Logarithm Base To 10)	-
5	EXP	자연 지수 연산(Natural Exponential)	-
삼각 펑션			
6	SIN	사인 값 연산(Sine)	-
7	COS	코사인 값 연산(Cosine)	-
8	TAN	탄젠트 값 연산(Tangent)	-
9	ASIN	아크 사인 값 연산(Arc Sine)	-
10	ACOS	아크 코사인 값 연산(Arc Cosine)	-
11	ATAN	아크 탄젠트 값 연산(Arc Tangent)	-
각도 펑션			
12	RAD_REAL	각도의 단위를 (°)에서 라디안(Radian)으로 변환	-
13	RAD_LREAL		
14	DEG_REAL	라디안(Radian)값을 각도(°)로 변환	-
15	DEG_LREAL		

2) 기본 수치 연산 펑션

No.	펑션 이름	기 능	비 고
입력 개수를 확장할 수 있는 연산 펑션(단, n은 8까지 가능)			
1	ADD	더하기 (OUT <= IN1 + IN2 + ... + INn)	-
2	MUL	곱하기 OUT <= IN1 * IN2 * ... * INn)	-
입력 개수가 일정한 연산 펑션			
3	SUB	빼기(OUT <= IN1 - IN2)	-
4	DIV	나누기(OUT <= IN1 / IN2)	-
5	MOD	나머지 구하기(OUT <= IN1 Modulo IN2)	-
6	EXPT	지수 연산(OUT <= $IN1^{IN2}$)	-
7	MOVE	데이터 복사(OUT <= IN)	-
입력 데이터 값 교환			
8	XCHG_***	입력 데이터 값을 서로 교환	-

6.1.3. 비트열 펑션

1) 비트 시프트 펑션

No.	펑션 이름	기 능	비 고
1	SHL	입력을 N 비트 왼쪽으로 이동(오른쪽은 0 으로 채움)	-
2	SHR	입력을 N 비트 오른쪽으로 이동(왼쪽은 0 으로 채움)	-
3	SHIFT_C_***	입력을 N 비트만큼 지정된 방향으로 이동(Carry 발생)	-
4	ROL	입력을 N 비트 왼쪽으로 회전	-
5	ROR	입력을 N 비트 오른쪽으로 회전	-
6	ROTATE_C_***	입력을 N 비트만큼 지정된 방향으로 회전(Carry 발생)	-

2) 비트 연산 펑션

No.	펑션 이름	기 능 (단, n은 8까지 가능)	비 고
1	AND	논리곱(OUT <= IN1 AND IN2 AND ... AND INn)	-
2	OR	논리합(OUT <= IN1 OR IN2 OR ... OR INn)	-
3	XOR	배타적 논리합(OUT <= IN1 XOR IN2 XOR ... XOR INn)	-
4	NOT	논리 반전(OUT <= NOT IN1)	-
5	XNR	배타적 논리곱(OUT <= IN1 XNR IN2 XNR ... XNR INn)	-

6.1.4. 선택 펑션

No.	펑션 이름	기 능 (단, n은 8까지 가능)	비 고
1	SEL	입력 IN0 와 IN1 중에 선택하여 출력	-
2	MAX	입력 IN1,...INn 중에 최대값 출력	-
3	MIN	입력 IN1,...INn 중에 최소값 출력	-
4	LIMIT	상, 하한 제한 값 출력	-
5	MUX	입력 IN0, ... INn 중 K 번째 입력을 출력	-

6.1.5. 데이터 교환 펑션

No.	펑션 이름	기 능	비 고
1	SWAP_BYTE	BYTE 의 상 · 하위 Nibble 을 교환하여 출력	-
	SWAP_WORD	WORD 의 상 · 하위 BYTE 를 교환하여 출력	-
	SWAP_DWORD	DWORD 의 상 · 하위 WORD 를 교환하여 출력	-
	SWAP_LWORD	LWORD 의 상 · 하위 DWORD 를 교환하여 출력	-
2	ARY_SWAP_BYTE	Array 로 입력된 BYTE 의 상 · 하위 Nibble 을 교환하여 출력	-
	ARY_SWAP_WORD	Array 로 입력된 WORD 의 상 · 하위 BYTE 를 교환하여 출력	-
	ARY_SWAP_DWORD	Array 로 입력된 DWORD 의 상 · 하위 WORD 를 교환하여 출력	-
	ARY_SWAP_LWORD	Array 로 입력된 LWORD 의 상 · 하위 DWORD 를 교환하여 출력	-

6.1.6. 비교 펑션

No.	펑션 이름	기 능 (단, n은 8까지 가능)	비 고
1	GT	'크다' 비교 OUT <= (IN1>IN2) & (IN2>IN3) & ... & (INn-1 > INn)	-
2	GE	'크거나 작다' 비교 OUT <= (IN1>=IN2) & (IN2>=IN3) & ... & (INn-1 >= INn)	-
3	EQ	'같다' 비교 OUT <= (IN1=IN2) & (IN2=IN3) & ... & (INn-1 = INn)	-
4	LE	'작거나 같다' 비교 OUT <= (IN1<=IN2) & (IN2<=IN3) & ... & (INn-1 <= INn)	-
5	LT	'작다' 비교 OUT <= (IN1<IN2) & (IN2<IN3) & ... & (INn-1 < INn)	-
6	NE	'같지 않다' 비교 OUT <= (IN1<>IN2) & (IN2<>IN3) & ... & (INn-1 <> INn)	-

6.1.7. 문자열 펑션

No.	펑션 이름	기 능	비 고
1	LEN	입력 문자열의 길이 구하기	-
2	LEFT	입력 문자열을 왼쪽으로부터 L 만큼 출력	-
3	RIGHT	입력 문자열을 오른쪽으로부터 L 만큼 출력	-
4	MID	입력 문자열의 P 번째부터 L 만큼 출력	-
5	CONCAT	입력 문자열을 붙여 출력	-

No.	펑션 이름	기 능	비 고
6	INSERT	첫 번째 입력 문자열의 P 번째 문자 뒤에 두 번째 입력 문자열을 삽입하여 출력	-
7	DELETE	입력 문자열의 P 번째 문자부터 L 개 문자를 삭제하여 출력	-
8	REPLACE	첫 번째 입력 문자열의 P 번째 문자부터 L 개 문자를 두 번째 입력 문자열로 대치하여 출력	-
9	FIND	첫 번째 입력 문자열중에 두 번째 입력 문자열 패턴과 동일한 부분을 찾아 시작 문자 위치를 출력	-

6.1.8. 날짜 시각 펑션

No.	펑션 이름	기 능	비 고
1	ADD_TIME	시간, 시각 및 날짜 시각에 시간 더하기	-
2	SUB_TIME	시간, 시각 및 날짜 시각에 시간 빼기	-
	SUB_DATE	날짜에서 날짜를 빼서 시간 산출하기	-
	SUB_TOD	시각에서 시각을 빼서 시간 산출하기	-
	SUB_DT	날짜 시각에서 날짜 시각을 빼서 시간 산출하기	-
3	MUL_TIME	시간에 숫자 곱하기	-
4	DIV_TIME	시간에 숫자 나누기	-
5	CONCAT_TIME	날짜와 시각을 붙여서 날짜 시각 만들기	-

6.1.9. 시스템 제어 펑션

No.	펑션 이름	기 능	비 고
1	DI	태스크 프로그램 기동 불허	-
2	EI	태스크 프로그램 기동 허가	-
3	STOP	프로그램에 의한 운전정지	-
4	ESTOP	프로그램에 의한 비상 운전정지	-
5	DIREC_IN	입력 데이터 즉시 갱신	-
6	DIREC_O	출력 모듈 데이터 즉시 갱신	-
7	WDT_RST	Watch_Dog 타이머 갱신	-
8	MCS	Master Control	-
9	MCSCLR	Master Control Clear	-
10	FALS	자가진단(고장표시)	-
11	OUTOFF	전출력 Off	-

6.1.10. 파일관련 펑션

No.	펑션 블록 이름	기 능	비 고
1	RSET	파일 레지스터 블록 번호 설정	-
2	EBCMP	블록 비교	-
3	EMOV	설정한 플래쉬 영역으로부터 데이터 읽기	-
4	EERRST	플래시 메모리관련 에러플래그 클리어	-

6.1.11. 데이터 조작 명령 펑션

No.	펑션 이름	기 능	비 고
1	MEQ_***	입력 값을 Masking 한 후 이 값들을 비교	-
2	DIS_***	입력 값들을 지정된 Bit 개수 단위로 출력	-
3	UNI_***	Array로 입력된 값을 지정된 Bit수만큼 결합	-
4	BIT_BYTE	8개의 Bit들을 BYTE로 모음	-
5	BYTE_BIT	BYTE를 8개의 Bit로 나눔	-
6	BYTE_WORD	2개의 BYTE들을 WORD로 모음	-
7	WORD_BYTE	WORD를 2개의 BYTE로 나눔	-
8	WORD_DWORD	2개의 WORD들을 DWORD로 모음	-
9	DWORD_WORD	DWORD를 2개의 WORD로 나눔	-
10	DWORD_LWORD	2개의 DWORD들을 LWORD로 모음	-
11	LWORD_DWORD	LWORD를 2개의 DWORD로 나눔	-
12	GET_CHAR	지정된 문자열로부터 한문자(Character)를 추출	-
13	PUT_CHAR	지정된 문자를 지정된 문자열에 쓰기	-
14	STRING_BYTE	지정된 문자열을 BYTE Array로 변환	-
15	BYTE_STRING	BYTE Array를 지정된 문자열로 변환	-

6.1.12. 스택 연산 명령 펑션

No.	펑션 이름	기 능	비 고
1	FIFO_***	선입 선출 명령	-
2	LIFO_***	후입 선출 명령	-

6.2. MK(MASTER-K) 펑션

No.	펑션 이름	기 능(단, n은 8까지 가능)	비 고
1	ENCO_B,W,D,L	On 된 비트 위치를 숫자로 출력	-
2	DECO_B,W,D,L	지정된 비트 위치를 On	-
3	BSUM_B,W,D,L	On 된 비트 개수를 숫자로 출력	-
4	SEG_WORD	BCD 또는 HEX 값을 7 세그먼트 디스플레이 코드로 변환	-
5	BMOV_B,W,D,L	비트 스트링의 일부분을 복사, 이동	-
6	INC_B,W,D,L	IN 데이터를 하나 증가	-
7	DEC_B,W,D,L	IN 데이터를 하나 감소	-

6.3. Array 연산 명령 펑션

No.	펑션 이름	기 능	비 고
1	ARY_MOVE	Array Type 의 데이터 복사(OUT <= IN)	-
2	ARY_CMP_***	2 개의 Array 로 입력된 값을 동일한 값이 있는지 비교	-
3	ARY_SCH_***	Array 내에서 입력된 값과 동일한 값을 찾아 출력	-
4	ARY_FLL_***	입력 데이터 값으로 Array 내부의 선택 영역을 채움.	-
5	ARY_AVE_***	Array 내부의 지정된 영역에 대하여 평균값을 구함	-
6	ARY_SFT_C_***	Array 의 Bit 들을 정해진 개수만큼 지정된 방향으로 이동	-
7	ARY_ROT_C_***	Array 의 Bit 들을 정해진 개수만큼 지정된 방향으로 회전	-
8	SHIFT_A_***	Array 블록 중 지정된 범위의 값들을 지정된 방향으로 이동	-
9	ROTATE_A_***	Array 블록 중 지정된 범위의 값들을 지정된 방향으로 회전	-

6.4. 기본 펑션 블록

6.4.1. 바이스테이블 펑션 블록

No.	펑션 블록 이름	기 능	비 고
1	SR	세트 우선 쌍안정 출력	-
2	RS	리셋 우선 쌍안정 출력	-
3	SEMA	시스템 자원 제어용 Semaphore	-

6.4.2. 에지 검출 펑션 블록

No.	펑션 블록 이름	기 능	비 고
1	R_TRIG	상승 에지 검출(Rising Edge Detector)	-
2	F_TRIG	하강 에지 검출(Falling Edge Detector)	-
3	FF	입력조건 상승 시 출력 반전	-

6.4.3. 카운터

No.	펑션 블록 이름	기 능	비 고
1	CTU_***	가산 카운터(Up Counter) INT,DINT,LINT,UINT,UDINT,ULINT	-
2	CTD_***	감산 카운터(Down Counter) INT,DINT,LINT,UINT,UDINT,ULINT	-
3	CTUD_***	가감산 카운터(Up Down Counter) INT,DINT,LINT,UINT,UDINT,ULINT	-
4	CTR	링 카운터(Ring Counter)	-

6.4.4. 타이머

No.	펑션 블록 이름	기 능	비 고
1	TP	펄스 타이머(Pulse Timer)	-
2	TON	On 딜레이 타이머(On-Delay Timer)	-
3	TOF	Off 딜레이 타이머(Off-Delay Timer)	-
4	TMR	적산 타이머(Integrating Timer)	-
5	TP_RST	펄스 타이머의 출력 Off가 가능한 노스테이블 타이머	-
6	TRTG	리트리거블 타이머(Retriggerable Timer)	-
7	TOF_RST	동작 중 출력 Off가 가능한 Off 딜레이 타이머(Off-Delay Timer)	-
8	TON_UINT	정수 설정 On 딜레이 타이머(On-Delay Timer)	-
9	TOF_UINT	정수 설정 Off 딜레이 타이머(Off-Delay Timer)	-
10	TP_UINT	정수 설정 펄스 타이머(Pulse Timer)	-
11	TMR_UINT	정수 설정 적산 타이머(Integrating Timer)	-
12	TMR_FLK	점멸 기능 타이머	-
13	TRTG_UINT	정수 설정 리트리거블 타이머	-

6.4.5. 파일관련 펑션 블록

No.	펑션 블록 이름	기 능	비 고
1	EBREAD	R영역 데이터를 플래시 영역에서 읽어오기	-
2	EBWRITE	R영역 데이터를 플래시 영역에 쓰기	-

6.4.6. 기타 펑션 블록

No.	펑션 블록 이름	기 능	비 고
1	SCON	순차 스텝 및 스텝 점프	-
2	DUTY	지정된 Scan마다 On/Off 반복	-
3	RTC_SET	시간 데이터 쓰기	-

6.4.7. 통신 펑션 블록

No.	펑션 블록 이름	기 능	비 고
1	P2PSN	국번 설정	-
2	P2PRD	읽기 영역 지정	-
3	P2PWR	쓰기 영역 지정	-
4	SEND_UDATA	사용자 정의 데이터 송신	-
5	RCV_UDATA	사용자 정의 데이터 수신	-
6	SEND_DTR	통신 준비 완료 신호 전달	-
7	SEND_RTS	수신 버퍼 상태 신호 전달	-

6.4.8. 특수 펑션 블록

No.	펑션 블록 이름	기 능	비 고
1	GET	특수 모듈 데이터 읽기	-
2	PUT	특수 모듈 데이터 쓰기	-
3	ARY_GET	특수 모듈 데이터 읽기(어레이)	-
4	ARY_PUT	특수 모듈 데이터 쓰기(어레이)	-
5	GETE	특수 모듈 데이터 읽기(상위 워드 Access 가능)	-
6	PUTE	특수 모듈 데이터 쓰기(상위 워드 Access 가능)	-
7	ARY_GETE	특수 모듈 데이터 읽기(어레이, 상위 워드 Access 가능)	-
8	ARY_PUTE	특수 모듈 데이터 쓰기(어레이, 상위 워드 Access 가능)	-

6.4.9. 모션 제어 펑션 블록

No.	펑션 블록 이름	기 능	비 고
1	GETM	모션 제어 모듈 데이터 읽기	-
2	PUTM	모션 제어 모듈 데이터 쓰기	-
3	ARY_GETM	모션 제어 모듈 데이터 읽기(어레이)	-
4	ARY_PUTM	모션 제어 모듈 데이터 쓰기(어레이)	-

6.4.10. 위치결정 펑션 블록 (APM)

No.	펑션 블록 이름	기 능	비 고
1	APM_ORG	원점 복귀 기동	-
2	APM_FLT	부동 원점 설정	-
3	APM_DST	직접 기동	-
4	APM_IST	간접 기동	-
5	APM_LIN	직선 보간 기동	-
6	APM_CIN	원호 보간 기동	-
7	APM_SST	동시 기동	-
8	APM_VTP	속도/위치 제어 전환	-
9	APM_PTV	위치/속도 제어 전환	-
10	APM_STP	감속 정지	-
11	APM_SKP	스킵 운전	-
12	APM_SSP	위치 동기	-
13	APM_SSS	속도 동기	-
14	APM_SSSP	위치지정 속도 동기	-
15	APM_POR	위치 오버라이드	-
16	APM_SOR	속도 오버라이드	-
17	APM_PSO	위치 지정 속도 오버라이드	-
18	APM_NMV	연속 운전	-
19	APM_INC	인칭 기동	-
20	APM_RTP	수동 운전 이전 위치로 복귀 기동	-
21	APM_SNS	기동 스텝 번호 변경	-
22	APM_SRS	반복 스텝 번호 변경	-
23	APM_MOF	M 코드 해제	-
24	APM_PRS	현재 위치 프리셋	-
25	APM_ZONE	Zone 출력 허용/금지	-
26	APM_EPRE	엔코더값 프리셋	-
27	APM_TEA	단독 티칭(ROM, RAM)	-
28	APM_ATEA	복수 티칭(ROM, RAM)	-
29	APM_SBP	기본 파라미터 설정	-
30	APM_SEP	확장 파라미터 설정	-
31	APM_SHP	원점 복귀 파라미터 설정	-
32	APM_SMP	수동 운전 파라미터 설정	-
33	APM_SIP	입력 신호 파라미터 설정	-
34	APM_SCP	공통 파라미터 설정	-

No.	펑션 블록 이름	기 능	비 고
35	APM_SMD	운전 데이터 설정	-
36	APM_EMG	비상 정지	-
37	APM_RST	에러 리셋 / 출력 금지 해제	-
38	APM_PST	포인트 운전	-
39	APM_WRT	파라미터/운전 데이터 저장	-
40	APM_CRD	운전 정보 읽기	-
41	APM_SRD	운전 정보 읽기	-
42	APM_ENCRD	엔코더값 읽기	-
43	APM_JOG	조그 기동	-
44	APM_MPG	수동 펄스 발생기(MPG) 운전	-
45	APM_RCP	현재 위치 구간 반복	
46	APM_VRD	가변 데이터 읽기	-
47	APM_VWR	가변 데이터 쓰기	-
48	APM_VTPP	위치지정 속도/위치 전환 제어	-

6.4.11. 위치결정 펑션 블록 (XPM)

No.	펑션 블록 이름	기 능	비 고
1	XPM_ORG	원점 복귀 기동	-
2	XPM_FLT	부동 원점 설정	-
3	XPM_DST	직접 기동	-
4	XPM_IST	간접 기동	-
5	XPM_SST	동시 기동	-
6	XPM_VTP	속도/위치 전환 제어	-
7	XPM_VTPP	위치지정 속도/위치 전환 제어	-
8	XPM_PTV	위치/속도 전환 제어	-
9	XPM_PTT	위치/토크 전환 제어	XGF-PN8A/B
10	XPM_STP	감속 정지	-
11	XPM_SKP	스킵 운전	-
12	XPM_SSP	위치 동기	-
13	XPM_SSS	속도 동기	-
14	XPM_SSSP	위치 지정 속도 동기	
15	XPM_POR	위치 오버라이드	-
16	XPM_SOR	속도 오버라이드	-
17	XPM_PSO	위치 지정 속도 오버라이드	-

No.	펑션 블록 이름	기 능	비 고
18	XPM_NMV	연속 운전	-
19	XPM_INC	인칭 운전	-
20	XPM_RTP	수동 운전 이전 위치로 복귀 기동	-
21	XPM_SNS	기동 스텝 번호 변경	-
22	XPM_SRS	반복 스텝 번호 변경	-
23	XPM_MOF	M 코드 해제	-
24	XPM_PRS	현재 위치 프리셋	-
25	APM_EPRE	엔코더 현재값 프리셋	-
26	APM_ATEA	복수 티칭	-
27	XPM_SBP	기본 파라미터 설정	-
28	XPM_SEP	확장 파라미터 설정	-
29	XPM_SHP	원점 복귀 파라미터 설정	XPM
30	XPM_SMP	수동 운전 파라미터 티칭	-
31	XPM_SIP	외부 신호 파라미터 티칭	XPM
32	XPM_SCP	공통 파라미터 티칭	-
33	XPM_SMD	운전 데이터 티칭	-
34	XPM_EMG	비상 정지	-
35	XPM_RST	에러 리셋	-
36	XPM_HRST	에러 히스토리 리셋	-
37	XPM_PST	포인트 운전	-
38	XPM_WRD	파라미터/운전 데이터 저장	-
39	XPM_CRD	운전 정보 읽기	-
40	XPM_SRD	운전 상태 읽기	-
41	XPM_ENCRD	엔코더값 읽기	-
42	XPM_SVERD	서보 에러 정보 읽기	XGF-PN8A/B
43	XPM_JOG	조그 기동	-
44	XPM_CAM	CAM 기동	-
45	XPM_CAMO	주축 옵셋 지정 CAM 기동	
45	XPM_ELIN	타원 보간 운동	-
47	XPM_VRD	가변 데이터 읽기	-
48	XPM_VWR	가변 데이터 쓰기	-
49	XPM_ECON	서보 통신 연결	XGF-PN8A/B
50	XPM_DCON	서보 통신 끊기	XGF-PN8A/B
51	XPM_SVON	서보 온	XGF-PN8A/B

No.	펑션 블록 이름	기 능	비 고
52	XPM_SVOFF	서보 오프	XGF-PN8A/B
53	XPM_SRST	서보 에러 리셋	XGF-PN8A/B
54	XPM_SHRST	서보 에러 히스토리 리셋	XGF-PN8A/B
55	XPM_RSTR	재기동	-
56	XPM_POE	설정 위치 출력 허용/금지	XPM
57	XPM_TRQ	토크 제어	XGF-PN8A/B
58	XPM_SVIRD	서보 외부 입력 정보 읽기	XGF-PN8B
59	XPM_SVPRD	서보드라이브 파라미터 읽기	XGF-PN8B
60	XPM_SVPWR	서보드라이브 파라미터 쓰기	XGF-PN8B
61	XPM_SVSAVE	서보드라이브 파라미터 저장	XGF-PN8B
62	XPM_PTT	토크 제어	XGF-PN8A/B
63	XPM_LRD	래치 위치 데이터 읽기	XGF-PN8A/B
64	XPM_LCLR	래치 리셋	XGF-PN8A/B
65	XPM_LSET	래치 설정	XGF-PN8B
66	XPM_STC	토크 동기	XGF-PN8A/B

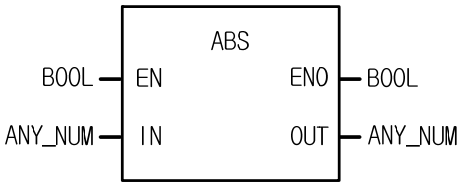
6.5. 확장 펑션

No.	펑션 이름	기 능	비 고
1	FOR	FOR ~ NEXT 구간을 n 번 실행	-
2	NEXT		-
3	BREAK	FOR ~ NEXT 구간을 빠져 나옴	-
4	CALL	SBRT 루틴 호출	-
5	SBRT	CALL 에 의해 호출될 루틴 지정	-
6	RET	RETURN	-
7	JMP	LABLE 위치로 점프	-
8	INIT_DONE	초기화 태스크 종료	-
9	END	프로그램의 종료	-

제7장 기본 펑션

1. 각각의 기본 펑션에 대한 설명입니다.
2. 기본 펑션을 사용하시기 전에 3.4.1 의 펑션에 대한 일반사항을 이해 하신 후 프로그램에 적용하시면, 프로그램 작성이 용이합니다.

ABS	적용 기종	발생플래그
절대값 연산	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 절대값 연산의 입력 값 **출력** ENO : 1을 출력 OUT : 절대값 IN, OUT은 같은 데이터 타입이어야 함.

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN						○	○	○	○	○	○	○	○	○	○					
	OUT						○	○	○	○	○	○	○	○	○	○					

■ 기능

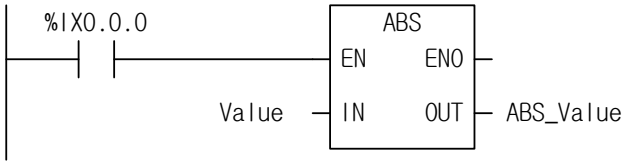
1. IN의 절대값을 OUT으로 출력시킵니다.
   OUT = |IN|
2. X의 절대값 |X| 는
   A. X≥0 이면 |X| = X 이고,
   B. X<0 이면 |X| = -X 입니다.

■ 플래그

플래그	설명
_ERR	IN의 값이 (-)최소값일 때 _ERR, _LER 플래그가 셋(Set)됩니다. 예) 데이터 타입이 SINT일 때 IN의 값이 -128이면 에러

■ 프로그램 예

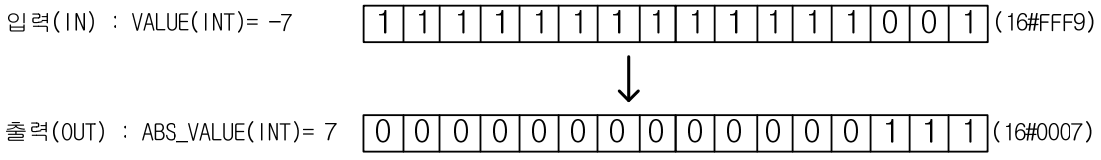
1. LD



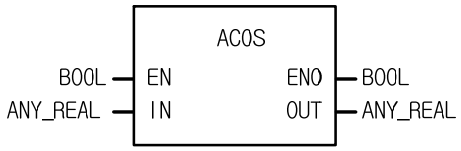
2. ST

```
ABS_Value := ABS(EN:=%IX0.0.0, IN:=Value);
```

(1) 실행조건(%IX0.0.0)이 On 하면 펑션 ABS 가 실행됩니다.
(2) VALUE = -7 이면, ABS_VALUE = |-7| = 7 이 됩니다.
VALUE = 200 이면, ABS_VALUE = |200| = 200 이 됩니다.
(3) INT 형의 음수 표시는 2 의 Complement 표현 (3.2.4. 데이터 타입별 구조 참조)



ACOS	적용 기종	발생플래그
Arc Cosine 연산	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : Arc Cosine 연산의 입력 값 **출력** ENO : EN 값이 그대로 출력 OUT : Arc Cosine 연산의 각도 결과 값(Radian) IN, OUT 은 같은 데이터 타입이어야 함

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

■ 기능

1. IN 의 Arc Cosine 값을 구해 OUT 으로 출력시킵니다. 출력 값은 0 에서 π 사이의 값이 됩니다.

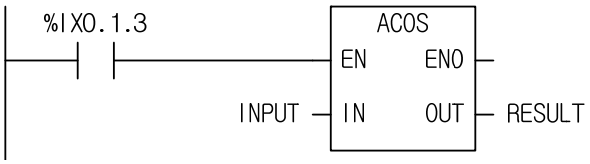
   OUT = ACOS (IN)

■ 플래그

플래그	설명
_ERR	입력 값이 범위가 -1.0 과 1.0 사이에 있지 않을 경우 _ERR, _LER 플래그가 셋(SET)됩니다.

■ 프로그램 예

1. LD



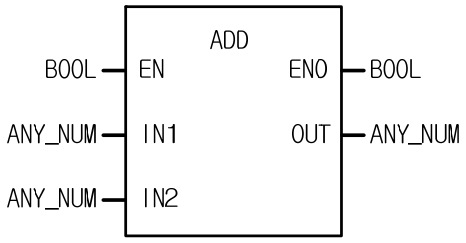
2. ST

```
RESULT := ACOS(EN:=%IX0.1.3, IN:=INPUT);
```

(1) 실행조건(%IX0.1.3)이 On 하면 펑션 Arc Cosine 연산 펑션 ACOS 가 실행됩니다.

(2) INPUT 으로 선언된 입력 변수의 값이 0.8660... ($\sqrt{3}/2$) 일 때 출력 변수로 선언된 RESULT 은 0.5235... ($\pi/6$ rad = 30°) 입니다.

ADD	적용 기종	발생플래그
더하기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 더할 값 IN2 : 더할 값 입력은 8개까지 확장 가능 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 더한 결과 값 IN1, IN2, ..., OUT에 연결되는 변수는 모두 같은 데이터 타입이어야 함.

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1						○	○	○	○	○	○	○	○	○	○					
	IN2						○	○	○	○	○	○	○	○	○	○					
	OUT						○	○	○	○	○	○	○	○	○	○					

■ 기능

1. IN1, IN2,..., INn (n은 입력 개수)를 더해서 OUT으로 출력시킵니다.

   OUT = IN1 + IN2 + ... + INn

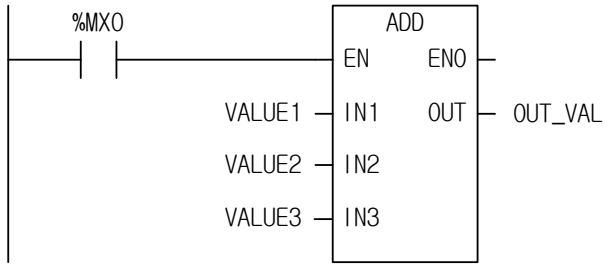
■ 플래그

플래그	설명
_ERR	출력 값이 해당 데이터 타입의 범위를 벗어날 경우 _ERR, _LER 플래그가 셋(Set)됩니다

☆ REAL, LREAL 타입 연산에서는 IN1 에서 IN8 로 순차적으로 연산을 하기 때문에 중간 연산과정에서 이미 REAL, LREAL 에 최대값이나 최소값을 넘게 되면 _ERR,_LER 플래그가 셋(Set)되고 결과값은 무한대 값 또는 비정상적인 값 (1.#INF000000000000e+000, 1.#SNAN00000000000e+000, 1.#QNAN00000000000e+000)으로 표시됩니다.

■ 프로그램 예

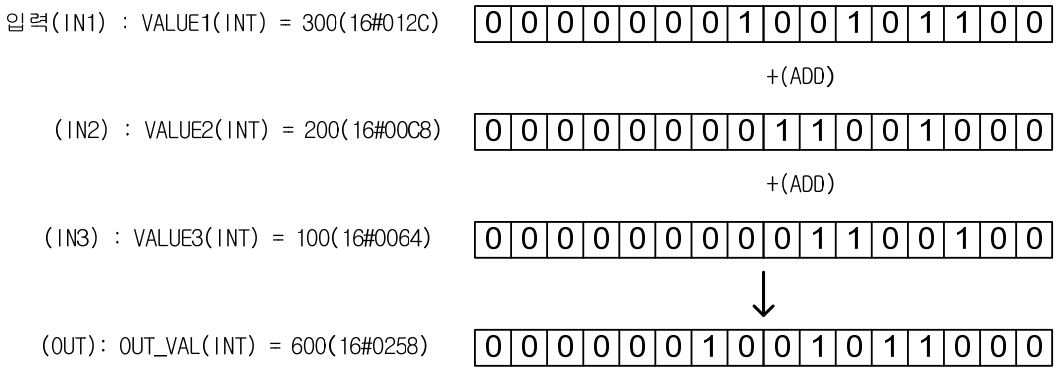
1. LD



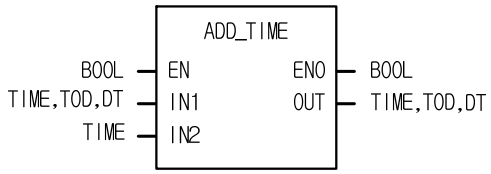
2. ST

```
OUT_VAL := ADD(EN:=%MX0, IN1:= VALUE1, IN2:= VALUE2, IN3:= VALUE3);
```

(1) 실행조건(%MX0)이 On 하면 더하기 펑션 ADD 가 실행됩니다.
(2) 입력변수 값이 VALUE1 = 300, VALUE2 = 200, VALUE3 = 100 이면, 출력변수로 설정한 OUT_VAL = 300 + 200 + 100 = 600 이 됩니다.



ADD_TIME	적용 기종	발생플래그
시간 더하기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 기준시각 또는 시간 IN2 : 더할 시간 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 더한 결과 시각 또는 시간 OUT 의 타입은 입력 IN1 의 타입을 따름. 즉 IN1 의 타입이 TIME_OF_DAY 이면, 출력 OUT 의 타입도 TIME_OF_DAY 임.

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1																○		○	○	
	OUT																○		○	○	

■ 기능

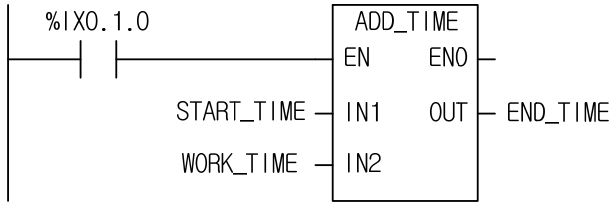
1. IN1 이 TIME 일 경우에는 시간과 시간을 더해서 합한 시간을 출력시킵니다.
2. IN1 이 TIME_OF_DAY 일 경우에는 기준시각에 시간을 더해서 하루 중의 시각을 출력시킵니다.
3. IN1 이 DATE_AND_TIME 일 경우에는 기준이 되는 날짜와 시각에 시간을 더해서 날짜와 시각을 출력시킵니다.

■ 플래그

플래그	설 명
_ERR	출력 값이 해당 데이터 타입의 범위를 벗어날 경우, _ERR, _LER 플래그가 셋(Set)됩니다. 시간과 시간을 더한 결과가 TIME 데이터 타입의 범위 T#49D17H2M47S295MS 를 넘거나 시각(TOD)과 시간을 더한 결과가 24 시를 넘을 경우, 또는 날짜와 시각(DT)과 시간을 더한 결과가 2163 년을 넘을 경우 에러가 됩니다.

■ 프로그램 예

1. LD



2. ST

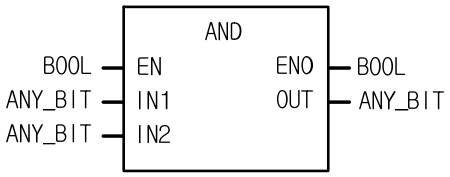
```
END_TIME := ADD_TIME(EN:= %IX0.1.0, IN1:= START_TIME, IN2:= WORK_TIME);
```

(1) 실행조건(%IX0.1.0)이 On 하면 시간 더하기 펑션 ADD_TIME 이 실행됩니다.
(2) 작업을 시작한 START_TIME 이 TOD#08:30:00 이고 작업한 시간 WORK_TIME 이 T#2H10M20S500MS 이면 작업이 종료된 시각 END_TIME 에는 TOD#10:40:20.5 가 출력됩니다.

```
입력 (IN1) : START_TIME(TOD)  = TOD#08:30:00
                                      + (ADD_TIME)
     (IN2) : WORK_TIME(TIME)  = T#2H10M20S500MS
                                      ↓
출력 (OUT) : END_TIME(TOD)    = TOD#10:40:20.5
```

AND	적용 기종	발생플래그
논리곱	XGI, XGR, XEC	-

펑 션



설 명

입력 N : 1일 때 펑션 실행
IN1 : AND될 값
IN2 : AND될 값
입력이 8개까지 확장 가능

출력 ENO : EN값이 그대로 출력
OUT : AND된 값

IN1, IN2, OUT은 모두 같은 타입이어야 함.

ANY 타입 변수설명

변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN1	○	○	○	○	○															
IN2	○	○	○	○	○															
OUT	○	○	○	○	○															

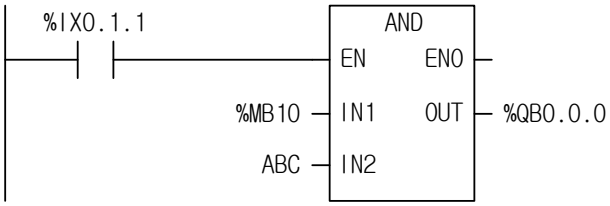
■ 기능

1. IN1을 IN2와 비트별로 AND해서 OUT으로 출력시킵니다.

IN1 1111 0000
&
IN2 1010 1010
OUT 1010 0000

■ 프로그램 예

1. LD



2. ST

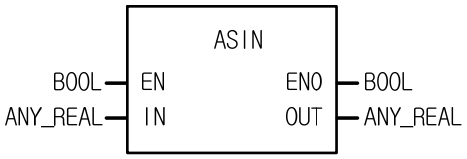
ST 언어는 AND 는 지원하지 않습니다.

AND2_BYTE 인 경우

```
%QB0.0.0 := AND2_BYTE(EN:=%IX0.1.1, IN1:= %MB10, IN2:= ABC);
```

(1) 실행조건(%IX0.1.1)이 On 하면 펑션 AND 가 실행됩니다.
(2) IN1= %MB10 과 IN2 = ABC 값을 AND 시킨 결과가 OUT = %QB0.0.0 에 출력됩니다.

ASIN	적용 기종	발생플래그
Arc Sine 연산	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : Arc Sine 연산의 입력 값 **출력** ENO : EN 값이 그대로 출력 OUT : Arc Sine 연산의 결과 값 (Radian) IN, OUT 은 같은 데이터 타입이어야 함

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

■ 기능

1. IN 의 Arc Sine 값을 구해 OUT 으로 출력시킵니다. 출력 값은 -π/2 에서 π/2 사이의 값이 됩니다.

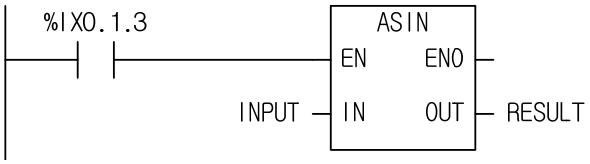
   OUT = ASIN (IN)

■ 에러

플래그	설명
_ERR	입력 값이 범위가 -1.0 과 1.0 사이에 있지 않을 경우 _ERR, _LER 플래그가 셋(SET)됩니다.

■ 프로그램 예

1. LD



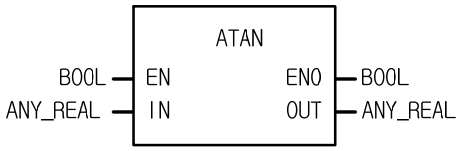
2. ST

```
RESULT := ASIN(EN:=%IX0.1.3, IN1:= INPUT);
```

(1) 실행조건(%IX0.1.3)이 On 하면 펑션 ASIN 이 실행됩니다.

(2) INPUT 으로 선언된 입력 변수의 값이 0.8660... ($\sqrt{3}/2$) 일 때 출력 변수로 선언된 RESULT 은 1.0471...
(π/3 rad = 60°) 입니다.

ATAN	적용 기종	발생플래그
Arc Tangent 연산	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 IN : Arc Tangent 연산의 입력 값 **출력** ENO : EN 값이 그대로 출력 OUT : Arc Tangent 연산의 결과 값(Radian) IN, OUT 은 같은 데이터 타입이어야 함

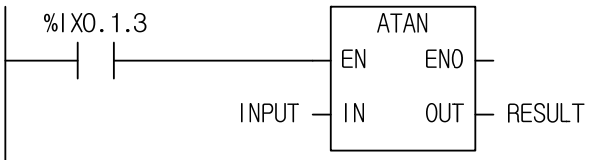
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

■ 기능

1. IN 의 Arc Tangent 값을 구해 OUT 으로 출력시킵니다. 출력 값은 -π/2 에서 π/2 사이의 값이 됩니다.
   OUT = ATAN (IN)

■ 프로그램 예

1. LD

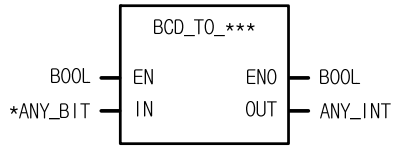


2. ST

```
RESULT := ATAN(EN:=%IX0.1.3, IN1:= INPUT);
```

(1) 실행조건(%IX0.1.3)이 On 하면 펑션 Arc Tangent 연산 펑션 ATAN 이 실행됩니다.
(2) INPUT 으로 선언된 입력 변수의 값이 1.0 일 때 출력 변수로 선언된 RESULT 은 0.7853...
(π/4 rad = 45°) 입니다.

BCD_TO_***	적용 기종	발생플래그
BCD 타입을 정수로 변환	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : BCD 형태의 데이터를 갖고 있는 ANY 타입입력 **출력** ENO : EN 값이 그대로 출력 OUT : 타입 변환된 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT						○	○	○	○	○	○	○	○							

*ANY_BIT : ANY_BIT 중 BOOL 타입제외

■ 기능

1. IN 을 타입 변환해서 OUT 으로 출력시킵니다.

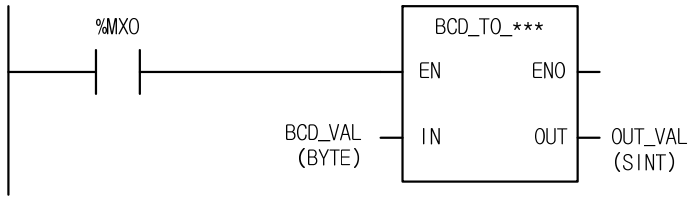
펑션	입력 타입	출력 타입	동작 설명
BYTE_BCD_TO_SINT	BYTE	SINT	BCD 를 출력 데이터 타입 타입으로 변환합니다. 입력이 BCD 값일 경우에만 정상 변환됩니다. (입력 데이터 타입이 WORD 일 경우 0~16#9999 값만 정상 변환됩니다.)
WORD_BCD_TO_INT	WORD	INT	
DWORD_BCD_TO_DINT	DWORD	DINT	
LWORD_BCD_TO_LINT	LWORD	LINT	
BYTE_BCD_TO_USINT	BYTE	USINT	
WORD_BCD_TO_UINT	WORD	UINT	
DWORD_BCD_TO_UDINT	DWORD	UDINT	
LWORD_BCD_TO_ULINT	LWORD	ULINT	

■ 플래그

플래그	설명
_ERR	IN 이 BCD 형태의 데이터가 아닌 경우, 출력은 0 이 되고 _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

1. LD



2. ST

ST 언어는 BCD_TO_***는 지원하지 않습니다.

BYTE_BCD_TO_SINT 인 경우

```
OUT_VAL := BYTE_BCD_TO_SINT(EN:=%MX0, IN:= BCD_VAL);
```

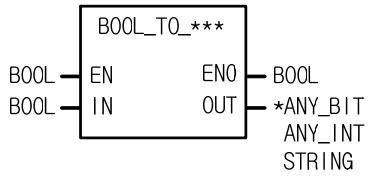
(1) 실행조건(%MX0)이 On 하면 펑션 BCD_TO_*** 이 실행됩니다.
(2) BCD_VAL(BYTE 타입) = 16#22(2#0010_0010)이면, 펑션의 출력 변수로 선언된 OUT_VAL(SINT 타입) = 22 (2#0001_0110)가 출력됩니다.

입력(IN1) :BCD_VAL(BYTE) = 16#22 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

↓ (BCD_TO_SINT)

출력(OUT): OUT_VAL(SINT) = 22 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |

BOOL_TO_***	적용 기종	발생플래그
BOOL 타입 변환	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 비트(1 비트) **출력** ENO : EN 값이 그대로 출력 OUT : 타입 변환된 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT		○	○	○	○	○	○	○	○	○	○	○	○							○

*ANY_BIT: ANY_BIT 타입 중 BOOL 제외

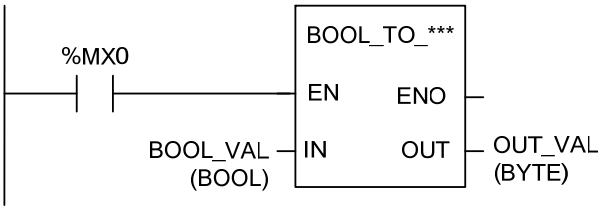
■ 기능

1. IN 을 타입 변환해서 OUT 으로 출력시킵니다.

펑 션	출력 타입	동작 설명
BOOL_TO_SINT	SINT	BOOL 입력의 값이 2#0 면 정수값 '0' 을, 2#1 이면 정수값 '1' 을 출력 데이터 타입에 맞추어 출력합니다.
BOOL_TO_INT	INT	
BOOL_TO_DINT	DINT	
BOOL_TO_LINT	LINT	
BOOL_TO_USINT	USINT	
BOOL_TO_UINT	UINT	
BOOL_TO_UDINT	UDINT	
BOOL_TO_ULINT	ULINT	
BOOL_TO_BYTE	BYTE	BOOL 을 상위 비트들을 0 으로 채운 출력 데이터 타입 타입으로 변환합니다.
BOOL_TO_WORD	WORD	
BOOL_TO_DWORD	DWORD	
BOOL_TO_LWORD	LWORD	
BOOL_TO_STRING	STRING	BOOL 을 STRING 타입으로 변환합니다. '0' 또는 '1' 로 변환합니다.

■ 프로그램 예

1. LD



2. ST

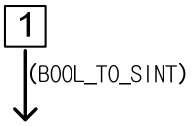
ST 언어는 BOOL_TO_***는 지원하지 않습니다.

BOOL_TO_BYTE 인 경우

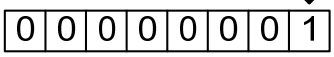
```
OUT_VAL := BOOL_TO_BYTE(EN:=%MX0, IN:= BOOL_VAL);
```

(1) 실행조건(%MX0)이 On 하면 펑션 BOOL_TO_***이 실행됩니다.
(2) 입력 변수로 선언된 BOOL_VAL(BOOL 타입) = 2#1 이면, 출력 변수로 선언된 OUT_VAL(BYTE 타입) = 2#0000_0001)이 출력 됩니다.

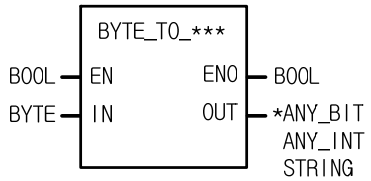
입력(IN) : BOOL_VAL(BOOL) = 2#1



출력(OUT): OUT_VAL(BYTE) = 16#1



BYTE_TO_***	적용 기종	발생플래그
BYTE 타입 변환	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 비트(1 비트) **출력** ENO : EN 값이 그대로 출력 OUT : 타입 변환된 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT		○	○	○	○	○	○	○	○	○	○	○	○							○

*ANY_BIT : ANY_BIT 타입중 BOOL 제외

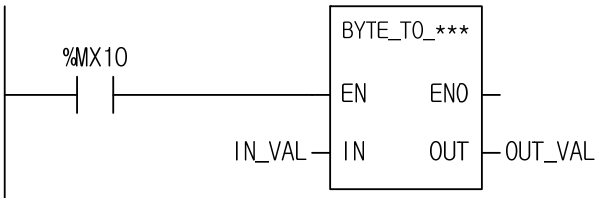
■ 기능

1. IN 을 타입 변환해서 OUT 으로 출력시킵니다.

펑션	출력 타입	동작 설명
BYTE _TO_SINT	SINT	내부 비트 배열의 변환 없이 SINT 타입으로 변환합니다.
BYTE _TO_INT	INT	상위비트를 0 으로 채워 INT 타입으로 변환합니다.
BYTE _TO_DINT	DINT	상위비트를 0 으로 채워 DINT 타입으로 변환합니다.
BYTE _TO_LINT	LINT	상위비트를 0 으로 채워 LINT 타입으로 변환합니다.
BYTE _TO_USINT	USINT	내부 비트 배열의 변환 없이 SINT 타입으로 변환합니다.
BYTE _TO_UINT	UINT	상위비트를 0 으로 채워 UINT 타입으로 변환합니다.
BYTE _TO_UDINT	UDINT	상위비트를 0 으로 채워 UDINT 타입으로 변환합니다.
BYTE _TO_ULINT	ULINT	상위비트를 0 으로 채워 ULINT 타입으로 변환합니다.
BYTE _TO_BOOL	BOOL	하위 1 비트를 취해 BOOL 타입으로 변환합니다.
BYTE _TO_WORD	WORD	상위비트를 0 으로 채워 WORD 타입으로 변환합니다.
BYTE _TO_DWORD	DWORD	상위비트를 0 으로 채워 DWORD 타입으로 변환합니다.
BYTE _TO_LWORD	LWORD	상위비트를 0 으로 채워 LWORD 타입으로 변환합니다.
BYTE _TO_STRING	STRING	입력 값을 STRING 타입으로 변환합니다.

■ 프로그램 예

1. LD



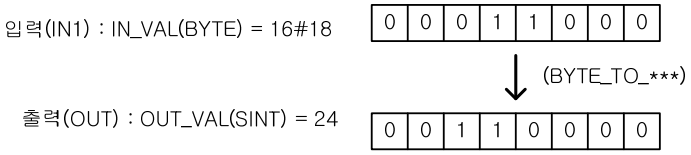
2. ST

ST 언어는 BYTE_TO_***는 지원하지 않습니다.

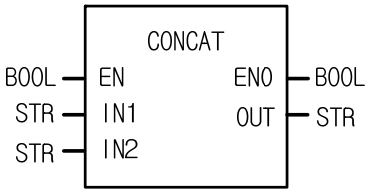
BYTE_TO_SINT 인 경우

```
OUT_VAL := BYTE_TO_SINT(EN:=%MX10, IN:= IN_VAL);
```

(1) 실행조건(%MX10)이 On 하면 펑션 BYTE_TO_***이 실행됩니다.
(2) IN_VAL(BYTE 타입) = 2#0001_1000 이면, OUT_VAL(SINT 타입) = 24(2#0011_0000)가 됩니다



CONCAT	적용 기종	발생플래그
문자열 연결하기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 입력 문자열 IN2 : 입력 문자열 입력 8개까지 확장 가능 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 출력 문자열

■ 기능

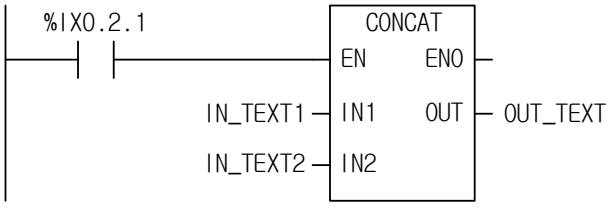
1. 입력 문자열 IN1, IN2, IN3,....., INn(n은 입력 개수)을 순서대로 붙여서 출력 문자열 OUT에 출력시킵니다.

■ 플래그

플래그	설명
_ERR	(각 입력 문자열의 문자수의 합) > 31 인 경우, OUT 값은 각 입력 문자열을 31 자까지 CONCAT한 값이 출력되고, _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

1. LD



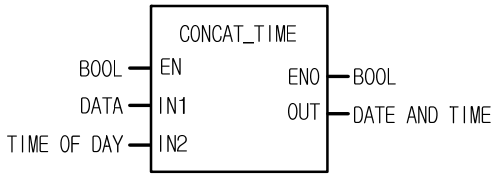
2. ST

```
OUT_TEXT := CONCAT(EN:=%IX0.2.1, IN1:= IN_TEXT1, IN2:= IN_TEXT2);
```

(1) 실행조건(%IX0.2.1)이 On 하면 펑션 CONCAT 이 실행됩니다.
(2) 펑션의 입력변수로 선언된 IN_TEXT1=`ABCD`, IN_TEXT2=`DEF`이면, 출력 변수로 선 OUT_TEXT=`ABCDDEF`가 됩니다.

```
입력(IN1) : IN_TEXT1(STRING) =    `ABCD`
                                (CONCAT)
    (IN2) : IN_TEXT2(STRING) =    `DEF`
                                    ↓
출력(OUT) : OUT_TEXT(STRING) =   'ABCDDEF'
```

CONCAT_TIME	적용 기종	발생플래그
날짜와 시각 연결하기	XGI, XGR, XEC	-

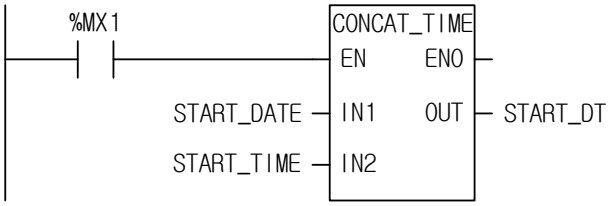
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 날짜 데이터 입력 IN2 : 시각 데이터 입력 **출력** ENO : EN 값이 그대로 출력 OUT : 날짜와 시각 출력

■ 기능

1. IN1(날짜)과 IN2(시각)를 붙여서 날짜와 시각(DATE_AND_TIME) OUT 으로 출력합니다.

■ 프로그램 예

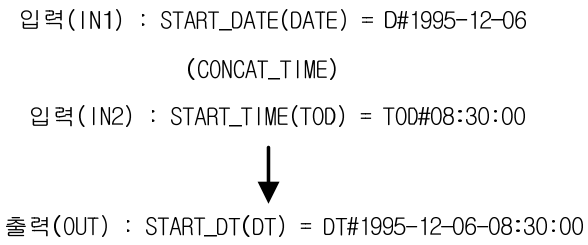
1. LD



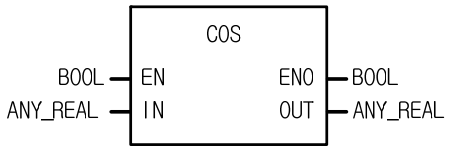
2. ST

```
START_DT := CONCAT_TIME(EN:=%MX1, IN1:= START_DATE, IN2:= START_TIME);
```

(1) 실행조건(%MX1)이 On 하면 펑션 CONCAT_TIME 이 실행됩니다.
(2) 운전시작 날짜 데이터 변수 START_DATE = D#1995-12-06 이고 운전시작 시각 START_TIME = TOD#08:30:00 이면 날짜와 시각이 합쳐진 START_DT 에는 DT#1995-12-06-08:30:00 이 출력됩니다.



COS	적용 기종	발생플래그
Cosin 연산	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : Cosine 연산의 각도 입력값(Radian) **출력** ENO : EN 값이 그대로 출력 OUT : Cosine 연산결과 값 IN, OUT 은 같은 데이터 타입이어야 함

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

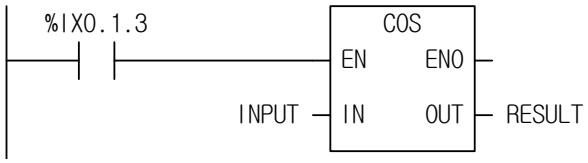
■ 기능

1. IN 의 Cosine 값을 구해 OUT 으로 출력시킵니다.

   OUT = COS (IN)

■ 프로그램 예

1. LD



2. ST

```
RESULT := COS(EN:=%IX0.1.3, IN:= INPUT);
```

(1) 실행조건(%IX0.1.3)이 On 하면 펑션 COS 이 실행됩니다.
(2) INPUT 으로 선언된 입력 변수의 값이 0.5235 (π/6 rad = 30°)일때 출력 변수로 선언된 RESULT 은 0.8660 .... ($\sqrt{3}$/2) 입니다.

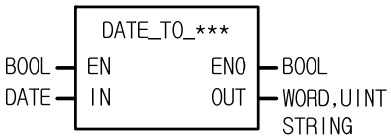
COS (π/6) = $\sqrt{3}$/2 = 0.866

입력(IN) : INPUT(REAL) = 0.5235

↓ (COS)

출력(OUT) : RESULT(REAL) = 8.66074800E-01

DATE_TO_***	적용 기종	발생플래그
DATE 타입변환	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 날짜 데이터 **출력** ENO : EN 값이 그대로 출력 OUT : 타입 변환된 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN			○								○									○

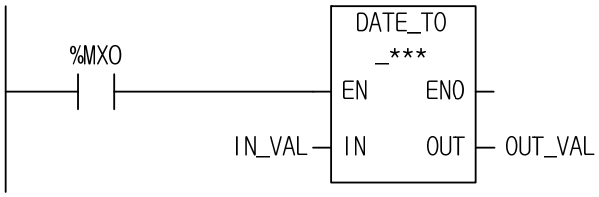
■ 기능

1. IN 을 타입 변환해서 OUT 으로 출력시킵니다.

펑션	출력 타입	동작 설명
DATE_TO_UINT	UINT	DATE 를 UINT 타입으로 변환합니다.
DATE_TO_WORD	WORD	DATE 를 WORD 타입으로 변환합니다.
DATE_TO_STRING	STRING	DATE 를 STRING 타입으로 변환합니다.

■ 프로그램 예

1. LD



2. ST

ST 언어는 DATE_TO_***는 지원하지 않습니다.

DATE_TO_STRING 인 경우

```
OUT_VAL := DATE_TO_STRING(EN:=%MX0, IN:= IN_VAL);
```

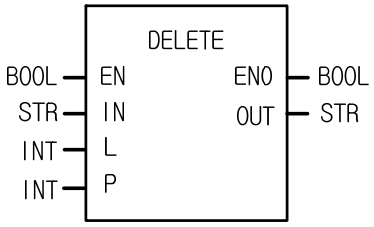
(1) 실행조건(%MX0)이 On 하면 데이터 타입 변환 펑션 DATE_TO_***이 실행됩니다.
(2) 입력 변수로 선언된 IN_VAL(DATE 타입)의 값이 D#1995-12-01 이면, 출력 변수로 선언된 OUT_VAL (STRING 타입)는 'D#1995-12-01' 이 됩니다.

입력(IN) : IN_VAL(DATE) = D#1995-12-01

↓ (DATE_TO_STRING)

출력(OUT) : OUT_VAL(STRING) = 'D#1995-12-01'

DELETE	적용 기종	발생플래그
문자열을 삭제하기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 입력 문자열 L : 삭제할 문자열 길이 P : 문자열의 삭제 위치 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 출력 문자열

■ 기능

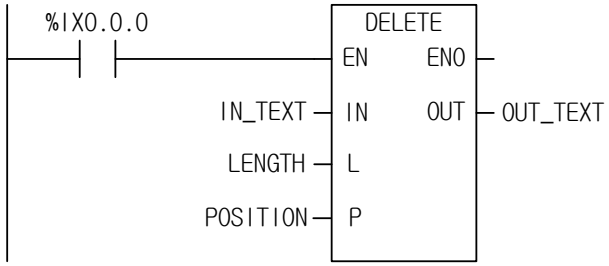
1. 문자열 IN의 P번째 문자부터 길이 L숫자만큼 삭제한 후, 문자열 OUT에 출력시킵니다.

■ 플래그

플래그	설명
_ERR	P≤ 0 또는 L< 0 인 경우 또는 P > (IN의 입력 문자열의 문자 수)인 경우, _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

1. LD



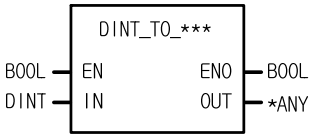
2. ST

```
OUT_TEXT := DELETE(EN:= %IX0.0.0, IN:= IN_TEXT, L:= LENGTH, P:= POSITION);
```

(1) 실행조건(%IX0.0.0)이 On 하면 문자열 삭제 DELETE 가 실행됩니다.
(2) 입력 변수의 값이 IN_TEXT(입력한 문자) = `ABCDEF`, LENGTH(삭제할 문자열 길이) = 3, POSITION(문자열의 삭제 위치) = 3 이면, 출력 변수로 선언된 OUT_TEXT(STRING 타입)는 `ABF`가 됩니다.

```
입력(IN) : IN_TEXT(STRING) =  `ABCDEF`

(L)  : LENGTH(INT)         =    3

(P)  : POSITION(INT)       =    3
                                 ↓ (DELETE)
출력(OUT): OUT_TEXT(STRING)  =  `ABF`
```

DINT_TO_***	적용 기종	발생플래그
DINT 타입 변환	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 Double Integer 값 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 타입 변환된 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT	○	○	○	○	○	○	○		○	○	○	○	○	○	○			○	○	○

*ANY: ANY 타입 중 DINT, TIME, DATE 제외

■ 기능

1. IN을 타입 변환해서 OUT으로 출력시킵니다.

펑션	출력타입	동작 설명
DINT_TO_SINT	SINT	입력이 -128 ~127 일 경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
DINT_TO_INT	INT	입력이 -32,768~32,767 일 경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
DINT_TO_LINT	LINT	LINT 타입으로 정상 변환합니다.
DINT_TO_USINT	USINT	입력이 0~255 일 경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
DINT_TO_UINT	UINT	입력이 0~65,535 일 경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
DINT_TO_UDINT	UDINT	입력이 0~2,147,483,647 일 경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
DINT_TO_ULINT	ULINT	입력이 0~2,147,483,647 일 경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
DINT_TO_BOOL	BOOL	하위 1 비트를 취해 BOOL 타입으로 변환합니다.
DINT_TO_BYTE	BYTE	하위 8 비트를 취해 BYTE 타입으로 변환합니다.
DINT_TO_WORD	WORD	하위 16 비트를 취해 WORD 타입으로 변환합니다.
DINT_TO_DWORD	DWORD	내부 비트 배열의 변화 없이 DWORD 타입으로 변환합니다.
DINT_TO_LWORD	LWORD	상위 비트를 0으로 채운 LWORD 타입으로 변환합니다.

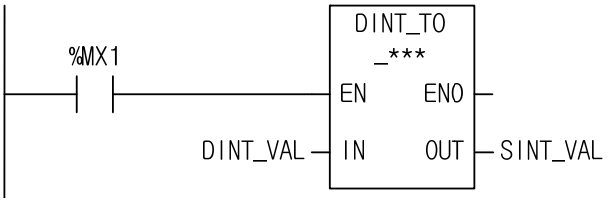
펑션	출력타입	동작 설명
DINT_TO_REAL	REAL	DINT 를 REAL 타입으로 변환합니다. 변환 중 정밀도에 따른 오차가 발생할 수 있습니다.
DINT_TO_LREAL	LREAL	DINT 를 LREAL 타입으로 변환합니다. 변환 중 정밀도에 따른 오차가 발생할 수 있습니다.
DINT_TO_STRING	STRING	입력 값을 STRING 타입으로 변환합니다.

■ 플래그

플래그	설명
_ERR	변환 에러 발생시 _ERR, _LER 플래그가 셋(Set)됩니다. 에러 발생시 출력 타입의 비트 수 만큼 하위 비트를 취해 내부 비트 배열의 변환 없이 출력 시킵니다

■ 프로그램 예

1. LD



2. ST

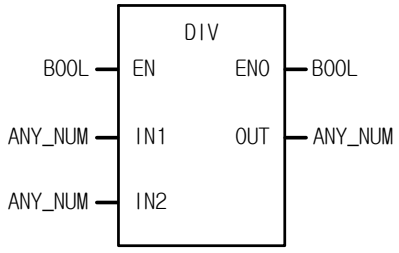
ST 언어는 DINT_TO_***는 지원하지 않습니다.

DINT_TO_SINT 인 경우

```
SINT_VAL := DINT_TO_SINT(EN:= %MX1, IN:= DINT_VAL);
```

(1) 실행조건(%MX1)이 On 하면 데이터 타입 변환 펑션 DINT_TO_***가 실행됩니다.
(2) INI = DINT_VAL(DINT 타입) = -77 이면, SINT_VAL(SINT 타입) = -77 이 됩니다.

DIV	적용 기종	발생플래그
나누기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 나누어질 값(피제수) IN2 : 나눌 값(제수) **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 나눈 결과 값(몫) IN1, IN2, OUT에 연결되는 변수는 모두 같은 데이터 타입이어야 함.

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1						○	○	○	○	○	○	○	○	○	○					
	IN2						○	○	○	○	○	○	○	○	○	○					
	OUT						○	○	○	○	○	○	○	○	○	○					

■ 기능

1. IN1을 IN2로 나눠서 그 몫 중에서 소수점 이하를 버린 값을 OUT으로 출력시킵니다.

   OUT = IN1/IN2

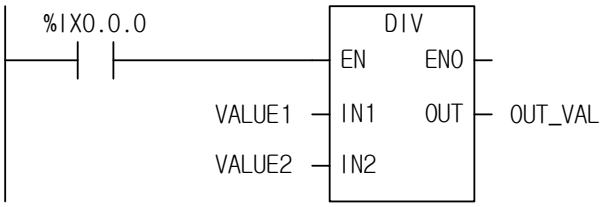
IN1	IN2	OUT	비 고
7	2	3	소수점 이하 버림
7	-2	-3	
-7	2	-3	
-7	-2	3	
7	0	×	에러

■ 플래그

플래그	설명
_ERR	나눌 값(제수)이 '0' 인 경우, 나눈 결과값이 각 타입의 최대값을 넘을 경우 _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

1. LD

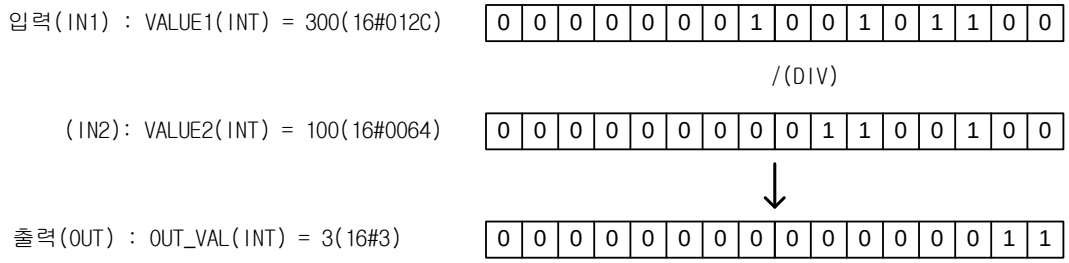


2. ST

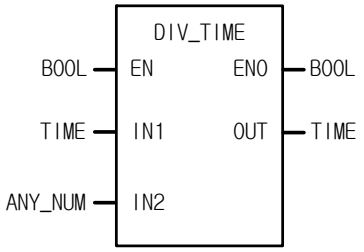
```
OUT_VAL := DIV(EN:= %IX0.0.0, IN1:= VALUE1, IN2:= VALUE2);
```

(1) 실행조건(%IX0.0.0)이 On 하면 나누기 펑션 DIV 가 실행됩니다.

(2) 입력 변수 VALUE1 = 300, VALUE2 = 100 이면, 출력 변수로 선언된 OUT_VAL = 300/100 = 3 이 출력됩니다.



DIV_TIME	적용 기종	발생플래그
시간 나누기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 나눌 시간 IN2 : 나눌 값 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 나눈 결과 시간

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN2						○	○	○	○	○	○	○	○	○	○					

■ 기능

1. IN1(시간)을 IN2(숫자)로 나누어서 나누어진 시간을 OUT 으로 출력시킵니다.

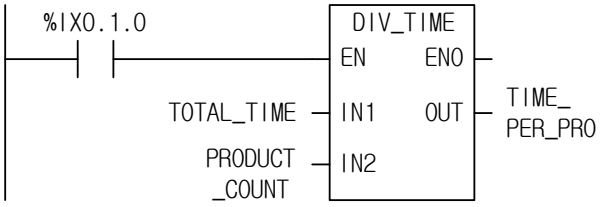
■ 플래그

플래그	설명
_ERR	제수(IN2)가 0 인 경우, 또는 0 보다 작은 경우 _ERR, _LER 플래그가 셋(Set)됩니다. IN2 에 음수 값 입력 시 _ERR, _LER 플래그가 ON 되고 결과에 0 을 출력합니다.

■ 프로그램 예

어느 생산 Line 의 하루 작업 시간이 12 시간 24 분 24 초이고, 하루 생산량이 12 개일 때 제품 한 개를 생산하는데 걸리는 시간을 계산하는 프로그램입니다.

1. LD



2. ST

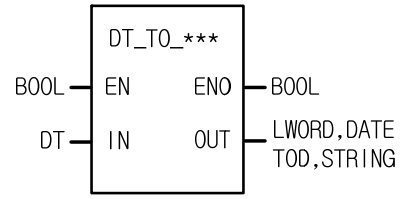
```
TIME_PER_PRO := DIV_TIME(EN:= %IX0.1.0, IN1:= TOTAL_TIME, IN2:= PRODUCT_COUNT);
```

(1) 실행 조건(%IX0.1.0)이 On 하면 시간 나누기 펑션 DIV_TIME 이 실행됩니다.

(2) 하루 작업 시간 TOTAL_TIME(T#12H24M24S)을 하루 제품 생산량 PRODUCT_COUNT(12)로 나누면, 제품 한 개당 걸리는 시간 TIME_PER_PRO(T#1H2M2S)이 출력됩니다. 즉, 1 시간 2 분 2 초에 한 개씩 생산됩니다.

```
입력(IN1) : TOTAL_TIME(TIME)   =   T#12H24M24S
                      /(DIV_TIME)
   (IN2) : PRODUCT_COUNT(INT) =        12
                      ↓
출력(OUT) : TIME_PER_PRO(TIME) =    T#1H2M2S
```

DT_TO_***	적용 기종	발생플래그
DT 타입변환	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 날짜와 시각 데이터 **출력** ENO : EN 값이 그대로 출력 OUT : 타입 변환된 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT					○												○	○		○

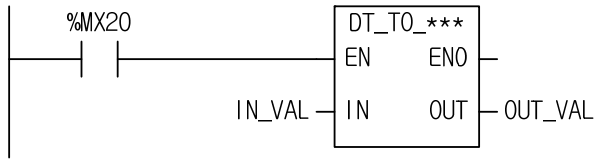
■ 기능

1. IN 을 타입 변환해서 OUT 으로 출력시킵니다.

펑션	출력타입	동작 설명
DT_TO_LWORD	LWORD	DT 를 LWORD 타입으로 변환합니다. (내부 데이터의 변환가 없어 역 변환이 가능합니다.)
DT_TO_DATE	DATE	DT 를 DATE 타입으로 변환합니다.
DT_TO_TOD	TOD	DT 를 TOD 타입으로 변환합니다.
DT_TO_STRING	STRING	DT 를 STRING 타입으로 변환합니다.

■ 프로그램 예

1. LD



2. ST

ST 언어는 DT_TO_***는 지원하지 않습니다.

DT_TO_DATE 인 경우

```
OUT_VAL := DT_TO_DATE(EN:= %MX20, IN1:= IN_VAL);
```

(1) 실행조건(%MX20)이 On 하면 DT 타입 변환 펑션 DT_TO_***가 실행됩니다.

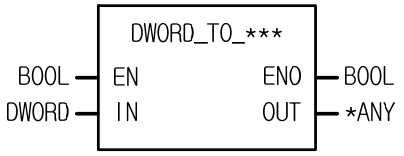
(2) 입력 변수로 선언된 IN_VAL(DT 타입) = DT#1995-12-01-12:00:00 이면, 출력 변수로 선언된 OUT_VAL (DATE 타입) = D#1995-12-01 이 됩니다.

입력(IN) : IN_VAL(DT) = DT#1995-12-01-12:00:00

↓ (DT_TO_DATE)

출력(OUT) : OUT_VAL(DATE) = D#1995-12-01

DWORD_TO_***	적용 기종	발생플래그
DWORD 타입변환	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 IN : 타입 변환할 비트열(32 비트) **출력** ENO : EN 값이 그대로 출력 OUT : 타입 변환된 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT	○	○	○		○	○	○	○	○	○	○	○	○	○		○		○	○	○

*ANY: ANY 타입 중 DWORD, LREAL, DATE 제외

■ 기능

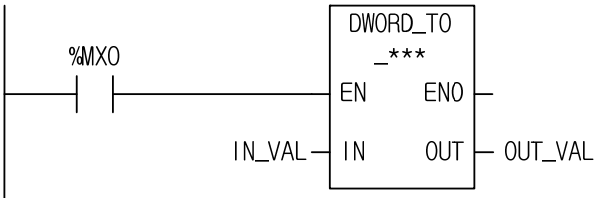
1. IN 을 타입 변환해서 OUT 으로 출력시킵니다.

펑션	출력타입	동작 설명
DWORD_TO_SINT	SINT	하위 8 비트를 취해 SINT 타입으로 변환합니다.
DWORD_TO_INT	INT	하위 16 비트를 취해 INT 타입으로 변환합니다.
DWORD_TO_DINT	DINT	내부 비트 배열의 변환 없이 DINT 타입으로 변환합니다.
DWORD_TO_LINT	LINT	상위 비트를 0 으로 채운 LINT 타입으로 변환합니다.
DWORD_TO_USINT	USINT	하위 8 비트를 취해 USINT 타입으로 변환합니다.
DWORD_TO_UINT	UINT	하위 16 비트를 취해 UINT 타입으로 변환합니다.
DWORD_TO_UDINT	UDINT	내부 비트 배열의 변환 없이 UDINT 타입으로 변환합니다.
DWORD_TO_ULINT	ULINT	상위 비트를 0 으로 채운 ULINT 타입으로 변환합니다.
DWORD_TO_BOOL	BOOL	하위 1 비트를 취해 BOOL 타입으로 변환합니다.
DWORD_TO_BYTE	BYTE	하위 8 비트를 취해 BYTE 타입으로 변환합니다.
DWORD_TO_WORD	WORD	하위 16 비트를 취해 WORD 타입으로 변환합니다.
DWORD_TO_LWORD	LWORD	상위 비트를 0 으로 채운 LWORD 타입으로 변환합니다.
DWORD_TO_REAL	REAL	내부 비트 배열의 변화 없이 REAL 타입으로 변환합니다.
DWORD_TO_TIME	TIME	내부 비트 배열의 변화 없이 TIME 타입으로 변환합니다.

펑션	출력타입	동작 설명
DWORD_TO_TOD	TOD	내부 비트 배열의 변화 없이 TOD 타입으로 변환합니다. 단, TOD 범위(TOD#23:59:59.999)를 벗어난 값 입력시 _ERR, _LER 플래그를 셋(SET)하고 TOD 범위 내에서 순환하여 변환합니다.
DWORD_TO_STRING	STRING	입력 값을 Decimal 로 바꾸어 STRING 타입으로 변환합니다.

■ 프로그램 예

1. LD



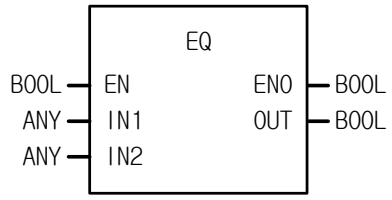
2. ST

ST 언어는 DWORD_TO_***는 지원하지 않습니다.
DWORD_TO_TOD 인 경우

```
OUT_VAL := DWORD_TO_***(EN:= %MX0, IN1:= IN_VAL);
```

(1) 실행조건(%MX0)이 On 하면 타입 변환 펑션 DWORD_TO_TOD 가 실행됩니다.
(2) IN_VAL(DWORD 타입) = 16#3E8(1000)이면, OUT_VAL(TOD 타입) = TOD#1S 가 됩니다.
(3) TIME, TOD 는 10 진 값을 MS 단위로 환산해 계산합니다. 즉, 1000 은 1000MS = 1S 로 계산합니다.
(3.2.4. 데이터 타입별 구조 참조)

EQ	적용 기종	발생플래그
'같다' 비교	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 비교할 값 IN2 : 비교할 값 입력은 8개까지 확장 가능 IN1, IN2, ...는 모두 같은 타입이어야 함. **출력** ENO : EN값이 그대로 출력 OUT : 비교 결과값

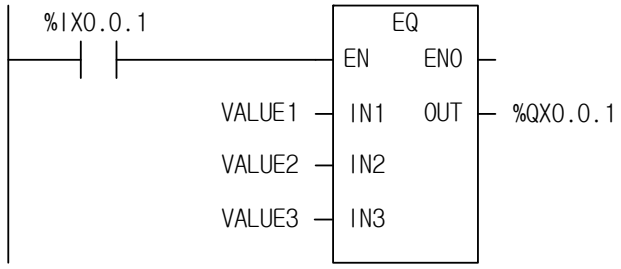
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	IN2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ 기능

1. IN1=IN2=IN3...=INn(n 은 입력개수)이면 OUT 으로 1 이 출력됩니다.
2. 다른 경우에는 OUT 으로 0 이 출력됩니다.

■ 프로그램 예

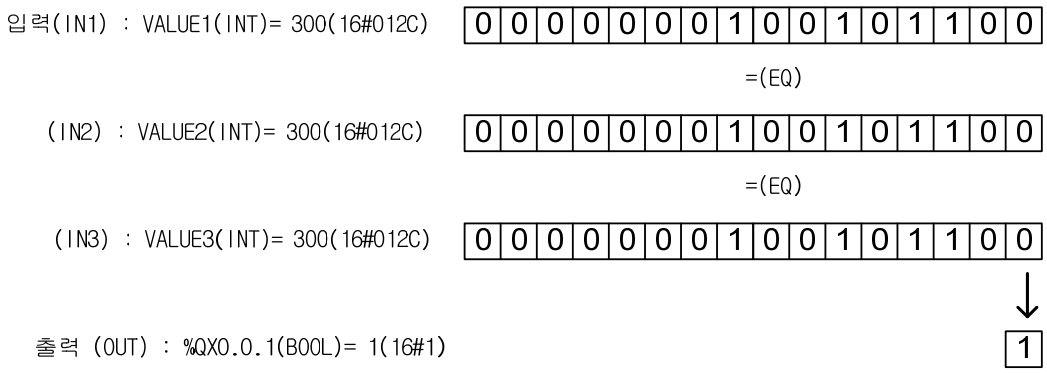
1. LD



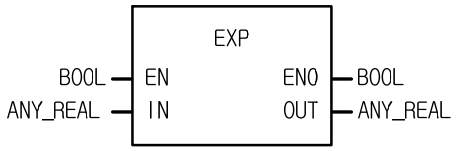
2. ST

```
%QX0.0.1 := EQ(EN:= %IX0.0.1, IN1:= VALUE1, IN2:= VALUE2, IN3:= VALUE3);
```

(1) 실행조건(%IX0.0.1)이 On 하면 비교 펑션 'EQ'가 실행됩니다.
(2) VALUE1 = 300, VALUE2 = 300, VALUE3 = 300 이면, 비교결과 VALUE1 = VALUE2 = VALUE3 이므로 출력 값 %QX0.0.1 = 1 이 됩니다.



EXP	적용 기종	발생플래그
EXP 연산	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 지수연산의 입력 값 **출력** ENO : EN값이 그대로 출력 OUT : 지수연산 결과 값 IN, OUT은 같은 데이터 타입이어야 함

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

■ 기능

1. IN의 지수연산 값을 구해 OUT으로 출력시킵니다.

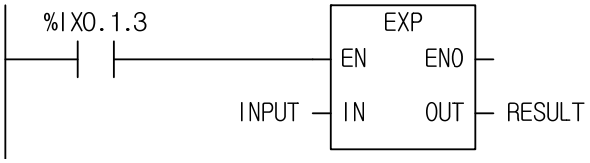
$$OUT = e^{IN}$$

■ 에러

플래그	설명
_ERR	출력 값이 해당 타입의 범위를 벗어 날 경우 _ERR, _LER 플래그가 셋(SET)됩니다.

■ 프로그램 예

1. LD



2. ST

```
RESULT := EXP(EN:= %IX0.1.3, IN1:= INPUT);
```

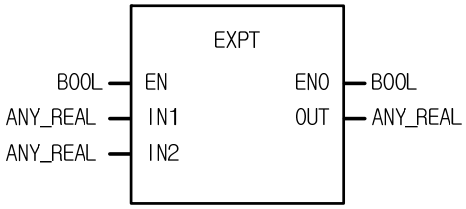
(1) 실행조건(%IX0.1.3)이 On 하면 펑션 EXP 가 실행됩니다.

(2) INPUT 으로 선언된 입력 변수의 값이 2.0 일 때 출력 변수로 선언된 RESULT 은 7.3890... 입니다.

$$RESULT = e^{INPUT}$$

INPUT = 2.0, RESULT = 7.3890...

EXPT	적용 기종	발생플래그
지수 연산	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 진수 IN2 : 지수 **출력** ENO : EN 값이 그대로 출력 OUT : 연산결과 값 IN1, OUT 은 같은 데이터 타입이어야 함

ANY 타입 변수설명

변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
IN1														○	○					
IN2														○	○					
OUT														○	○					

■ 기능

1. IN1 에 IN2 를 지수승하여 OUT 으로 출력시킵니다.

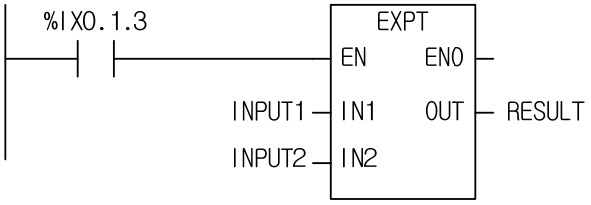
$$OUT = IN1^{IN2}$$

■ 에러

플래그	설명
_ERR	출력 값이 해당 타입의 범위를 벗어 날 경우 _ERR, _LER 플래그가 셋(SET)됩니다.

■ 프로그램 예

1. LD



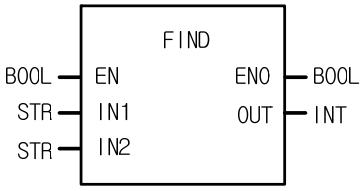
2. ST

```
RESULT := EXPT(EN:= %IX0.1.2, IN1:= INPUT1, IN2:= INPUT2);
```

(1) 실행조건(%IX0.1.3)이 On 하면 펑션 EXPT 가 실행됩니다.

(2) INPUT 으로 선언된 입력 변수의 값 INPUT1 = 1.5, INPUT2 = 3 이면 출력 변수로 선언된 RESULT 는 3.375 가 됩니다.

$$3.375 = 1.5^{3}$$

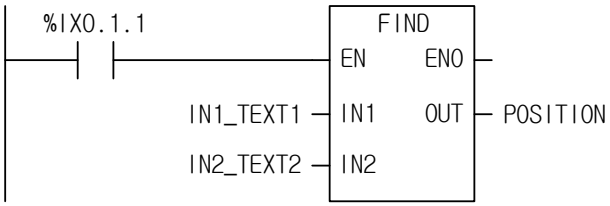
FIND	적용 기종	발생플래그
문자열 찾기	XGI, XGR, XEC	-
펑 션	설 명	
	**입력** EN : 실행 허용 IN1 : 입력 문자열 IN2 : 찾을 문자열 **출력** ENO : EN 값이 그대로 출력 OUT : 찾는 문자열의 위치	

■ 기능

1. 입력 문자열 IN1 에서 문자열 IN2 의 위치를 찾습니다. 찾으면 문자열 IN1 에서 문자열 IN2 가 있는 첫번째 문자위치를 OUT 에 출력시키고, 없으면 0 을 OUT 에 출력시킵니다.

■ 프로그램 예

1. LD



2. ST

```
POSITION := FIND(EN:= %IX0.1.2, IN1:= IN1_TEXT1, IN2:= IN2_TEXT2);
```

(1) 실행 조건(%IX0.1.1)이 On 하면 FIND(문자열 찾기) 펑션이 실행됩니다.

(2) 입력 변수로 선언된 입력문자열이 IN_TEXT1 = `ABCEF`, 찾을 문자열이 IN_TEXT2 = `BC`이면, 출력 변수로 선언된 POSITION = 2 가 선언됩니다.

(3) 입력문자열 IN_TEXT1 = `ABCEF`에서 찾을 문자열 IN_TEXT2 = `BC`의 위치는 2 번째 입니다.

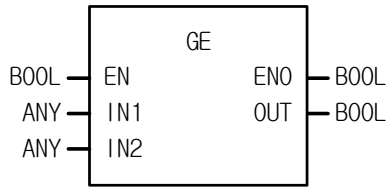
입력(IN1) : IN_TEXT1(STRING) = 'ABCEF'

(IN2) : IN_TEXT2(STRING) = 'BC'

(FIND)

출력(OUT) : POSITION(INT) = 2

GE	적용 기종	발생플래그
'크거나 같다' 비교	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 IN1 : 비교할 값 IN2 : 비교할 값 입력은 8 개까지 확장 가능 IN1, IN2,...는 모두 같은 타입이어야 함. **출력** ENO : EN 값이 그대로 출력 OUT : 비교 결과값

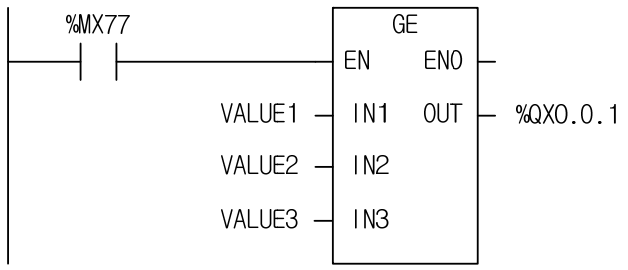
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	IN2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ 기능

1. IN1≥IN2≥IN3...≥INn(n 은 입력개수)이면 OUT 으로 1 이 출력됩니다.
2. 다른 경우에는 OUT 으로 0 이 출력됩니다.

■ 프로그램 예

1. LD

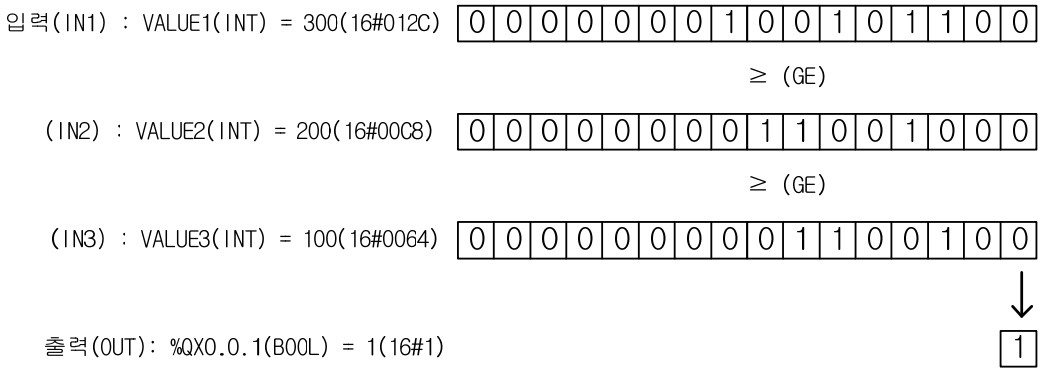


2. ST

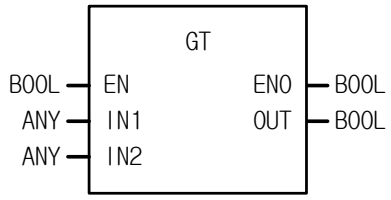
```
%QX0.0.1 := GE(EN:= %MX77, IN1:= VALUE1, IN2:= VALUE2, IN3:= VALUE3);
```

(1) 실행 조건(%MX77)이 On 하면 GE(비교:크거나 같다) 펑션이 실행됩니다.

(2) 입력변수로 선언된 VALUE1=300, VALUE2=200, VALUE3=100 이면, 비교 결과 VALUE1 ≥ VALUE2 ≥ VALUE3 이므로, 출력 결과 값 %QX0.0.1=1 이 됩니다.



GT	적용 기종	발생플래그
'크다' 비교	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 비교될 값 IN2 : 비교할 값 입력은 8개까지 확장 가능 IN1, IN2, ...는 모두 같은 타입이어야 함. **출력** ENO : EN값이 그대로 출력 OUT : 비교 결과 값

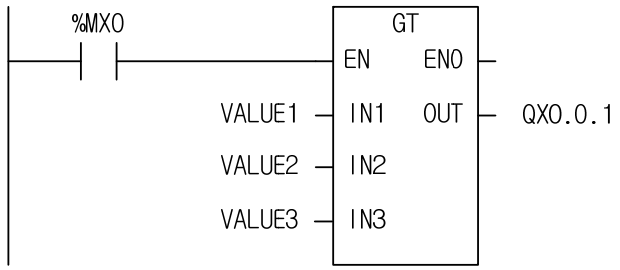
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	IN2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ 기능

1. IN1>IN2>IN3...>INn(n 은 입력 개수)이면 OUT 으로 1 이 출력됩니다.
2. 다른 경우에는 OUT 으로 0 이 출력됩니다.

■ 프로그램 예

1. LD



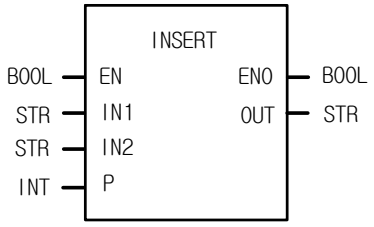
2. ST

```
%QX0.0.1 := GT(EN:= %MX0, IN1:= VALUE1, IN2:= VALUE2, IN3:= VALUE3);
```

(1) 실행조건(%MX0)이 On 하면 GT(비교: 크다) 펑션이 실행됩니다.

(2) 입력변수로 선언된 VALUE1 = 300, VALUE2 = 200, VALUE3 = 100 이면, 비교결과 VALUE1 > VALUE2 > VALUE3 이므로 출력결과값 %QX0.0.1 = 1 이 됩니다.

INSERT	적용 기종	발생플래그
문자열 삽입하기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 삽입될 문자열 IN2 : 삽입할 문자열 P : 문자열을 삽입할 위치 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 출력 문자열

■ 기능

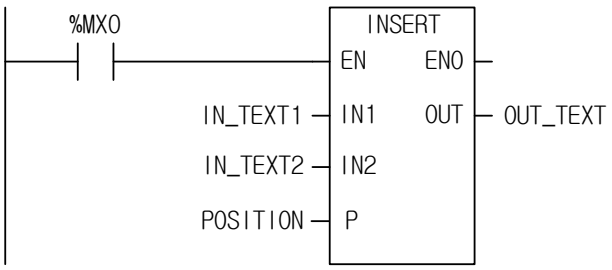
1. 문자열 IN1의 P번째 문자 뒤에 문자열 IN2를 삽입한 후 문자열 OUT에 출력시킵니다.

■ 플래그

플래그	설명
_ERR	P ≤ 0 이거나 (변수 IN1 의 문자 수) < P 인 경우, 또는 연산결과 문자수가 31 자를 넘어서 OUT으로 32까지 출력된 경우, _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

1. LD

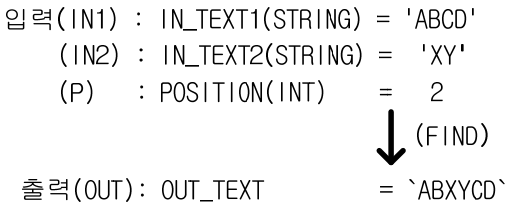


2. ST

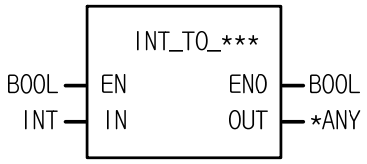
```
OUT_TEXT := INSERT(EN:= %MX0, IN1:= IN_TEXT1, IN2:= IN_TEXT2, P:= POSITION);
```

(1) 실행조건(%MX0)이 On 하면, INSERT(문자열 삽입하기) 펑션이 실행됩니다.

(2) 입력변수로 선언된 변수 IN_TEXT1 = `ABCD`, IN_TEXT2 = `XY`, POSITON = 2 이면, 출력변수 OUT_TEXT = `ABXYCD`가 됩니다.



INT_TO_***	적용 기종	발생플래그
INT 타입변환	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 Integer 값 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 타입 변환된 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT	○	○	○	○	○	○		○	○	○	○	○	○	○	○					○

*ANY: ANY 타입 중 INT, TIME, DATE, TOD, DT 제외

■ 기능

1. IN을 타입 변환해서 OUT으로 출력시킵니다.

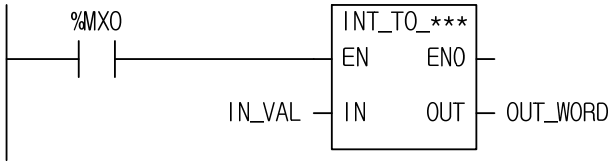
펑션	출력타입	동작 설명
INT_TO_SINT	SINT	입력이 -128 ~ 127일 경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
INT_TO_DINT	DINT	DINT 타입으로 정상 변환합니다.
INT_TO_LINT	LINT	LINT 타입으로 정상 변환합니다.
INT_TO_USINT	USINT	입력이 0 ~ 255일 경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
INT_TO_UINT	UINT	입력이 0 ~ 32767일 경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
INT_TO_UDINT	UDINT	입력이 0 ~ 32767일 경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
INT_TO_ULINT	ULINT	입력이 0 ~ 32767일 경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
INT_TO_BOOL	BOOL	하위 1비트를 취해 BOOL 타입으로 변환합니다.
INT_TO_BYTE	BYTE	하위 8비트를 취해 BYTE 타입으로 변환합니다.
INT_TO_WORD	WORD	내부 비트 배열의 변화 없이 WORD 타입으로 변환합니다.
INT_TO_DWORD	DWORD	상위 비트들을 0으로 채운 DWORD 타입으로 변환합니다.
INT_TO_LWORD	LWORD	상위 비트들을 0으로 채운 LWORD 타입으로 변환합니다.
INT_TO_REAL	REAL	INT를 REAL 타입으로 정상 변환합니다.
INT_TO_LREAL	LREAL	INT를 LREAL 타입으로 정상 변환합니다.
INT_TO_STRING	STRING	INT를 STRING 타입으로 정상 변환합니다.

■ 플래그

플래그	설명
_ERR	변환에러 발생시 _ERR, _LER 플래그가 셋(Set)됩니다. 에러 발생시 출력 타입의 비트 수 만큼 하위 비트를 취해 내부 비트 배열의 변환 없이 출력 시킵니다

■ 프로그램 예

1. LD



2. ST

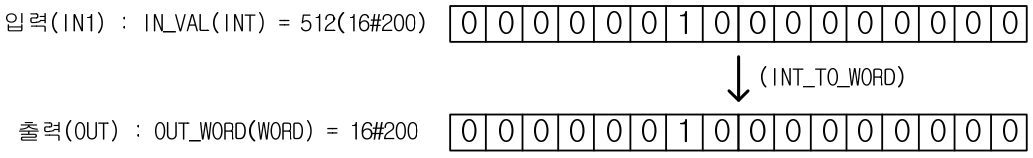
ST 언어는 INT_TO_***는 지원하지 않습니다.

INT_TO_WORD 인 경우

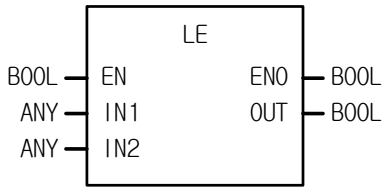
```
OUT_WORD := INT_TO_WORD(EN:= %MX0, IN1:= IN_VAL);
```

(1) 입력조건(%MX0)이 On 하면 INT_TO_*** 펑션이 실행됩니다.

(2) 입력 변수로 선언된 IN_VAL(INT 타입) = 512(16#200)이면, 출력 변수로 선언된 OUT_WORD(WORD 타입) =16#200 이 됩니다.



LE	적용 기종	발생플래그
'작거나 같다' 비교	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 비교할 값 IN2 : 비교할 값 입력은 8개까지 확장 가능 IN1, IN2, ...는 모두 같은 타입이어야 함. **출력** ENO : EN값이 그대로 출력 OUT : 비교 결과

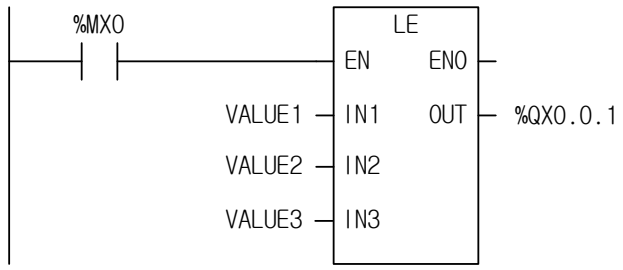
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	IN2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ 기능

1. IN1 ≤ IN2 ≤ IN3... ≤ INn(n 은 입력 개수)이면 OUT 으로 1 이 출력됩니다.
2. 다른 경우에는 OUT 으로 0 이 출력됩니다.

■ 프로그램 예

1. LD



2. ST

```
%QX0.0.1 := LE(EN:= %MX0, IN1:= VALUE1, IN2:= VALUE2, IN3:= VALUE3);
```

(1) 실행조건(%MX0)이 On 하면 LE(비교: 작거나 같다) 펑션이 실행됩니다.

(2) 입력 변수로 선언된 VALUE1 = 100, VALUE2 = 200, VALUE3 = 300 이면, 비교결과 VALUE1 ≤ VALUE2 ≤ VALUE3 이므로, 출력 결과값 %QX0.0.1 = 1 이 됩니다.

입력(IN1) : VALUE1(INT) = 100(16#0064) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

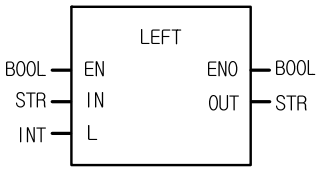
(IN2) : VALUE2(INT) = 200(16#00C8) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |

(IN3) : VALUE3(INT) = 300(16#012C) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |

↓

출력(OUT): %QX0.0.1(BOOL) = 1(16#1) | 1 |

LEFT	적용 기종	발생플래그
문자열의 왼쪽을 취하기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 입력 문자열 L : 출력할 문자열 길이 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 출력 문자열

■ 기능

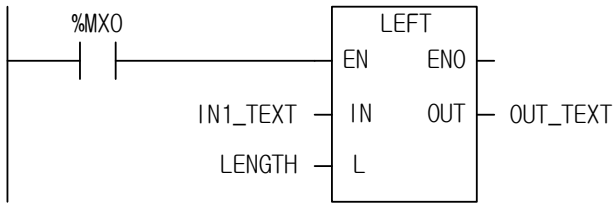
1. 입력 문자열 IN에 대하여 왼쪽부터 문자열 길이 L만큼 출력 문자열 OUT에 출력시킵니다.

■ 플래그

플래그	설명
_ERR	L < 0 인 경우, _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

1. LD

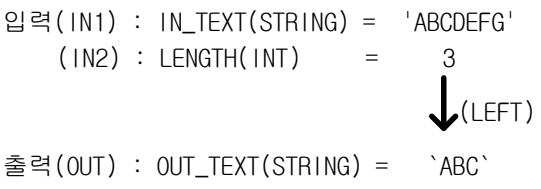


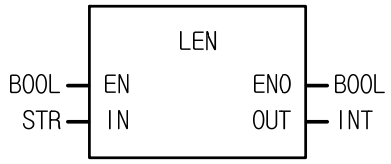
2. ST

```
OUT_TEXT:= LEFT(EN:= %MX0, IN:= IN1_TEXT, L:= LENGTH);
```

(1) 실행조건(%MX0)이 On 하면 LEFT(문자열의 왼쪽 취하기) 펑션이 실행됩니다.

(2) 입력 변수로 선언된 문자열이 IN_TEXT = `ABCDEFG`이고, 출력할 문자열의 길이 LENGTH = 3 이면, 출력 문자열 변수로 지정된 OUT_TEXT = `ABC`가 됩니다.



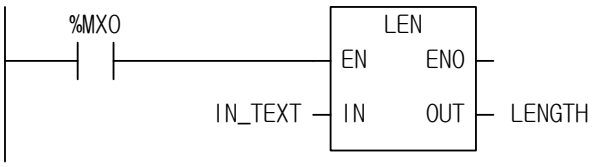
LEN	적용 기종	발생플래그
문자열 길이 구하기	XGI, XGR, XEC	-
펑 션	설 명	
	**입력** EN : 1일 때 펑션 실행 IN : 입력 문자열 **출력** ENO : EN 값이 그대로 출력 OUT : 문자열 길이	

■ 기능

1. 입력 문자열(IN)의 길이(문자 수)가 OUT 으로 출력됩니다.

■ 프로그램 예

1. LD



2. ST

```
LENGTH := LEN(EN:= %MX0, IN1:= IN_TEXT);
```

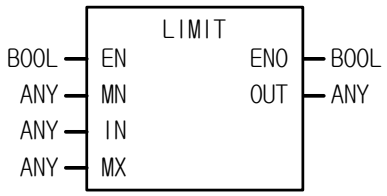
(1) 실행조건(%MX0)이 On 하면 LEN(문자열 길이 구하기) 펑션이 실행됩니다.
(2) 입력 변수로 선언된 IN_TEXT = `ABCD`이면, 출력 변수로 선언된 LENGTH = 4 가 됩니다.

입력 (IN) : IN_TEXT(STRING) = ' ABCD'

↓

출력 (OUT) : LENGTH(INT) = 4

LIMIT	적용 기종	발생플래그
상하한 제한	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 MN : 최소값 IN : 제한될 값 MX : 최대값 **출력** ENO : EN 값이 그대로 출력 OUT : 범위 안에 든 값 MN, IN, MX, OUT 은 데이터 타입이 모두 같아야 함

ANY 타입 변수설명

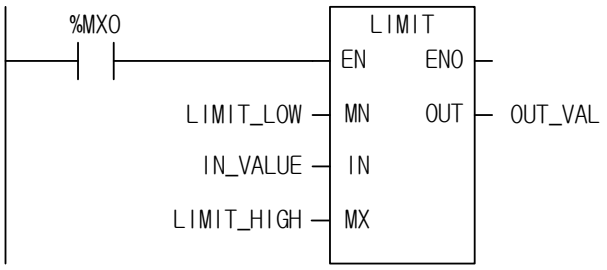
변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
MN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
IN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
MX	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ 기능

1. 입력 IN 값이 MN 과 MX 사이에 있으면, OUT 으로 IN 이 출력됩니다. 즉, MN ≤ IN ≤ MX 이면 OUT = IN
2. 입력 IN 값이 MN 보다 작으면, OUT 으로 MN 이 출력됩니다. 즉, IN < MN 이면 OUT = MN
3. 입력 IN 값이 MX 보다 크면, OUT 으로 MX 가 출력됩니다. 즉, IN > MX 이면 OUT = MX

■ 프로그램 예

1. LD



2. ST

```
OUT_VAL := LIMIT(EN:= %MX0, MX:= LIMIT_LOW, IN:= IN_VALUE, MX:= LIMIT_HIGH);
```

(1) 실행조건(%MX0)이 On 하면 LIMIT(상하한 제한) 펑션이 실행합니다.

(2) 하한 값의 입력 변수(LIMIT_LOW), 상한 값의 입력변수(LIMIT_HIGH), 제한된 값의 입력변수(IN_VALUE)에 대한 출력 변수(OUT_VAL)의 값은 아래와 같습니다.

LIMIT_LOW	IN_VALUE	LIMIT_HIGH	OUT_VAL
1000	2000	3000	2000
1000	500	3000	1000
1000	4000	3000	3000

입력(MN) : LIMIT_LOW (INT) = 1000

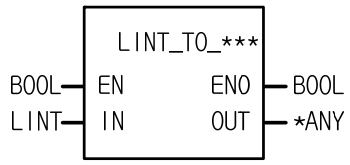
(IN) : IN_VALUE (INT) = 4000

(MX) : LIMIT_HIGH(INT) = 3000

↓ (LIMIT)

출력(OUT) : OUT_VAL (INT) = 3000

LINT_TO_***	적용 기종	발생플래그
LINT 타입변환	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 Long Integer 값 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 타입 변환된 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT	○	○	○	○	○	○	○	○		○	○	○	○	○	○					○

*ANY: ANY 타입 중 LINT, TIME, DATE, TOD, DT 제외

■ 기능

1. IN을 타입 변환해서 OUT으로 출력시킵니다.

펑션	출력타입	동작 설명
LINT_TO_SINT	SINT	입력이 -128 ~ 127일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
LINT_TO_INT	INT	입력이 -32,768~32,767 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
LINT_TO_DINT	DINT	입력이 -2^{31} ~ $2^{31}-1$일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
LINT_TO_USINT	USINT	입력이 0~ 255일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
LINT_TO_UINT	UINT	입력이 0~ 65,535일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
LINT_TO_UDINT	UDINT	입력이 0 ~ $2^{32}-1$일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
LINT_TO_ULINT	ULINT	입력이 0 ~ $2^{63}-1$일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
LINT_TO_BOOL	BOOL	하위 1비트를 취해 BOOL 타입으로 변환합니다.
LINT_TO_BYTE	BYTE	하위 8비트를 취해 BYTE 타입으로 변환합니다.
LINT_TO_WORD	WORD	하위 16비트를 취해 WORD 타입으로 변환합니다.
LINT_TO_DWORD	DWORD	하위 32비트를 취해 DWORD 타입으로 변환합니다.
LINT_TO_LWORD	LWORD	내부 비트 배열의 변화 없이 LWORD 타입으로 변환합니다.
LINT_TO_REAL	REAL	LINT를 REAL 타입으로 변환합니다. 변환 중 정밀도에 따른 오차가 발생할 수 있습니다.

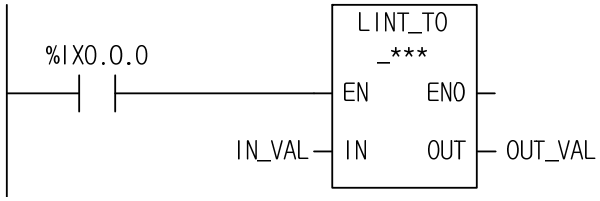
펑션	출력타입	동작 설명
LINT_TO_LREAL	LREAL	LINT 를 LREAL 타입으로 변환합니다. 변환 중 정밀도에 따른 오차가 발생할 수 있습니다.
LINT_TO_STRING	STRING	입력 값을 STRING 타입으로 변환합니다.

■ 플래그

플래그	설명
_ERR	변환에러 발생시 _ERR, _LER 플래그가 셋(Set)됩니다. 에러 발생시 출력 타입의 비트 수만큼 하위 비트를 취해 내부 비트 배열의 변환 없이 출력 시킵니다.

■ 프로그램 예

1. LD

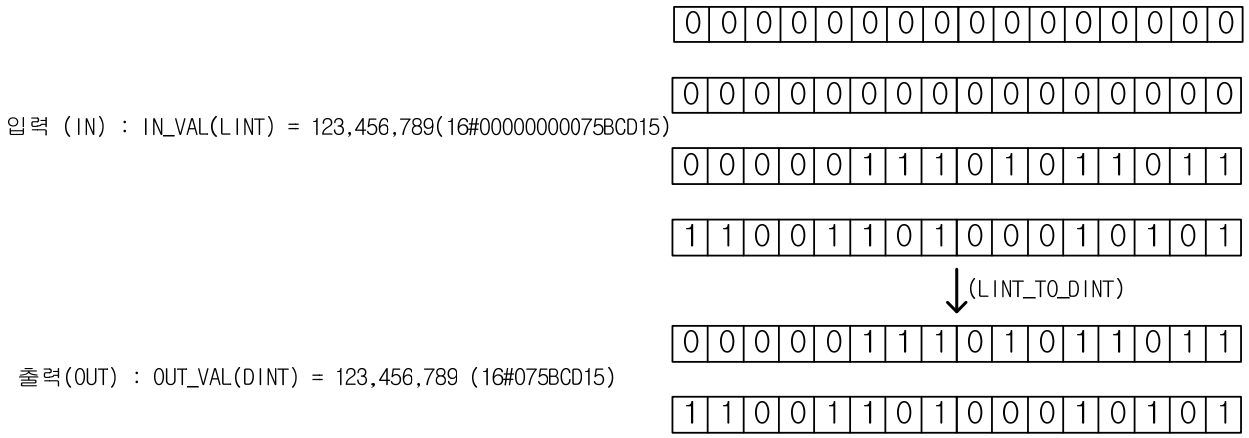


2. ST

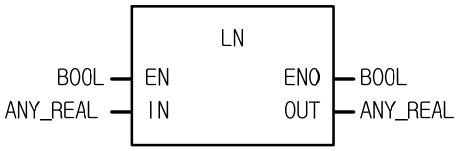
ST 언어는 LINT_TO_***는 지원하지 않습니다.
LINT_TO_DINT 인 경우

```
OUT_VAL := LINT_TO_DINT(EN:= %IX0.0.0, IN:= IN_VAL);
```

(1) 입력조건 (%IX0.0.0)이 On 하면 LINT_TO_*** 펑션이 실행됩니다.
(2) 입력변수로 선언된 IN_VAL(LINT 타입) = 123,456,789 이면, OUT_VAL(DINT 타입) = 123,456,789 가 됩니다.


0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
입력 (IN) : IN_VAL(LINT) = 123,456,789(16#00000000075BCD15)
0 0 0 0 0 1 1 1 0 1 0 1 1 0 1 1
1 1 0 0 1 1 0 1 0 0 0 1 0 1 0 1
(LINT_TO_DINT)
0 0 0 0 0 1 1 1 0 1 0 1 1 0 1 1
출력(OUT) : OUT_VAL(DINT) = 123,456,789 (16#075BCD15)
1 1 0 0 1 1 0 1 0 0 0 1 0 1 0 1

LN	적용 기종	발생플래그
LN 연산	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 자연대수 연산의 입력 값 **출력** ENO : EN값이 그대로 출력 OUT : 자연대수 값 IN, OUT은 같은 데이터 타입이어야 함

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

■ 기능

1. IN의 자연대수 값을 구해 OUT으로 출력시킵니다.

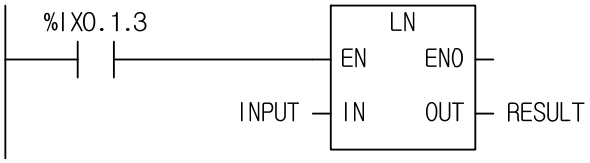
   OUT = ln (IN)

■ 에러

플래그	설명
_ERR	입력 값이 0 또는 음수일 때 _ERR, _LER 플래그가 셋(SET)됩니다.

■ 프로그램 예

1. LD



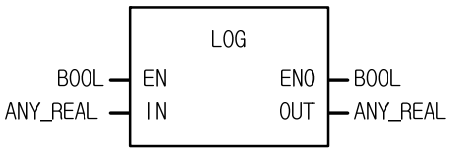
2. ST

```
RESULT := LN(EN:= %UX0.1.3, IN1:= INPUT);
```

(1) 실행조건(%IX0.1.3)이 On 하면 LN 펑션이 실행됩니다.

(2) INPUT 으로 선언된 입력 변수의 값이 2.0 일 때 출력 변수로 선언된 RESULT 은 0.6931... 입니다.
ln(2.0) = 0.6931...

LOG	적용 기종	발생플래그
LOG 연산	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 상용대수 연산의 입력 값 **출력** ENO : EN값이 그대로 출력 OUT : 상용대수 연산결과 값 IN, OUT은 같은 데이터 타입이어야 함

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

■ 기능

1. IN의 상용대수 값을 구해 OUT으로 출력시킵니다.

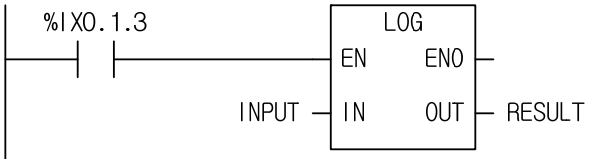
   $OUT = \log_{10}(IN) = \log(IN)$

■ 에러

플래그	설명
_ERR	입력 값이 0 또는 음수일 때 _ERR, _LER 플래그가 셋(SET)됩니다.

■ 프로그램 예

1. LD



2. ST

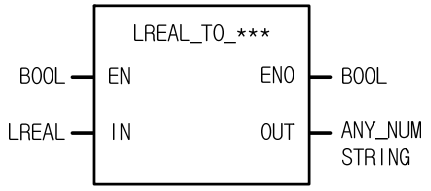
```
RESULT := LOG(EN:= %IX0.1.3, IN:= INPUT);
```

(1) 실행조건(%IX0.1.3)이 On 하면 LOG 펑션이 실행됩니다.

(2) INPUT 으로 선언된 입력 변수의 값이 2.0 일 때 출력 변수로 선언된 RESULT 은 0.3010... 입니다.

$Log_{10}(2.0) = 0.3010...$

LREAL_TO_***	적용 기종	발생플래그
LREAL 타입변환	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 LREAL 값 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 타입 변환된 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT					○	○	○	○	○	○	○	○	○	○						○

■ 기능

1. IN을 타입 변환해서 OUT으로 출력시킵니다.

펑션	출력타입	동작 설명
LREAL_TO_SINT	SINT	입력의 정수 부분이 -128 ~ 127 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다. (소수점 이하는 반올림)
LREAL_TO_INT	INT	입력의 정수 부분이 -32,768 ~ 32,767 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다. (소수점 이하는 반올림)
LREAL_TO_DINT	DINT	입력의 정수 부분이 -2^{31} ~ $2^{31}-1$ 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다. (소수점 이하는 반올림)
LREAL_TO_LINT	LINT	입력의 정수 부분이 -2^{63} ~ $2^{63}-1$ 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다. (소수점 이하는 반올림)
LREAL_TO_USINT	USINT	입력의 정수 부분이 0 ~ 255 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다. (소수점 이하는 반올림)
LREAL_TO_UINT	UINT	입력의 정수 부분이 0 ~ 65,535 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다. (소수점 이하는 반올림)
LREAL_TO_UDINT	UDINT	입력의 정수 부분이 0 ~ $2^{32}-1$ 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다. (소수점 이하는 반올림)
LREAL_TO_ULINT	ULINT	입력의 정수 부분이 0 ~ $2^{64}-1$ 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다. (소수점 이하는 반올림)
LREAL_TO_LWORD	LWORD	내부 비트 배열의 변화 없이 LWORD 타입으로 변환합니다.

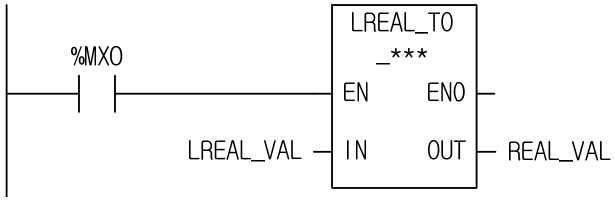
펑션	출력타입	동작 설명
LREAL_TO_REAL	REAL	LREAL 을 REAL 타입으로 정상 변환합니다. 변환 중 정밀도에 따른 오차가 발생할 수 있습니다.
LREAL_TO_STRING	STRING	LREAL 을 STRING 타입으로 정상 변환합니다.

■ 플래그

플래그	설명
_ERR	입력 값이 출력 타입에 저장할 수 있는 값보다 커서 오버 플로(Overflow) 발생시 _ERR, _LER 플래그가 셋(Set)됩니다. 에러 발생시 결과값에 0 을 출력합니다.

■ 프로그램 예

1. LD



2. ST

ST 언어는 LREAL_TO_***는 지원하지 않습니다.

LREAL_TO_REAL 인 경우

```
REAL_VAL :=  LREAL_TO_REAL(EN:= %MX0, IN:= LREAL_VAL);
```

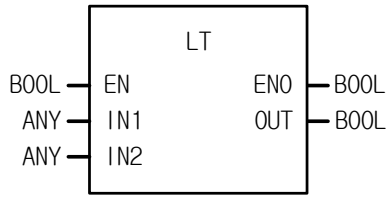
(1) 입력조건(%MX0)이 On 하면, LREAL_TO_*** 펑션이 실행됩니다.
(2) 입력변수로 선언된 LREAL_VAL(LREAL 타입) = -1.34E-12 이면,
출력변수로 선언된 REAL_VAL(REAL 타입) = -1.34E-12 가 됩니다.

입력(IN) : LREAL_VAL (LREAL) = -1.34E-12

↓ (LREAL_TO_REAL)

출력(OUT) : REAL_VAL (REAL) = -1.34E-12

LT	적용 기종	발생플래그
'작다' 비교	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 비교할 값 IN2 : 비교할 값 입력은 8개까지 확장 가능 IN1, IN2, ...는 모두 같은 타입이어야 함. **출력** ENO : EN값이 그대로 출력 OUT : 비교 결과값

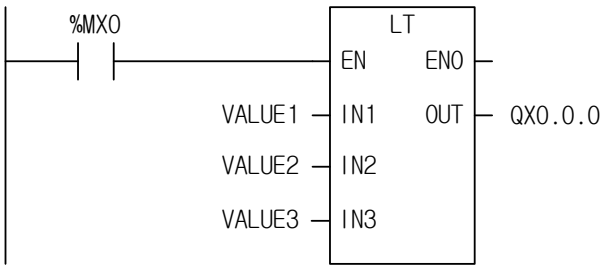
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	IN2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ 기능

1. IN1<IN2<IN3...<INn(n 은 입력개수)이면 OUT 으로 1 이 출력됩니다.
2. 다른 경우에는 OUT 으로 0 이 출력됩니다.

■ 프로그램 예

1. LD



2. ST

```
%QX0.0.0 := LT(EN:= %MX0, IN1:= VALUE1, IN2:= VALUE2, IN3:= VALUE3);
```

(1) 실행조건(%MX0)이 On 하면 LT(비교: 작다) 펑션이 실행됩니다.

(2) 입력변수로 선언된 VALUE1 = 100, VALUE2 = 200, VALUE3 = 300 이면, 비교결과 VALUE1 < VALUE2 < VALUIE3 이므로 출력 결과값 %QX0.0.0 = 1 이 됩니다.

입력(IN1) : VALUE1(INT) = 100(16#0064)

0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0

<(LT)

(IN2) : VALUE2(INT) = 200(16#00C8)

0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0

<(LT)

(IN3) : VALUE3(INT) = 300(16#012C)

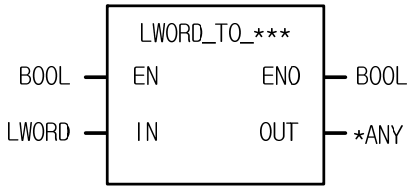
0	0	0	0	0	0	0	1	0	0	1	0	1	1	0	0

↓

출력(OUT) : %QX0.0.0(BOOL) = 1(16#0001)

1

LWORD_TO_***	적용 기종	발생플래그
LWORD 타입변환	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 비트 열(64 비트) **출력** ENO : EN 값이 그대로 출력 OUT : 타입 변환된 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT	○	○	○	○		○	○	○	○	○	○	○	○		○				○	○

*ANY: ANY 타입 중 LWORD, REAL, TIME, DATE, TOD 제외

■ 기능

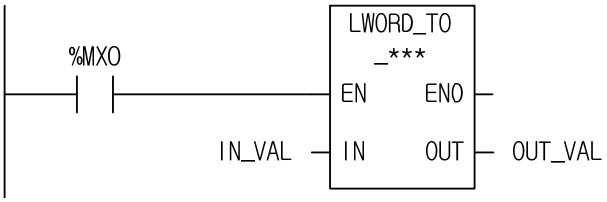
1. IN 을 타입 변환해서 OUT 으로 출력시킵니다.

펑션	출력타입	동작 설명
LWORD_TO_SINT	SINT	하위 8 비트를 취하여 SINT 타입으로 변환합니다.
LWORD_TO_INT	INT	하위 16 비트를 취하여 INT 타입으로 변환합니다.
LWORD_TO_DINT	DINT	하위 32 비트를 취하여 DINT 타입으로 변환합니다.
LWORD_TO_LINT	LINT	내부 비트 배열의 변화 없이 LINT 타입으로 변환합니다.
LWORD_TO_USINT	USINT	하위 8 비트를 취하여 USINT 타입으로 변환합니다.
LWORD_TO_UINT	UINT	하위 16 비트를 취하여 UINT 타입으로 변환합니다.
LWORD_TO_UDINT	UDINT	하위 32 비트를 취하여 UDINT 타입으로 변환합니다.
LWORD_TO_ULINT	ULINT	내부 비트 배열의 변화 없이 ULINT 타입으로 변환합니다.
LWORD_TO_BOOL	BOOL	하위 1 비트를 취하여 BOOL 타입으로 변환합니다.
LWORD_TO_BYTE	BYTE	하위 8 비트를 취하여 BYTE 타입으로 변환합니다.
LWORD_TO_WORD	WORD	하위 16 비트를 취하여 WORD 타입으로 변환합니다.
LWORD_TO_DWORD	DWORD	하위 32 비트를 취하여 DWORD 타입으로 변환합니다.
LWORD_TO_LREAL	LREAL	LWORD 를 LREAL 타입으로 변환합니다.
LWORD_TO_DT	DT	내부 비트 배열의 변화 없이 DT 타입으로 변환합니다. 단 DT 범위

펑션	출력타입	동작 설명
		(DT#2163-12-31-23:59:59:999)를 벗어난 값 입력시 _ERR, _LER 플래그를 SET 하고 DT 범위 내에서 순환하여 변환합니다.
LWORD_TO_STRING	STRING	입력 값을 STRING 타입으로 변환합니다.

■ 프로그램 예

1. LD



2. ST

ST 언어는 LWORD_TO_***는 지원하지 않습니다.

LWORD_TO_LINT 인 경우

```
OUT_VAL := LWROD_TO_LINT(EN:= %MX0, IN:= IN_VAL);
```

(1) 입력조건(%MX0)이 On 하면 LWORD_TO_*** 펑션이 실행됩니다.

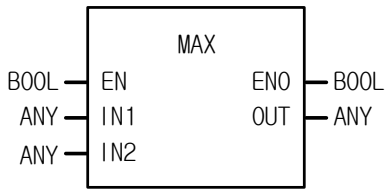
(2) 입력변수로 선언된 IN_VAL(LWORD 타입) = 16#FFFF_FFFF_FFFF_FFFF 면, 출력변수로 선언된 OUT_VAL(LINT 타입) = -1(16#FFFF_FFFF_FFFF_FFFF)이 됩니다.

입력(IN) : IN_VAL(LWORD) = 16#FFFFFFFFFFFFFFFF

↓(LWORD_TO_LINT)

출력(OUT) : OUT_VAL(LINT) = -1

MAX	적용 기종	발생플래그
최대값	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 비교될 값 IN2 : 비교될 값 입력 8개까지 확장 가능 **출력** ENO : EN값이 그대로 출력 OUT : 입력 값 중 최대값 IN1, IN2, ..., OUT은 모두 같은 타입이어야 함.

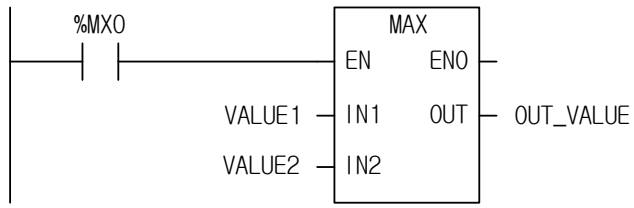
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	IN2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ 기능

1. 입력 IN1, IN2, ..., INn(n은 입력 개수)중에서 최대값을 OUT으로 출력시킵니다.

■ 프로그램 예

1. LD

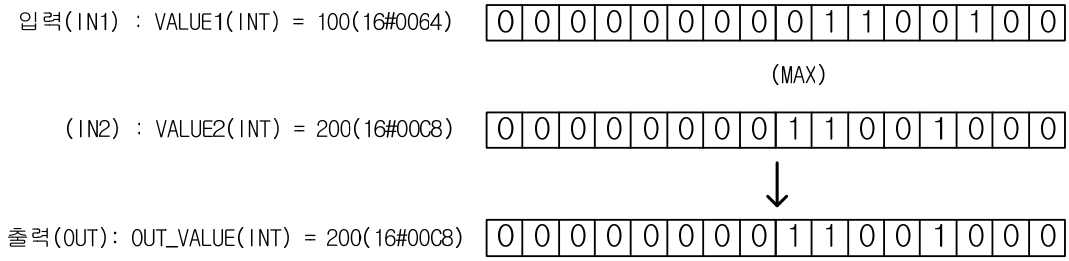


2. ST

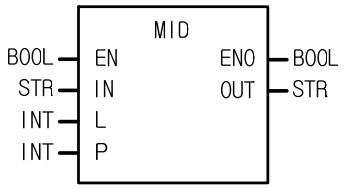
```
OUT_VALUE := MAX(EN:= %MX0, IN1:= VALUE1, IN2:= VALUE2);
```

(1) 실행조건(%MX0)이 On하면 MAX(최대값) 펑션이 실행됩니다.

(2) 입력변수로 선언된 VALUE1 = 100, VALUE2 = 200을 비교결과 최대값이 200이므로 출력변수로 선언된 OUT_VALUE = 200으로 출력됩니다.


입력(IN1) : VALUE1(INT) = 100(16#0064)
0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0
(MAX)
(IN2) : VALUE2(INT) = 200(16#00C8)
0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0
출력(OUT): OUT_VALUE(INT) = 200(16#00C8)
0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0

MID	적용 기종	발생플래그
문자열의 중간을 취하기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 입력 문자열 L : 출력할 문자열 길이 P : 출력할 문자열의 시작 위치 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 출력 문자열

■ 기능

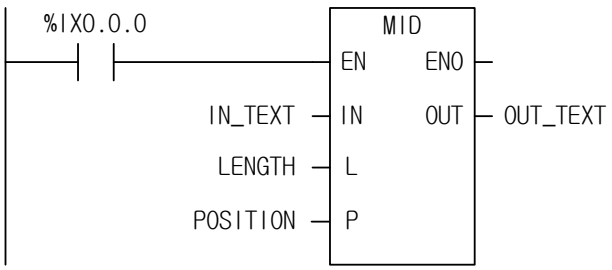
1. 입력 문자열 IN에 대하여 입력 문자열의 P번째 문자부터 길이L 만큼을 출력 문자열 OUT에 출력시킵니다.

■ 플래그

플래그	설명
_ERR	(변수 IN 의 문자 수) < P 인 경우, 또는 P <= 0 및 L < 0 인 경우 _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

1. LD



2. ST

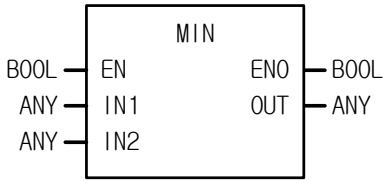
```
OUT_TEXT := MID(EN:= %IX0.0.0, IN:= IN_TEXT, L:= LENGTH, P:= POSITION);
```

(1) 실행조건(%IX0.0.0)이 On 하면 MID(문자열의 중간을 취하기) 펑션이 실행됩니다.

(2) 입력된 문자열 IN_TEXT = `ABCDEFG`이고, 출력할 문자열의 길이 LENGTH = 3, 출력할 문자열의 시작위치 POSITION = 2 이면, 출력 문자열 변수로 선언된 OUT_TEXT = `BCD`가 됩니다.

```
입력(IN) : IN_TEXT(STRING) = 'ABCDEFG'

     (L) :  LENGTH(INT)    =    3

     (P) :  POSITION(INT)  =    2
                                 ↓ (MID)
출력(OUT) : OUT_TEXT       =  `BCD`
```

MIN	적용 기종	발생플래그
최소값	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 비교될 값 IN2 : 비교될 값 입력 8개까지 확장 가능 **출력** ENO : EN값이 그대로 출력 OUT : 입력 값 중 최소값 IN1, IN2, ..., OUT은 모두 같은 타입이어야 함.

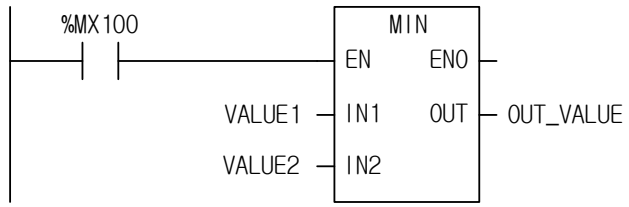
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	IN2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ 기능

1. 입력 IN1, IN2, ..., INn(n은 입력 개수) 중에서 최소값을 OUT으로 출력시킵니다.

■ 프로그램 예

1. LD

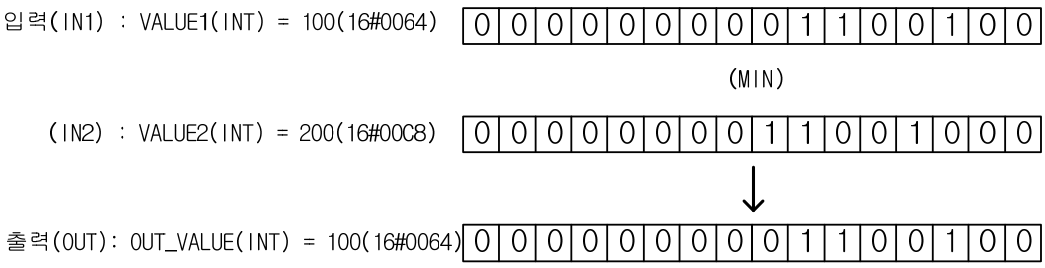


2. ST

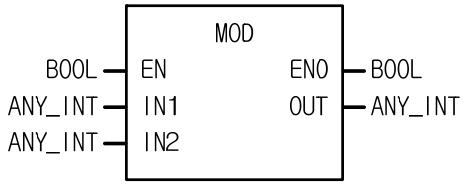
```
OUT_VALUE := MIN(EN:= %MX100, IN1:= VALUE1, IN2:= VALUE2);
```

(1) 실행조건(%MX100)이 On 하면 MIN(최소값) 펑션이 실행됩니다.

(2) 입력변수로 선언된 VALUE1 = 100, VALUE2 = 200 을 비교결과 최소값이 100 이므로 출력변수로 선언된 OUT_VALUE = 100 이 출력됩니다.



MOD	적용 기종	발생플래그
나머지 구하기	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 나누어 질 값(피제수) IN2 : 나눌 값(제수) **출력** ENO : EN 값이 그대로 출력 OUT : 나눈 결과값(나머지) IN1, IN2, OUT 에 연결되는 변수는 모두 같은 데이터 타입이어야 함.

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1						○	○	○	○	○	○	○	○							
	IN2						○	○	○	○	○	○	○	○							
	OUT						○	○	○	○	○	○	○	○							

■ 기능

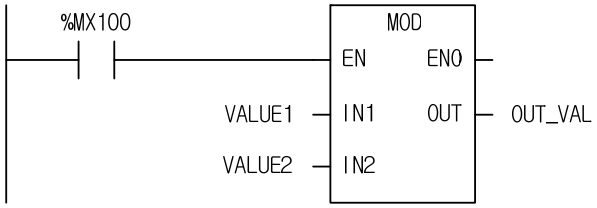
1. IN1 을 IN2 로 나눠서 그 나머지를 OUT 으로 출력시킵니다.

   OUT = IN1 - (IN1/IN2) × IN2 (단, IN2 = 0 이면 OUT = 0)

IN1	IN2	OUT
7	2	1
7	-2	1
-7	2	-1
-7	-2	-1
7	0	0

■ 프로그램 예

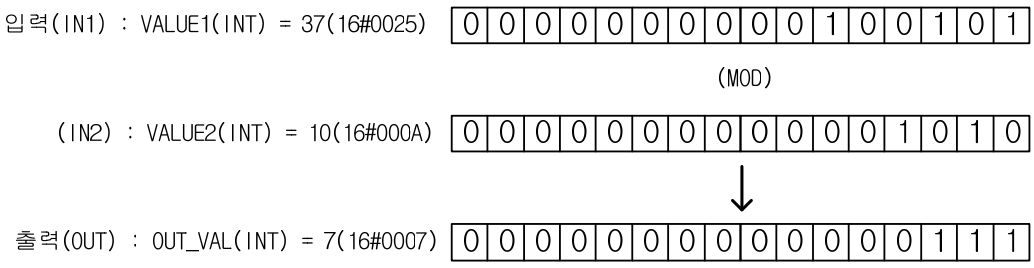
1. LD



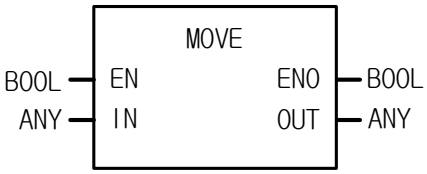
2. ST

```
OUT_VAL := MOD(EN:= %MX100, IN1:= VALUE1, IN2:= VALUE2);
```

(1) 실행조건(%MX100)이 On 하면 MOD(나머지 구하기) 펑션이 실행합니다.
(2) 입력변수 중 나누어질 값 VALUE1 = 37 이고, 나눌 값 VALUE2 = 10 이면, 출력변수로 선언된 OUT_VAL 의 값은 37 을 10 으로 나눈 나머지 7 이 됩니다.



MOVE	적용 기종	발생플래그
데이터 복사	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : MOVE할 값 **출력** ENO : EN값이 그대로 출력 OUT : MOVE된 값 IN, OUT에 연결되는 변수는 같은 데이터 타입이어야 함

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

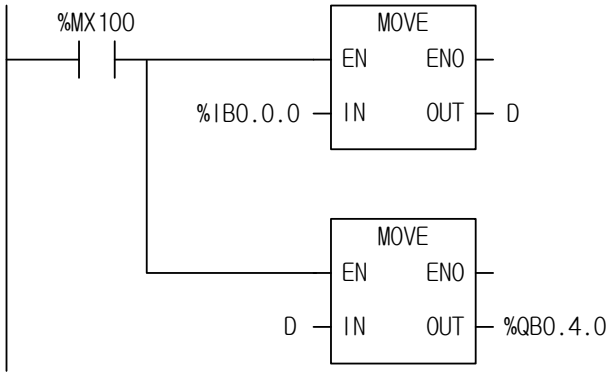
■ 기능

1. IN의 값을 OUT으로 이동합니다.

■ 프로그램 예

입력 %IX0.0.0~%IX0.0.7 의 8 점의 입력상태를 변수 D 로 전송한 후, 전송된 데이터를 출력 %QX0.4.0~%QX0.4.7 의 8 점으로 출력시키는 프로그램

1. LD

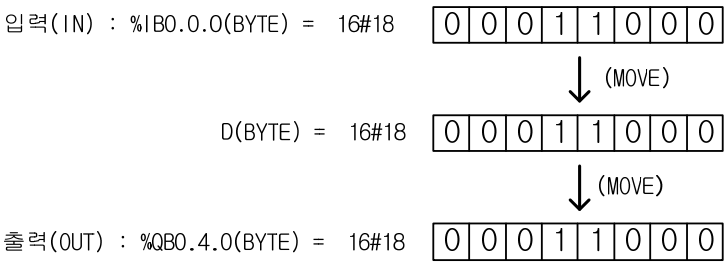


2. ST

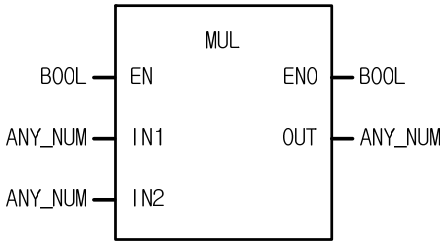
```
D := MOVE(EN:= %MX100, IN:= %IB0.0.0);
%QB0.4.0 := MOVE(EN:= %MX100, IN:= D);
```

(1) 실행조건(%MX100)가 On 되면 MOVE(데이터 복사) 펑션이 실행됩니다.

(2) 첫번째 MOVE 펑션에 의해 입력모듈의 8 점 입력 데이터가 변수 D 영역으로 옮겨지고 두번째 MOVE 펑션에 의해 변수 D 에 저장된 입력모듈의 상태가 출력모듈로 출력됩니다.



MUL	적용 기종	발생플래그
곱하기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 곱해질 값 (피승수) IN2 : 곱할 값 (승수) 입력은 8개까지 확장 가능 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 곱한 결과 값 IN1, IN2, ..., OUT 에 연결되는 변수는 모두 같은 데이터 타입이어야 함

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1						○	○	○	○	○	○	○	○	○	○					
	IN2						○	○	○	○	○	○	○	○	○	○					
	OUT						○	○	○	○	○	○	○	○	○	○					

■ 기능

1. IN1, IN2,..., INn (n 은 입력 개수)를 곱해서 OUT 으로 출력시킵니다.

   OUT = IN1 × IN2 × ... × INn

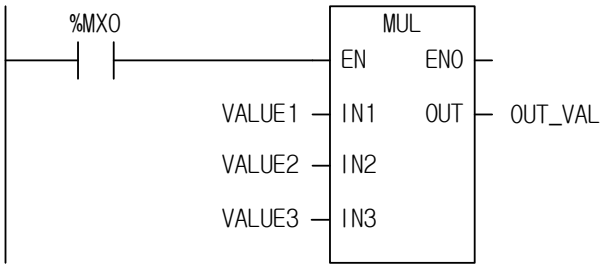
■ 플래그

플래그	설명
_ERR	출력 값이 해당 데이터 타입의 범위를 벗어날 경우 _ERR, _LER 플래그가 셋(Set)됩니다.

☆ REAL, LREAL 타입 연산에서는 IN1 에서 IN8 로 순차적으로 연산을 하기 때문에 중간 연산과정에서 이미 REAL, LREAL 에 최대값이나 최소값을 넘게 되면 _ERR, _LER 플래그가 셋(Set)되고 결과값은 무한대 값 또는 비정상적인 값 (1.#INF000000000000e+000, 1.#SNAN00000000000e+000, 1.#QNAN00000000000e+000)으로 표시됩니다.

■ 프로그램 예

1. LD



2. ST

```
OUT_VAL := MUL(EN:= %MX0, IN1:= VALUE1, IN2:= VALUE2, IN3:= VALUE3);
```

(1) 실행조건(%MX0)이 On 하면 MUL(곱하기) 펑션이 On 합니다.

(2) MUL 펑션의 입력변수로 선언된 VALUE1 = 30, VALUE2 = 20, VALUE3 = 10 이면, 출력변수로 선언된 OUT_VAL = 30 × 20 × 10 = 6,000 이 됩니다.

입력(IN1) : VALUE1(INT) = 30(16#001E) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 |

+(MUL)

(IN2) : VALUE2(INT) = 20(16#0014) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |

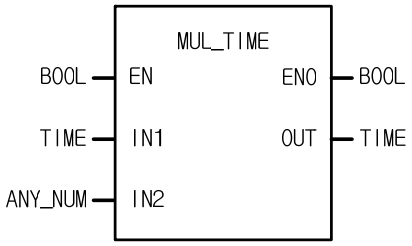
+(MUL)

(IN3) : VALUE3(INT) = 10(16#000A) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

↓

출력(OUT) : OUT_VAL(INT) = 6,000(16#1770) | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

MUL_TIME	적용 기종	발생플래그
시간 곱하기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 곱할 시간 IN2 : 곱할 값 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 곱한 결과 시간

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN2						○	○	○	○	○	○	○	○	○	○					

■ 기능

1. IN1(시간)을 IN2(숫자)로 곱해서 결과 시간을 OUT 으로 출력시킵니다.

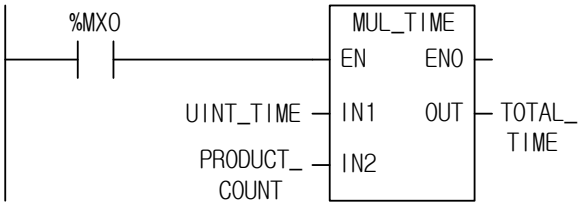
■ 플래그

플래그	설명
_ERR	출력 값이 TIME 데이터 타입의 범위를 벗어날 경우, _ERR, _LER 플래그가 셋(Set)됩니다. IN2 에 음수 값 입력 시 _ERR, _LER 플래그가 ON 되고 IN2 값을 16 진수 값으로 변환 후 곱셈 결과를 출력합니다.

■ 프로그램 예

어떤 제품생산 LINE 에서 생산되는 제품의 단위 제품당 평균 계획시간이 20 분 2 초이고, 하루 생산할 제품의 수가 20 개 일 때 작업 소요시간을 설정하는 프로그램

1. LD



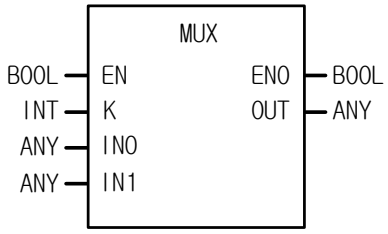
2. ST

```
TOTAL_TIME := MUL_TIME(EN:= %MX0, IN1:= UINT_TIME, IN2:= PRODUCT_COUNT);
```

(1) 입력변수(1N1: 단위 제품당 제작 시간) UNIT_TIME: T#20M2S 을 입력합니다.
(2) 입력변수(1N2: 생산수량) PRODUCT_COUNT: 20 을 입력합니다.
(3) 출력변수 (OUT: 총 작업 소요시간)에 TOTAL_TIME 을 입력합니다.
(4) 실행조건(%MX0)이 On 되면 출력변수로 설정한 TOTAL_TIME 에 T#6H40M40S 가 출력됩니다.

```
입력(IN1): UNIT_TIME(TIME)     =   T#20MS2S
                                   (MUL_TIME)
    (IN2): PRODUCT_COUNT(INT) =     16#18
                                     ↓
출력(OUT): TOTAL_TIME(TIME)    =  T#6H40M40S
```

MUX	적용 기종	발생플래그
여러 개 중 선택	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 K : 선택 IN0 : 선택될 값 IN1 : 선택될 값 입력은 8 개까지 확장 가능(IN0, IN1,..., IN7) **출력** ENO : 에러 없이 실행되면 1 을 출력 OUT : 선택된 값 IN0, IN1, ..., OUT 은 모두 같은 타입이어야 함

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN0	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ 기능

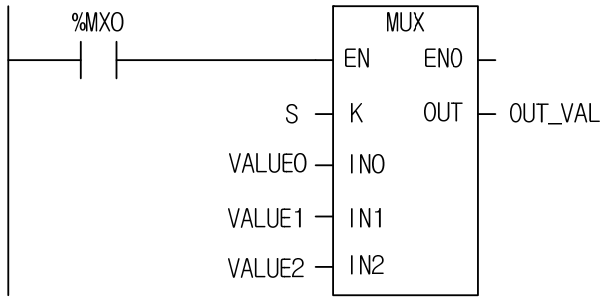
1. K 값으로 여러 입력(IN0, IN1,..., INn) 중 하나를 선택하여 출력시킵니다.
2. K = 0 이면 IN0 이, K = 1 이면 IN1 이, K = n 이면 INn 이 OUT 으로 출력됩니다.

■ 플래그

플래그	설명
_ERR	K 의 값이 입력 변수 INn 의 개수보다 크거나 같은 경우에 OUT 으로는 IN0 값이 출력되고, _ERR, _LER 플래그가 셋(Set)됩니다. K 값이 음수일 때 _ERR, _LER 플래그가 셋(SET)됩니다.

■ 프로그램 예

1. LD



2. ST

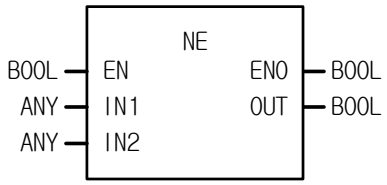
```
OUT_VAL := MUX(EN:= %MX0, K:= S, IN0:= VALUE0, IN1:= VALUE1, IN2:= VALUE2);
```

(1) 실행조건(%MX0)이 On 하면 MUX(여러 개를 선택) 펑션이 실행됩니다.

(2) 입력변수로 설정된 VALUE0, 1, 2 중 선택변수 S 의 값에 따라 선택되어 출력변수로 선언된 OUT 으로 출력시킵니다.

```
입력 (K) : S(INT) =  2
   (IN0) : VALUE0(WORD) =    16#0011
   (IN1) : VALUE1(WORD) =    16#0022
   (IN2) : VALUE2(WORD) =    16#0033
                              ↓ (MUX)
출력 (OUT) : OUT_VAL(WORD) = 16#0033
```

NE	적용 기종	발생플래그
'같지 않다' 비교	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 실행 허용 IN1 : 비교될 값 IN2 : 비교될 값 IN1, IN2 는 같은 타입이어야 함. **출력** ENO : EN 값이 그대로 출력 OUT : 비교 결과값

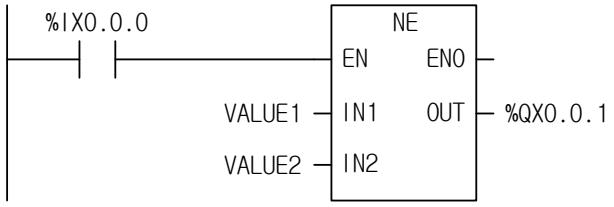
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	IN2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ 기능

1. IN1 이 IN2 와 같지 않으면 OUT 으로 1 이 출력됩니다.
2. 같으면 OUT 으로 0 이 출력됩니다.

■ 프로그램 예

1. LD

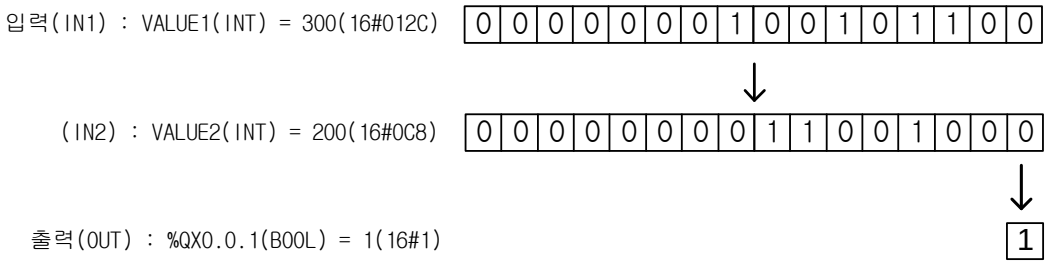


2. ST

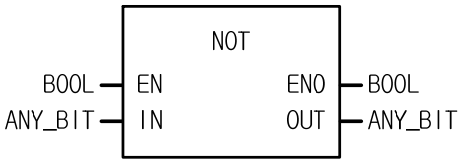
```
%QX0.0.1 := NE(EN:= %IX0.0.0, IN1:= VALUE1, IN2:= VALUE2);
```

(1) 실행조건(%IX0.0.0)이 On 하면 NE(비교: 같지 않다) 펑션이 실행됩니다.

(2) 입력변수로 선언된 VALUE1 = 300, VALUE2 = 200 이면, 비교결과 VALUE1 과 VALUE2 가 다르므로 출력결과값 %QX0.0.1 = 1 이 됩니다


입력(IN1) : VALUE1(INT) = 300(16#012C)
0 0 0 0 0 0 0 1 0 0 1 0 1 1 0 0
(IN2) : VALUE2(INT) = 200(16#0C8)
0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0
출력(OUT) : %QX0.0.1(BOOL) = 1(16#1)
1

NOT	적용 기종	발생플래그
논리 반전	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : NOT 될 값 **출력** ENO : EN 값이 그대로 출력 OUT : NOT 된 값 IN, OUT 은 모두 같은 타입이어야 함.

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN	○	○	○	○	○															
	OUT	○	○	○	○	○															

■ 기능

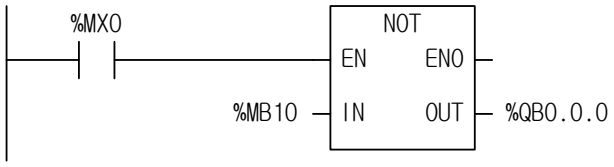
1. IN 을 비트별로 NOT(반전)해서 OUT 으로 출력시킵니다.

   IN 1100 1010

   OUT 0011 0101

■ 프로그램 예

1. LD

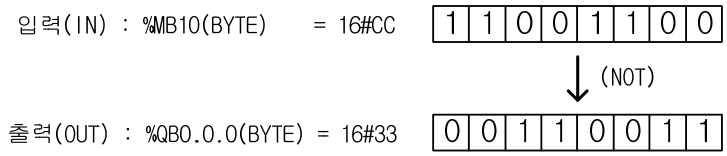


2. ST

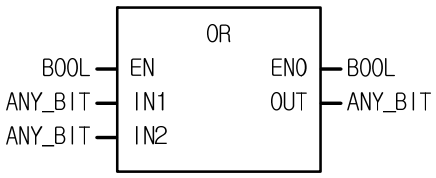
```
%QB0.0.0 := NOT_BYTE(EN:= %MX0, IN1:=MB10);
```

(1) 실행조건(%MX0)이 On 하면 NOT(논리반전) 펑션이 실행됩니다.

(2) NOT 펑션이 실행되면 입력변수로 선언된 %MB10 의 데이터 값이 반전되어 출력변수로 선언된 %QB0.0.0 에 출력됩니다.


입력(IN) : %MB10(BYTE) = 16#CC
1 1 0 0 1 1 0 0
(NOT)
출력(OUT) : %QB0.0.0(BYTE) = 16#33
0 0 1 1 0 0 1 1

OR	적용 기종	발생플래그
논리합	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : OR될 값 IN2 : OR될 값 입력 8개까지 확장 가능 **출력** ENO : EN값이 그대로 출력 OUT : OR된 값 IN1, IN2, OUT은 모두 같은 타입이어야 함.

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN	○	○	○	○	○															
	OUT	○	○	○	○	○															

■ 기능

1. IN1을 IN2와 비트별로 OR해서 OUT으로 출력시킵니다.

   IN1 1111 0000

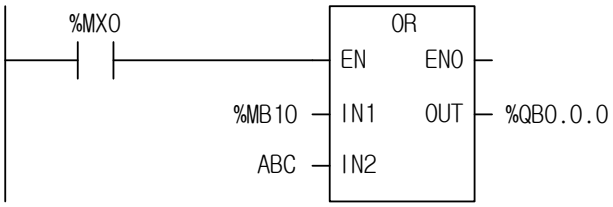
   OR

   IN2 1010 1010

   OUT 1111 1010

■ 프로그램 예

1. LD



2. ST

```
%QB0.0.0 := OR2_BYTE(EN:=%MX0, IN1:=%MB10, IN2:=ABC);
```

(1) 실행조건(%MX0)이 On 하면 OR 펑션이 실행됩니다.

(2) %MB10 = 2#1100_1100 을 ABC = 2#1111_0000 와 OR 시킨 결과가 %QB0.0.0 = 2#1111_1100 로 출력됩니다.

입력(IN1) : %IB0.0.0(BYTE) = 16#CC | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |

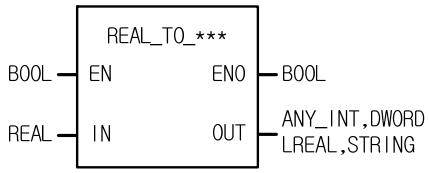
& (OR)

(IN2) : ABC(BYTE) = 16#F0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

↓

출력(OUT) : %QB0.0.0(BYTE) = 16#FC | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

REAL_TO_***	적용 기종	발생플래그
REAL 타입 변환	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 REAL 값 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 타입 변환된 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT				○		○	○	○	○	○	○	○	○		○					○

■ 기능

1. IN을 타입 변환해서 OUT으로 출력시킵니다.

펑션	출력 타입	동작 설명
REAL_TO_SINT	SINT	입력의 정수 부분이 -128 ~ 127 일 경우 정상 변환되나, 그 외 값은 에러가 발생 합니다. (소수점 이하는 반올림)
REAL_TO_INT	INT	입력의 정수 부분이 -32768 ~ 32767 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다. (소수점 이하는 반올림)
REAL_TO_DINT	DINT	입력의 정수 부분이 -2^{31} ~ $2^{31}-1$ 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다. (소수점 이하는 반올림)
REAL_TO_LINT	LINT	입력의 정수 부분이 -2^{63} ~ $2^{63}-1$ 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다. (소수점 이하는 반올림)
REAL_TO_USINT	USINT	입력의 정수 부분이 0 ~ 255 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다. (소수점 이하는 반올림)
REAL_TO_UINT	UINT	입력의 정수 부분이 0 ~ 65,535 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다. (소수점 이하는 반올림)
REAL_TO_UDINT	UDINT	입력의 정수 부분이 0 ~ $2^{32}-1$ 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다. (소수점 이하는 반올림)
REAL_TO_ULINT	ULINT	입력의 정수 부분이 0 ~ $2^{64}-1$ 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다. (소수점 이하는 반올림)
REAL_TO_DWORD	DWORD	내부 비트 배열의 변화 없이 DWORD 타입으로 변환합니다.
REAL_TO_LREAL	LREAL	REAL을 LREAL 타입으로 정상 변환합니다.

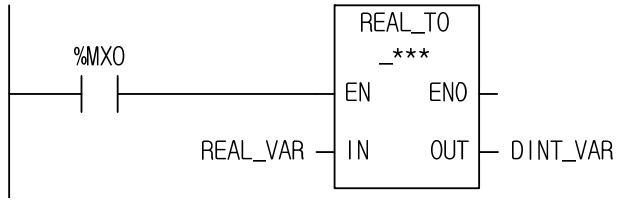
펑션	출력 타입	동작 설명
REAL_TO_STRING	STRING	REAL 을 STRING 타입으로 정상 변환합니다.

■ 플래그

플래그	설명
_ERR	입력값이 출력타입에 저장할 수 있는 값보다 커서 오버 플로(Overflow)가 발생하면 _ERR, _LER 플래그가 셋(Set)됩니다. 에러 발생시 결과값에 0 을 출력합니다.

■ 프로그램 예

1. LD



2. ST

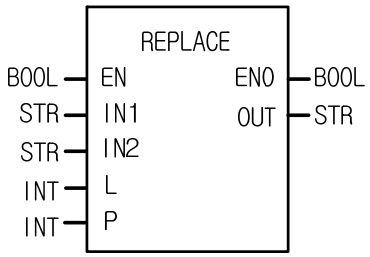
ST 언어는 REAL_TO_***은 지원하지 않습니다.
REAL_TO_DINT 인 경우

```
DINT_VAR := REAL_TO_DINT(EN:=%MX0, IN:=REAL_VAR);
```

(1) 실행조건(%MX0)이 On 하면 REAL_TO_*** 펑션이 실행됩니다.
(2) REAL_VAL(REAL 타입) = 1.234E4 이면, DINT_VAL(DINT 타입) = 12,340 이 됩니다.

입력(IN) : REAL_VAL(REAL) = 1.234E4
↓ (REAL_TO_DINT)
출력(OUT) : DINT_VAL(DINT) = 12,340

REPLACE	적용 기종	발생플래그
문자열 대체하기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 대체될 문자열 IN2 : 대체할 문자열 L : 대체될 문자열의 길이 P : 대체될 문자열의 위치 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 출력 문자열

■ 기능

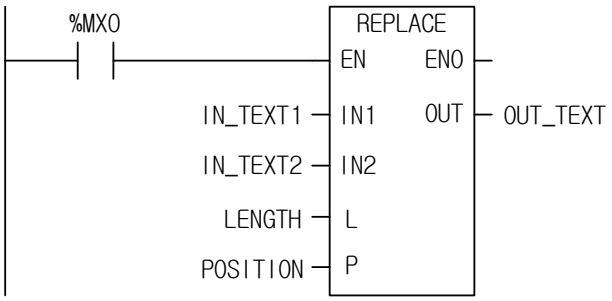
1. 문자열 IN1의 P번째 문자부터 길이 L만큼의 문자를 문자열 IN2로 대체한 후 문자열 OUT에 출력시킵니다.

■ 플래그

플래그	설명
_ERR	아래와 같은 경우 _ERR, _LER 플래그가 셋(Set)됩니다. P ≤ 0 또는 L < 0, P > (IN1의 입력 문자열의 문자 수) 연산 결과 문자 수 > 30

■ 프로그램 예

1. LD



2. ST

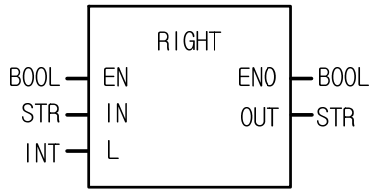
```
OUT_TEXT := REPLACE(EN:=%MX0, IN1:=IN_TEXT1, IN2:= IN_TEXT2, L:=LENGTH, P:=POSITION);
```

(1) 실행조건(%MX0)이 On 하면 REPLACE(문자열 대체하기) 펑션이 실행됩니다.

(2) 대체될 문자열 입력변수가 IN_TEXT1 = `ABCDEF`이고, 대체할 문자열 입력변수 IN_TEXT2 = `X`이며, 대체될 문자열의 길이 입력변수 LENGTH=3, 대체될 문자열 위치 지정 입력변수 POSITION=2 이면 IN_TEXT1 의 `BCD`가 IN_TEXT2 의 `X`로 대체되어 출력변수 OUT_TEXT 에 `AXEF`가 출력됩니다.

```
입력(IN1) : IN_TEXT1(STRING) = `ABCDEF`
    (IN2) : IN_TEXT2(STRING) =  ` X`
      (L) : LENGTH(INT) =         3
      (P) : POSITION(INT) =       2
                                  ↓
출력(OUT) : OUT_TEXT(STRING) =  `AXEF`
```

RIGHT	적용 기종	발생플래그
문자열의 오른쪽을 취하기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 입력 문자열 L : 출력할 문자열 길이 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 출력 문자열

■ 기능

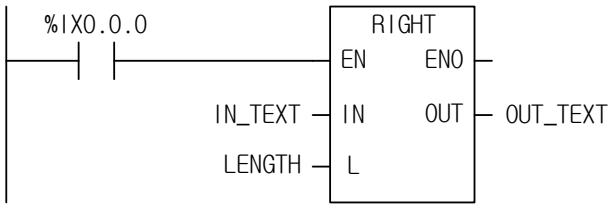
1. 입력 문자열 IN에 대하여 오른쪽부터 문자열 길이 L 만큼을 출력 문자열 OUT에 출력시킵니다.

■ 플래그

플래그	설명
_ERR	L < 0 인 경우, _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

1. LD



2. ST

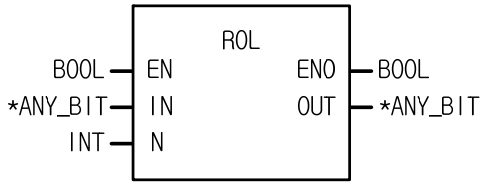
```
OUT_TEXT := RIGHT(EN:=%IX0.0.0, IN:=IN_TEXT, L:=LENGTH);
```

(1) 실행조건(%IX0.0.0)이 On 하면 RIGHT(문자열의 오른쪽 취하기) 펑션이 실행됩니다.

(2) 입력변수로 선언된 문자열이 IN_TEXT = `ABCDEFG`이고, 출력할 문자열의 길이 LENGTH = 3 이면, 출력 문자열 변수로 지정된 OUT_TEXT = `EFG`가 됩니다.

```
입력(IN) : IN_TEXT(STRING)  = `ABCDEFG`
     (L) :  LENGTH(INT)  =        3
                                  ↓ (RIGHT)
출력(OUT) : OUT_TEXT(STRING)  =  `EFG`
```

ROL	적용 기종	발생플래그
왼쪽으로 회전 (Rotate Left)	XGI, XGR, XEC	-

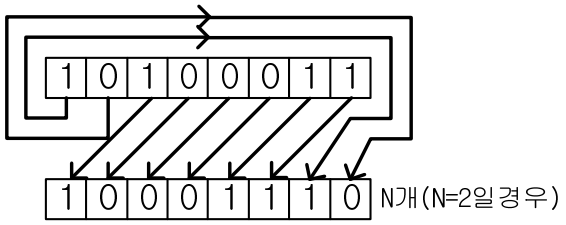
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 회전될 값 N : 회전할 비트 수 **출력** ENO : EN값이 그대로 출력 OUT : 회전된 값

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*ANY_BIT: ANY_BIT 중 BOOL 제외

■ 기능

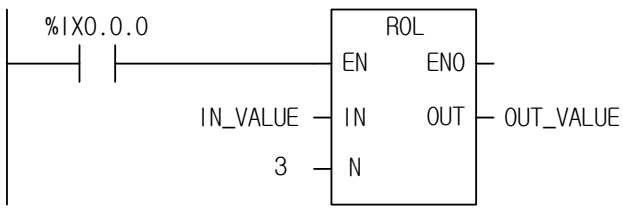
1. 입력 IN을 N 비트 수만큼 왼쪽으로 회전시킵니다.



■ 프로그램 예

입력 %IX0.0.0이 On하면 입력 데이터(2#1100_1100_1100_1100:16#CCCC)의 값을 좌로 3비트 만큼 회전시키는 프로그램

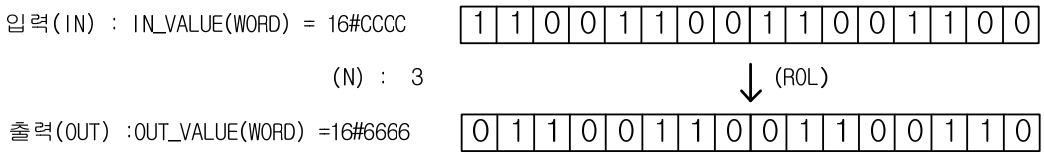
1. LD



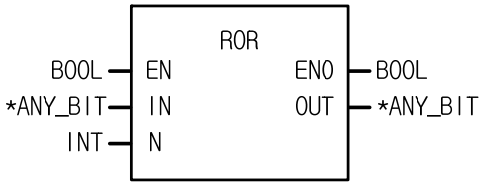
2. ST

```
OUT_VALUE := ROL(EN:=%IX0.0.0, IN:=IN_VALUE, N:=3);
```

(1) 회전할 데이터 값을 입력한 변수 IN_VALUE 로 설정한다.
(2) 좌회전할 비트 수 3 을 회전할 비트 수 지정 입력(N)에 쓴다.
(3) 회전된 데이터 값을 출력할 출력변수를 OUT_VALUE 로 설정한다.
(4) 실행조건 %IX0.0.0 이 On 하면 ROL(왼쪽으로 회전) 펑션이 실행되어 입력변수로 설정된 데이터 비트를 좌로 3 비트 회전하여 출력변수로 선언된 OUT_VALVE 값에 출력된다.



ROR	적용 기종	발생플래그
오른쪽으로 회전 (Rotate Right)	XGI, XGR, XEC	-

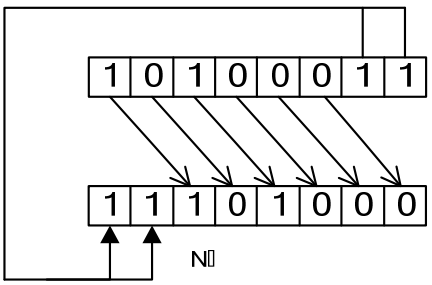
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 회전될 값 N : 회전할 비트 수 **출력** ENO : EN 값이 그대로 출력 OUT : 회전된 값

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*ANY_BIT: ANY 타입 중 BOOL 제외

■ 기능

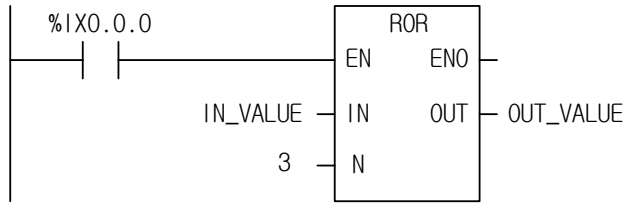
1. 입력 IN 을 N 비트 수만큼 오른쪽으로 회전시킵니다.



■ 프로그램 예

입력 %IX0.0.0 이 On 하면 입력 데이터 값(2#1110_0011_0011_0001: 16#E331)을 우로 3 비트 만큼 회전시키는 프로그램

1. LD



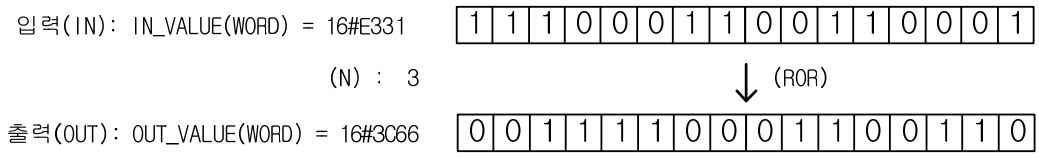
2. ST

```
OUT_VALUE := ROR(EN:=%IX0.0.0, IN:=IN_VALUE, N:=3);
```

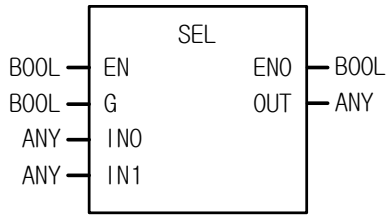
(1) 회전할 데이터 값을 입력한 변수를 IN_VALUE 로 설정한다.

(2) 우회전할 비트 수 3 을 회전할 비트 수 지정 입력(N)에 설정한다.

(3) 실행조건 %IX0.0.0 이 On 하면 ROR(오른쪽으로 회전) 펑션이 실행되어 입력변수로 설정된 데이터 비트가 우로 3 비트만큼 회전되어 출력변수로 선언된 OUT_VALUE 값에 출력된다.



SEL	적용 기종	발생플래그
둘 중 선택	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 G : 선택 IN0 : 선택될 값 IN1 : 선택될 값 **출력** ENO : EN 값이 그대로 출력 OUT : 선택된 값 IN1, IN2, OUT 은 모두 같은 타입이어야 함.

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN0	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

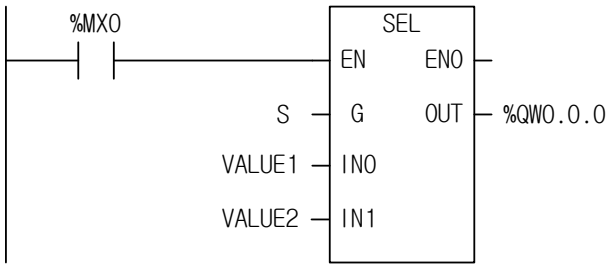
■ 기능

1. G가 0 이면 IN0 이 OUT 으로, G가 1 이면 IN1 이 OUT 으로 출력됩니다.

■ 프로그램 예

입력 %MX0 이 On 하면 VALUE1 과 VALUE2 의 데이터 값 중 S 에 입력된 값에 따라 하나의 데이터 값을 출력하는 프로그램.

1. LD



2. ST

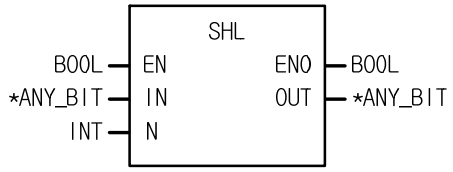
```
%QW0.0.0 := SEL(EN:=%MX0, G:=S, IN0:=VALUE1, IN1:=VALUE2);
```

(1) 실행조건(%MX0)이 On 하면 SEL(둘 중 선택) 펑션이 실행됩니다.

(2) SEL 펑션이 실행되면 S = 1일 때, VALUE1 = 16#1110, VALUE2 = 16#FF00 이면 %QW0.0.0 = 16#FF00 이 됩니다.

```
입력(G)   : S = 1
  (IN0)   : VALUE1(WORD) = 16#1110
  (IN1)   : VALUE2(WORD) = 16#FF00
                                ↓ (SEL)
출력(OUT)  : %QW0.0.0(WORD) = 16#FF00
```

SHL	적용 기종	발생플래그
왼쪽으로 이동 (Shift Left)	XGI, XGR, XEC	-

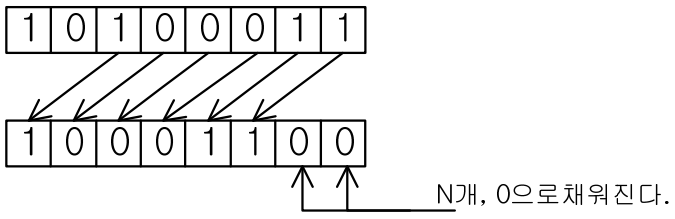
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 이동될 비트 열 N : 이동할 비트 수 **출력** ENO : EN 값이 그대로 출력 OUT : 이동된 값

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*ANY_BIT: ANY_BIT 중 BOOL 제외

■ 기능

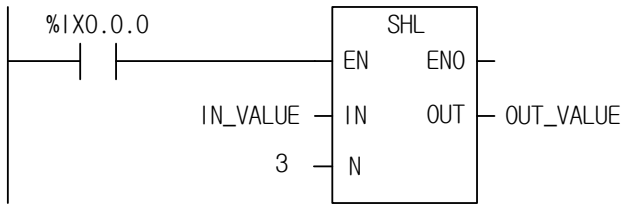
1. 입력 IN을 N 비트 수만큼 왼쪽으로 이동합니다.
2. 입력 IN의 맨 오른쪽에 있는 N개 비트는 0으로 채워집니다.



■ 프로그램 예

입력 %IX0.0.0 이 On 하면 입력 데이터 값(2#1100_1100_1100_1100: 16#CCCC)을 좌로 3 비트 만큼 이동시키는 프로그램

1. LD



2. ST

```
OUT_VALLUE := SHL(EN:=%IX0.0.0, IN:=IN_VALUE, N:=3);
```

(1) 이동할 데이터 값을 입력할 변수를 IN_VALUE(2#1100_1100_1100_1100: 16#CCCC)로 설정한다.

(2) 좌로 이동한 비트 수 3 을 지정 입력(N)에 쓴다. (변수 지정 후 쓰기로 가능)

(3) 실행조건(%IX0.0.0)이 On 하면 SHL(왼쪽으로 이동) 펑션이 실행되어 입력변수로 설정된 데이터 비트가 좌로 3 비트 이동하여, 출력변수로 선언된 OUT_VALUE 에 출력됩니다.

입력(IN) : IN_VALUE(WORD) = 16#CCCC

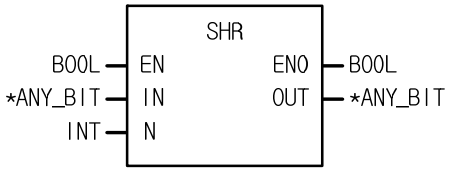
1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0

(N) : 3 ↓ (ROL)

출력(OUT) : OUT_VALUE(WORD) = 16#6660

0	1	1	0	0	1	1	0	0	1	1	0	0	0	0	0

SHR	적용 기종	발생플래그
오른쪽으로 이동 (Shift Right)	XGI, XGR, XEC	-

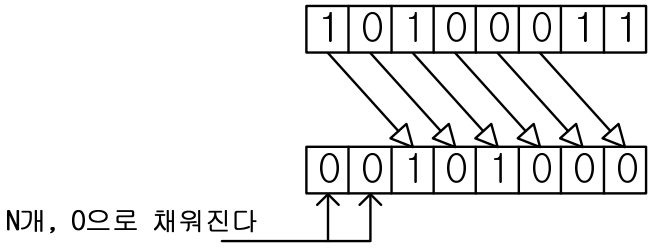
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 이동될 비트 열 N : 이동할 비트 수 **출력** ENO : EN 값이 그대로 출력 OUT : 이동된 값

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*ANY_BIT: ANY_BIT 중 BOOL 제외

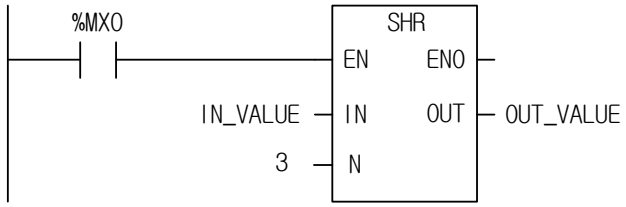
■ 기능

1. 입력 IN을 N 비트 수만큼 오른쪽으로 이동합니다.
2. 입력 IN의 맨 왼쪽에 있는 N개 비트는 0으로 채워집니다.



■ 프로그램 예

1. LD

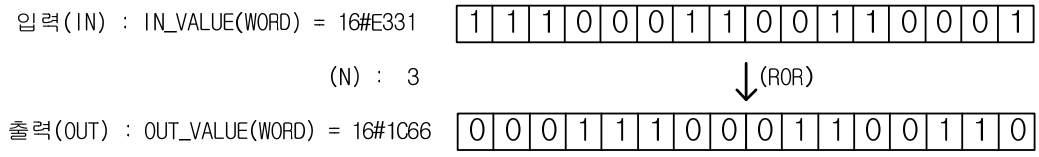


2. ST

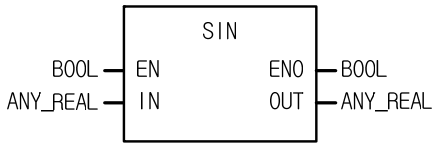
```
OUT_VALUE := SHR(EN:=%MX0, IN:=IN_VALUE, N:=3);
```

(1) 실행조건(%MX0)이 On 하면 SHR(오른쪽으로 이동) 펑션이 실행됩니다.

(2) 입력변수로 설정된 데이터 비트가 우로 3 비트 이동하여, 출력변수로 선언된 OUT_VALUE 에 출력됩니다.



SIN	적용 기종	발생플래그
Sine 연산	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : Sine 연산의 각도 입력 값(Radian) **출력** ENO : EN값이 그대로 출력 OUT : Sine 연산결과 값 IN, OUT은 같은 데이터 타입이어야 함.

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

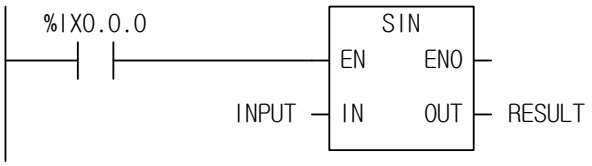
■ 기능

1. IN의 Sine값을 구해 OUT으로 출력시킵니다.

   OUT = SIN (IN)

■ 프로그램 예

1. LD



2. ST

```
RESULT := SIN(EN:=IX0.0.0, IN:=INPUT);
```

(1) 실행조건(%IX0.0.0)이 On 하면 SIN(Sine 연산) 펑션이 실행됩니다.

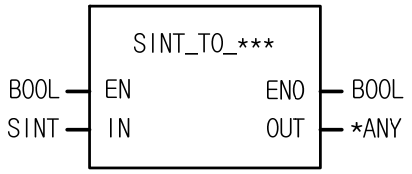
(2) INPUT 으로 선언된 입력변수의 값이 1.0471 .... ($\pi/3$ rad = 60°) 일 때 출력변수로 선언된 RESULT 는 0.8660 .... ($\sqrt{3}/2$)이 출력됩니다. $SIN(\pi/3) = \sqrt{3}/2 = 0.8660$

입력(IN) : INPUT(REAL) = 1.0471

↓(SIN)

출력(OUT) : RESULT(REAL) = 8.65976572E-01

SINT_TO_***	적용 기종	발생플래그
SINT 타입 변환	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 Short Integer 값 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 타입 변환된 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT	○	○	○	○	○		○	○	○	○	○	○	○	○	○					○

*ANY: ANY 타입 중 SINT, TIME, DATE, TOD, DT 제외

■ 기능

1. IN 을 타입 변환해서 OUT 으로 출력시킵니다.

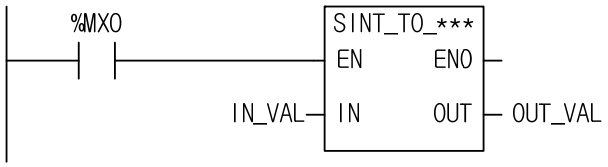
펑 션	출력 타입	동작 설명
SINT_TO_INT	INT	INT 타입으로 정상 변환합니다.
SINT_TO_DINT	DINT	DINT 타입으로 정상 변환합니다.
SINT_TO_LINT	LINT	LINT 타입으로 정상 변환합니다.
SINT_TO_USINT	USINT	입력이 0 ~ 127 일 경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
SINT_TO_UINT	UINT	입력이 0 ~ 127 일 경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
SINT_TO_UDINT	UDINT	입력이 0 ~ 127 일 경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
SINT_TO_ULINT	ULINT	입력이 0 ~ 127 일 경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
SINT_TO_BOOL	BOOL	하위 1 비트를 취해 BOOL 타입으로 변환합니다.
SINT_TO_BYTE	BYTE	내부 비트 배열의 변화 없이 BYTE 타입으로 변환합니다.
SINT_TO_WORD	WORD	상위 비트들을 0 으로 채운 WORD 타입으로 변환합니다.
SINT_TO_DWORD	DWORD	상위 비트들을 0 으로 채운 DWORD 타입으로 변환합니다.
SINT_TO_LWORD	LWORD	상위 비트들을 0 으로 채운 LWORD 타입으로 변환합니다.
SINT_TO_REAL	REAL	SINT 를 REAL 타입으로 정상 변환합니다.
SINT_TO_LREAL	LREAL	SINT 를 LREAL 타입으로 정상 변환합니다.
SINT_TO_STRING	STRING	SINT 를 STRING 타입으로 정상 변환합니다.

■ 플래그

플래그	설명
_ERR	변환에러 발생시 _ERR, _LER 플래그가 셋(Set)됩니다. 에러 발생시 출력 타입의 비트 수만큼 하위 비트를 취해 내부 비트 배열의 변환 없이 출력 시킵니다.

■ 프로그램 예

1. LD



2. ST

ST 언어는 SINT_TO_***는 지원하지 않습니다.
SINT_TO_BYTE 인 경우

```
OUT_VAL := SINT_TO_BYTE(EN:=%MX0, IN:=IN_VAL);
```

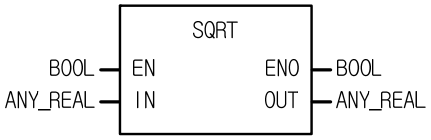
(1) 입력조건(%MX0)이 On 하면 SINT_TO_*** 펑션이 실행됩니다.
(2) 입력변수로 선언된 IN_VAL(SINT 타입)=64(2#0100_0000)이면, OUT_VAL(BYTE 타입)=16#40(2#0100_0000)가 됩니다.

입력(IN) : IN_VAL(SINT) = 64(16#40) | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

↓(SINT_TO_BYTE)

출력(OUT) : OUT_VAL(BYTE) = 16#64(16#64) | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

SQRT	적용 기종	발생플래그
제곱근 연산	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 제곱근 연산의 입력 값 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 제곱근 값 IN, OUT은 같은 데이터 타입이어야 함.

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

■ 기능

1. IN의 제곱근 값을 구해 OUT으로 출력시킵니다.

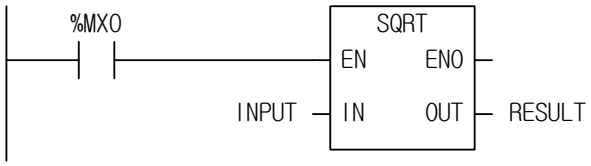
$OUT = \sqrt{IN}$

■ 플래그

플래그	설명
_ERR	IN의 값이 음수일 때 _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

1. LD



2. ST

```
RESULT := SQRT(EN:=%MX0, IN:=INPUT);
```

(1) 실행조건(%MX0)이 On 하면 SQRT(제곱근 연산) 펑션이 실행됩니다.

(2) INPUT 으로 선언된 입력변수의 값이 9.0 일 때 출력변수로 선언된 RESULT 는 3.0 이 출력됩니다.

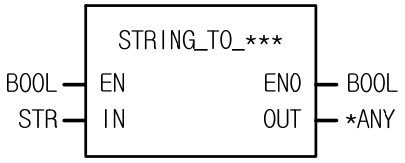
$\sqrt{9.0} = 3.0$

입력(IN) : INPUT(REAL) = 9.0

↓ (SQRT)

출력(OUT) : RESULT(REAL) = 3.0

STRING_TO_***	적용 기종	발생플래그
STRING 타입 변환	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 문자열 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 타입 변환된 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ANY: ANY 타입 중 STRING 제외

■ 기능

1. IN을 타입 변환해서 OUT으로 출력시킵니다.

펑 션	출력 타입	동작 설명
STRING_TO_SINT	SINT	STRING을 SINT 타입으로 변환합니다.
STRING_TO_INT	INT	STRING을 INT 타입으로 변환합니다.
STRING_TO_DINT	DINT	STRING을 DINT 타입으로 변환합니다.
STRING_TO_LINT	LINT	STRING을 LINT 타입으로 변환합니다.
STRING_TO_USINT	USINT	STRING을 USINT 타입으로 변환합니다.
STRING_TO_UINT	UINT	STRING을 UINT 타입으로 변환합니다.
STRING_TO_UDINT	UDINT	STRING을 UDINT 타입으로 변환합니다.
STRING_TO_ULINT	ULINT	STRING을 ULINT 타입으로 변환합니다.
STRING_TO_BOOL	BOOL	STRING을 BOOL 타입으로 변환합니다.
STRING_TO_BYTE	BYTE	STRING을 BYTE 타입으로 변환합니다.
STRING_TO_WORD	WORD	STRING을 WORD 타입으로 변환합니다.
STRING_TO_DWORD	DWORD	STRING을 DWORD 타입으로 변환합니다.
STRING_TO_LWORD	LWORD	STRING을 LWORD 타입으로 변환합니다.
STRING_TO_REAL	REAL	STRING을 REAL 타입으로 변환합니다.
STRING_TO_LREAL	LREAL	STRING을 LREAL 타입으로 변환합니다.
STRING_TO_DT	DT	STRING을 DT 타입으로 변환합니다.
STRING_TO_DATE	DATE	STRING을 DATE 타입으로 변환합니다.

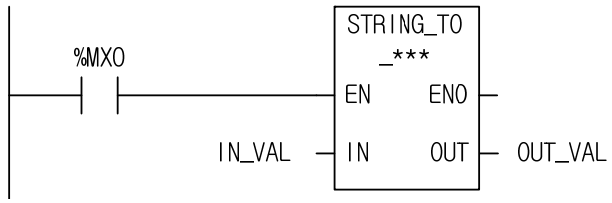
펑 션	출력 타입	동작 설명
STRING_TO_TOD	TOD	STRING을 TOD 타입으로 변환합니다.
STRING_TO_TIME	TIME	STRING을 TIME 타입으로 변환합니다.

■ 플래그

플래그	설명
_ERR	입력 문자 패턴이 출력 데이터 타입과 맞지 않을 때 _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

1. LD



2. ST

ST 언어는 STRING_TO_***는 지원하지 않습니다.
STRING_TO_REAL 인 경우

```
OUT_VAL := STRING_TO_REAL(EN:=%MX0, IN:=IN_VAL);
```

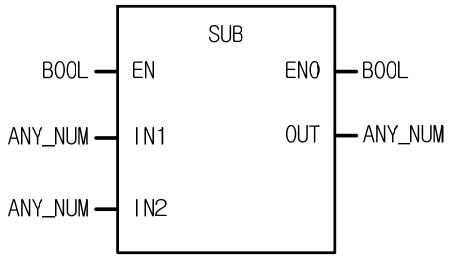
(1) 입력조건(%MX0)이 On 하면 STRING_TO_*** 펑션이 실행됩니다.
(2) 입력변수로 선언된 IN_VAL(STRING 타입) = '-1.34E12' 면, 출력변수로 선언된 OUT_VAL(REAL = -1.34E12)가 됩니다.

입력(IN) : IN_VAL(STRING) = '-1.34E12'

↓ (STRING_TO_REAL)

출력(OUT) : OUT_VAL(REAL) = -1.34E12

SUB	적용 기종	발생플래그
빼기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 피감수 IN2 : 감수 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 뺀 결과 값 IN1, IN2, OUT에 연결되는 변수는 모두 같은 데이터 타입이어야 함.

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1						○	○	○	○	○	○	○	○	○	○					
	IN2						○	○	○	○	○	○	○	○	○	○					
	OUT						○	○	○	○	○	○	○	○	○	○					

■ 기능

1. IN1에서 IN2를 빼서 OUT으로 출력시킵니다.

   OUT = IN1 − IN2

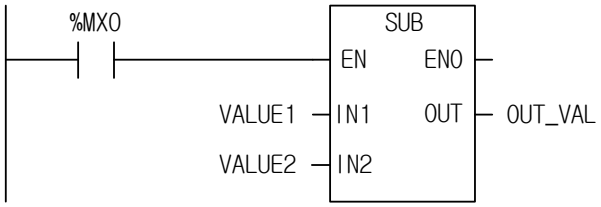
■ 플래그

플래그	설명
_ERR	출력 값이 해당 데이터 타입의 범위를 벗어날 경우 _ERR, _LER 플래그가 셋(Set)됩니다.

☆ REAL, LREAL 타입 연산에서는 IN1 에서 IN8 로 순차적으로 연산을 하기 때문에 중간 연산과정에서 이미 REAL, LREAL 의 최대값이나 최소값을 넘게 되면 _ERR, _LER 플래그가 셋(Set)되고 결과값은 무한대 값 또는 비정상적인 값 (1.#INF000000000000e+000, 1.#SNAN00000000000e+000, 1.#QNAN00000000000e+000)으로 표시됩니다.

■ 프로그램 예

1. LD

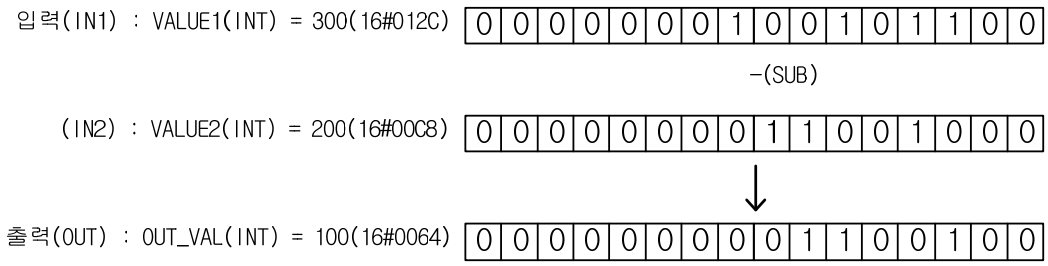


2. ST

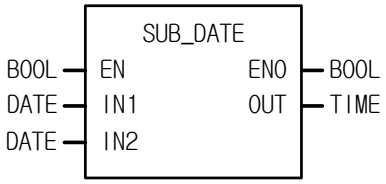
```
OUT_VAL := SUB(EN:=%MX0, IN1:=VALUE1, IN2:=VALUE2);
```

(1) 실행조건(%MX0)이 On 하면 SUB(빼기) 펑션이 실행됩니다.

(2) 입력변수 값이 VALUE1 = 300, VALUE2 = 200 이면, 출력변수로 설정한 OUT_VAL 는 연산을 실시한 결과 (300-200=100)가 출력됩니다.



SUB_DATE	적용 기종	발생플래그
날짜 빼기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 기준 날짜 IN2 : 뺄 날짜 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 두 날짜간의 차이를 시간으로 출력

■ 기능

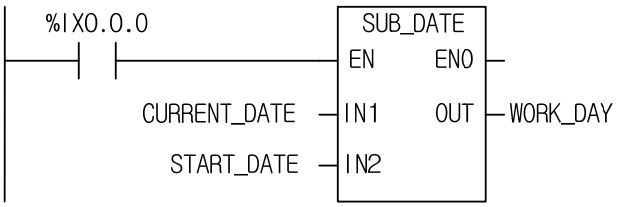
1. IN1(기준 날짜)에서 IN2(특정 날짜)를 빼서 날짜 차이를 OUT 으로 출력시킵니다.

■ 플래그

플래그	설명
_ERR	출력 값이 TIME 데이터 타입의 범위를 벗어날 경우, _ERR, _LER 플래그가 셋(Set)됩니다. 날짜 차이가 TIME 데이터 타입의 범위 T#49D17H2M47S295MS 를 넘거나 날짜 연산의 결과가 음수가 되면 에러가 됩니다.

■ 프로그램 예

1. LD



2. ST

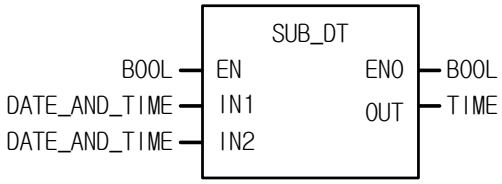
```
WORK_DAY := SUB_DATE(EN:=%IX0.0.0, IN1:=CURRENT_DATE, IN2:=START_DATE);
```

(1) 실행조건(%IX0.0.0)이 On 하면 SUB_DATE(날짜 빼기) 펑션이 실행됩니다.

(2) 입력변수로 선언한 현재의 날짜 CURRENT_DATE 가 D#1995-12-15 이고 시스템이 가동을 시작한 날짜 START_DATE 가 D#1995-11-1 이면, 출력변수로 선언한 조업한 날짜 WORK_DAY 에는 T#44D 가 출력됩니다.

```
입력(IN1) : CURRENT_DATE(DATE) = D#1995-12-15
                                 (SUB_DATE)
     (IN2) : START_DATE(DATE)  = D#1995-11-1
                                    ↓
출력(OUT) : WORK_DAY(TIME)     =    T#44D
```

SUB_DT	적용 기종	발생플래그
시간과 날짜 빼기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 기준 날짜와 시각 IN2 : 뺄 날짜와 시각 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 뺀 결과 시간

■ 기능

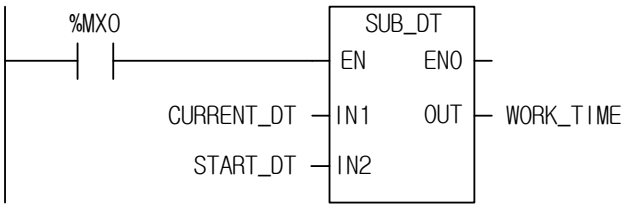
1. IN1(기준 날짜와 시각)에서 IN2(특정 날짜와 시각)를 빼서 시간 차이를 OUT 으로 출력시킵니다.

■ 플래그

플래그	설명
_ERR	출력 값이 TIME 데이터 타입의 범위를 벗어날 경우, _ERR, _LER 플래그가 셋(Set)됩니다. 날짜와 시각 빼기 연산의 결과가 음수가 되면 에러가 됩니다.

■ 프로그램 예

1. LD



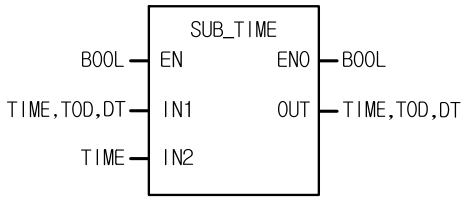
2. ST

```
WORK_TIME := SUB_DT(EN:=%MX0, IN1:=CURRNET_DT, IN2:=START_DT);
```

(1) 실행조건(%MX0)이 On 하면 SUB_DT(시간과 날짜빼기) 펑션이 실행됩니다.
(2) 입력변수로 선언한 현재의 날짜와 시각 CURRENT_DT 가 DT#1995-12-15-14:30:00 이고 조업이 재개된 날짜와 시각 START_DT 가 DT#1995-12-13-12:00:00 이면, 출력변수로 선언된 연속 조업한 시간 WORK_TIME 에는 T#2D2H30M 가 출력됩니다.

```
입력(IN1) : CURRENT_DT(DT) =  DT#1995-12-15-14:30:00
                                          (SUB_DATE)
    (IN2) : START_DT(DT)  =    DT#1995-12-13-12:00:00
                                          ↓
 출력(OUT) : WORK_TIME(TIME) =       T#2D2H30M
```

SUB_TIME	적용 기종	발생플래그
시간과 빼기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 기준 시각 또는 시간 IN2 : 뺄 시간 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 뺀 결과 시각 또는 시간 OUT의 타입은 입력 IN1의 타입을 따름. 즉 IN1의 타입이 TIME이면, 출력 OUT의 타입도 TIME임.

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1																○		○	○	
	OUT																○		○	○	

■ 기능

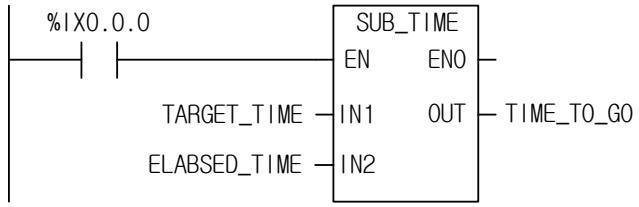
1. IN1이 TIME일 경우에는 시간에서 시간을 빼서 차이 시간을 출력시킵니다.
2. IN1이 TIME_OF_DAY일 경우에는 기준시각에서 시간을 빼서 하루 중의 시각을 출력시킵니다.
3. IN1이 DATE_AND_TIME일 경우에는 기준이 되는 날짜와 시각에서 시간을 빼서 날짜와 시각을 출력시킵니다.

■ 플래그

플래그	설명
_ERR	출력 값이 해당 데이터 타입의 범위를 벗어날 경우, _ERR, _LER 플래그가 셋(Set)됩니다. 시간에서 시간을 뺀 결과가 음수가 되거나 시각(TOD)에서 시간을 뺀 결과가 음수가 되면 에러가 됩니다.

■ 프로그램 예

1. LD



2. ST

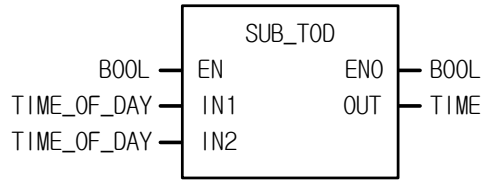
```
TIME_TO_GO := SUB_TIME(EN:=%IX0.0.0, IN1:=TARGET_TIME, IN2:=ELABSED_TIME);
```

(1) 실행조건(%IX0.0.0)이 On 하면 SUB_TIME(시간빼기) 펑션이 실행됩니다.

(2) 입력변수로 선언된 전체 작업시간 TARGET_TIME 이 T#2H30M 이고, 경과된 시간 ELAPSED_TIME 이 T#1H10M30S300MS 면 출력변수로 선언된 남아 있는 작업시간 TIME_TO_GO 에는 T#1H19M29S700MS 가 출력됩니다.

```
입력(IN1) : TARGET_TIME(TIME) =    T#2H30M
                                        (SUB_DATE)
    (IN2) : ELAPSED_TIME(TIME)= T#1H10M30S300MS
                                        ↓
출력(OUT) : TIME_TO_GO(TIME) =   T#1H19M29S700MS
```

SUB_TOD	적용 기종	발생플래그
시각에서 시각빼기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 기준 시각 IN2 : 뺄 시각 **출력** ENO : 에러 없이 실행되면 1 출력 OUT : 뺀 결과 시간

■ 기능

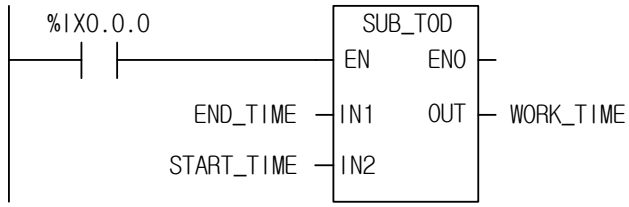
1. IN1(기준 시각)에서 IN2(특정 시각)를 빼서 시간 차이를 OUT 으로 출력시킵니다.

■ 플래그

플래그	설명
_ERR	시각에서 시각을 뺀 결과가 음수가 되면 에러가 됩니다.

■ 프로그램 예

1. LD



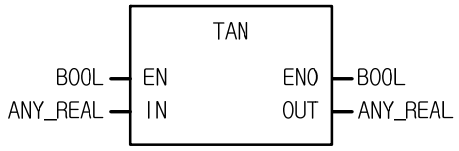
2. ST

```
WORK_TIME := SUB_TOD(EN:=%IX0.0.0, IN1:=END_TIME, IN2:=START_TIME);
```

(1) 실행조건(%IX0.0.0)이 On 하면 SUB_TOD(시각에서 시각빼기) 펑션이 실행됩니다.
(2) 입력변수로 선언된 END_TIME 이 TOD#14:20:30.500 이고 작업을 시작한 START_TIME 이 TOD#12:00:00 이면, 출력변수로 선언된 작업에 소요된 시간 WORK_TIME 에는 T#2H20M30S500MS 가 출력됩니다.

```
입력(IN1) : END_TIME(TOD) =    TOD#14:20:30.500
                                        (SUB_TOD)
   (IN2) : START_TIME(TOD)  =   TOD#12:00:00
                                        ↓
출력(OUT) : WORK_TIME(TIME) =  T#2H20M30S500MS
```

TAN	적용 기종	발생플래그
Tangent 연산	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : Tangent 각도입력 값(Radian) **출력** ENO : EN값이 그대로 출력 OUT : Tangent 연산결과 값 IN, OUT은 같은 데이터 타입이어야 함.

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

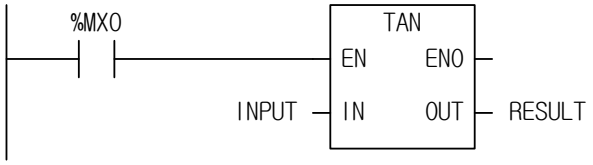
■ 기능

1. IN의 Tangent 값을 구해 OUT으로 출력시킵니다.

   OUT = TAN(IN)

■ 프로그램 예

1. LD



2. ST

```
RESULT := TAN(EN:=%MX0, IN:=INPUT);
```

(1) 실행조건(%MX0)이 On 하면 TAN(Tangent 연산) 펑션이 실행합니다.

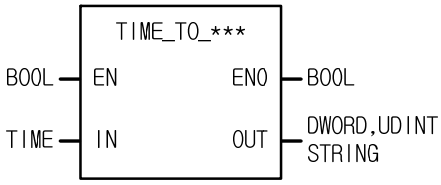
(2) INPUT 으로 선언된 입력변수의 값이 0.7853 .... (π/4 rad = 45°)일 때 출력 변수로 선언된 RESULT 는 1.0000 입니다.

TAN(π/4) = 1

입력(IN) : INPUT(REAL) = 0.7853

출력(OUT) : RESULT(REAL) = 9.99803722E-01

TIME_TO_***	적용 기종	발생플래그
TIME 타입변환	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 시간 데이터 **출력** ENO : EN 값이 그대로 출력 OUT : 타입 변환된 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT				○								○								○

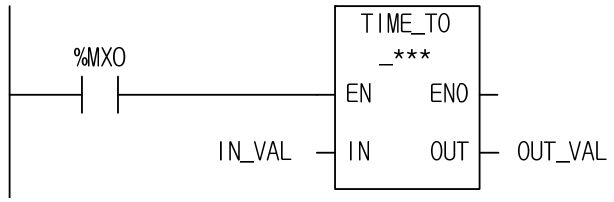
■ 기능

1. IN 을 타입 변환해서 OUT 으로 출력시킵니다.

펑 션	출력 타입	동작 설명
TIME_TO_UDINT	UDINT	TIME 를 UDINT 타입으로 변환합니다. 데이터(내부 비트 배열 상태)의 변화 없이 데이터 타입만 변환합니다.
TIME_TO_DWORD	DWORD	TIME 를 DWORD 타입으로 변환합니다. 데이터(내부 비트 배열 상태)의 변화 없이 데이터 타입만 변환합니다.
TIME_TO_STRING	STRING	TIME 를 STRING 타입으로 변환합니다.

■ 프로그램 예

1. LD



2. ST

ST 언어는 TIME_TO_***는 지원하지 않습니다.
TIME_TO_UDINT 인 경우

```
OUT_VAL := TIME_TO_UDINT(EN:=%MX0, IN:=IN_VAL);
```

(1) 입력조건(%MX0)이 On 하면 TIME_TO_*** 펑션이 실행됩니다.
(2) 입력변수로 선언된 IN_VAL(TIME 타입) = T#120MS 이면, 출력변수로 선언된 OUT_VAL(UDINT 타입) = 120 이 됩니다.

입력(IN) : IN_VAL(TIME) = T#120MS(16#78)

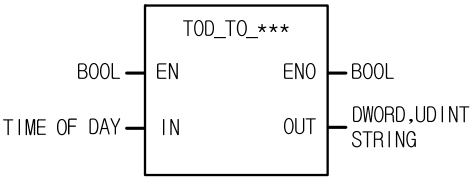
0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0

↓ (TIME_TO_UDINT)

출력(OUT) : OUT_VAL(UDINT) = 120(16#78)

0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0

TOD_TO_***	적용 기종	발생플래그
TOD 타입변환	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 하루 중 시각 데이터 **출력** ENO : EN 값이 그대로 출력 OUT : 타입 변환된 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT				○								○								○

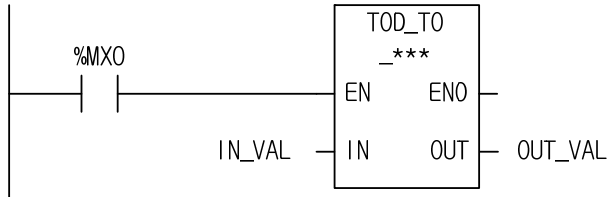
■ 기능

1. IN을 타입 변환해서 OUT으로 출력시킵니다.

펑 션	출력 타입	동작 설명
TOD_TO_UDINT	UDINT	TOD를 UDINT 타입으로 변환합니다. 데이터(내부 비트 배열 상태)의 변화 없이 데이터 타입만 변환합니다.
TOD_TO_DWORD	DWORD	TOD를 DWORD 타입으로 변환합니다. 데이터(내부 비트 배열 상태)의 변화 없이 데이터 타입만 변환합니다.
TOD_TO_STRING	STRING	TOD를 STRING 타입으로 변환합니다.

■ 프로그램 예

1. LD



2. ST

ST 언어는 TIME_TO_***는 지원하지 않습니다.
TIME_TO_UDINT 인 경우

```
OUT_VAL := TOD_TO_STRING(EN:=%MX0, IN:=IN_VAL);
```

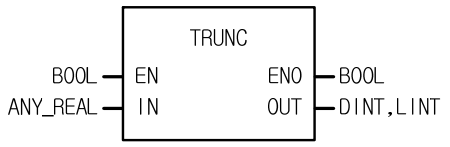
(1) 입력조건(%MX0)이 On 하면 TOD_TO_*** 펑션이 실행됩니다.
(2) 입력변수로 선언된 IN_VAL(TOD 타입) = TOD#12:00:00 이면, 출력변수로 선언된 OUT_VAL(STRING 타입) = 'TOD#12:00:00' 이 됩니다.

입력(IN) : IN_VAL(TOD) = TOD#12:00:00

↓ (TOD_TO_STRING)

출력(OUT) : OUT_VAL(STRING) = 'TOD#12:00:00'

TRUNC	적용 기종	발생플래그
소수점 이하 값을 버리고 정수로 변환	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 변환될 Real 값 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 정수로 변환된 값

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT								○	○											

■ 기능

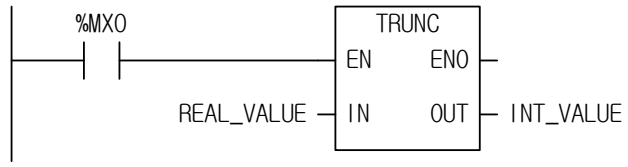
펑션	입력 타입	출력 타입	동 작 설 명
TRUNC	REAL	DINT	입력 IN으로 들어온 부동소수의 소수점 이하 값은 버리고 OUT으로 정수 값을 출력합니다.
	LREAL	LINT	

■ 플래그

플래그	설명
_ERR	변환된 값이 OUT 에 연결된 데이터 타입의 최대값보다 큰 경우 또는 OUT 에 연결된 변수가 Unsigned Integer 이고 변환된 출력 값이 음수일 때 OUT 으로 0 을 출력한 경우에는 _ERR, _LER 플래그가 셋(Set)됩니다

■ 프로그램 예

1. LD



2. ST

```
INT_VALUE:=TRUNC(EN:=%MX0, IN:=REAL_VALUE);
```

(1) 실행조건(%MX0)이 On 하면 TRUNC(소수점 이하 값을 버리고 정수 변환) 펑션이 실행합니다.

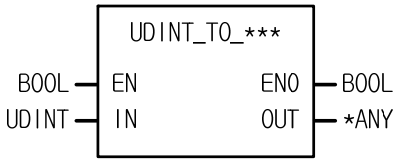
(2) 입력변수로 선언된 REAL_VALUE(REAL 타입) = 1.6 이면 INT_VALUE(INT 타입) = 1 이 됩니다.
REAL_VALUE(REAL 타입) = -1.6 이면 INT_VALUE(INT 타입) = -1 이 됩니다.

입력(IN) : REAL_VALUE(REAL) = 1.6

↓ (TRUNC)

출력(OUT) : INT_VALUE(INT) = 1

UDINT_TO_***	적용 기종	발생플래그
UDINT 타입 변환	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 Unsigned Double Integer값 **출력** ENO : EN값이 그대로 출력 OUT : 타입 변환된 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT	○	○	○	○	○	○	○	○	○	○	○		○	○	○	○		○		○

*ANY: ANY 타입 중 UDINT, DATE, DT 제외

■ 기능

1. IN을 타입 변환해서 OUT으로 출력시킵니다.

펑 션	출력 타입	동작 설명
UDINT_TO_SINT	SINT	입력이 0~127일 경우 정상 변환되나 그 외 값은 에러가 발생합니다.
UDINT_TO_INT	INT	입력이 0~32,767일 경우 정상 변환되나 그 외 값은 에러가 발생합니다.
UDINT_TO_DINT	DINT	입력이 0~2,147,483,647 일 경우 정상 변환되나 그 외 값은 에러가 발생합니다.
UDINT_TO_LINT	LINT	UDINT를 LINT 타입으로 정상 변환합니다.
UDINT_TO_USINT	USINT	입력이 0~255일 경우 정상 변환되나 그 외 값은 에러가 발생합니다.
UDINT_TO_UINT	UINT	입력이 0~65,535일 경우 정상 변환되나 그 외 값은 에러가 발생합니다.
UDINT_TO_ULINT	ULINT	UDINT를 ULINT 타입으로 정상 변환합니다.
UDINT_TO_BOOL	BOOL	하위 1비트를 취해 BOOL 타입으로 변환합니다.
UDINT_TO_BYTE	BYTE	하위 8비트를 취해 BYTE 타입으로 변환합니다.
UDINT_TO_WORD	WORD	하위 16비트를 취해 WORD 타입으로 변환됩니다.
UDINT_TO_DWORD	DWORD	내부 비트 배열의 변환 없이 DWORD 타입으로 변환합니다.
UDINT_TO_LWORD	LWORD	상위 비트를 0으로 채운 LWORD 타입으로 변환합니다.
UDINT_TO_REAL	REAL	UDINT를 REAL 타입으로 변환합니다. 변환 중 정밀도에 따른 오차가 발생할 수 있습니다.
UDINT_TO_LREAL	LREAL	UDINT를 LREAL 타입으로 변환합니다. 변환 중 정밀도에 따른 오차가 발생할 수 있습니다.

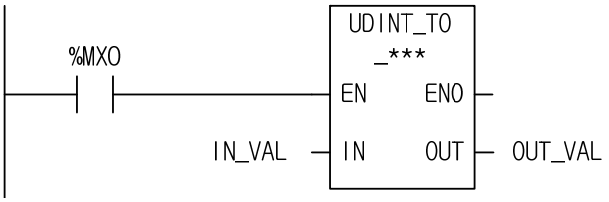
펑 션	출력 타입	동작 설명
UDINT_TO_TOD	TOD	내부 비트 배열의 변환 없이 TOD 타입으로 변환합니다. 단, TOD범위 (TOD#23:59:59.999)를 벗어난 값 입력시 에러를 발생하고 TOD 범위 내에서 순환하여 변환합니다.
UDINT_TO_TIME	TIME	내부 비트 배열의 변환 없이 TIME 타입으로 변환합니다.
UDINT_TO_STRING	STRING	UDINT 를 STRING 타입으로 변환합니다.

■ 플래그

플래그	설명
_ERR	변환에러 발생시 _ERR, _LER 플래그가 셋(Set)됩니다. 에러 발생시 출력 타입의 비트 수만큼 하위 비트를 취해 내부 비트 배열의 변환 없이 출력시킵니다.

■ 프로그램 예

1. LD



2. ST

ST 언어는 UDINT_TO_***는 지원하지 않습니다.
UDINT_TO_TIME 인 경우

```
OUT_VAL := UDINT_TO_TIME(EN:=%MX0, IN:=IN_VAL);
```

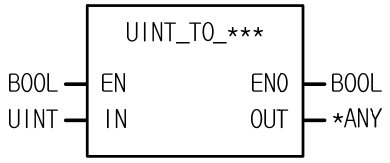
(1) 입력조건(%MX0)이 On 하면 UDINT_TO_*** 펑션이 실행됩니다.
(2) 입력변수로 선언된 IN_VAL(UDINT 타입) = 123 이면, 출력변수로 선언된 OUT_VAL(TIME 타입) = T#123MS 가 됩니다.

입력(IN) : IN_VAL(UDINT) = 123

↓

출력(OUT) : OUT_VAL(TIME) = T#123MS

UINT_TO_***	적용 기종	발생플래그
UINT 타입 변환	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 IN : 타입 변환할 Unsigned Integer 값 **출력** ENO : EN 값이 그대로 출력 OUT : 타입 변환된 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT	○	○	○	○	○	○	○	○	○	○		○	○	○	○		○			○

*ANY: ANY 타입 중 UINT, TIME, TOD, DT 제외

■ 기능

1. IN 을 타입 변환해서 OUT 으로 출력시킵니다.

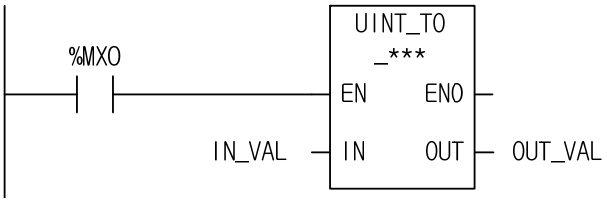
펑 션	출력타입	동작 설명
UINT_TO_SINT	SINT	입력이 0~127 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
UINT_TO_INT	INT	입력이 0~32,767 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
UINT_TO_DINT	DINT	UINT 를 UDINT 타입으로 정상 변환합니다.
UINT_TO_LINT	LINT	UINT 를 ULINT 타입으로 정상 변환합니다.
UINT_TO_USINT	USINT	입력이 0~255 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
UINT_TO_UDINT	UDINT	UINT 를 UDINT 타입으로 정상 변환합니다.
UINT_TO_ULINT	ULINT	UINT 를 ULINT 타입으로 정상 변환합니다.
UINT_TO_BOOL	BOOL	하위 1 비트를 취해 BOOL 타입으로 변환합니다.
UINT_TO_BYTE	BYTE	하위 8 비트를 취해 BYTE 타입으로 변환합니다.
UINT_TO_WORD	WORD	내부 비트 배열의 변환 없이 WORD 타입으로 변환합니다.
UINT_TO_DWORD	DWORD	상위 비트를 0 으로 채운 DWORD 타입으로 변환합니다.
UINT_TO_LWORD	LWORD	상위 비트를 0 으로 채운 LWORD 타입으로 변환합니다.
UINT_TO_REAL	REAL	UINT 를 REAL 타입으로 변환합니다.
UINT_TO_LREAL	LREAL	UINT 를 LREAL 타입으로 변환합니다.
UINT_TO_DATE	DATE	내부 비트 배열의 변환 없이 DATE 타입으로 변환합니다.
UINT_TO_STRING	STRING	UINT 를 STRING 타입으로 변환합니다.

■ 플래그

플래그	설명
_ERR	변환에러 발생시 _ERR, _LER 플래그가 셋(Set)됩니다. 에러 발생시 출력 타입의 비트 수만큼 하위 비트를 취해 내부 비트 배열의 변환 없이 출력시킵니다.

■ 프로그램 예

1. LD

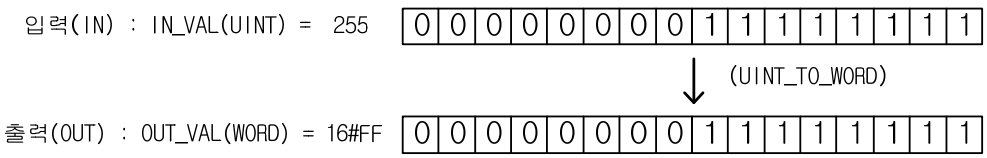


2. ST

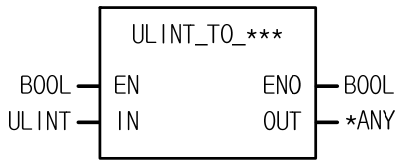
ST 언어는 UINT_TO_***는 지원하지 않습니다.
UINT_TO_WORD 인 경우

```
OUT_VAL := UINT_TO_WORD(EN:=%MX0, IN:=IN_VAL);
```

(1) 입력조건(%MX0)이 On 하면 UINT_TO_*** 펑션이 실행됩니다.
(2) 입력변수로 선언된 IN_VAL(UINT 타입) = 255(2#0000_0000_1111_1111)이면, 출력변수로 선언된 OUT_VAL(WORD 타입) = 2#0000_0000_1111_1111 이 됩니다.



ULINT_TO_***	적용 기종	발생플래그
ULINT 타입 변환	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 IN : 타입 변환할 Unsigned Long Integer 값 **출력** ENO : EN 값이 그대로 출력 OUT : 타입 변환된 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT	○	○	○	○	○	○	○	○	○	○		○	○	○	○		○			○

*ANY: ANY 타입 중 UINT, TIME, TOD, DT 제외

■ 기능

1. IN 을 타입 변환해서 OUT 으로 출력시킵니다.

펑 션	출력 타입	동작 설명
ULINT_TO_SINT	SINT	입력이 0~127 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
ULINT_TO_INT	INT	입력이 0~32,767 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
ULINT_TO_DINT	DINT	입력이 0~2^{31}-1 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
ULINT_TO_LINT	LINT	입력이 0~2^{63}-1 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
ULINT_TO_USINT	USINT	입력이 0~255 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
ULINT_TO_UINT	UINT	입력이 0~65,535 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
ULINT_TO_UDINT	UDINT	입력이 0~2^{32}-1 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
ULINT_TO_BOOL	BOOL	하위 1 비트를 취해 BOOL 타입으로 변환합니다.
ULINT_TO_BYTE	BYTE	하위 8 비트를 취해 BYTE 타입으로 변환합니다.
ULINT_TO_WORD	WORD	하위 16 비트를 취해 WORD 타입으로 변환합니다.
ULINT_TO_DWORD	DWORD	하위 32 비트를 취해 DWORD 타입으로 변환합니다.
ULINT_TO_LWORD	LWORD	내부 비트 배열의 변환 없이 LWORD 타입으로 변환합니다.
ULINT_TO_REAL	REAL	ULINT 를 REAL 타입으로 변환합니다. 변환 중 정밀도에 따른 오차가 발생할 수 있습니다.
ULINT_TO_LREAL	LREAL	ULINT 를 LREAL 타입으로 변환합니다. 변환 중 정밀도에 따른 오차가 발생할 수 있습니다.

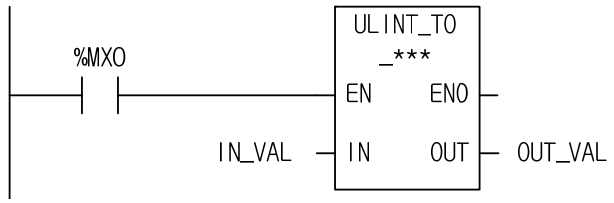
펑 션	출력 타입	동작 설명
ULINT_TO_STRING	STRING	ULINT 를 STRING 타입으로 변환합니다.

■ 플래그

플래그	설명
_ERR	변환에러 발생시 _ERR, _LER 플래그가 셋(Set)됩니다. 에러 발생시 출력 타입의 비트 수만큼 하위 비트를 취해 내부 비트 배열의 변환 없이 출력 시킵니다.

■ 프로그램 예

1. LD



2. ST

ST 언어는 ULINT_TO_***는 지원하지 않습니다.
ULINT_TO_LINT 인 경우

```
OUT_VAL := ULINT_TO_LINT(EN:=%MX0, IN:=IN_VAL);
```

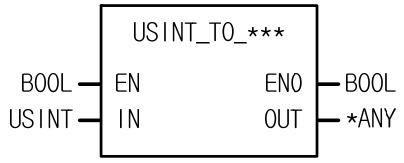
(1) 입력조건(%MX0)이 On 하면 ULINT_TO_*** 펑션이 실행됩니다.
(2) 입력변수로 선언된 IN_VAL(ULINT 타입) = 123,567,899 이면, 출력변수로 선언된 OUT_VAL(LINT 타입) = 123,567,899 가 됩니다.

입력(IN) : IN_VAL(ULINT) = 123,567,899

↓ (ULINT_TO_LINT)

출력(OUT) : OUT_VAL(LINT) = 123,567,899

USINT_TO_***	적용 기종	발생플래그
USINT 타입 변환	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 Unsigned Short Integer 값 **출력** ENO : EN 값이 그대로 출력 OUT : 타입 변환된 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT	○	○	○	○	○	○	○	○	○		○	○	○	○	○					○

*ANY : ANY 타입 중 USINT, TIME, DATE, TOD, DT 제외

■ 기능

1. IN 을 타입 변환해서 OUT 으로 출력시킵니다.

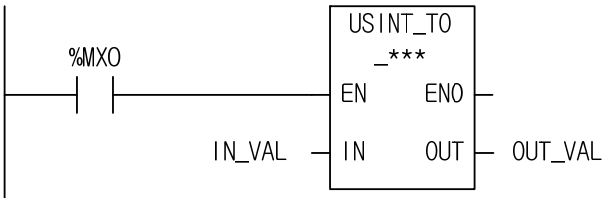
펑 션	출력 타입	동작 설명
USINT_TO_SINT	SINT	입력이 0~127 일 경우 정상 변환되나 그 외 값은 에러가 발생합니다.
USINT_TO_INT	INT	입력을 INT 타입으로 정상 변환합니다.
USINT_TO_DINT	DINT	입력을 DINT 타입으로 정상 변환합니다.
USINT_TO_LINT	LINT	입력을 LINT 타입으로 정상 변환합니다.
USINT_TO_UINT	UINT	입력을 UINT 타입으로 정상 변환합니다.
USINT_TO_UDINT	UDINT	입력을 UDINT 타입으로 정상 변환합니다.
USINT_TO_ULINT	ULINT	입력을 ULINT 타입으로 정상 변환합니다.
USINT_TO_BOOL	BOOL	하위 1 비트를 취해 BOOL 타입으로 변환합니다.
USINT_TO_BYTE	BYTE	내부 비트 배열의 변환 없이 BYTE 타입으로 변환합니다.
USINT_TO_WORD	WORD	상위 비트를 0 으로 채운 WORD 타입으로 변환합니다.
USINT_TO_DWORD	DWORD	상위 비트를 0 으로 채운 DWORD 타입으로 변환합니다.
USINT_TO_LWORD	LWORD	상위 비트를 0 으로 채운 LWORD 타입으로 변환합니다.
USINT_TO_REAL	REAL	USINT 를 REAL 타입으로 변환합니다.
USINT_TO_LREAL	LREAL	USINT 를 LREAL 타입으로 변환합니다.
USINT_TO_STRING	STRING	USINT 를 STRING 타입으로 변환합니다.

■ 플래그

플래그	설명
_ERR	변환에러 발생시 _ERR, _LER 플래그가 셋(Set)됩니다. 에러 발생시 출력 타입의 비트 수만큼 하위 비트를 취해 내부 비트 배열의 변환 없이 출력시킵니다.

■ 프로그램 예

1. LD

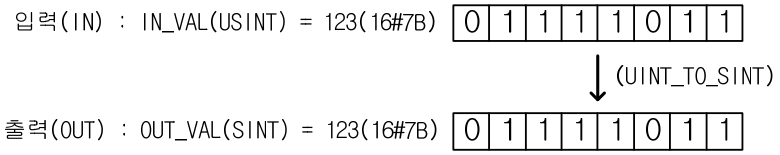


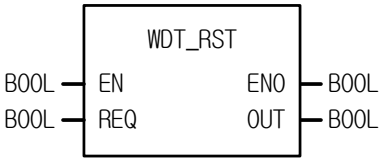
2. ST

ST 언어는 USINT_TO_***는 지원하지 않습니다.
USINT_TO_SINT 인 경우

```
OUT_VAL := USINT_TO_SINT(EN:=%MX0, IN:=IN_VAL);
```

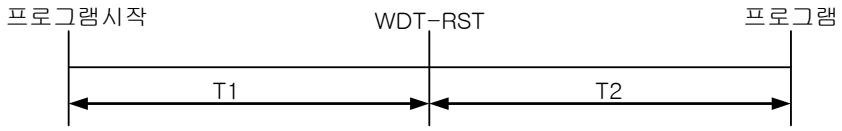
(1) 입력조건(%MX0)이 On 하면 ULINT_TO_*** 펑션이 실행됩니다.
(2) 입력변수로 선언된 IN_VAL(USINT 타입) = 123 이면, 출력변수로 선언된 OUT_VAL(SINT 타입) = 123 이 됩니다.



WDT_RST	적용 기종	발생플래그
Watch_Dog 타이머 초기화	XGI, XGR, XEC	-
펑 션	설 명	
	**입력** EN : 1일 때 펑션 실행 REQ : Watch_Dog 타이머 초기화 요구 **출력** ENO : EN 값이 그대로 출력 OUT : Watch_Dog 타이머를 초기화 후 1출력	

■ 기능

1. 프로그램 중에서 Watch-Dog Timer 를 Reset 시킵니다.
2. 프로그램에서 스캔 타임이 조건에 의해 설정된 Watch-Dog Time 시간을 넘는 경우 사용합니다.
3. 스캔 타임이 매번 설정된 스캔 Watch-Dog Time 을 넘는 경우는 스캔 워치 독 타임의 설정 값으로 변경해 주십시오.
4. 프로그램의 0 Line 부터 WDT_RST 펑션까지의 시간(T1)과 WDT_RST 펑션부터 프로그램 끝까지의 시간(T2) 어느 쪽도 스캔 워치 독 타임 설정 값을 넘지 않도록 설정해야 합니다.

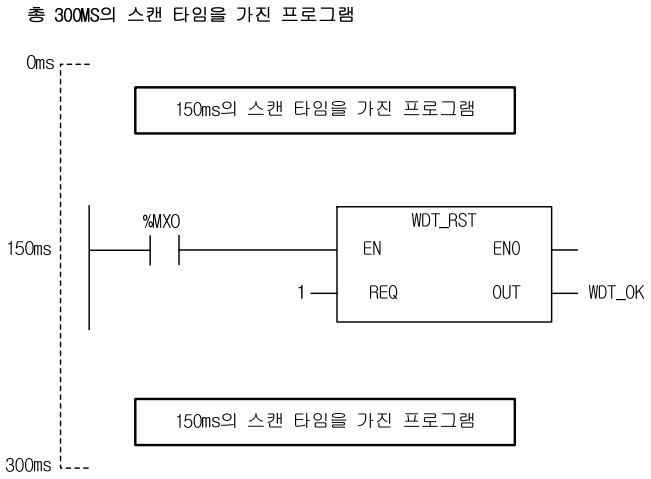


5. 프로그램의 WDT_RST 펑션은 1스캔 중에 여러 번 사용 가능합니다.

■ 프로그램 예

스캔 워치 독 타임이 200ms 로 설정된 프로그램에서 실행조건에 따라 프로그램의 수행 시간이 300ms 가 될 때의 프로그램.

1. LD



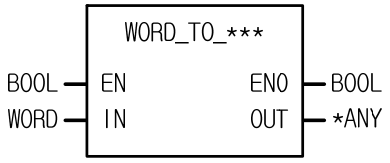
2. ST

```
WDT_OK := WDT_RST(EN:=%MX0, REQ:=%MX0);
```

(1) 실행조건(%MX0)이 On 하면 WDT_RST(Watch Dog 타이머 초기화) 펑션이 실행합니다.

(2) WDT_RST 펑션이 실행되면, 프로그램의 실행조건에 따라 스캔 타임이 300ms 까지 늘어나는 프로그램을 스캔 워치 독 타임 이내(200ms)으로 맞출 수 있습니다.

WORD_TO_***	적용 기종	발생플래그
WORD 타입변환	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 비트 열(16비트) **출력** ENO : EN값이 그대로 출력 OUT : 타입 변환된 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT	○	○		○	○	○	○	○	○	○	○	○	○				○			○

*ANY: ANY 타입 중 WORD, REAL, LREAL, TIME, TOD, DT 제외

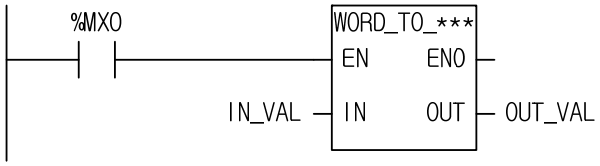
■ 기능

1. IN을 타입 변환해서 OUT으로 출력시킵니다.

펑 션	출력 타입	동작 설명
WORD _TO_SINT	SINT	하위 8비트를 취해 SINT 타입으로 변환합니다.
WORD _TO_INT	INT	내부 비트 배열의 변환 없이 INT 타입으로 변환합니다.
WORD _TO_DINT	DINT	상위 비트를 0으로 채운 DINT 타입으로 변환합니다.
WORD _TO_LINT	LINT	상위 비트를 0으로 채운 LINT 타입으로 변환합니다.
WORD _TO_USINT	USINT	하위 8비트를 취해 SINT 타입으로 변환합니다.
WORD _TO_UINT	UINT	내부 비트 배열의 변환 없이 INT 타입으로 변환합니다.
WORD _TO_UDINT	UDINT	상위 비트를 0으로 채운 DINT 타입으로 변환합니다.
WORD _TO_ULINT	ULINT	상위 비트를 0으로 채운 LINT 타입으로 변환합니다.
WORD _TO_BOOL	BOOL	하위 1비트를 취해 BOOL 타입으로 변환합니다.
WORD _TO_BYTE	BYTE	하위 8비트를 취해 SINT 타입으로 변환합니다.
WORD _TO_DWORD	DWORD	상위 비트를 0으로 채운 DWORD 타입으로 변환합니다.
WORD _TO_LWORD	LWORD	상위 비트를 0으로 채운 LWORD 타입으로 변환합니다.
WORD _TO_DATE	DATE	내부 비트 배열의 변환 없이 DATE 타입으로 변환합니다.
WORD _TO_STRING	STRING	WORD를 STRING 타입으로 변환합니다.

■ 프로그램 예

1. LD



2. ST

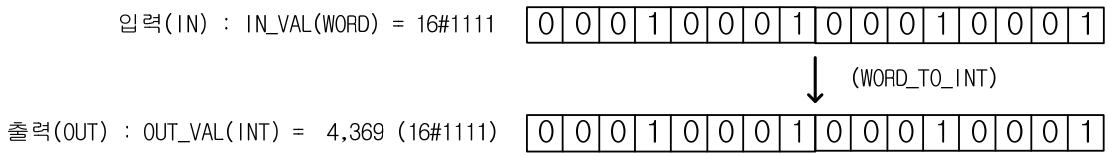
ST 언어는 WORD_TO_***는 지원하지 않습니다.

WORD_TO_INT 인 경우

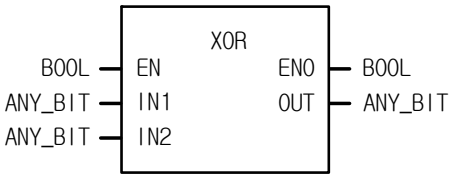
```
OUT_VAL := WORD_TO_INT(EN:=%MX0, IN:=IN_VAL);
```

(1) 입력조건(%MX0)이 On 하면 WORD_TO_*** 펑션이 실행됩니다.

(2) 입력변수로 선언된 IN_VAL(WORD 타입) = 2#0001_0001_0001_0001 이면, 출력변수로 선언된 OUT_VAL(INT 타입) = 4,096 + 256 + 16 + 1 = 4,369 가 됩니다.



XOR	적용 기종	발생플래그
배타적 논리합	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : XOR될 값 IN2 : XOR될 값 입력 8개까지 확장 가능 **출력** ENO : EN값이 그대로 출력 OUT : XOR 된 값 IN1, IN2, OUT은 모두 같은 타입이어야 함.

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN	○	○	○	○	○															
	OUT	○	○	○	○	○															

■ 기능

1. IN1을 IN2와 비트별로 XOR해서 OUT으로 출력시킵니다.

   IN1 1111 0000

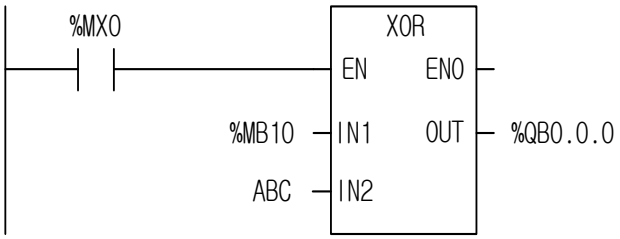
   XOR

   IN2 1010 1010

   OUT 0101 1010

■ 프로그램 예

1. LD



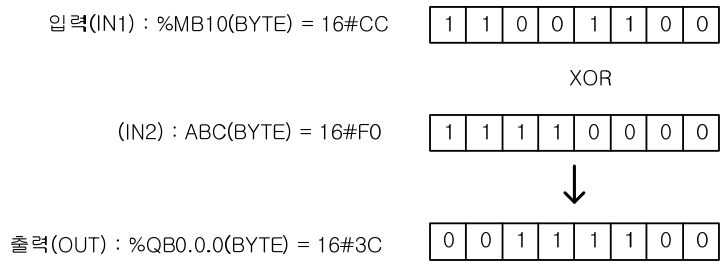
2. ST

ST 언어는 XOR 는 지원하지 않습니다.
XOR2_BYTE 인 경우

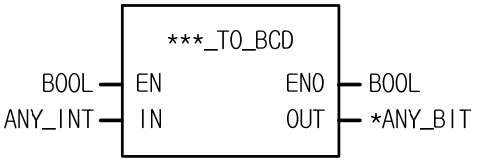
```
%QB0.0.0 := XOR2_BYTE(EN:=%MX0, IN1:=%MB10, IN2:=ABC);
```

(1) 실행조건(%MX0)이 On 하면 XOR(배타적 논리합) 펑션이 실행됩니다.

(2) 입력변수 %MB10 = 1100_1100, ABC = 1111_0000 이면, 두 값을 XOR 시킨 결과가 출력변수 %QB0.0.0 = 0011_1100 로 출력됩니다.



***_TO_BCD	적용 기종	발생플래그
ANY 타입을 BCD 타입 변환	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : BCD 형태의 데이터를 갖고 있는 ANY_BIT 입력 **출력** ENO : EN 값이 그대로 출력 OUT : 타입 변환된 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN						○	○	○	○	○	○	○	○							
	OUT		○	○	○	○															

*ANY_BIT: ANY_BIT 중 BOOL 타입 제외

■ 기능

1. IN 을 타입 변환해서 OUT 으로 출력시킵니다.

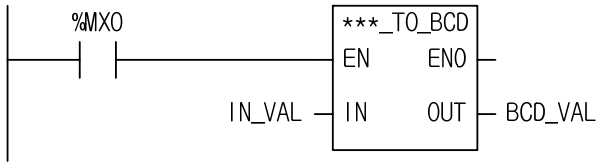
펑션	입력 타입	출력 타입	동작 설명
SINT_TO_BCD_BYTE	SINT	BYTE	ANY 타입 BCD 타입으로 변환합니다. 입력이 BCD 값일 경우에만 정상적으로 변환됩니다. (입력 데이터 타입이 WORD 일 경우 0~16#9999 값만 정상적으로 변환됩니다.)
INT_TO_BCD_WORD	INT	WORD	
DINT_TO_BCD_DWORD	DINT	DWORD	
LINT_TO_BCD_LWORD	LINT	LWORD	
USINT_TO_BCD_BYTE	USINT	BYTE	
UINT_TO_BCD_WORD	UINT	WORD	
UDINT_TO_BCD_DWORD	UDINT	DWORD	
ULINT_TO_BCD_LWORD	ULINT	LWORD	

■ 플래그

플래그	설명
_ERR	IN 이 BCD 범위의 데이터가 아닌 경우, 출력은 0 이 되고 _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

1. LD



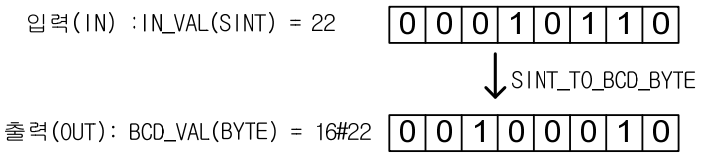
2. ST

ST 언어는 ***_TO_BCD 는 지원하지 않습니다.

SINT_TO_BCD_BYTE 인 경우

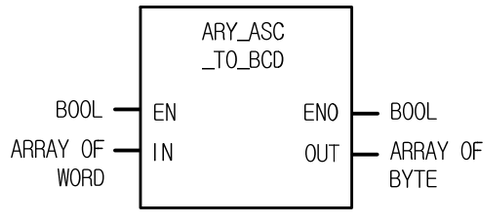
```
BCD_VAL := SINT_TO_BCD_BYTE(EN:=%MX0, IN:=IN_VAL);
```

(1) 실행조건(%MX0)이 On 하면 SINT_TO_BCD 펑션이 실행됩니다.
(2) IN_VAL(SINT 타입) = 16#22(2#0001_0110)이면, 펑션의 출력 변수로 선언된 BCD_VAL(BYTE 타입) = 16#22 (2#0010_0010)가 출력됩니다.



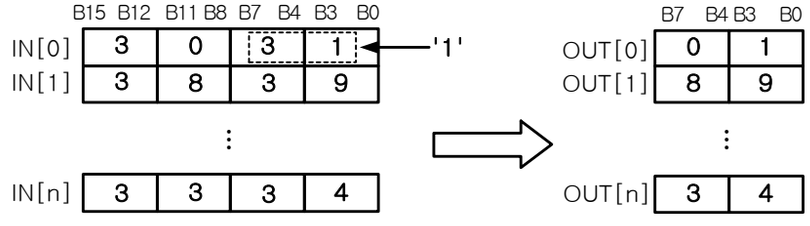
제8장 응용 펑션

7 장에서 설명한 기본 펑션과는 다른 응용 펑션에 대하여 설명합니다.

ARY_ASC_TO_BCD	적용 기종	발생플래그
입력 ASCII Array, 출력 BCD Array	XGI, XGR, XEC	_ERR, _LER
펑 션	설 명	
	**입력** EN : 1일 때 펑션 실행 IN : ASCII Array 입력 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : BCD Array 출력	

■ 기능

1. ASCII 데이터를 지니고 있는 WORD Array를 입력 받아서, BCD(Binary Coded Decimal) 값인 BYTE Array로 변환합니다.



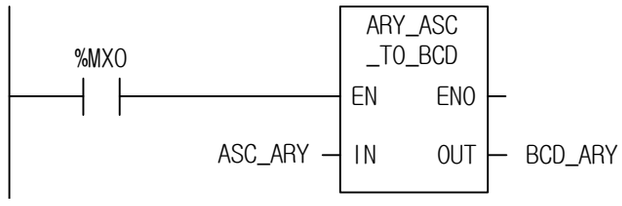
■ 플래그

플래그	설명
_ERR	입, 출력단의 Array 의 개수가 서로 다를 경우, OUT 값에는 변화가 없으며 _ERR, _LER 플래그가 셋(Set)됩니다. IN Array 원소의 값이 16 진수 값으로 '0' ~ '9' 이외의 값이 입력되면, IN 에 해당하는 OUT Array 원소의 값은 16#00 이 되고 나머지 원소들의 값은 정상적으로 변환이 이루어 지지만 _ERR, _LER 플래그가 셋(Set)됩니다.

☆ 입, 출력단의 Array 의 개수가 서로 다르면 _ERR, _LER 플래그가 발생하는데 출력단 Array 변수 생략시 어레이의 개수를 0 으로 보아 _ERR, _LER 플래그가 발생합니다.

■ 프로그램 예

1. LD



2. ST

```
CD_ARY := ARY_ASC_TO_BCD(EN:=%MX0, IN:=ASC_ARY);
```

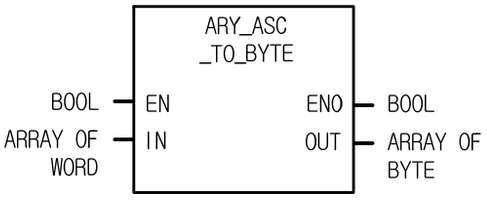
(1) 실행조건(%MX0)이 On 되면, ARY_ASC_TO_BCD 펑션이 실행됩니다.
(2) 펑션의 입력 변수로 선언된 ASC_ARY 값이 다음과 같이 선언되어 있다면

ASC_ARY[0]	3031
ASC_ARY[1]	3839
ASC_ARY[2]	3334

펑션이 실행된 후에 출력 변수로 선언된 BCD_ARY 의 값은 다음과 같이 출력됩니다.

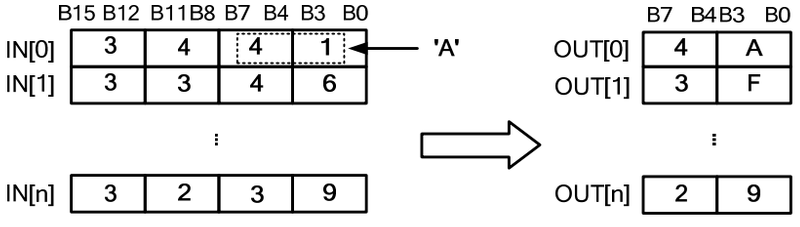
BYTE_ARY[0]	01
BYTE_ARY[1]	89
BYTE_ARY[2]	34

ARY_ASC_TO_BYTE	적용 기종	발생플래그
입력 ASCII Array, 출력 BYTE Array	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : ASCII Array 입력 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : BYTE Array 출력

■ 기능

1. ASCII 데이터를 지니고 있는 WORD Array를 입력 받아서, 16진수(HEX) 값인 BYTE Array로 변환합니다.



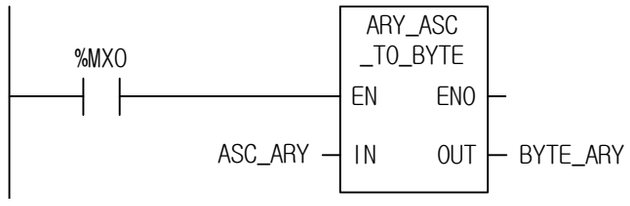
■ 플래그

플래그	설명
_ERR	입, 출력단의 Array 의 개수가 서로 다를 경우, OUT 값에는 변화가 없으며 _ERR, _LER 플래그가 셋(Set)됩니다. IN Array 원소의 값이 16 진수 값으로 '0' ~ 'F' 이외의 값이 입력되면, OUT에 해당하는 Array 원소의 값은 0 이 되고 나머지 원소들의 값은 정상적으로 변환이 이루어 지지만 _ERR, _LER 플래그가 셋(Set)됩니다.

☆ 입, 출력단의 Array의 개수가 서로 다르면 _ERR, _LER 플래그가 발생하는데 출력단 Array 변수 생략시 어레이의 개수를 0으로 보아 _ERR, _LER 플래그가 발생합니다.

■ 프로그램 예

1. LD



2. ST

```
YTE_ARY := ARY_ASC_TO_BYTE(EN:=%MX0, IN:=ASC_ARY);
```

(1) 실행조건(%MX0)이 On 되면, ARY_ASC_TO_BYTE 펑션이 실행됩니다.

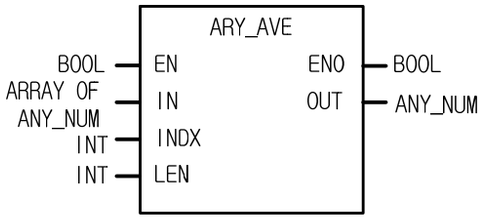
(2) 펑션의 입력 변수로 선언된 ASC_ARY 값이 다음과 같이 선언되어 있다면

ASC_ARY[0]	3441
ASC_ARY[1]	3346
ASC_ARY[2]	3239

펑션이 실행된 후에 출력 변수로 선언된 BYTE_ARY 의 값은 다음과 같이 출력됩니다.

BYTE_ARY[0]	4A
BYTE_ARY[1]	3F
BYTE_ARY[2]	29

ARY_AVE	적용 기종	발생플래그
Array 평균값 구하기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 평균을 위한 데이터 Array INDX : Array 내에서 연산을 시작할 위치 LEN : 평균값을 구할 Array 원소의 개수 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 연산후의 평균값을 출력

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN						○	○	○	○	○	○	○	○	○	○					
	OUT						○	○	○	○	○	○	○	○	○	○					

■ 기능

1. ARY_AVE 펑션은 Array 내부의 지정된 영역에 대하여 평균값을 구합니다.
2. 출력 타입은 입력 Array 타입과 동일하게 설정되어 있습니다.
3. LEN 이 마이너스일 경우에는 Array 인덱스부터 (Array 인덱스 - |LEN|) 사이의 원소들에 대한 평균값을 구합니다. 평균값은 반올림하여 출력합니다.

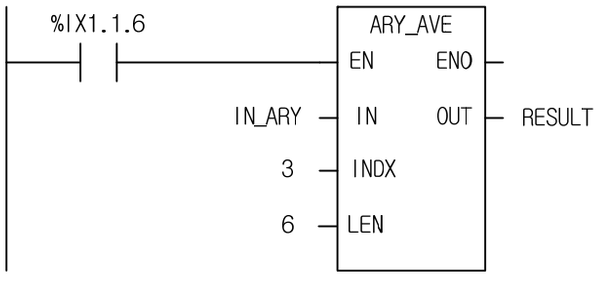
펑션	출력 타입	동작 설명
ARY_AVE	SINT	SINT 값들의 평균값을 구합니다. (소수점 이하는 반올림)
ARY_AVE	INT	INT 값들의 평균값을 구합니다. (소수점 이하는 반올림)
ARY_AVE	DINT	DINT 값들의 평균값을 구합니다. (소수점 이하는 반올림)
ARY_AVE	LINT	LINT 값들의 평균값을 구합니다. (소수점 이하는 반올림)
ARY_AVE	USINT	USINT 값들의 평균값을 구합니다. (소수점 이하는 반올림)
ARY_AVE	UINT	UINT 값들의 평균값을 구합니다. (소수점 이하는 반올림)
ARY_AVE	UDINT	UDINT 값들의 평균값을 구합니다. (소수점 이하는 반올림)
ARY_AVE	ULINT	ULINT 값들의 평균값을 구합니다. (소수점 이하는 반올림)
ARY_AVE	REAL	REAL 값들의 평균값을 구합니다.
ARY_AVE	LREAL	LREAL 값들의 평균값을 구합니다.

■ 플래그

플래그	설명
_ERR	Array 의 범위를 초과하여 지정한 경우 _ERR, _LER 플래그가 셋(Set)됩니다. 에러발생시 OUT 에는 0 이 출력됩니다. ※에러가 발생하는 범위 지정 INDX < 0 또는 INDX > IN 의 최대 원소번호 INDX + LEN > IN 의 최대 원소번호

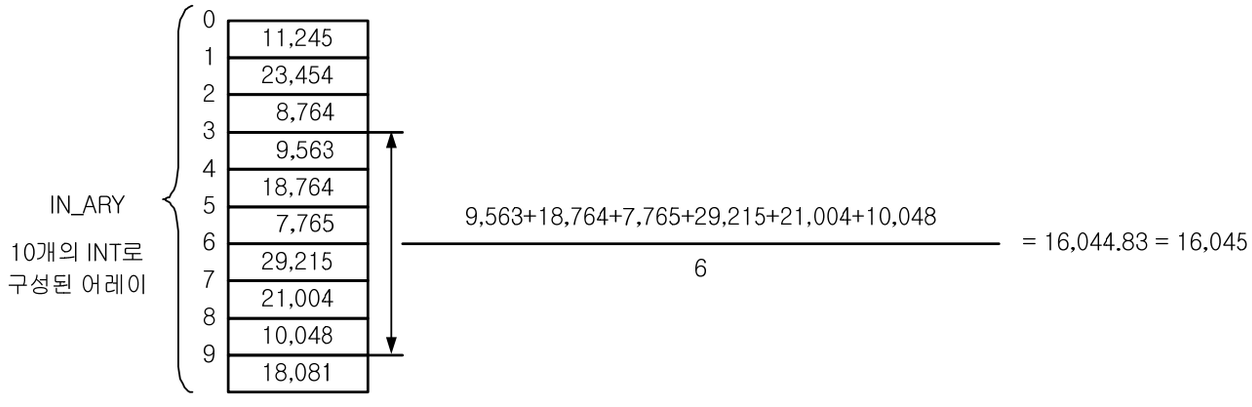
■ 프로그램 예

1. LD



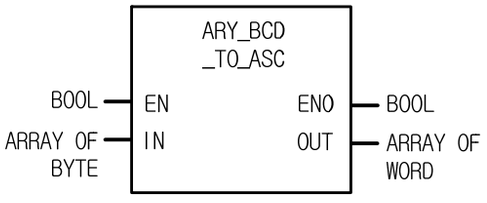
2. ST

```
RESULT := ARY_AVE(EN:=%IX1.1.6, IN:=IN_ARY, INDX:=3, LEN:=6);
```



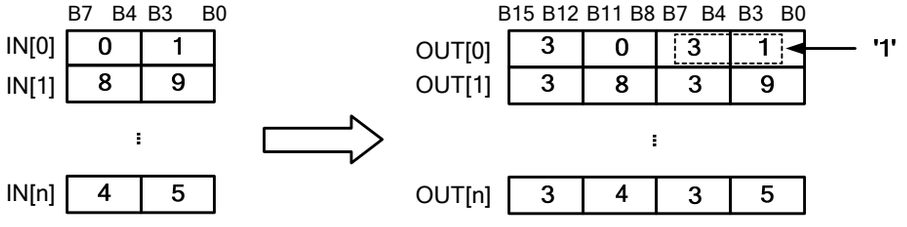
(1) 입력조건 (%IX1.1.6)이 On 되면, ARY_AVE 펑션의 INT 타입이 실행됩니다.
(2) ARRAY 내의 값이 위의 그림과 같을 경우 Array 인덱스 3 번째로부터 6 개의 원소에 대한 평균값을 구합니다
(3) 평균값이 16,044.8 이지만 출력 타입이 INT 이므로 반올림을 수행하여 16,045 를 출력합니다.

ARY_BCD_TO_ASC	적용 기종	발생플래그
입력 BCD Array, 출력 ASCII Array	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : BCD Array 입력 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : ASCII Array 출력

■ 기능

1. BCD 값을 지닌 BYTE Array 를 입력 받아서, ASCII 데이터를 지니고 있는 WORD Array 로 변환합니다.



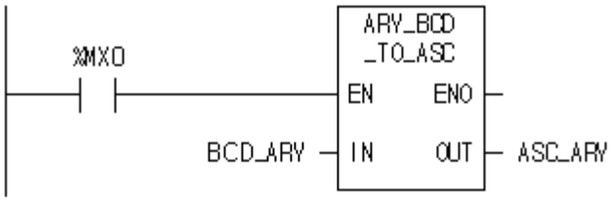
■ 플래그

플래그	설명
_ERR	입, 출력 단의 Array 의 개수가 서로 다를 경우, OUT 값에는 변화가 없으며 _ERR, _LER 플래그가 셋(Set)됩니다. IN Array 원소의 값이 16 진 값으로 '0' ~ '9' 이외의 값이 입력되면, IN 에 해당하는 OUT Array 원소의 값은 0 이 되고 나머지 원소들의 값은 정상적으로 변환이 이루어 지지만 _ERR, _LER 플래그가 셋(Set)됩니다.

☆ 입, 출력단의 Array 의 개수가 서로 다르면 _ERR, _LER 플래그가 발생하는데 출력단 Array 변수 생략시 어레이의 개수를 0 으로 보아 _ERR, _LER 플래그가 발생합니다.

■ 프로그램 예

1. LD



2. ST

```
ASC_ARY := ARY_BCD_TO_ASC(EN:=%MX0, IN:=BCD_ARY);
```

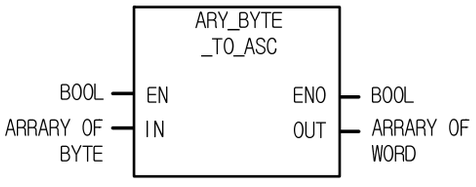
(1) 실행조건(%MX0)이 On 되면, ARY_BCD_TO_ASC 펑션이 실행 됩니다.

(2) 펑션의 입력 변수로 선언된 BYTE_ARY 의 값이 다음과 같이 선언되어 있다면

BYTE_ARY[0]	01
BYTE_ARY[1]	89
BYTE_ARY[2]	45

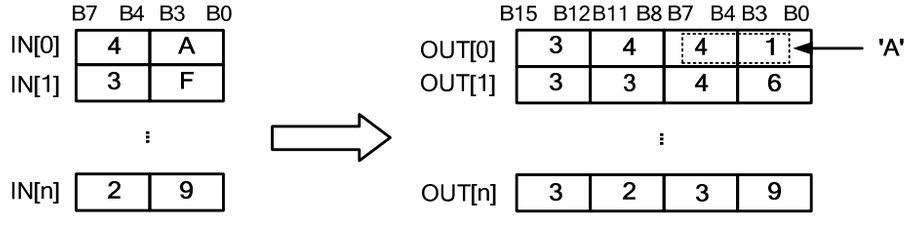
펑션이 실행된 후에 출력 변수로 선언된 ASC_ARY 의 값은 다음과 같이 출력됩니다.

ASC_ARY[0]	3031
ASC_ARY[1]	3839
ASC_ARY[2]	3435

ARY_BYTE_TO_ASC	적용 기종	발생플래그
입력 BYTE Array, 출력 ASCII Array	XGI, XGR, XEC	_ERR, _LER
펑 션	설 명	
	**입력** EN : 1일 때 펑션 실행 IN : BYTE Array 입력 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : ASCII Array 출력	

■ 기능

1. 16진(HEX)값을 가지고 있는 BYTE Array를 입력 받아서, ASCII 데이터를 지니고 있는 WORD Array로 변환합니다.



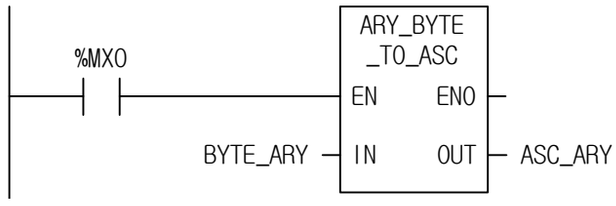
■ 플래그

플래그	설명
_ERR	입, 출력단의 Arrary의 개수가 서로 다를 경우 OUT값에는 변화가 없으며 _ERR, _LER 플래그가 셋(SET)됩니다.

☆ 입, 출력단의 Array의 개수가 서로 다르면 _ERR, _LER 플래그가 발생하는데 출력단 Array 변수 생략시 어레이의 개수를 0으로 보아 _ERR, _LER 플래그가 발생합니다.

■ 프로그램 예

1. LD



2. ST

```
ASC_ARY := ARY_BYTE_TO_ASC(EN:=%MX0, IN:=BYTE_ARY);
```

(1) 실행조건(%MX0)이 On 되면, ARY_BYTE_TO_ASC 펑션이 실행됩니다.

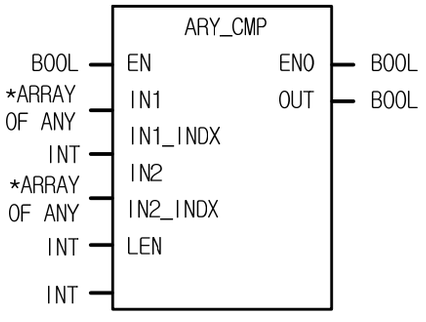
(2) 펑션의 입력 변수로 선언된 BYTE_ARY 의 값이 다음과 같이 선언되어 있다면

BYTE_ARY[0]	4A
BYTE_ARY[1]	3F
BYTE_ARY[2]	29

펑션이 실행된 후에 출력 변수로 선언된 ASC_ARY 의 값은 다음과 같이 출력됩니다.

ASC_ARY[0]	3441
ASC_ARY[1]	3346
ASC_ARY[2]	3239

ARY_CMP	적용 기종	발생플래그
Array 비교	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 비교할 첫번째 Array IN1_INDX : 첫번째 Array에서 비교할 시작 위치 IN2 : 비교할 두번째 Array IN2_INDX : 두번째 Array에서 비교할 시작 위치 LEN : 비교할 원소의 개수 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : Array 비교값이 동일하면 1을 출력

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	IN2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ARRAY OF ANY: ANY 타입 중 STRING 제외

■ 기능

1. ARY_CMP 펑션은 2개의 Array를 입력 받아 서로 동일한 값을 가지고 있는지를 비교합니다.
2. LEN이 마이너스일 경우에는 Array 인덱스부터 (Array 인덱스 - |LEN|) 사이의 원소들을 비교합니다.

펑션	입력 Array 타입	동작 설명
ARY_CMP	BOOL	2개의 BOOL Array를 서로 비교합니다.
ARY_CMP	BYTE	2개의 BYTE Array를 서로 비교합니다.
ARY_CMP	WORD	2개의 WORD Array를 서로 비교합니다.
ARY_CMP	DWORD	2개의 DWORD Array를 서로 비교합니다.
ARY_CMP	LWORD	2개의 LWORD Array를 서로 비교합니다.
ARY_CMP	SINT	2개의 SINT Array를 서로 비교합니다.
ARY_CMP	INT	2개의 INT Array를 서로 비교합니다.
ARY_CMP	DINT	2개의 DINT Array를 서로 비교합니다.
ARY_CMP	LINT	2개의 LINT Array를 서로 비교합니다.
ARY_CMP	USINT	2개의 USINT Array를 서로 비교합니다.
ARY_CMP	UINT	2개의 UINT Array를 서로 비교합니다.

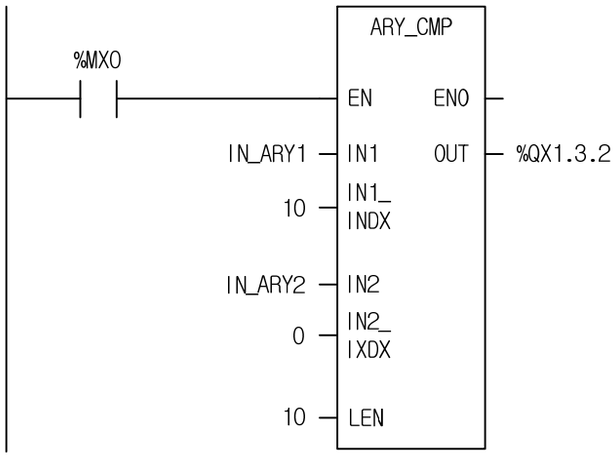
펑션	입력 Array 타입	동작 설명
ARY_CMP	UDINT	2 개의 UDINT Array 를 서로 비교합니다.
ARY_CMP	ULINT	2 개의 ULINT Array 를 서로 비교합니다.
ARY_CMP	REAL	2 개의 REAL Array 를 서로 비교합니다.
ARY_CMP	LREAL	2 개의 LREAL Array 를 서로 비교합니다.
ARY_CMP	TIME	2 개의 TIME Array 를 서로 비교합니다.
ARY_CMP	DATE	2 개의 DATE Array 를 서로 비교합니다.
ARY_CMP	TOD	2 개의 TOD Array 를 서로 비교합니다.
ARY_CMP	DT	2 개의 DT Array 를 서로 비교합니다.

■ 플래그

플래그	설명
_ERR	Array 의 범위를 초과하여 지정한 경우 _ERR, _LER 플래그가 셋(Set)됩니다. ※ 에러가 발생하는 범위 지정 IN1_INDX < 0 또는 IN1_INDX > IN1 최대 원소 번호 IN2_INDX < 0 또는 IN2_INDX > IN2 최대 원소 번호 IN1_INDX + LEN ≥ IN1 최대 원소 번호 IN2_INDX + LEN ≥ IN2 최대 원소 번호

■ 프로그램 예

1. LD



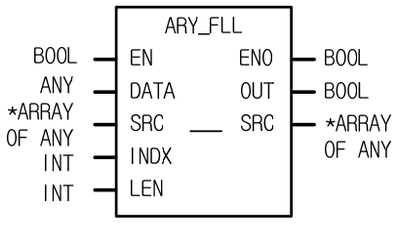
2. ST

```
%QX1.3.2 := ARY_CMP(EN:=%MX0, IN1:=IN_ARY1, IN1_INDX:=10, IN2:=IN_ARY2, IN2_INDX:=0, LEN:=10);
```

(1) 입력조건(%MX0)이 On 되면 ARY_CMP 펑션이 실행됩니다.

(2) IN_ARY1 이 100 개의 원소를 지닌 TIME Array 이고, IN_ARY2 가 10 개의 원소를 지닌 TIME Array 일 경우 IN_ARY1 의 11 번째 원소로부터 20 번째까지 10 개의 원소를 IN_ARY 2 의 첫번째 원소로부터 10 번째까지 10 개의 원소와 각각 비교하여 모두 동일하면 접점 출력 %QX1.3.2 가 On 됩니다.

ARY_FLL	적용 기종	발생플래그
Array 내부 영역 채우기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 DATA : Array 내를 채울 값 INDX : 값을 채울 Array 내의 처음 위치 LEN : 값을 채울 Array 원소의 개수 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 동작이 성공적으로 끝나면 1을 출력 **입출력** SRC : 값이 채워질 Array

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	DATA	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	SRC	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ARRAY OF ANY: ANY 타입 중 STRING 제외

■ 기능

1. ARY_FLL 펑션은 입력 DATA 값으로 Array 내부의 선택 영역을 채웁니다.
2. LEN 이 마이너스일 경우에는 Array 인덱스부터 (Array 인덱스 - |LEN|) 사이의 원소들을 비교합니다.

펑션	입출력 Array 타입	동작 설명
ARY_FLL	BOOL	BOOL Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	BYTE	BYTE Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	WORD	WORD Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	DWORD	DWORD Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	LWORD	LWORD Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	SINT	SINT Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	INT	INT Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	DINT	DINT Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	LINT	LINT Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	USINT	USINT Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	UINT	UINT Array 내부를 지정된 값으로 채웁니다.

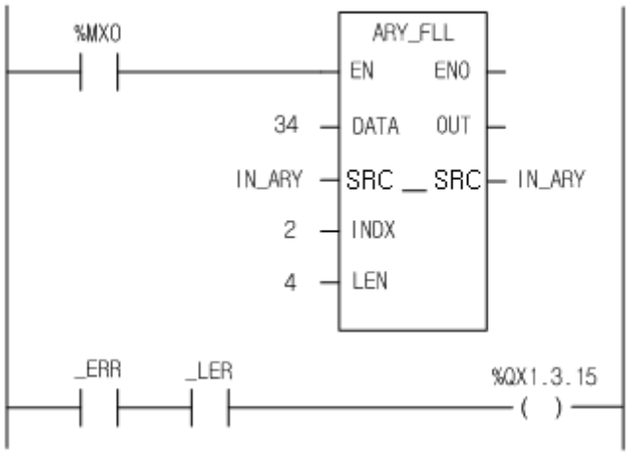
펑션	입출력 Array 타입	동작 설명
ARY_FLL	UDINT	UDINT Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	ULINT	ULINT Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	REAL	REAL Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	LREAL	LREAL Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	TIME	TIME Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	DATE	DATE Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	TOD	TOD Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	DT	DT Array 내부를 지정된 값으로 채웁니다.

■ 플래그

플래그	설명
_ERR	Array 의 범위를 초과하여 지정한 경우 _ERR, _LER 플래그가 셋(Set)됩니다. 에러 발생시 OUT 은 Off 되고, IN 의 Array 값은 변하지 않습니다. ※ 에러가 발생하는 범위 지정 INDX < 0 또는 INDX > IN 의 최대 원소번호 INDX + LEN ≥ IN 의 최대 원소번호

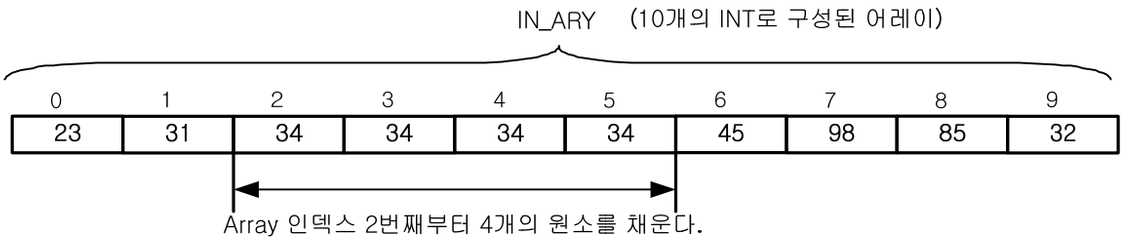
■ 프로그램 예

1. LD



2. ST

```
OUT :=ARY_FLL(EN:=%MX0, DATA:=34, SRC:=IN_ARY, INDX:=2, LEN:=4);
IF _ERR = 1 AND _LER = 1 THAN %QX1.3.15 := 1;
END_IF;
```

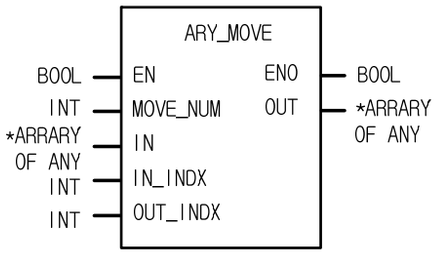


(1) 입력조건(%MX0)이 On 되면, ARY_FLL 펑션이 실행됩니다.

(2) Array 인덱스 2 번째부터 4 개의 원소를 지정된 값 34 로 채웁니다.

(3) 만약 LEN 을 9 로 대체하면 Array 의 전체 개수를 초과하므로 에러가 발생하여 _ERR 과 _LER 플래그가 On 되므로 출력 접점 %QX1.3.15 가 On 됩니다.

ARY_MOVE	적용 기종	발생플래그
Array 복사	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1 일때 펑션 실행 MOVE_NUM: Move 할 어레이 개수 IN : Move 할 어레이 변수(STRING Type 은 사용불가) N_INDX : 어레이 변수의 Move 할 시작 Pointer 위치 OUT_INDX: 어레이 변수의 Move 될 시작 Pointer 위치 **출력** ENO : 에러 없이 실행되면 1 을 출력 OUT : Move 될 어레이 변수(STRING Type 은 사용불가)

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ARRAY OF ANY: ANY 타입 중 STRING 제외

■ 기능

1. EN 이 1 이면, IN 어레이 변수의 데이터를 OUT 어레이 변수에 복사합니다.
2. IN 의 IN_INDX 번째 값부터 MOVE_NUM 개수만큼 데이터를 복사하여, OUT 의 OUT_INDX 번째 값부터 붙여넣기를 실행합니다.
3. MOVE 가 가능하기 위해서는 IN 과 OUT 의 어레이 데이터 타입과 Size 가 동일해야 합니다. 단, IN 과 OUT 의 어레이 개수는 다를 수 있습니다.
4. 데이터 타입의 Size 는 아래표와 같습니다.

데이터 Size	변수 타입
1 Bit	BOOL
8 Bit	BYTE/ SINT/ USINT
16 Bit	WORD / INT / UINT / DATE
32 Bit	DWORD / DINT / UDINT / TIME / TOD
64 Bit	DT

■ 플래그

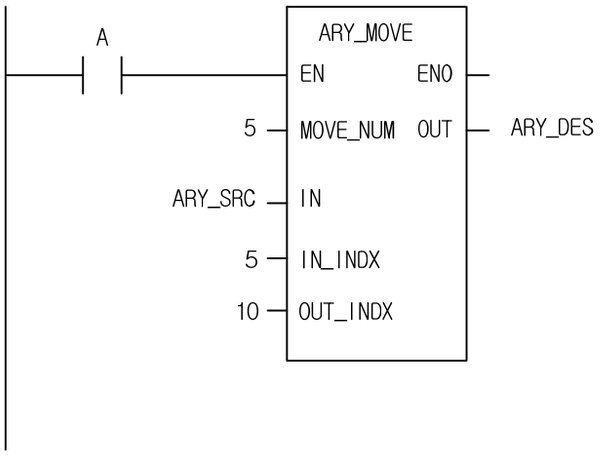
플래그	설명
_ERR	IN 과 OUT 어레이 데이터 타입의 Size 가 서로 다른 경우 에러가 발생합니다. IN 의 Array 개수가 (IN_INDX + MOVE_NUM) 보다 작은 경우나, OUT 의 Array 개수가 (OUT_INDX + MOVE_NUM) 보다 작은 경우 에러가 발생합니다. 이때, 데이터 복사는 수행하지 않고 OUT 은 0 이 됩니다. 또한, ENO 가 Off 되고 _ERR, _LER 플래그가 셋(Set)됩니다.

☆ 입, 출력단의 Array 의 개수가 서로 다르면 _ERR, _LER 플래그가 발생하는데 출력단 Array 변수 생략시 어레이의 개수를 0 으로 보아 _ERR, _LER 플래그가 발생합니다.

■ 프로그램 예

변수명	변수 타입	어레이 개수
ARY_SRC	INT	10
ARY_DES	WORD	15

1. LD



2. ST

```
ARY_DES := ARY_MOVE(EN:=A, MOVE_NUM:=5, IN:=ARY_SRC, IN_INDX:=5, OUT_INDX:=10);
```

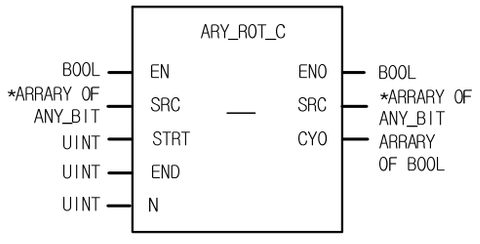
(1) 실행조건(A)이 On 되면 ARY_MOVE 펑션이 실행됩니다.

(2) 아래표와 같이 ARY_SRC[5]부터 5 개의 데이터를 ARY_DES[10]에 복사합니다.
이 때, ARY_DES 의 데이터 타입이 WORD 이므로 16 진수로 저장됩니다.

실행전				실행후			
ARY_SRC[0]	0	ARY_DES[0]	16#0	ARY_SRC[0]	0	ARY_DES[0]	16#0
ARY_SRC[1]	11	ARY_DES[1]	16#1	ARY_SRC[1]	11	ARY_DES[1]	16#1

실행전				실행후			
ARY_SRC[2]	22	ARY_DES[2]	16#2	ARY_SRC[2]	22	ARY_DES[2]	16#2
ARY_SRC[3]	33	ARY_DES[3]	16#3	ARY_SRC[3]	33	ARY_DES[3]	16#3
ARY_SRC[4]	44	ARY_DES[4]	16#4	ARY_SRC[4]	44	ARY_DES[4]	16#4
ARY_SRC[5]	55	ARY_DES[5]	16#5	ARY_SRC[5]	55	ARY_DES[5]	16#5
ARY_SRC[6]	66	ARY_DES[6]	16#6	ARY_SRC[6]	66	ARY_DES[6]	16#6
ARY_SRC[7]	77	ARY_DES[7]	16#7	ARY_SRC[7]	77	ARY_DES[7]	16#7
ARY_SRC[8]	88	ARY_DES[8]	16#8	ARY_SRC[8]	88	ARY_DES[8]	16#8
ARY_SRC[9]	99	ARY_DES[9]	16#9	ARY_SRC[9]	99	ARY_DES[9]	16#9
-	-	ARY_DES[10]	16#A	-	-	ARY_DES[10]	16#37
-	-	ARY_DES[11]	16#B	-	-	ARY_DES[11]	16#42
-	-	ARY_DES[12]	16#C	-	-	ARY_DES[12]	16#4D
-	-	ARY_DES[13]	16#D	-	-	ARY_DES[13]	16#58
-	-	ARY_DES[14]	16#E	-	-	ARY_DES[14]	16#63

ARY_ROT_C	적용 기종	발생플래그
Array의 Bit Rotate with Carry	XGI, XGR, XEC	_ERR, _LER

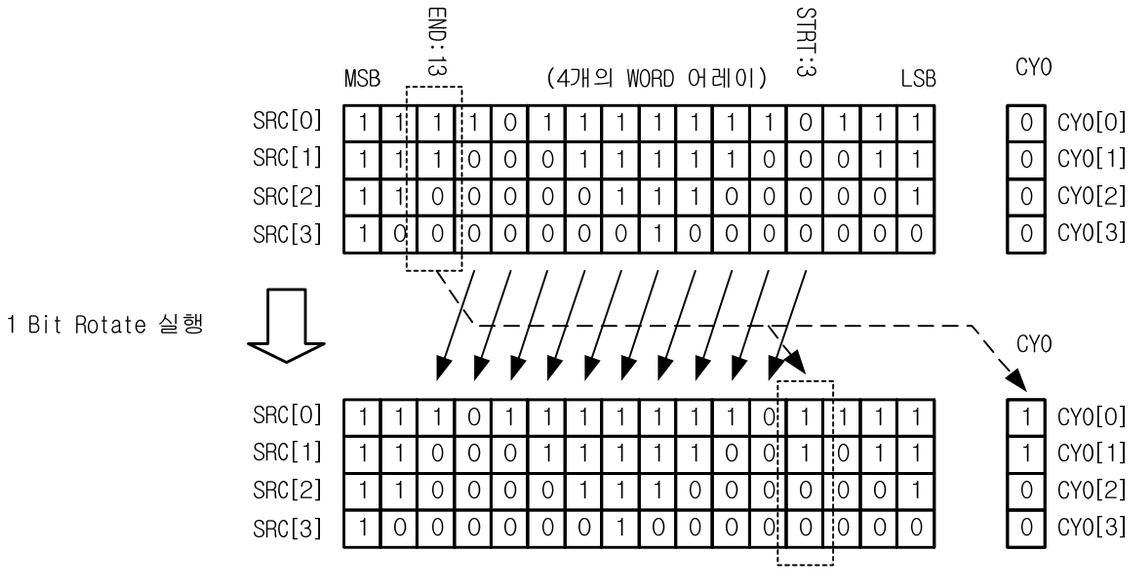
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 STRT : Rotate할 시작위치 END : Rotate할 종료위치 N : Rotate 시킬 개수 **출력** ENO : 에러 없이 실행되면 1을 출력 CYO : Rotate후에 출력되는 Carry bit Array **입출력** SRC : Rotate 시킬 Source Array

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	SRC		○	○	○	○															

*ARRAY OF ANY_BIT: ANY_BIT 타입 중 BOOL 제외

■ 기능

1. ARY_ROT_C 펑션은 지정된 범위의 Array 원소들의 bit 들을 정해진 개수만큼 Rotate 시킵니다.
2. 동작의 지정
   - 범위 지정: STRT 와 END 로 이동할 비트의 범위를 지정합니다.
   - 이동 방향 및 횟수: STRT 에서 END 방향으로 지정된 횟수(N)만큼 Rotate 합니다.
   - 출력: 데이터 조작의 결과는 SRC 에 지정된 어레이에 저장되며, 회전동작으로 END 위치에서 STRT 로 돌아 들어가는 비트열 데이터는 CYO 비트 Array 로 출력됩니다.



펑션	입출력 Array 타입	동작 설명
ARY_ROT_C	BYTE	각 타입 Array 원소들의 비트들을 지정된 범위 내에서 지정된 비트 수만큼 Rotate 시킵니다.
ARY_ROT_C	WORD	
ARY_ROT_C	DWORD	
ARY_ROT_C	LWORD	

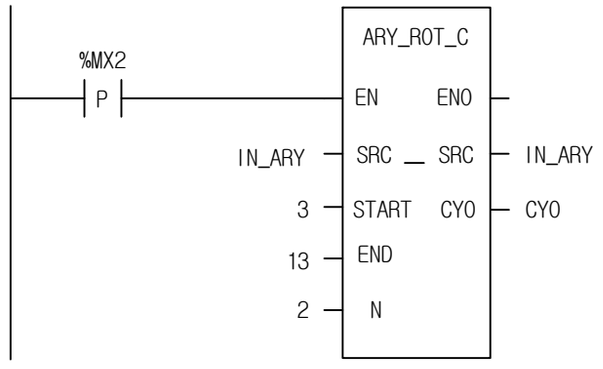
■ 플래그

플래그	설명
_ERR	SRC 과 CY0 Array 의 개수가 서로 동일하지 않을 경우 _ERR, _LER 플래그가 셋(Set)됩니다. STRT 와 END 가 SRC 원소의 비트 범위를 벗어나면 에러가 발생합니다. 에러발생시 SRC 와 CY0 는 변하지 않습니다.

☆ 입, 출력단의 Array 의 개수가 서로 다르면 _ERR, _LER 플래그가 발생하는데 출력단 Array 변수 생략시 어레이의 개수를 0 으로 보아 _ERR, _LER 플래그가 발생합니다.

■ 프로그램 예

1. LD



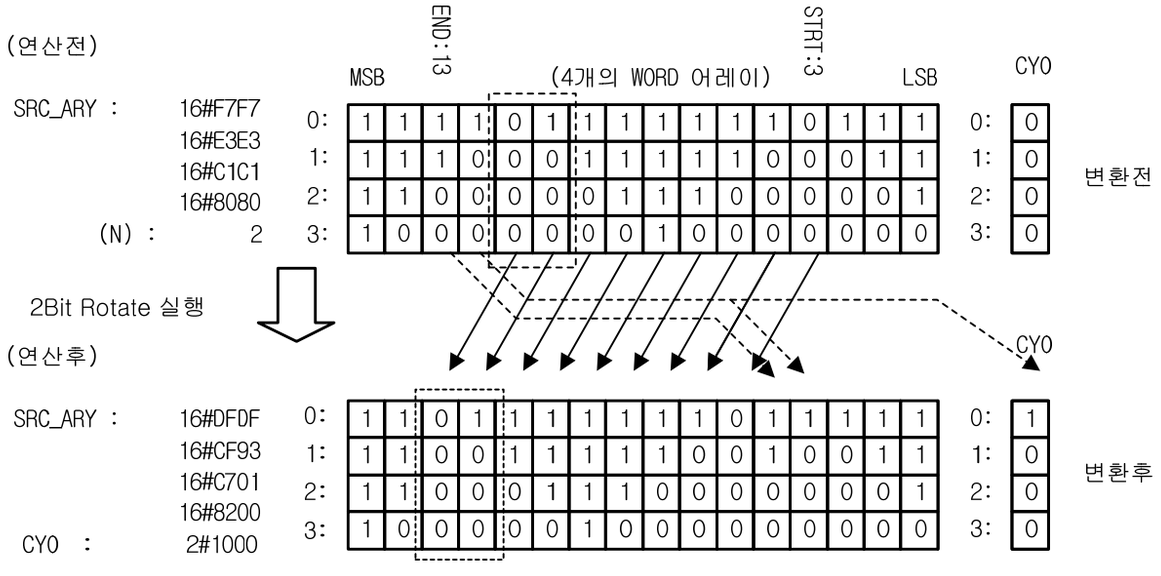
2. ST

```
ARY_ROT_C(EN:=%MX2, SRC:=IN_ARY, STRT:=3, END:=13, N:=2, CYO=>CYO);
```

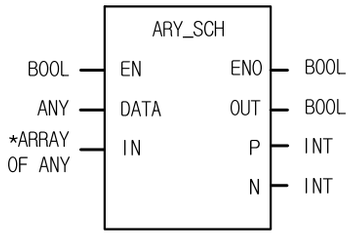
(1) 입력조건(%MX2)이 On 되면, ARY_ROT_C 펑션이 실행됩니다.

(2) IN_ARY 에 STRT 인 3 번째 비트열과 END 인 13 번째 비트열로 지정된 범위에서 STRT 로부터 END 방향으로 2 번 회전합니다.

(3) 회전된 결과는 다시 IN_ARY 에 저장되며 이때 출력되는 캐리 비트열은 CYO Array 로 출력됩니다.



ARY_SCH	적용 기종	발생플래그
Array 탐색(search)	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 DATA : 탐색할 DATA IN : 탐색할 Array **출력** ENO : EN 값을 그대로 출력 OUT : 원하는 값을 찾으면 1을 출력 P : 목표값에 해당하는 Array 의 첫번째 위치 N : 목표값과 동일한 Array 원소들의 개수

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	DATA	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	IN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ARRAY OF ANY: ANY 타입 중 STRING 제외

■ 기능

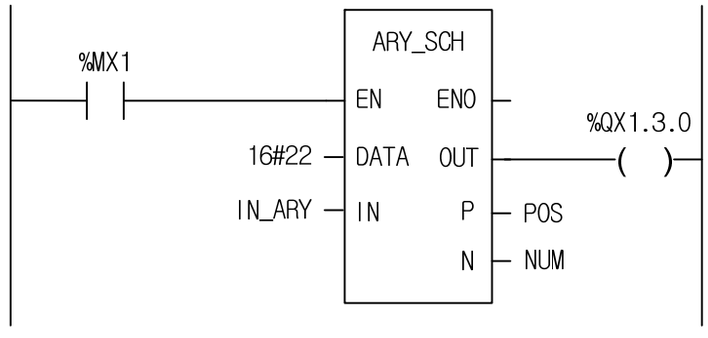
1.ARY_SCH 펑션은 Array 내에서 입력된 값과 동일한 값을 찾아 Array 내에서의 처음 위치와 전체 개수를 출력합니다. Array 내에서 목표 값과 동일한 값이 하나 이상일 경우 OUT 에 1을 출력합니다.

펑션	입력 Array 타입	동작 설명
ARY_SCH	BOOL	BOOL Array 내에서 탐색합니다.
ARY_SCH	BYTE	BYTE Array 내에서 탐색합니다.
ARY_SCH	WORD	WORD Array 내에서 탐색합니다.
ARY_SCH	DWORD	DWORD Array 내에서 탐색합니다.
ARY_SCH	LWORD	LWORD Array 내에서 탐색합니다.
ARY_SCH	SINT	SINT Array 내에서 탐색합니다.
ARY_SCH	INT	INT Array 내에서 탐색합니다.
ARY_SCH	DINT	DINT Array 내에서 탐색합니다.
ARY_SCH	LINT	LINT Array 내에서 탐색합니다.
ARY_SCH	USINT	USINT Array 내에서 탐색합니다.
ARY_SCH	UINT	UINT Array 내에서 탐색합니다.
ARY_SCH	UDINT	UDINT Array 내에서 탐색합니다.

펑션	입력 Array 타입	동작 설명
ARY_SCH	ULINT	ULINT Array 내에서 탐색합니다.
ARY_SCH	REAL	REAL Array 내에서 탐색합니다.
ARY_SCH	LREAL	LREAL Array 내에서 탐색합니다.
ARY_SCH	TIME	TIME Array 내에서 탐색합니다.
ARY_SCH	DATE	DATE Array 내에서 탐색합니다.
ARY_SCH	TOD	TOD Array 내에서 탐색합니다.
ARY_SCH	DT	DT Array 내에서 탐색합니다.

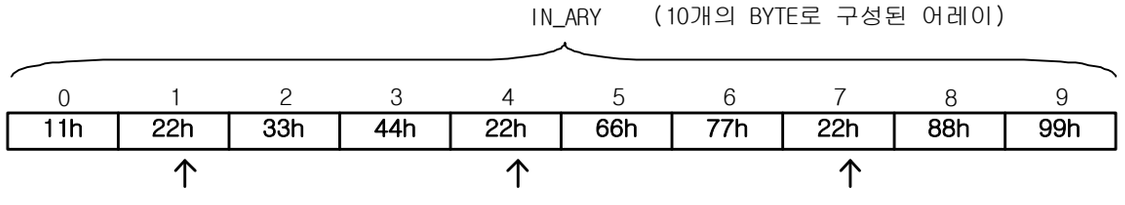
■ 프로그램 예

1. LD



2. ST

```
%QX1.3.0 := ARY_SCH(EN:=%MX1, DATA:=16#22, IN:=IN_ARY, P=>POS, N=>NUM);
```

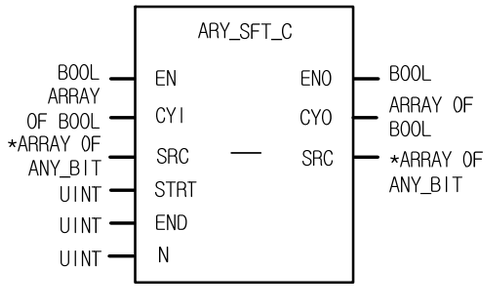


(1) 입력조건(%MX1)이 On 되면 ARY_SCH 펑션이 실행됩니다.

(2) 위의 그림에서처럼 IN_ARY 가 10 개의 바이트 Array 로 구성되어 있고 이 Array 내부에서 22h 이라는 값을 탐색하면 화살표로 표시된 바와 같이 3 개가 탐색 됩니다.

(3) POS 에는 Array 내부의 첫번째 위치인 1 이 출력되고, NUM 에는 전체 개수인 3 이 출력됩니다. 펑션 출력으로는 하나 이상의 값이 탐색 되었으므로 1 이 출력되어 출력접점 %QX1.3.0 은 On 이 됩니다.

ARY_SFT_C	적용 기종	발생플래그
Array의 Bit Shift Left with Carry	XGI, XGR, XEC	_ERR, _LER

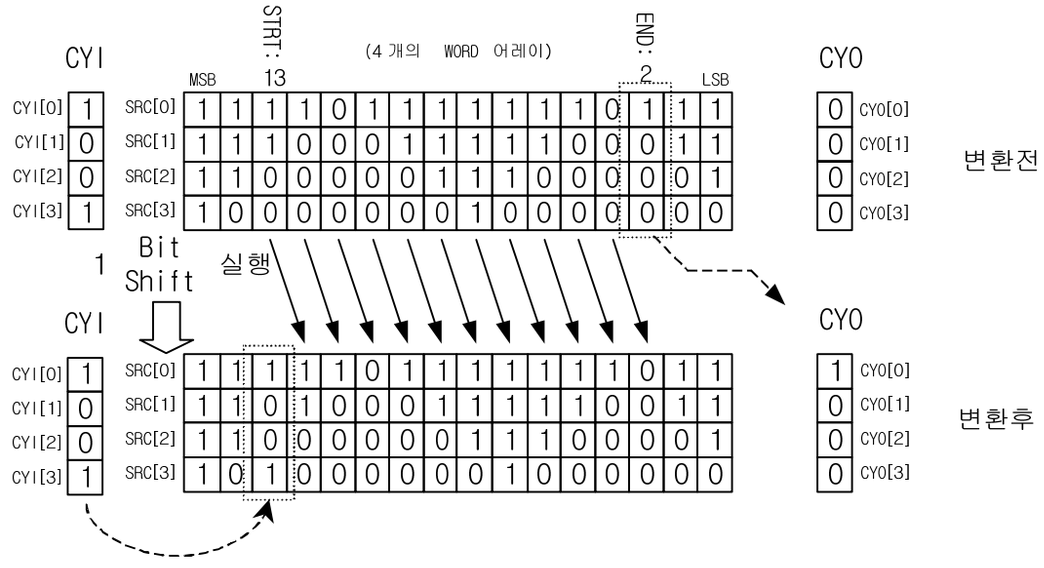
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 CYI : Array에 입력되는 Carry bit Array STRT : Shift할 bit의 시작 END : Shift할 bit의 종료위치 N : Shift 시킬 개수 **출력** ENO : 에러 없이 실행되면 1을 출력 CYO : Shift 후에 출력되는 Carry bit Array **입출력** SRC : Shift 시킬 Source Array

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	SRC		○	○	○	○															

*ARRAY OF ANY_BIT: ANY_BIT 타입 중 BOOL 제외

■ 기능

1. ARY_SFT_C 펑션은 Array 원소들의 bit 들을 정해진 개수만큼 지정된 방향으로 이동시킵니다.
2. 동작의 지정
   - 범위 지정: STRT와 END로 이동할 비트의 범위를 지정됩니다.
   - 이동 방향 및 횟수: STRT에서 END방향으로 지정된 횟수(N)만큼 이동합니다.
   - 입력 데이터 지정: Shift한 후에 비워지는 위치를 입력 데이터(CYI)로 채웁니다.
   - 출력: 데이터 조작의 결과는 SRC에 지정된 어레이에 저장되며, END 위치에서 Shift 동작으로 밀려 나온 비트열 데이터는 CYO로 출력됩니다.



펑션	입출력 Array 타입	동작 설명
ARY_SFT_C	BYTE	각 타입 Array 원소들의 비트들을 지정된 범위 내에서 지정된 비트 수만큼 Shift 시킵니다.
ARY_SFT_C	WORD	
ARY_SFT_C	DWORD	
ARY_SFT_C	LWORD	

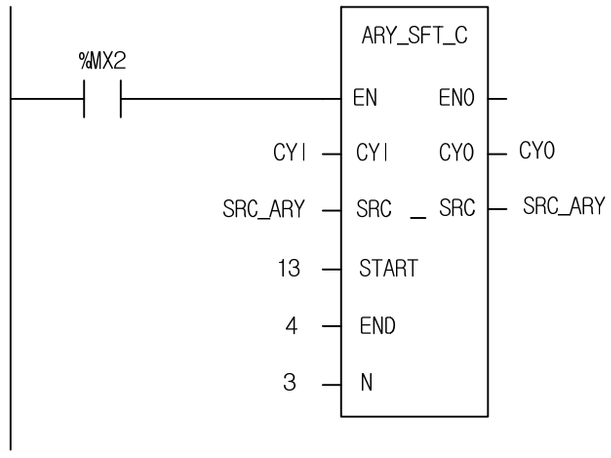
■ 플래그

플래그	설명
_ERR	CYI, SRC, CYO Array 의 개수가 모두 동일하지 않을 경우 _ERR, _LER 플래그가 셋(Set) 됩니다. STRT 와 END 가 SRC 원소의 비트 범위를 벗어나면 에러가 발생합니다. 에러발생시 SRC 와 CYO 는 변하지 않습니다.

☆ 입, 출력단의 Array 의 개수가 서로 다르면 _ERR, _LER 플래그가 발생하는데 출력단 Array 변수생략시 어레이의 개수를 0 으로 보아 _ERR, _LER 플래그가 발생합니다.

■ 프로그램 예

1. LD



2. ST

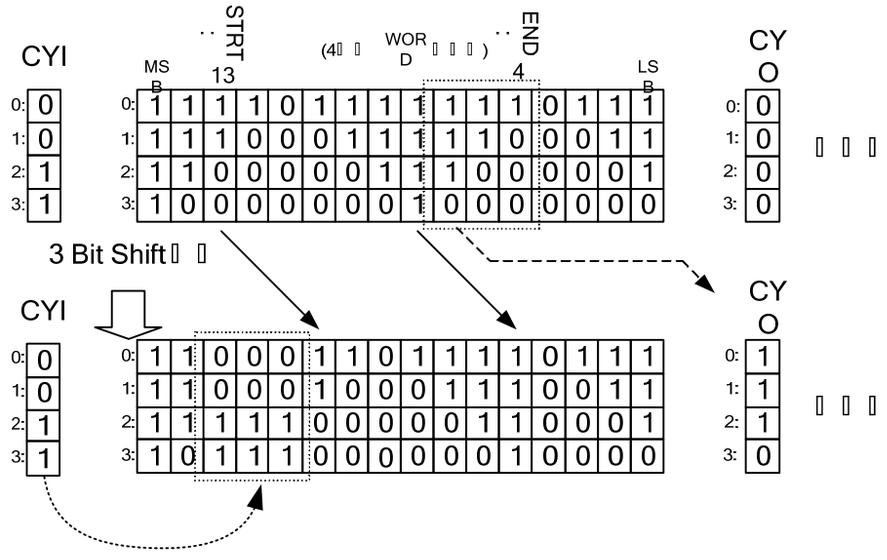
```
ARY_SFT_C(EN:=%MX2, CYI:=CYO, SRC:=SRC_ARY, STRT:=13, END:=4, N:=2, CYO=>CYO);
```

(1) 입력조건(%MX2)이 On 되면, ARY_SFT_C 펑션이 실행됩니다.

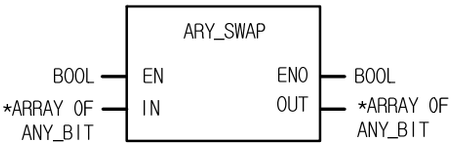
(2) STRT 인 13 번째 bit 열과 END 인 4 번째 비트열로 지정된 범위에서 STRT 로부터 END 방향으로 3 번 shift 합니다.

(3) Shift 후에 비워지는 비트열은 CYI(2#0011)로 채워집니다.

(4) Shift 후의 결과는 SRC_ARY 로 다시 출력되고 캐리 비트열은 CYO 로 출력됩니다.



ARY_SWAP	적용 기종	발생플래그
Array 원소 데이터의 상위 하위 바꾸기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : Array 입력 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : Swap 된 Array 출력

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*ARRAY OF ANY_BIT: ANY_BIT 타입 중 BOOL 제외

■ 기능

1. 입력된 Array 원소를 2 개의 크기로 구분하여 상위와 하위를 서로 교환합니다.

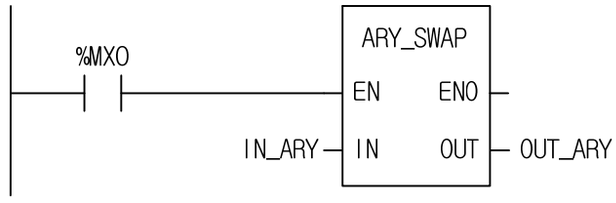
■ 플래그

플래그	설명
_ERR	입, 출력단의 Array 의 개수가 서로 다를 경우에는 OUT Array 의 원소 값에는 변화가 없으며 _ERR, _LER 플래그가 셋(Set)됩니다.

☆ 입, 출력단의 Array 의 개수가 서로 다르면 _ERR, _LER 플래그가 발생하는데 출력단 Array 변수생략시 어레이의 개수를 0 으로 보아 _ERR, _LER 플래그가 발생합니다.

■ 프로그램 예

1. LD



2. ST

```
OUT_ARY := ARY_SWAP(EN:=%MX0, IN:=IN_ARY);
```

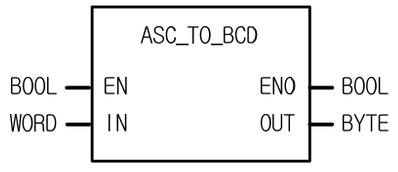
(1) 실행조건(%MX0)이 On 되면, ARY_SWAP 펑션의 WORD 타입이 실행됩니다.

(2) 펑션의 입력 변수로 선언된 IN_ARY 의 값이 다음과 같을 경우

IN_ARY[0]	12AB
IN_ARY[1]	23BC
IN_ARY[2]	34CD

펑션이 실행된 후에 출력 변수로 선언된 OUT_ARY 의 값은 다음과 같이 출력됩니다.

OUT_ARY[0]	AB12
OUT_ARY[1]	BC23
OUT_ARY[2]	CD34

ASC_TO_BCD	적용 기종	발생플래그
ASCII 를 BCD 로 변환	XGI, XGR, XEC	_ERR, _LER
펑 션	설 명	
	**입력** EN : 1 일 때 펑션 실행 IN : ASCII 입력 **출력** ENO : 에러 없이 실행되면 1 을 출력 OUT : BCD 출력	

■ 기능

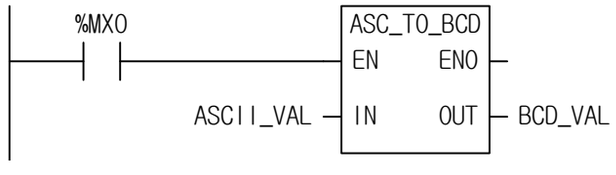
1. 2 개의 ASCII 값을 받아서 2 자리의 BCD(Binary Coded Decimal)로 출력합니다.

■ 플래그

플래그	설명
_ERR	입력 IN 값이 16진수(HEX)로 '0' ~ '9' 이외의 값이 입력 되면 출력 OUT는 00이 되고 _ERR, _LER의 에러 플래그가 셋(SET)됩니다.

■ 프로그램 예

1. LD

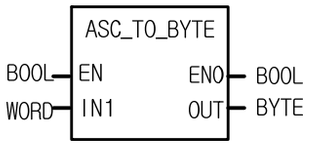


2. ST

```
BCD_VAL := ASC_TO_BCD(EN:=%MX0, IN:=ASCII_VAL);
```

(1) 조건(%MX0)이 On 되면, ASC_TO_BCD 펑션이 실행됩니다.

(2) 펑션 입력으로 선언된 ASCII_VAL(WORD 타입) = 16#3732 = '72' 일 경우, 펑션의 출력 변수로 선언된 BCD_VAL(BYTE 타입) = 16#72 가 출력됩니다.

ASC_TO_BYTE	적용 기종	발생플래그
ASCII 를 BYTE 로 변환	XGI, XGR, XEC	_ERR, _LER
펑 션	설 명	
	**입력** EN : 1 일 때 펑션 실행 IN : ASCII 입력 **출력** ENO : 에러 없이 실행되면 1 을 출력 OUT : BYTE 출력	

■ 기능

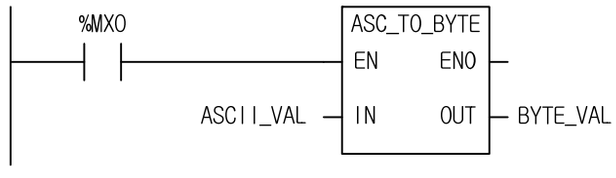
1. 2 개의 ASCII 데이터를 입력을 받아서 2 자리의 16 진(HEX) 값으로 출력합니다.

■ 플래그

플래그	설명
_ERR	입력 IN값이 '0' ~ 'F' 이외의 값이 입력으로 들어오면 출력 OUT값은 00이 되고 에러 플래그 _ERR, LER이 셋(SET)됩니다.

■ 프로그램 예

1. LD

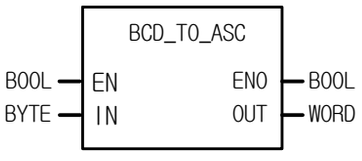


2. ST

```
BYTE_VAL := ASC_TO_BYTE(EN:=%MX0, IN:=ASCII_VAL);
```

(1) 실행조건(%MX0)이 On 되면, ASC_TO_BYTE 펑션이 실행됩니다.

(2) 펑션의 입력변수로 선언된 ASCII_VAL(WORD 타입)=16#4339 일 경우, 펑션의 출력 변수로 선언된 BYTE_VAL(BYTE 타입) = 16#C9 가 출력됩니다.

BCD_TO_ASC	적용 기종	발생플래그
BCD 를 ASCII 로 변환	XGI, XGR, XEC	_ERR, _LER
펑 션	설 명	
	**입력** EN : 1 일 때 펑션 실행 IN : BCD 입력 **출력** ENO : 에러 없이 실행되면 1 을 출력 OUT : ASCII 출력	

■ 기능

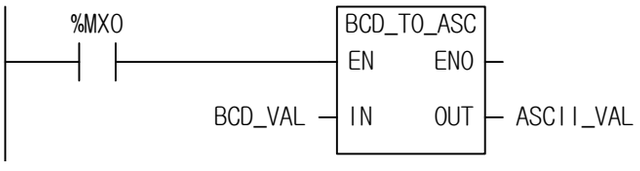
1. 2 자리의 BCD(Binary Coded Decimal) 값을 받아서 2 개의 아스키 값으로 출력합니다.

■ 플래그

플래그	설명
_ERR	IN값을 통해 16진수 값으로 0 ~ 9이외의 값이 입력되면 출력은 16#3030(“00”)이 출력되고 에러 플래그 _ERR, _LER이 셋(SET)됩니다.

■ 프로그램 예

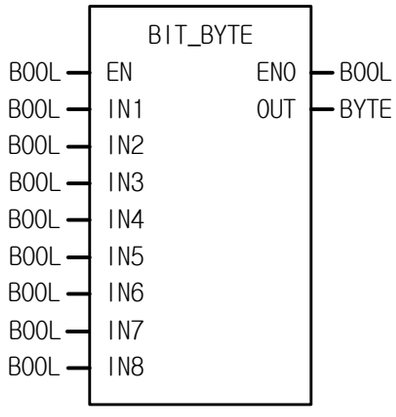
1. LD



2. ST

```
ASCII_VAL := BCD_TO_ASC(EN:=%MX0, IN:=BCD_VAL);
```

(1) 실행조건(%MX0)가 On 되면 BCD_TO_ASC 펑션이 동작합니다.
(2) 입력 BCD_VAL(BYTE 타입) = 16#85일 경우, 펑션의 출력 변수로 선언된 ASCII_VAL(WORD 타입) = 16#3835 = “85” 가 출력됩니다.

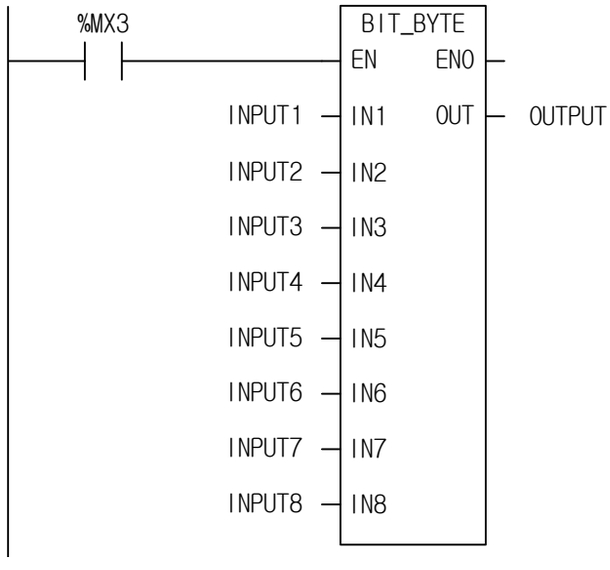
BIT_BYTE	적용 기종	발생플래그
8 개 BIT 들을 BYTE 로 묶음	XGI, XGR, XEC	_ERR, _LER
펑 션	설 명	
	**입력** EN : 1 일 때 펑션 실행 IN1 ~ IN8 : Bit 입력 **출력** ENO : EN 값을 그대로 출력 OUT : Byte 출력	

■ 기능

1. 8 개의 비트를 하나의 바이트로 조합합니다.
   IN8: 최상의 비트(MSB), IN1: 최하의 비트(LSB)

■ 프로그램 예

1. LD

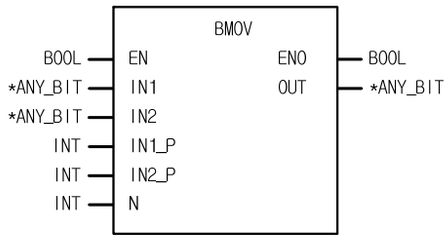


2. ST

```
OUTPUT := BIT_BYTE(EN:=%MX3, IN1:=INPUT1, IN2:=INPUT2, IN3:=INPUT3, IN4:=INPUT4, IN5:=INPUT5, IN6:=INPUT6,
                   IN7:=INPUT7, IN8:=INPUT8);
```

(1) 실행조건(%MX3)이 On 되면 BIT_BYTE 펑션이 실행됩니다.
(2) 8개의 입력이 IN1~ IN8까지 {0,1,1,0,1,1,0,0}이면 출력변수로 선언된 OUTPUT(BYTE타입) = 2#0110_1100 입니다.

BMOV	적용 기종	발생플래그
비트 스트링의 일부분을 복사, 이동	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 조합할 비트 데이터를 가진 스트링 데이터 IN2 : 조합할 비트 데이터를 가진 스트링 데이터 IN1_P : IN1 지정 데이터상의 시작 비트 위치 IN2_P : IN2 지정 데이터상의 시작 비트 위치 N : 조합할 비트의 수 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 조합된 비트 스트링 데이터 출력

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1		○	○	○	○															
	IN2		○	○	○	○															
	OUT		○	○	○	○															

*ANY_BIT: ANY_BIT 타입 중 BOOL 제외

■ 기능

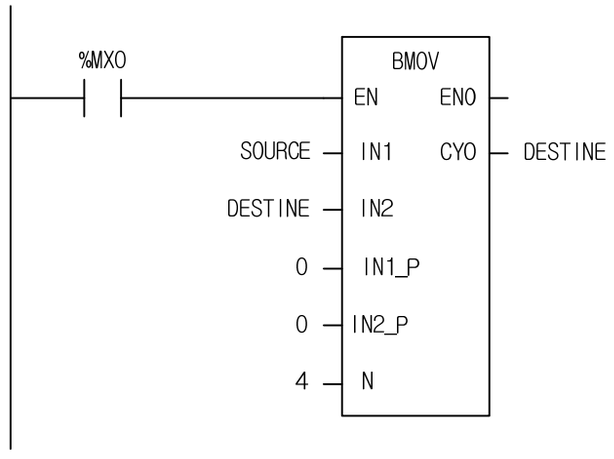
1. EN이 1이 되면 IN1의 비트 스트링 중 IN1_P로 지정된 비트 위치부터 큰 방향으로 N개의 비트를 취하여, IN2의 비트 스트링에서 IN2_P로 지정된 비트 위치부터 큰 방향으로 대치한 후 OUT으로 출력합니다.
2. IN1 = 1111_0000_1111_0000, IN2 = 0000_1010_1010_1111 이고 IN1_P = 4, IN2_P = 8, N = 4 면, 출력되는 데이터는 OUT = 0000_1111_1010_1111 이 됩니다.입력에는 B(BYTE), W(WORD), D(DWORD), L(LWORD) 타입의 데이터가 접속 가능합니다.

■ 플래그

플래그	설명
_ERR	IN1_P, IN2_P 값이 데이터 범위를 벗어나거나, N의 값이 음수 또는 IN1_P, IN2_P로부터 N개를 취한 것이 데이터 범위를 벗어나면 결과를 출력하지 않고 에러플래그 _ERR, _LER이 셋(SET)됩니다.

■ 프로그램 예

1. LD



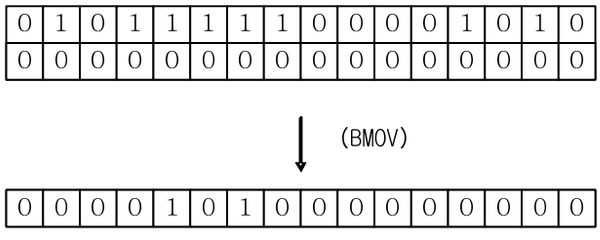
2. ST

```
DESTINE := BMOV(EN:=%MX0, IN1:=SOURCE, IN2:=DESTINE, IN1_P:=0, IN2_P:=0, N:=4);
```

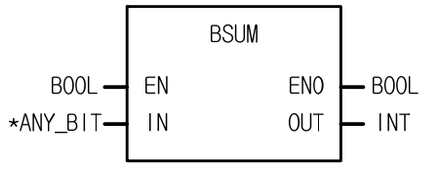
(1) 실행조건(%MX0)이 On 되면 BMOV 펑션이 실행됩니다.

(2) 입력변수로 선언된 SOURCE = 2#0101_1111_0000_1010, DESTINE = 2#0000_0000_0000_0000 이고, IN1_ P = 0, IN2_P = 8, N = 4 이므로 연산 결과는 2#0000_1010_0000_0000 이 되고, 출력을 DESTINE 으로 지정하였으므로 DESTINE = 2#0000_1010_0000_0000 으로 바뀌게 됩니다.

```
입력(IN1) : SOURCE (WORD) = 16#5F0A
    (IN2) : DESTINE(WORD) = 16#0000
    (IN1_ P) = 0
    (IN2_P) = 8
    (N) =    4

출력(OUT) : DESTINE(WORD) = 16#0A00
```



BSUM	적용 기종	발생플래그
On 된 비트 개수를 숫자로 출력	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : On 비트를 검색할 입력 데이터 **출력** ENO : EN 값이 그대로 출력 OUT : On 된 비트 개수를 합한 결과 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															

*ANY_BIT: ANY_BIT 타입 중 BOOL 제외

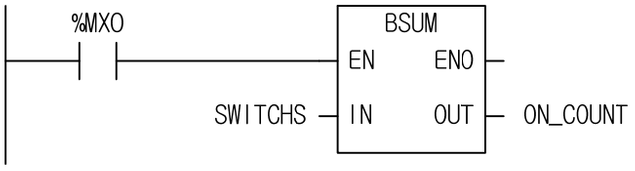
■ 기능

1. EN 이 1 이면, IN 의 비트 스트링 데이터 중, 1 로 되어있는 비트의 숫자를 세어서 OUT 으로 출력합니다.
2. 입력에는 BYTE, WORD, DWORD, LWORD 타입의 데이터가 접속 가능합니다.

FUNCTION	IN 변수 타입	동작 설명
BSUM	BYTE	입력 데이터에 맞추어 4 가지 타입 중 하나를 사용합니다.
BSUM	WORD	
BSUM	DWORD	
BSUM	LWORD	

■ 프로그램 예

1. LD



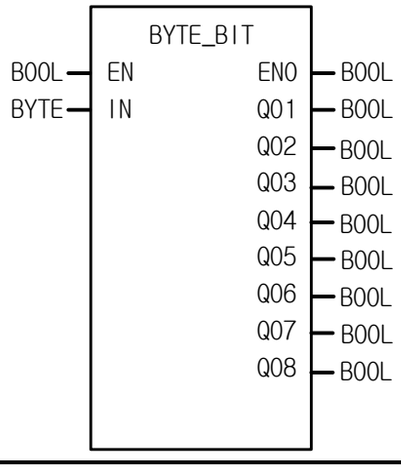
2. ST

```
ON_COUNT := BSUM(EN:=%MX0, IN:=SWITCHS);
```

(1) 실행조건(%MX0)이 On 되면 BSUM 펑션이 실행됩니다.

(2) 입력변수로 선언된 SWITCHS(WORD 타입) = 2#0000_0100_0010_1000 이라면, 출력변수(On_COUNT)는 On 되어 있는 비트의 개수를 출력합니다. 즉, '3' 을 출력하여 ON_COUNT(INT 타입)에 정수값 '3' 이 저장됩니다.

BYTE_BIT	적용 기종	발생플래그
BYTE 를 8 개 BIT 들로 나눔	XGI, XGR, XEC	-

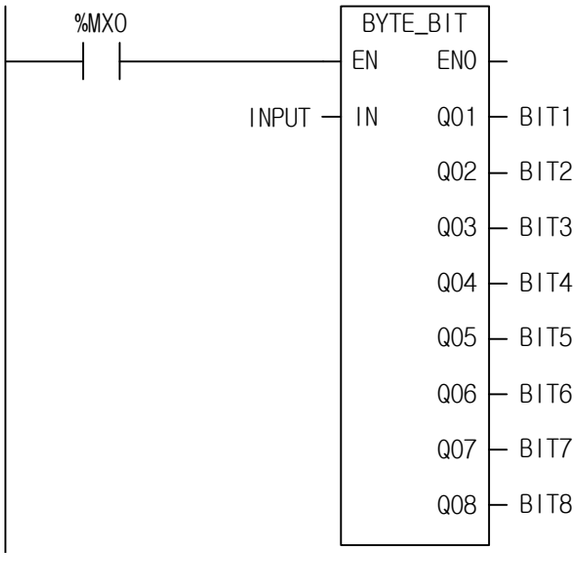
펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 IN : Byte 입력 **출력** ENO : EN 값을 그대로 출력 Q01~8 : Bit 출력

■ 기능

1. 1 개의 바이트를 출력 변수로 선언된 8 개의 비트(Q01~Q02)로 분산합니다.
2. Q08: 최상위 비트(MSB), Q01: 최하위 비트(LSB)

■ 프로그램 예

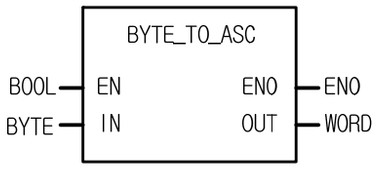
1. LD



2. ST

```
BYTE_BIT(EN:=%MX0, IN:= INPUT, Q01=> BIT1, Q02=> BIT2, Q03=> BIT3, Q04=> BIT4, Q05=> BIT5, Q06=> BIT6, Q07=> BIT7, Q08=>
BIT8);
```

(1) 실행조건(%MX0)가 On되면 BYTE_BIT 펑션이 실행 됩니다.
(2) 입력으로 선언된 INPUT = 16#AC = 2#1010_1100 이면 출력 변수로 선언된 Q01 ~ Q08까지 Q01부터 순서대로 2#{0,0,1,1,0,1,0,1}이 저장됩니다.

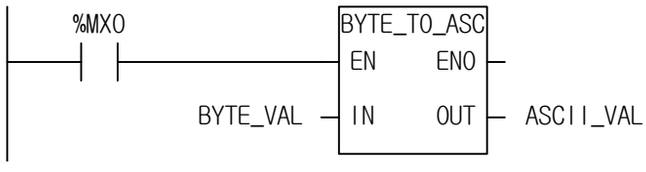
BYTE_TO_ASC	적용 기종	발생플래그
BYTE 를 ASCII 로 변환	XGI, XGR, XEC	-
펑 션	설 명	
	**입력** EN : 1 일 때 펑션 실행 IN : BYTE 입력 **출력** ENO : EN 값을 그대로 출력 OUT : ASCII 출력	

■ 기능

1. 2 자리의 16 진(HEX) 데이터를 입력 받아서 2 개의 아스키 값으로 출력합니다.
   예) 16#12 -> 3132
2. 16#A~F 는 대문자의 아스키 값으로 출력합니다.

■ 프로그램 예

1. LD

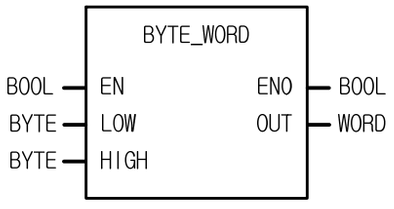


2. ST

```
ASCII_VAL := BYTE_TO_ASC(EN:=%MX0, IN:=BYTE_VAL);
```

(1) 실행조건(%MX0)가 On 되면 BYTE_TO_ASC 펑션이 동작합니다.
(2) 펑션의 입력 변수로 선언된 BYTE_VAL(BYTE 타입) = 16#3A 일 경우, 펑션의 출력 변수로 선언된 ASCII_VAL(WORD 타입) = 16#3341 = '3', 'A' 가 출력됩니다.

BYTE_WORD	적용 기종	발생플래그
2 개의 BYTE 를 WORD 로 모음	XGI, XGR, XEC	-

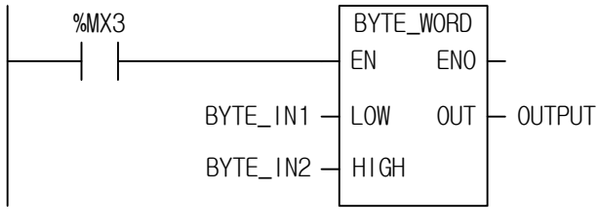
펑 션	설 명
	**입력** EN : 1 일 때 펑 션 실행 LOW : 하위 BYTE 입력 HIGH : 상위 BYTE 입력 **출력** ENO : EN 값을 그대로 출력 OUT : WORD 출력

■ 기능

1. 2 개의 바이트를 하나의 워드로 조합합니다.
   LOW: 하위 바이트 입력, HIGH: 상위 바이트 입력

■ 프로그램 예

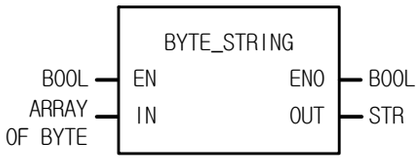
1. LD



2. ST

```
OUTPUT := BYTE_WORD(EN:=%MX3, LOW:=BYTE_IN1, HIGH:=BYTE_IN2);
```

(1) 실행조건(%MX3)이 On 되면 BYTE_WORD 펑션이 실행됩니다.
(2) 입력변수로 선언된 BYTE_IN1 = 16#56 이고 BYTE_IN2 = 16#AD 이면 출력변수로 선언된 OUTPUT = 16#AD56 입니다.

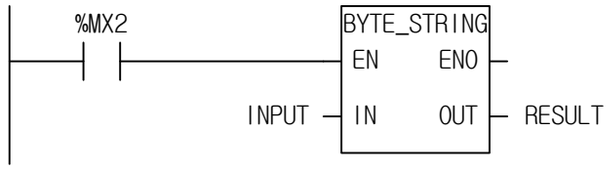
BYTE_STRING	적용 기종	발생플래그
Byte Array 를 문자열변환	XGI, XGR, XEC	-
펑 션	설 명	
	**입력** EN : 1일 때 펑션 실행 IN : Byte Array 입력 **출력** ENO : EN 값을 그대로 출력 OUT : 변환된 String 출력	

■ 기능

1. Byte Array 를 하나의 String 으로 변환합니다.

■ 프로그램 예

1. LD

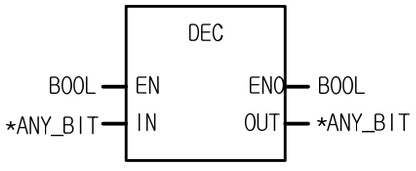


2. ST

```
RESULT := BYTE_STRING(EN:=%MX2, IN:=INPUT);
```

(1) 실행조건(%MX2)이 On 되면 BYTE_STRING 펑션이 실행됩니다.
(2) 입력 INPUT 어레이 변수를 3 개를 설정했을 때 INPUT[0] = 16#41, INPUT[1] = 16#31, INPUT[2] = 16#35 를 입력하면 출력 RESULT = 'A15' 입니다.

DEC	적용 기종	발생플래그
데이터를 하나 감소	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 감소시킬 입력 데이터 **출력** ENO : EN값이 그대로 출력 OUT : 감소시킨 결과 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*ANY_BIT: ANY_BIT 타입 중 BOOL 제외

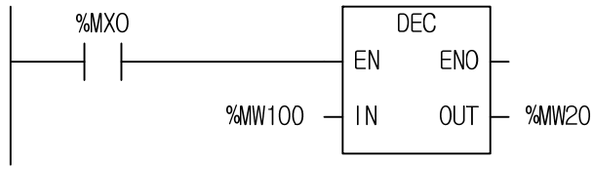
■ 기능

1. EN이 1이면, IN의 비트스트링 데이터를 1만큼 감소시켜서 OUT으로 출력합니다.
2. 언더플로우가 발생해도 에러는 발생하지 않으며, 결과는 16#0000인 경우에 16#FFFF가 됩니다.
3. 입력에는 BYTE, WORD, DWORD, LWORD 타입의 데이터가 접속 가능합니다.

FUNCTION	IN/OUT 변수 타입	동작 설명
DEC	BYTE	입출력 데이터에 맞추어 4가지 타입 중 하나를 사용합니다.
DEC	WORD	
DEC	DWORD	
DEC	LWORD	

■ 프로그램 예

1. LD



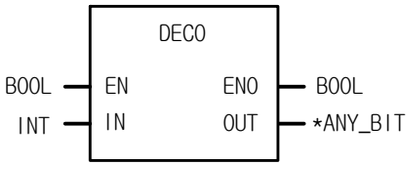
2. ST

```
%MW20 := DEC(EN:=%MX0, IN:=%MW100);
```

(1) 실행조건(%MX0)이 On 되면 DEC 펑션이 실행됩니다.

(2) 입력 변수로 선언된 %MW100 = 16#0007(2#0000_0000_0000_0111) 이라면, 연산이 실행된 후에는 %MW20 = 16#0006(2#0000_0000_0000_0110)이 됩니다.

DECO	적용 기종	발생플래그
지정된 비트 위치를 On	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : Decoding할 입력 데이터 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : Decoding한 결과 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*ANY_BIT: ANY_BIT 타입 중 BOOL 제외

■ 기능

1. EN 이 1 이면, IN 의 값 즉 비트 위치지정 데이터에 따라서 출력의 비트 스트링 데이터 중 지정된 위치의 비트만 1로 하여 출력합니다.
2. 출력에는 BYTE, WORD, DWORD, LWORD 타입의 데이터가 접속 가능합니다.

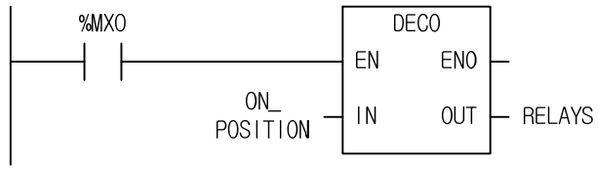
FUNCTION	OUT 변수 타입	동작 설명
DECO	BYTE	출력 테이터에 맞추어 4가지 타입 중 하나를 사용합니다.
DECO	WORD	
DECO	DWORD	
DECO	LWORD	

■ 플래그

플래그	설명
_ERR	입력데이터가 음수이거나, 비트 위치지정 데이터가 출력타입의 비트한계를 넘으면(DECO 의 WORD 타입인 경우 16 이상), OUT 은 0 이 되고 _ERR, _LER 플래그가 셋(SET)됩니다.

■ 프로그램 예

1. LD



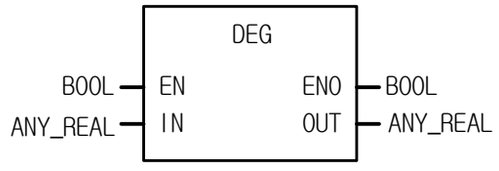
2. ST

```
RELAYS := DECO(EN:=%MX0, IN:=ON_POSITION);
```

(1) 실행조건(%MX0)이 On 되면 DECO 펑션이 실행됩니다.

(2) 입력변수로 선언된 ON_POSITON(INT 타입) = 5 라면, 출력의 5 번 비트만 On 되므로,
RELAYS(WORD 타입) = 2#0000_0000_0010_0000 이 됩니다.

DEG	적용 기종	발생플래그
Radian 값을 각도로 변환	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 IN : 라디안(Radian)값 입력 **출력** ENO : EN 값을 그대로 출력 OUT : 각도 출력

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

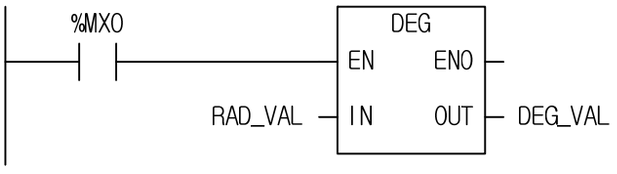
■ 기능

1. 라디안(Radian) 값을 입력 받아서 각도(Degree)로 출력 합니다.

펑션	입력 타입	출력 타입	동작 설명
DEG	REAL	REAL	라디안(Radian)값을 출력 데이터 타입에 해당하는 각도 값으로 변환 합니다.
DEG	LREAL	LREAL	

■ 프로그램 예

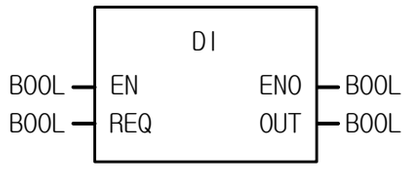
1. LD



2. ST

```
DEG_VAL := DEG(EN:=%MX0, IN:=RAD_VAL);
```

(1) 실행조건(%MX0)이 On 되면 DEG 펑션이 실행됩니다.
(2) 펑션의 입력변수로 선언된 RAD_VAL = 1.0일 경우 펑션의 출력변수 DEG_VAL = 5.72957795130785 50E+001가 됩니다.

DI	적용 기종	발생플래그
태스크 프로그램 기동 불허	XGI, XGR, XEC	-
펑 션	설 명	
	**입력** EN : 1 일 때 펑션 실행 REQ : 태스크 프로그램 기동 불허 요구 **출력** ENO : EN 값이 그대로 출력 OUT : DI 동작이 실행되면 1 출력	

■ 기능

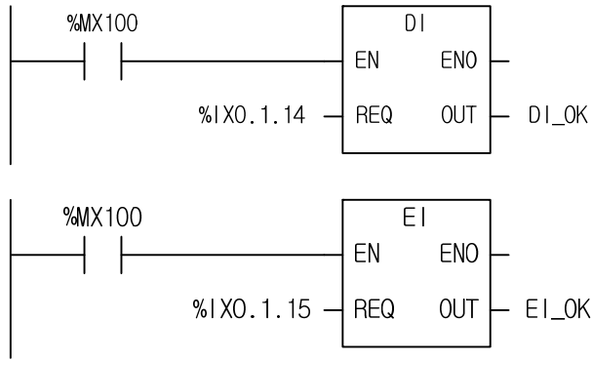
1. EN 이 1 이고 REQ 에 1 의 값이 들어오면 사용자에 의해서 작성된 태스크 프로그램(싱글, 인터벌, 인터럽트)의 기동을 막습니다.
2. 한번 'DI'명령이 수행되면 REQ 입력이 0 이 되어도 태스크 프로그램은 기동되지 않습니다.
3. 태스크 프로그램이 정상적으로 기동하도록 할 때는 'EI' 펑션을 사용하여 주십시오.
   프로그램의 수행 도중에 타 태스크 프로그램의 수행으로 연산 처리의 연속성을 잃을 경우 문제가 되는 부분에 대하여 부분적으로 태스크 프로그램의 수행을 막고자 할 때 사용할 수 있습니다.
4. 태스크 프로그램의 기동불허 상태에서 발생한 태스크들은 태스크 종류에 따라 다음과 같이 수행됩니다.
   - 싱글 태스크: 'EI'펑션 수행 후 또는 현재 수행 중인 태스크 프로그램의 종료 후에 수행됩니다.
     이때 싱글 변수의 상태 변화 횟수만큼 태스크 프로그램을 반복하여 수행합니다.
   - 인터벌 태스크, 인터럽트: 태스크 프로그램의 기동 불허 상태에서 발생한 태스크는 'EI'펑션 수행 후 또는 현재 수행 중인 태스크 프로그램의 종료 후에 수행됩니다. 그러나, 태스크가 2 번 이상 발생한 경우에는 태스크 충돌 경고(TASK_ERR)가 발생하고, 충돌 횟수(TC_CNT)를 Count 합니다.

■ 프로그램 예

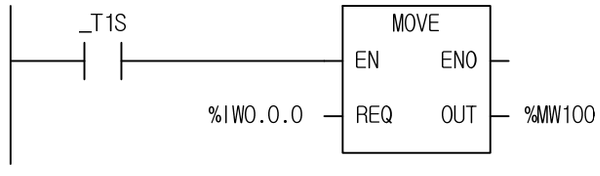
1 초마다 값을 증가하는 태스크 프로그램을 태스크 프로그램 기동 불허 펑션 DI 와 태스크 프로그램 기동허가 펑션 EI 를 이용하여 제어하는 프로그램

1. LD

Scan 프로그램(TASK 프로그램 제어)



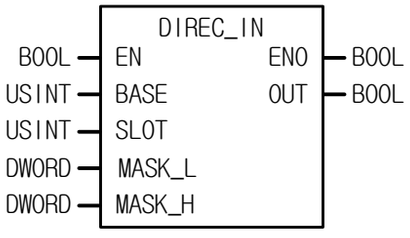
1 초마다 실행하여 증가하는 태스크 프로그램



2. ST

```
Scan 프로그램(TASK 프로그램 제어)
  %IX0.1.14 := DI(EN:=%MX100, REQ:=DI_OK);
  %IX0.1.15 := EI(EN:=%MX100, REQ:=EI_OK);

1 초마다 실행하여 증가하는 태스크 프로그램
  %MW100 := MOVE(EN:=_T1S, IN:=%IW0.0.0);
```

(1) DI(태스크 프로그램 기동불허 펑션)의 기동불허 요구인 REQ (직접변수 %IX0.1.14 로 지정)가 On 되면 펑션 DI 가 실행되어 출력 변수로 설정된 DI_OK 의 값이 1 이 됩니다.
(2) 펑션 DI 가 실행되면 1 초마다 실행되던 태스크 프로그램의 실행이 정지됩니다.
(3) EI (태스크 프로그램 기동허가 펑션)의 기동허가 요구인 REQ (직접변수 %IX0.1.15 로 지정)가 On 되면 펑션 EI 가 실행되어 출력 변수로 설정한 EI_OK 의 값이 1 이 됩니다.
(4) 펑션 EI 가 실행하면 펑션 DI 로 정지되었던 태스크 프로그램이 재실행 됩니다.

DIREC_IN	적용 기종	발생플래그
입력데이터 즉시갱신	XGI, XGR, XEC	_ERR, _LER
펑 션	설 명	
	**입력** EN : 1일 때 펑션 실행 BASE : 입력모듈이 장착된 베이스의 위치번호 SLOT : 입력모듈이 장착된 슬롯의 위치번호 MASK_L : 입력하위 32 비트 데이터 중 갱신하지 않을 비트 지정 MASK_H : 입력상위 32 비트 데이터 중 갱신하지 않을 비트 지정 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 입력데이터 갱신이 완료되면 1출력	

■ 기능

1. 펑션 DIREC_IN(입력 데이터 즉시 갱신)은 스캔 도중에 EN 이 1 이 되면 BASE, SLOT 에 지정된 위치의 입력 모듈의 64 비트 데이터를 읽어서 입력 이미지 영역에 갱신하여 넣습니다.
2. 이때 이미지 영역에는 해당 슬롯에 꽂혀있는 입력모듈의 점수만큼만 갱신됩니다.
3. 펑션 DIREC_IN 은 스캔 중에 입력(%I)의 On/Off 상태를 변화시키고 싶을 때 사용이 가능합니다.
4. 통상 스캔 동기 일괄처리방식은 입력데이터 읽기와 출력 데이터의 출력을 스캔 프로그램의 종료 후에 일괄 처리하기 때문에, 1Scan 도중에 외부로부터 의 입력된 데이터의 갱신이 불가능합니다.
5. 펑션 DIREC_IN 을 사용하면 프로그램 실행도중에 해당하는 입력을 갱신할 수 있습니다

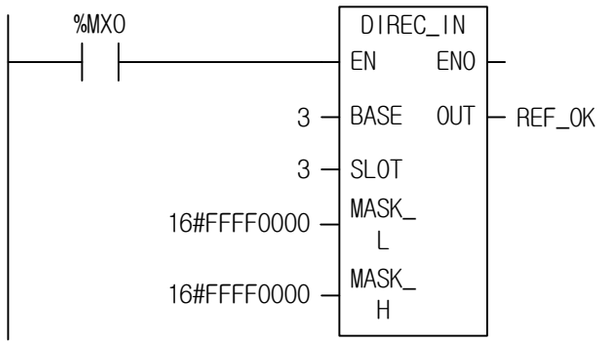
■ 플래그

플래그	설명
_ERR	BASE, SLOT 입력 값이 범위를 초과하거나 입/출력 데이터 갱신 시 에러가 발생되면 결과를 출력하지 않고 에러플래그 _ERR, _LER이 셋(SET)됩니다.

■ 프로그램 예

1. 3 번째 증설 Base, 3 번 Slot 에 꽂힌 모듈이 16 점 모듈이고, 입력 데이터가 2#1010_1010_1110_1011 로 즉시 갱신하는 프로그램

1. LD



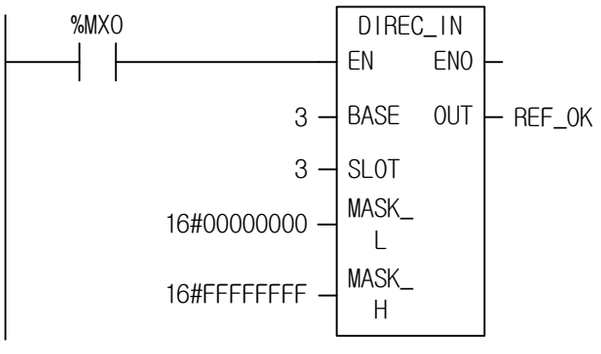
2. ST

```
REF_OK := DIREC_IN(EN:=%MX0, BASE:=3, SLOT:=3, MASK_L:=16#FFFF0000, MASK_H:=16#FFFF0000);
```

(1) 입력조건(%MX0)가 On 되면 DIREC_IN(입력데이터 즉시 갱신) 펑션이 실행합니다.
(2) 장착된 모듈이 16 점 모듈이므로 갱신대상 이미지 영역은 %IW3.3.0 이 되고 갱신하지 않을 비트의 지정변수 MASK_L(입력하위 32 비트)의 값에서 하위 16Bit 가 갱신허용으로 설정되어 있으므로 %IW3.3.0 은 스캔 도중 2#1010_1010_1110_1011 로 갱신됩니다.
(3) 비트의 지정 변수는 MASK_H(입력 상위 32 비트)의 설정 값은 현재 설정된 베이스, 슬롯에 16 점 모듈이 꽂혀 있으므로 무시됩니다.

2. 3 번째 증설 Base, 3 번 Slot 에 꽂힌 모듈이 32 점 모듈이고, 입력 데이터가 2#0000_0000_1111_1111_1100_1100_0011_0011 일 때 하위 32Bit 를 즉시 갱신하는 프로그램

1. LD



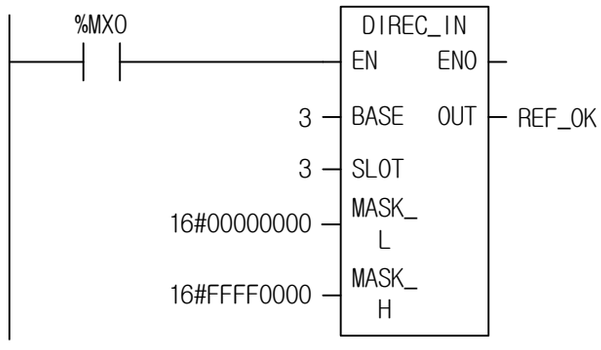
2. ST

```
REF_OK := DIREC_IN(EN:=%MX0, BASE:=3, SLOT:=3, MASK_L:=16#00000000, MASK_H:=16#FFFFFFFF);
```

(1) 입력조건(%MX0)가 On 되면 DIREC_IN(입력데이터 즉시 갱신) 펑션이 실행합니다.
(2) 장착된 모듈이 32 점 모듈이므로 갱신대상 이미지 영역은 %ID3.3.0 이 되나, 갱신하지 않을 때 비트 지정 MASK_L(입력하위 32 비트)의 값에서 하위 32 비트가 갱신허용으로 설정되어 있으므로 %ID3.3.0 이 2#0000_0000_1111_1111_1100_1100_0011_0011 로 갱신됩니다.

3. 3 번째 증설 Base, 3 번 Slot 에 꽂힌 모듈이 64 점 모듈이고 입력데이터가 16#0000_FFFF_AAAA_7777 (2#0000_0000_0000_0000_1111_1111_1111_1111_1010_1010_1010_1010_0111_0111_0111_0111)일 때 64 비트 중 하위 48 비트만 즉시 갱신할 프로그램.

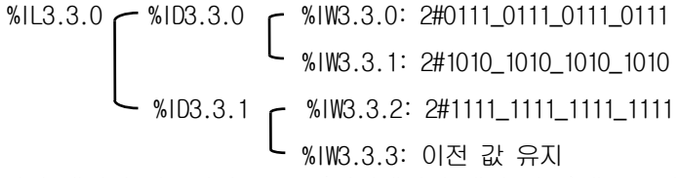
1. LD



2. ST

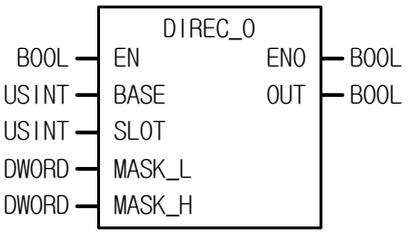
```
REF_OK := DIREC_IN(EN:=%MX0, BASE:=3, SLOT:=3, MASK_L:=16#00000000, MASK_H:=16#0000FFFF);
```

(1) 입력조건(%MX0)가 On 되면 DIREC_IN(입력데이터 즉시 갱신) 펑션이 실행합니다.
(2) 장착된 모듈이 64 점이므로 갱신 이미지 영역은 %IL3.3.0 즉 %ID3.3.0 과 %ID3.3.1 이 됩니다.
(3) 하위 32 비트(MASK_L) 모두는 갱신 허용으로 되어있으므로 %ID3.3.0 은 모두 갱신됩니다.
(4) 상위 32 비트(MASK_H)는 이중 하위 16Bit 만 갱신 허용으로 되어 있으므로 %ID3.3.1 은 %IW.3.3.2 는 갱신되고, %IW3.3.3 은 갱신되지 않습니다.
(5) 따라서 이미지 영역의 데이터의 갱신은 아래와 같습니다.



(6) 입력 갱신이 완료되면 REF_OK(입력데이터 갱신 완료)에는 1 이 출력됩니다.

DIREC_O	적용 기종	발생플래그
출력모듈 데이터 즉시갱신	XGI, XGR, XEC	_ERR, _LER
펑 션	설 명	



입력

EN : 1일 때 펑션 실행

BASE : 출력모듈이 장착된 베이스의 위치번호

SLOT : 출력모듈이 장착된 슬롯의 위치번호

MASK_L : 출력하위 32 비트 데이터 중 갱신하지 않을 비트 지정

MASK_H : 출력상위 32 비트 데이터 중 갱신하지 않을 비트 지정

출력 ENO : 에러 없이 실행되면 1을 출력

OUT : 출력데이터 갱신이 완료되면 1 출력

■ 기능

1. 펑션 DIREC_O(출력데이터 즉시 갱신)는 스캔 도중에 EN(DIREC_O 실행 조건)이 1 이 되면 BASE 와 SLOT 이 지정된 위치의 출력모듈의 64 비트 데이터를 읽어서 MASK(1)되지 않은 비트만을 출력모듈에 즉시 출력합니다.
2. 펑션 DIREC_O 는 1 스캔 중에 출력(Q 영역)의 On/Off 상태를 변화시키고 싶을 때 사용이 가능합니다.
3. 통상 스캔 중에 일괄 처리 방식은 입력 데이터 읽기와 출력 데이터의 출력을 스캔 프로그램의 종료 후에 일괄 처리하기 때문에 1 스캔 도중에 외부로 신호를 출력하는 것이 불가능합니다.
4. 펑션 DIREC_O 를 사용하면 프로그램 실행도중에 해당하는 비트의 데이터를 외부로 출력하는 것이 가능합니다.
5. 해당위치에 다른 타입의 모듈이 꽂혀 있거나, 출력모듈에 데이터가 정상적으로 써지지 않으면 ENO 와 OUT 을 0 으로 출력합니다. (정상 동작 시 1 출력)

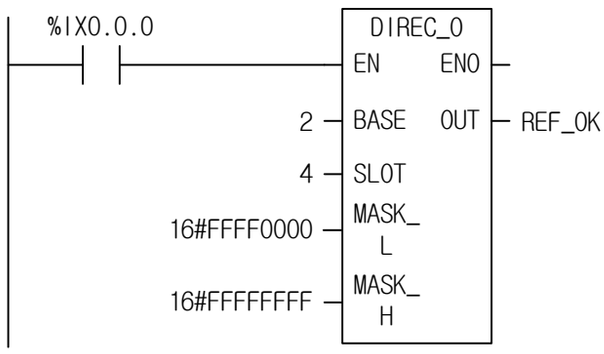
■ 플래그

플래그	설명
_ERR	BASE, SLOT 입력 값이 범위를 초과하거나 입/출력 데이터 갱신 시 에러가 발생되면 결과를 출력하지 않고 에러플래그 _ERR, _LER이 셋(SET)됩니다.

■ 프로그램 예

1. 2 번째 증설베이스 4 번 Slot 에 장착된 32 점 Relay 출력 모듈에 스캔 도중 즉시 출력하고 싶은 임의의 출력 데이터 값이 2#0111_0111_0111_0111 을 출력하는 프로그램.

1. LD



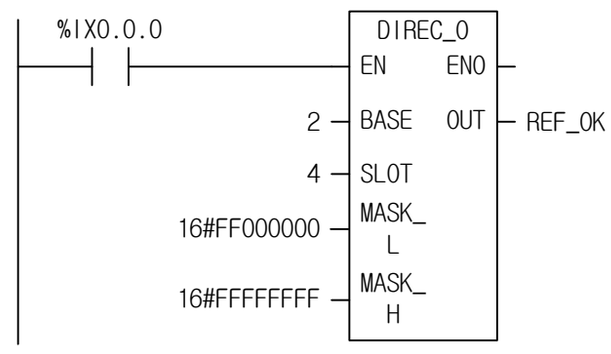
2. ST

```
REF_OK := DIREC_0(EN:=%IX0.0.0, BASE:=2, SLOT:=4, MASK_L:=16#FFFF0000, MASK_H:=16#FFFFFFFF);
```

(1) 출력모듈이 장착된 Base 의 위치번호 2 와 SLOT 번호 4 를 입력합니다.
(2) 스캔 도중 출력하고자 하는 데이터가 16 비트이므로 MASK_L 의 값 중 하위 16 비트만 출력 허용 값으로 설정합니다. (16#FFFF_0000)
(3) 실행조건(%IX0.0.0)이 On 되면 DIREC_0(출력모듈 데이터 즉시 갱신) 펑션이 실행되어 스캔 도중에 출력 모듈의 데이터가 2#0111_0111_0111_0111 로 출력됩니다.

2. 2 번째 증설베이스 4 번 Slot 에 장착된 32 점 TR 출력 모듈 중 하위 24 비트만 스캔 도중 임의의 출력데이터 값을 2#1111_0000_1111_0000_1111_0000 로 변경 출력하는 프로그램.

1. LD



2. ST

```
REF_OK := DIREC_0(EN:=%IX0.0.0, BASE:=2, SLOT:=4, MASK_L:=16#00000000, MASK_H:=16#FFFFFFFF);
```

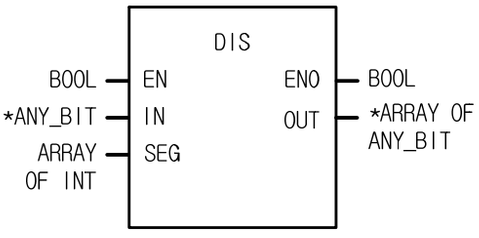
(1) 출력모듈이 장착된 Base 의 위치번호 2 와 SLOT 번호 4 를 입력합니다.

(2) 스캔 도중 출력 하고자 하는 데이터가 24 비트이므로 MASK_L 의 값 중 하위 24 비트만 출력 허용 값으로 설정합니다. (16#FF00_0000)

(3) 실행 조건(%IX0.0.0)가 On 되면 DIREC_O(출력모듈 데이터 즉시 갱신) 펑션이 실행되어, 스캔 도중에 출력 모듈의 데이터가 2#□□□□_□□□□_1111_0000_1111_0000_1111_0000 로 출력됩니다.

이전 값 유지

DIS	적용 기종	발생플래그
데이터 분산(Distribution)	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 입력 데이터 SEG : 데이터 분산 비트수 지정 어레이 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 분산된 어레이 출력

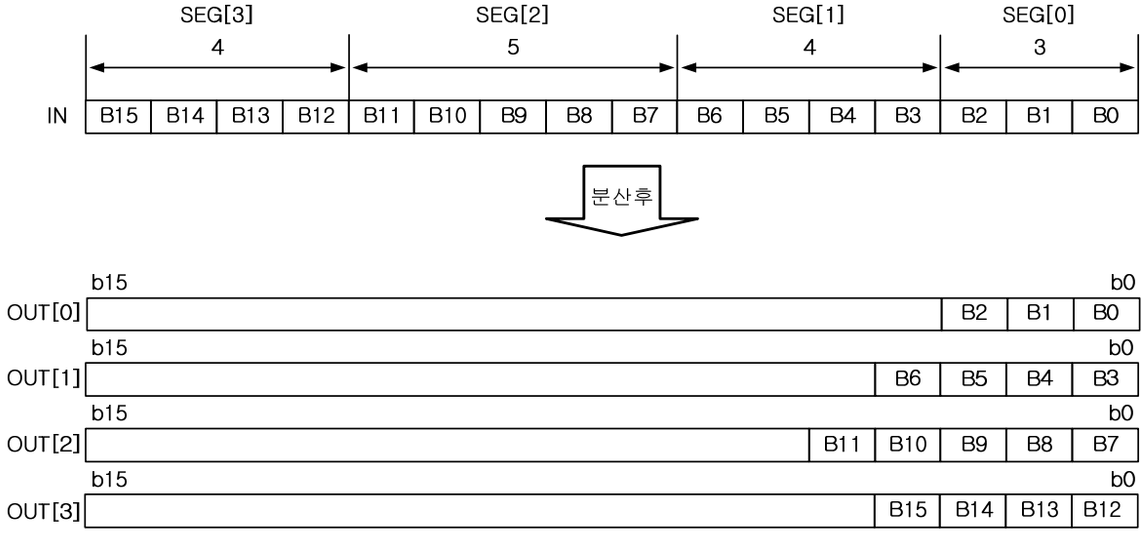
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*ANY_BIT: ANY_BIT 타입 중 BOOL 제외

■ 기능

1. 입력 데이터를 SEG 에서 지정된 비트 개수 단위로 구분하여 OUT 으로 출력합니다.

펑션	입력타입	동작 설명
DIS	BYTE	각 타입에 해당하는 IN 입력을 SEG 입력 어레이 값으로 비트열을 구분하여 IN과 동일한 데이터 타입의 OUT 어레이로 출력합니다.
DIS	WORD	
DIS	DWORD	
DIS	LWORD	



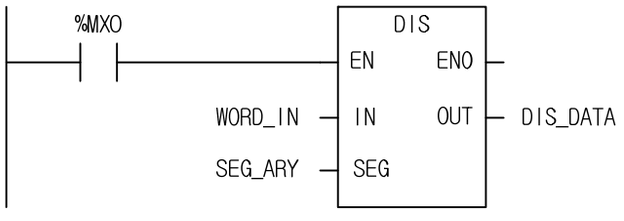
■ 플래그

플래그	설명
_ERR _LER	SEG로 지정된 값들의 합이 입력 변수 타입의 비트 수 보다 크거나 작을 경우 _ERR, _LER 에러 플래그가 셋(SET)됩니다.

☆ 출력단 Array 생략시 어레이의 개수를 0 으로 보아 _ERR, _LER 플래그가 발생합니다.

■ 프로그램 예

1. LD

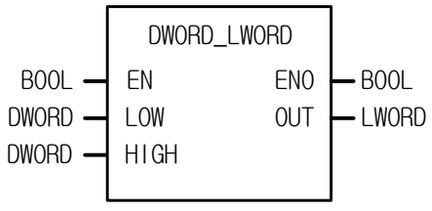


2. ST

```
DIS_DATA := DIS(EN:=%MX0, IN:= WORD_IN, SEG:=SEG_ARY);
```

(1) 실행조건(%MX0)이 On 되면, DIS 펑션이 실행됩니다.

(2) 입력변수로 선언된 WORD_IN(WORD 타입)의 값이 16#3456 이고,SEG_ARY={3,4,5,4}이면, 펑션이 실행된 후에 DIS_DATA 로 출력되는 값은 DIS_DATA[0]=16#0006
DIS_DATA[1]=16#000A
DIS_DATA[2]=16#0008
DIS_DATA[3]=16#0003 가 출력됩니다.

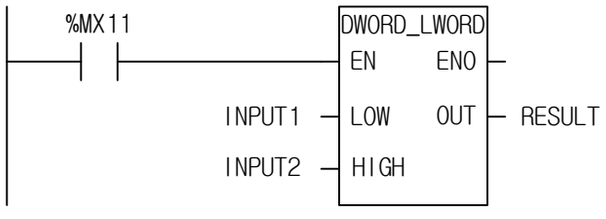
DWORD_LWORD	적용 기종	발생플래그
2 개의 DWORD 를 LWORD 로 모음	XGI, XGR, XEC	-
펑 션	설 명	
	**입력** EN : 1 일 때 펑션 실행 LOW : 하위 DWORD 입력 HIGH : 상위 DWORD 입력 **출력** ENO : EN 값을 그대로 출력 OUT : LWORD 출력	

■ 기능

1. 2 개의 DWORD 를 하나의 LWORD 로 조합합니다.
   LOW: 하위 더블워드 입력, HIGH: 상위 더블워드 입력

■ 프로그램 예

1. LD

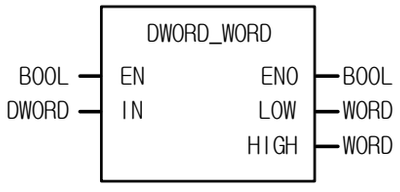


2. ST

```
RESULT := DWORD_LWORD(EN:=%MX11, LOW:=INPUT1, HIGH:=INPUT2);
```

(1) 실행조건(%MX11)이 On 되면 DWORD_LWORD 펑션이 실행됩니다.
(2) 입력변수로 선언된 INPUT1=16#1A2A_3A4A이고, INPUT2=16#8C7C_6C5C 일 때 출력변수로 선언된 RESULT = 16#8C7C_6C5C_1A2A_3A4A가 출력됩니다.

DWORD_WORD	적용 기종	발생플래그
DWORD 를 2 개의 WORD 로 나눔	XGI, XGR, XEC	-

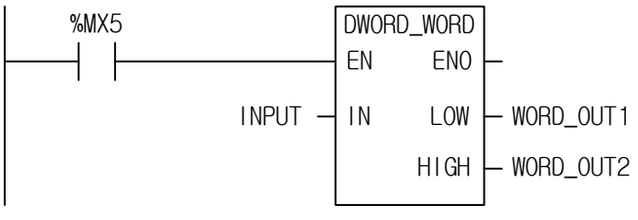
펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 IN : DWORD 입력 **출력** ENO : EN 값을 그대로 출력 LOW : 하위 WORD 출력 HIGH : 상위 WORD 출력

■ 기능

1. 하나의 DWORD 를 2 개의 WORD 로 분산합니다.
   LOW: 하위 워드 출력, HIGH: 상위 워드 출력

■ 프로그램 예

1. LD

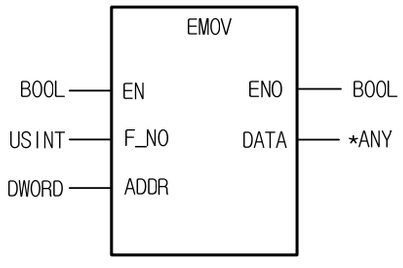


2. ST

```
DWORD_WORD(EN:=%MX5, IN:=INPUT, LOW=>WORD_OUT1, HIGH=>WORD_OUT2);
```

(1) 실행조건(%MX5)이 On 되면 DWORD_WORD 펑션이 실행됩니다.
(2) 입력변수로 선언된 INPUT=16#1122_AABB 일 때 입출력 변수로 선언된 WORD_OUT1 = 16#AABB,
WORD_OUT2 = 16#1122이 저장됩니다.

EMOV	적용 기종	발생플래그
설정한 플래쉬 영역으로부터 데이터 읽기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** REQ : 1일 때 펑션 실행 F_NO : Move할 데이터가 있는 블록 NO(0~31). ADDR : B_NO로 설정된 블록의 바이트 주소 **출력** ENO : 에러 없이 수행되면 1을 출력 DATA : 데이터 저장 영역 (BOOL과 STRING을 제외한 모든 변수 사용가능)

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	DATA		○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ANY: ANY 타입 중 BOOL, STRING 제외

■ 기능

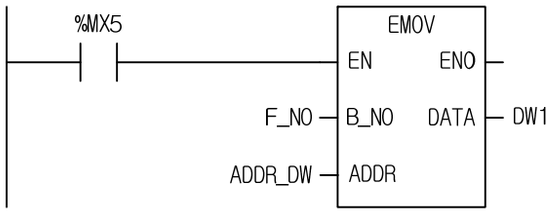
1. 플래시 메모리에 있는 32개의 블록의 데이터 중 원하는 데이터 하나를 MOVE하는 명령입니다.
2. 설정한 F_NO(플래시 넘버)의 블록에서 ADDR 위치의 데이터를 DATA 에 설정된 타입에 맞춰 MOVE 합니다. 이때 MOVE 된 데이터는 DATA 변수에 들어갑니다.
3. DATA로 선언한 변수의 타입과 ADDR 변수의 타입이 맞지 않을 경우, 에러는 없으나 의도하지 않은 데이터가 MOVE 되니, DATA 타입에 맞춰 ADDR 값을 설정해야 합니다. 예를 들어, 4BYTE(DWORD, UDINT, DINT, REAL ...) 형태의 변수를 DATA에 선언했다면, ADDR 변수도 4BYTE형 변수를 사용해야 합니다.
4. F_NO이 31이상이거나, ADDR값이 65,535를 초과할 경우, _ERR, _LER이 SET 됩니다.

■ 플래그

플래그	설명
_ERR	F_NO값이 31 이상이거나, ADDR 값이 65,535 초과할 경우

■ 프로그램 예

1. LD

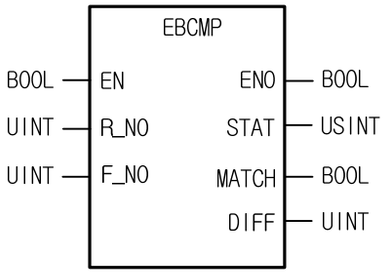


2. ST

```
EMOV(EN:=%MX5, F_NO:= F_NO, ADDR:= ADDR_DW, DATA=> DW1);
```

(1) 실행조건(%MX5)이 On 되면 EMOV 펑션이 실행됩니다.
(2) F_NO = 1, ADDR_DW(DWORD 타입) = 4로 설정할 때 1번 플래시 블록의 4BYTE OFFSET 위치의 DWORD DATA를 DW1(DWORD)에 이동합니다.

EBCMP	적용 기종	발생플래그
내용 비교 후 일치 여부 확인	XGI, XGR, XEC	-

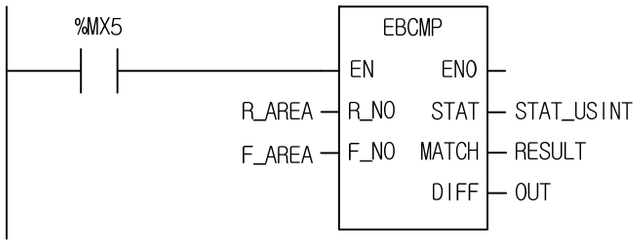
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 R_NO : R 디바이스의 블록 번호 F_NO : 플래시 메모리의 블록 번호 **출력** ENO : 비교 동작 완료시 On STAT : 에러 상태 MATCH: 비교한 결과가 같으면 On DIFF : 불일치 개수 (DWORD 단위)

■ 기능

1. 입력 접점이 On 되어 있는 동안 R 디바이스의 한 블록과 플래시 메모리의 한 블록의 내용을 비교하여 일치 여부를 확인하는 명령어 입니다. 비교시 DWORD 단위로 데이터를 비교합니다.
2. STAT 는 에러상태를 나타내며 R_NO 입력시 1 보다 크면 STAT = 1, F_NO 입력시 31 보다 크면 STAT = 2, 전체 비교 후 한 개의 오류가 나타나도 STAT = 3 의 에러를 표시합니다.
3. 불일치할 경우에는 DIFF 에 불일치 개수를 저장합니다.

■ 프로그램 예

1. LD

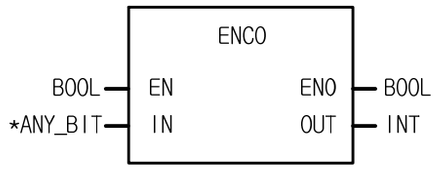


2. ST

```
EBCMP(EN:=%MX5, R_NO:=R_AREA, F_NO:=F_AREA, STAT=>STAT_USINT, MATCH=>RESULT, DIFF=>OUT);
```

(1) 실행조건(%MX5)이 On되면 EBCMP 펑션이 실행됩니다.
(2) R_AREA = 0, F_AREA = 1로 설정할 때 R 디바이스 블록번호 0번 내용과 플래시 블록번호 1번 내용이 동일하면 RESULT(BOOL)는 On되며 OUT(불일치 개수) = 0을 나타냅니다.

ENCO	적용 기종	발생플래그
On 된 비트 위치를 숫자로 출력	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 IN : Encoding 할 입력 데이터 **출력** ENO : 에러 없이 실행되면 1 을 출력 OUT : Encoding 한 결과 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT							○													

*ANY_BIT: ANY_BIT 타입 중 BOOL 제외

■ 기능

1. EN 이 1 이면, IN 의 비트 스트링 데이터 중, 1 로 되어있는 비트 중 최상위 비트의 위치를 OUT 으로 출력합니다.
2. 입력에는 B(BYTE), W(WORD), D(DWORD), L(LWORD) 타입의 데이터가 접속 가능합니다.

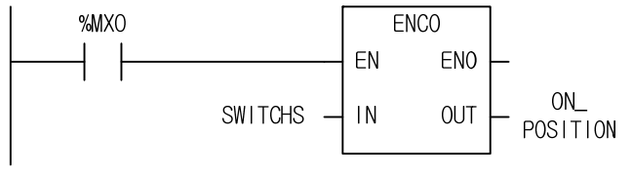
FUNCTION	IN 변수 타입	동작 설명
ENCO	BYTE	각 입력 변수 타입에 따라 원하는 ENCO 펑션의 타입을 사용합니다.
ENCO	WORD	
ENCO	DWORD	
ENCO	LWORD	

■ 플래그

플래그	설명
_ERR	입력데이터 중 하나의 비트도 1 이 되어있지 않은 경우는 OUT 은 -1 이 되고, _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예

1. LD

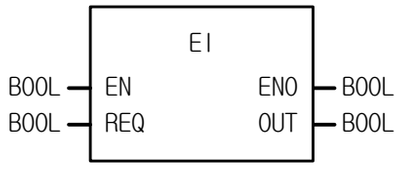


2. ST

```
ON_POSITION := ENCO(EN:=%MX0, IN:=SWITCHS);
```

(1) 실행조건(%MX0)이 On 되면 ENCO 펑션이 실행됩니다.

(2) SWITCHS(WORD 타입) = 2#0000_1000_0000_0010 이라면, On 되어 있는 2 비트의 위치, 즉 '11' 과 '1' 중 상위 위치인 '11' 을 출력하여 ON_POSITON(INT 타입)에 정수값 '11' 이 저장됩니다.

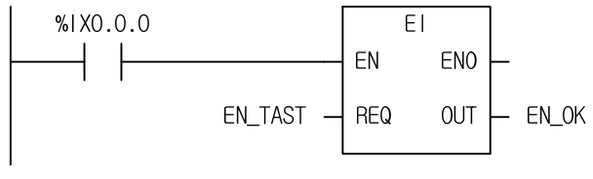
EI	적용 기종	발생플래그
태스크 프로그램 기동허가 (DI 의 해제)	XGI, XGR, XEC	-
펑 션	설 명	
	**입력** EN : 1 일 때 펑션 실행 REQ : 태스크 프로그램 기동 허가 요구 **출력** ENO : EN 값이 그대로 출력 OUT : EI 동작이 실행되면 1 출력	

■ 기능

1. EN 이 1 이고 REQ 에 1 의 값이 들어오면 'DI'펑션에 의해 막혀있던 태스크 프로그램이 정상적으로 기동합니다.
2. 한번 'EI'펑션이 수행되면 REQ 입력이 0 이 되어도 태스크 프로그램은 정상 기동합니다.
3. 태스크 프로그램의 기동불허 상태에서 발생한 태스크들은 'EI'펑션 수행 후 또는 현재 수행중인 태스크 프로그램의 종료 후에 수행됩니다.

■ 프로그램 예

1. LD



2. ST

```
EN_OK := EI(EN:=%IX0.0.0, REQ:=EN_TAST);
```

(1) EN_TASK 가 1 이 되면 태스크 프로그램이 정상 기동 됩니다.
(2) 'EI'펑션에 의해서 태스크 실행 허가 상태가 되면 EN_OK 에는 1 이 출력됩니다.

ESTOP	적용 기종	발생플래그
프로그램에 의한 비상 운전정지	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 REQ : 프로그램에 의한 비상 운전정지 요구 **출력** ENO : EN 값이 그대로 출력. 기능 1번 참조 OUT : ESTOP 동작이 실행되면 1 출력

■ 기능

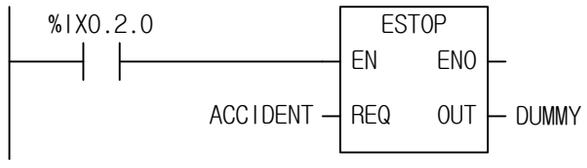
1. 펑션 실행 조건 EN 이 1 이고, 프로그램에 의한 비상 운전정지 요구 신호 REQ 가 1 이 되면 현재 수행중인 프로그램의 운전을 즉시 강제 정지하고 STOP 모드로 갑니다. 이 경우 ENO 값은 On 되지 않습니다. 그 이유는 즉시 강제 정지하기 때문입니다.
2. 'ESTOP'펑션에 의해 정지된 경우는 전원을 재 투입하여도 기동되지 않습니다.
3. 운전모드를 STOP 으로 하였다가 RUN 으로 하면 재 기동이 됩니다.
4. 'ESTOP' 펑션이 수행되면 수행중인 프로그램을 중도에 중지하기 때문에 재 기동시 콜드 리스타트 모드가 아닐 경우 데이터의 연속성에 오류가 있을 수 있습니다.

■ 관련 플래그

플래그	설명
_ESTOP_ON	ESTOP 명령어에 의해 STOP 된 경우 On 됩니다. 이 상태에서 다시 런 진입시 Off 됩니다.

■ 프로그램 예

1. LD



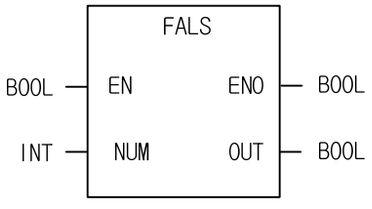
2. ST

```
DUMMY := ESTOP(EN:=%IX0.2.0, REQ:=ACCIDENT);
```

(1) 실행조건(%IX0.2.0)이 On 되면 프로그램에 의한 비상 운전정지 펑션 'ESTOP' 이 실행됩니다.

(2) 펑션 'ESTOP' 의 ACCIDENT 가 1 이 되면 실행중인 프로그램을 즉시 중지하고 STOP 모드로 됩니다.

※ 비상 사태 발생시 기계적 인터럽트와 함께 이중 안전장치로 사용할 수 있습니다.

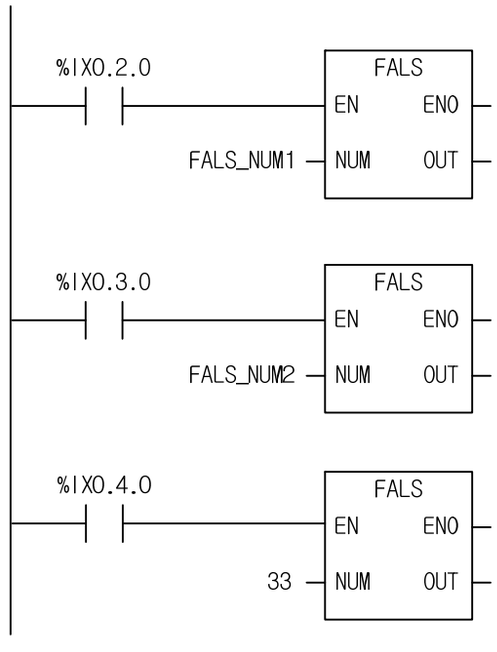
FALS	적용 기종	발생플래그
사용자가 정한 상수(N)를 F 영역의 지정된 주소에 저장(_FALS_NUM)	XGI, XGR, XEC	-
펑 션	설 명	
	**입력** EN : 1 일 때 펑션 실행 NUM : F 영역에 저장될 번호 **출력** ENO : EN 값이 그대로 출력 OUT : 정상 동작시 On 출력	

■ 기능

1. 사용자가 정한 상수(N)를 F 영역의 지정된 주소(_FALS_NUM)에 저장합니다.
2. NUM 은 16#0000 ~ 16#FFFF 까지 지정이 가능하며 해제되기 전까지는 최초에 발생한 NUM 이 저장됩니다.
3. FALS 의 해제는 FALS 0000 으로 실행합니다.

■ 프로그램 예

1. LD



2. ST

```
OUT1 := FALS(EN:=%IX0.2.0, NUM:=FALS_NUM1);
OUT2 := FALS(EN:=%IX0.3.0, NUM:=FALS_NUM2);
OUT3 := FALS(EN:=%IX0.4.0, NUM:=33);
```

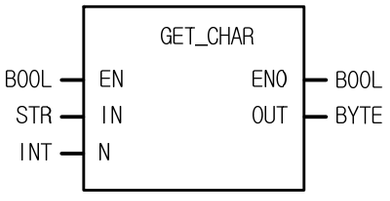
(1) 실행조건이 On 되면 각 FALS 펑션이 실행됩니다. (예: FALS_NUM1=31, FALS_NUM2=32)

(2) 해당되는 실행조건(%IX0.2.0, %IX0.3.0, %IX0.4.0)에 따라 _FALS_NUM 플래그에 값이 저장되며 최초 _FALS_NUM 플래그에 값이 저장되면 그 다음 값은 FALS 의 해제를 하기 전까지는 저장되지 않습니다.

(3) FALS 의 해제는 NUM 값에 0000 을 설정하여 실행하면 됩니다.

(4) FALS 펑션은 특수 상황에 따른 해당되는 값을 설정 함으로서 프로그램 실행 후 _FALS_NUM 플래그를 확인하여 해당되는 부분의 상태를 보기에 편리합니다.

GET_CHAR	적용 기종	발생플래그
문자열로부터 한 문자(CHAR)추출	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 IN : STRING 입력 N : STRING 내의 위치 지정 **출력** ENO : EN 값을 그대로 출력 OUT : BYTE 출력

■ 기능

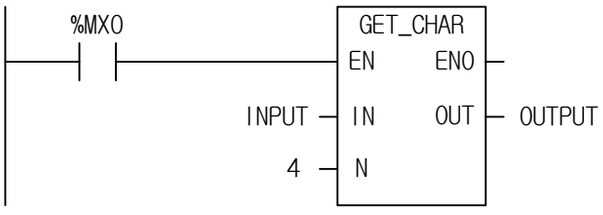
1. STRING 의 지정된 위치로부터 하나의 바이트를 추출합니다.

■ 플래그

플래그	설명
_ERR	N값이 스트링의 바이트 개수를 넘는 경우 _ERR, _LER 플래그가 셋(SET)됩니다. 에러가 발생했을 경우 16#00이 출력됩니다.

■ 프로그램 예

1. LD



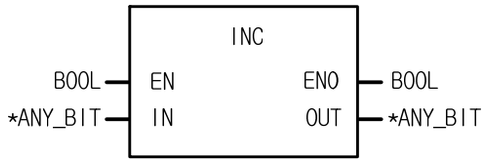
2. ST

```
OUTPUT := GET_CHAR(EN:=%MX0, IN:=INPUT, N:=4);
```

(1) 실행조건(%MX0)이 On 되면, GET_CHAR 펑션이 실행됩니다.

(2) 입력 변수로 선언된 INPUT(STRING 타입) = “LS XGI PLC” 일 때 이 String 의 4 번째 문자를 추출하게 되면 출력 변수로 선언된 OUTPUT 으로 16#58(“X”)가 출력됩니다.

INC	적용 기종	발생플래그
데이터를 하나 증가	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 증가시킬 입력 데이터 **출력** ENO : EN 값이 그대로 출력 OUT : 증가시킨 결과 데이터

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*ANY_BIT: ANY_BIT 타입 중 BOOL 제외

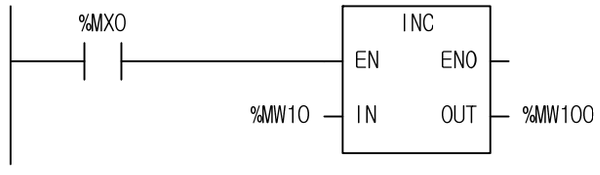
■ 기능

1. EN 이 1 이면, IN 의 비트스트링 데이터를 1 만큼 증가시켜서 OUT 으로 출력합니다.
2. 오버플로우가 발생해도 에러는 발생하지 않으며, 결과는 16#FFFF 인 경우에 16#0000 이 됩니다.
3. 입력에는 BYTE, WORD, DWORD, LWORD 타입의 데이터가 접속 가능합니다.

FUNCTION	IN/OUT 변수 타입	동작 설명
INC	BYTE	입출력 데이터에 맞추어 4 가지 펑션 중 하나를 사용합니다.
INC	WORD	
INC	DWORD	
INC	LWORD	

■ 프로그램 예

1. LD

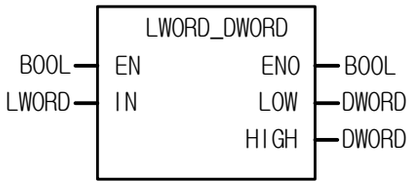


2. ST

```
%MW100 := INC(EN:=%MX0, IN:=%MW10);
```

(1) 실행조건(%MX0)이 On 되면 INC 펑션이 실행됩니다.

(2) 입력변수 %MW10 = 16#0007(2#0000_0000_0000_0111) 이라면, 연산이 실행된 후에는
%MW100 = 16#0008(2#0000_0000_0000_1000)이 됩니다.

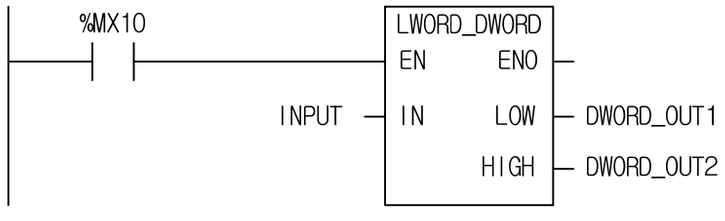
LWORD_DWORD	적용 기종	발생플래그
LWORD 를 2 개의 DWORD 로 나눔	XGI, XGR, XEC	-
펑 션	설 명	
	**입력** EN : 1 일 때 펑션 실행 IN : LWORD 입력 **출력** ENO : EN 값을 그대로 출력 LOW : 하위 DWORD 출력 HIGH : 상위 DWORD 출력	

■ 기능

1. 하나의 LWORD 를 2 개의 DWORD 로 분산합니다.
   LOW: 하위 더블워드 출력, HIGH: 상위 더블워드 출력

■ 프로그램 예

1. LD

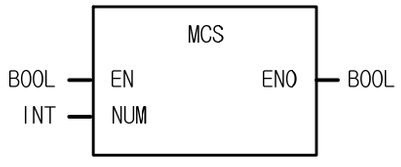


2. ST

```
LWORD_DWORD(EN:=%MX10, IN:=INPUT, LOW=>DWORD_OUT1, HIGH=>DWORD_OUT2);
```

(1) 실행조건(%MX10)이 On 되면, LWORD_DWORD 펑션이 실행됩니다.

(2) 입력변수로 선언된 INPUT=16#AAAA_BBBB_CCCC_DDDD 일 때, 출력 변수로 선언된 DWORD_OUT1=16#CCCC_DDDD, DWORD_OUT2=16#AAAA_BBBB 가 출력됩니다.

MCS	적용 기종	발생플래그
Master Control	XGI, XGR, XEC	-
펑 션	설 명	
	**입력** EN : 1 일 때 펑션 실행 NUM : Nesting (0~15) **출력** ENO : MCS 명령이 실행되면 1 을 출력	

■ 기능

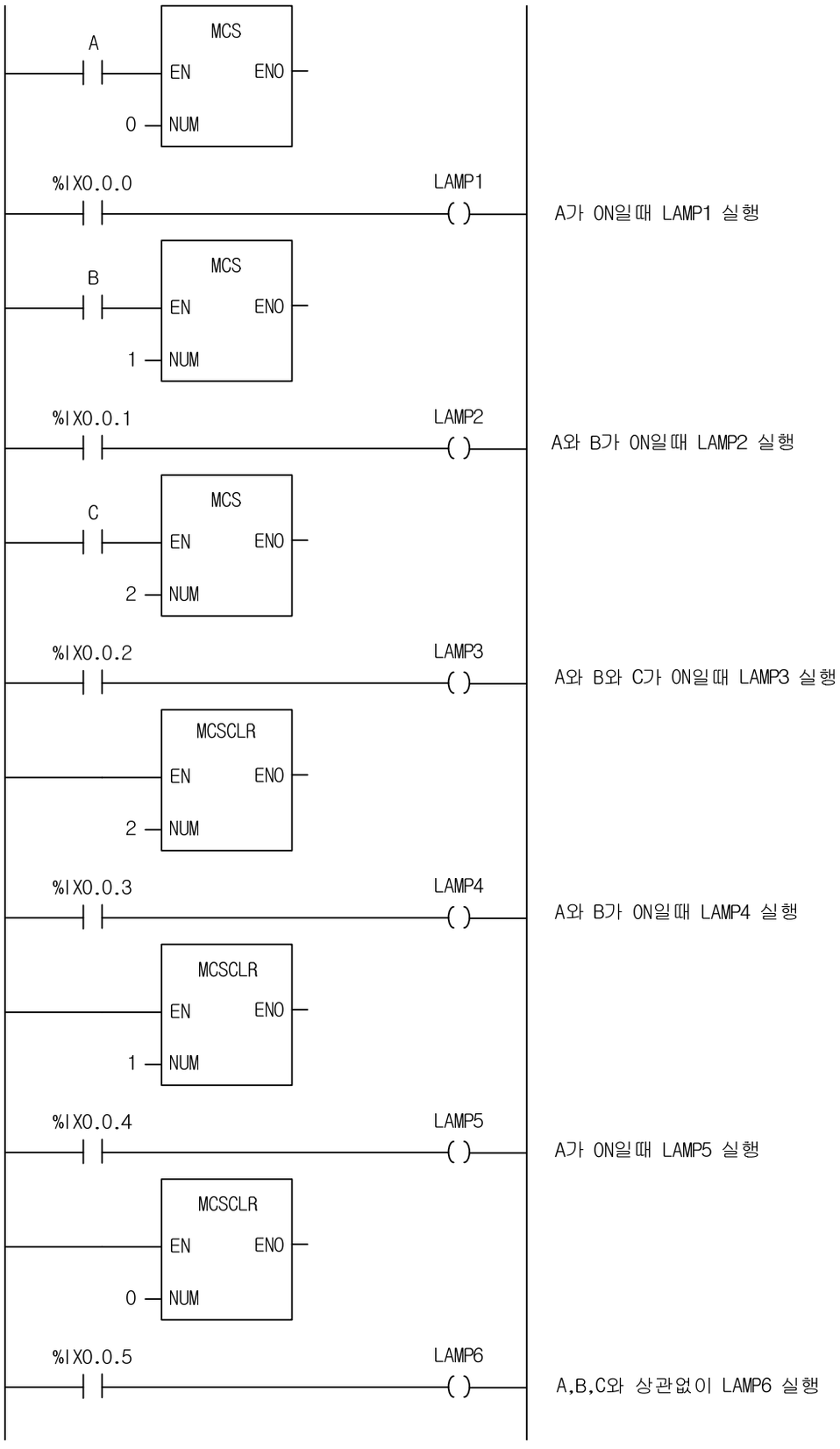
1. EN 이 On 이면, Master Control 이 수행됩니다. 이 경우, MCS 펑션에서 MCSCLR 펑션 사이의 프로그램은 정상적으로 수행됩니다.
2. EN 이 Off 인 경우, MCS 펑션에서 MCSCLR 펑션 사이의 프로그램은 아래와 같이 수행됩니다.

명령어	명령어 상태
Timer	현재값은 0 이 되고, 출력(Q)은 Off 됩니다.
Counter	출력(Q)은 Off 되고, 현재값은 현재 상태를 유지합니다.
코일	모두 Off 됩니다.
역코일	모두 Off 됩니다.
셋코일, 리셋코일	현재 값을 유지합니다.
펑션, 펑션 블록	현재 값을 유지합니다.

3. EN 이 Off 인 경우에도 MCS 펑션에서 MCSCLR 펑션 사이의 명령들이 위와 같이 수행되기 때문에 스캔 타임이 감소되지 않습니다.
4. Master Control 명령은 Nesting 해서 사용될 수 있습니다. 즉, Master Control 영역이 Nesting(NUM)에 의해 구분될 수 있습니다. Nesting(NUM)은 0 에서 15 까지 설정이 가능하고, 만약 16 이상으로 설정한 경우 Master Control 이 정상적으로 동작하지 않습니다.
5. MCSCLR 없이 MCS 명령을 사용한 경우, MCS 펑션에서 프로그램의 마지막 행까지 Master Control 이 수행되니 주의 바랍니다.

■ 프로그램 예

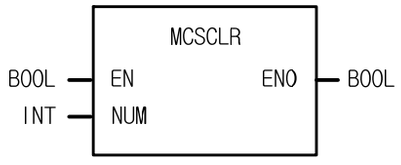
1. LD


A
MCS
EN
ENO
0
NUM
%IX0.0.0
LAMP1
A가 ON일때 LAMP1 실행
B
MCS
EN
ENO
1
NUM
%IX0.0.1
LAMP2
A와 B가 ON일때 LAMP2 실행
C
MCS
EN
ENO
2
NUM
%IX0.0.2
LAMP3
A와 B와 C가 ON일때 LAMP3 실행
MCSCLR
EN
ENO
2
NUM
%IX0.0.3
LAMP4
A와 B가 ON일때 LAMP4 실행
MCSCLR
EN
ENO
1
NUM
%IX0.0.4
LAMP5
A가 ON일때 LAMP5 실행
MCSCLR
EN
ENO
0
NUM
%IX0.0.5
LAMP6
A,B,C와 상관없이 LAMP6 실행

2. ST

```
MCS(EN:=A, NUM:=0);
LAMP1 := %IX0.0.0;          // A 가 ON 일 때 LAMP1 실행

MCS(EN:=B, NUM:=1);
LAMP2 := %IX0.0.1;          // A 와 B 가 ON 일 때 LAMP1 실행

MCS(EN:=C, NUM:=2);
LAMP3 := %IX0.0.2;          // A 와 B 와 C 가 ON 일 때 LAMP1 실행

MCSCLR(NUM:=2);
LAMP4 := %IX0.0.3;          // A 와 B 가 ON 일 때 LAMP1 실행

MCSCLR(NUM:=1);
LAMP5 := %IX0.0.4;          // A 가 ON 일 때 LAMP1 실행

MCSCLR(NUM:=0);
LAMP6 := %IX0.0.5;          // A, B, C 상관없이 ON 일 때 LAMP1 실행
```

(1) 각각의 MCS 펑션의 NUM 에 해당하는 값은 같은 NUM 에 해당하는 MCSCLR 과 짝을 이루어 영역을 설정합니다.
NESTING(NUM)은 0~15 까지 설정될 수 있으며 그 이상으로 설정 할 수 없습니다. MCS 와 MCSCLR 펑션을 같이 짝을 이루어 사용하지 않으면 프로그램의 마지막까지 MCS 펑션이 실행되므로 주의바랍니다.

MCSCLR	적용 기종	발생플래그
Master Control 해제 명령	XGI, XGR, XEC	-
펑 션	설 명	
	**입력** EN : 1일 때 펑션 실행 NUM : Nesting (0~15) **출력** ENO : MCSCLR 명령이 실행되면 1을 출력	

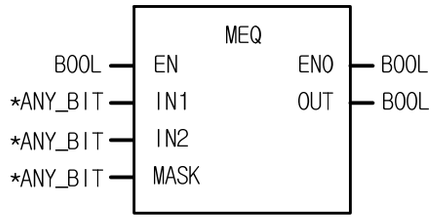
■ 기능

1. Master Control 명령을 해제합니다. 그리고, Master Control 영역의 마지막을 가리킵니다.
2. MCSCLR 펑션 동작시 Nesting(NUM)의 값보다 같거나 작은 모든 MCS 명령을 해제합니다.
3. MCSCLR 펑션 앞에는 접점을 사용하지 않습니다.

■ 프로그램 예

MCS 펑션의 프로그램 예를 참조 바랍니다.

MEQ	적용 기종	발생플래그
Masked Equal	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 IN1 : 입력 1 IN2 : 입력 2 MASK : masking 할 입력값 **출력** ENO : EN 값을 그대로 출력 OUT : 동일하면 1 을 출력

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1		○	○	○	○															
	IN2		○	○	○	○															
	MASK		○	○	○	○															

*ANY_BIT: ANY_BIT 타입 중 BOOL 제외

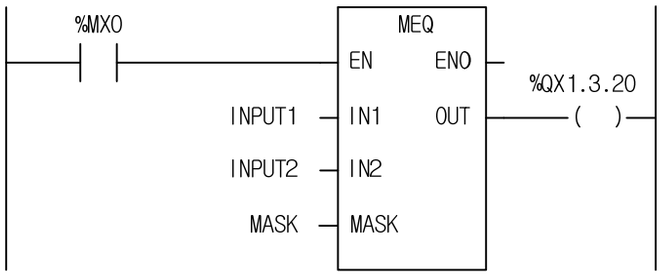
■ 기능

1. 입력된 변수값들을 Masking 한 후 두개의 변수값이 서로 동일한지를 비교합니다. 만약 8 비트 변수값을 2#1111_1100 으로 Masking 하면 하위 2 비트는 입력값 비교에서 제외됩니다.
2. 하나의 변수값에서 특정 비트들이 On 되어 있는지 검사하는 용도로도 사용이 가능합니다. 즉, 8 비트 변수 비교시 IN1 에 비교하고자 하는 변수를 입력하고 IN2 에는 16#FF 로 설정한 후 MASK 에 검색하고자 하는 비트조합을 입력하면(i.e. 2#0010_1100) 입력되는 변수와 비교하여 동일한 경우 On 이 출력됩니다.

펑션	입력변수 타입	동작 설명
MEQ	BYTE	입력값들을 Masking한 후 이 값들이 서로 동일한 지를 비교합니다.
MEQ	WORD	
MEQ	DWORD	
MEQ	LWORD	

■ 프로그램 예

1. LD



2. ST

```
%QX1.3.20 := MEQ(EN:=%MX0, IN1:=INPUT1, IN2:=INPUT2, MASK:=MASK);
```

(1) 실행조건(%MX0)이 On 되면, MEQ_BYTE 펑션이 실행됩니다.

(2) 입력변수 INPUT1(BYTE 타입) = 2#0101_1100

INPUT2(BYTE 타입) = 2#0111_0101

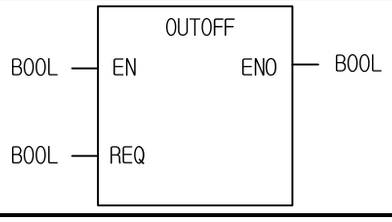
MASK(BYTE 타입) = 2#1101_0110 일 경우 Making 된 후의 입력 변수들의 비교할 비트는

INPUT1(BYTE 타입) = 2#0101_0100

INPUT2(BYTE 타입) = 2#0101_0100

과 같이 되어 서로 동일한 값을 가지므로 출력접점 %QX1.3.20 이 On 됩니다.

OUTOFF	적용 기종	발생플래그
입력 조건이 성립하면 전 출력을 Off	XGI, XGR, XEC	-

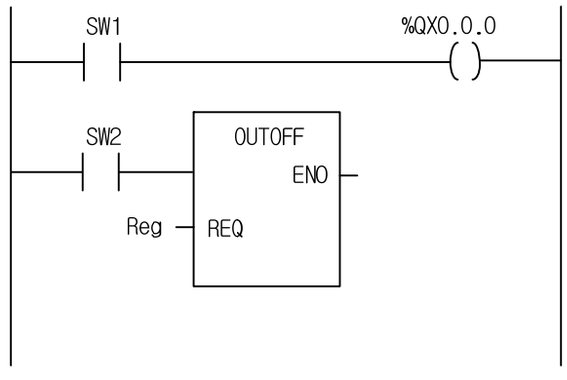
펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 REQ : 프로그램에 의한 전 출력 정지 **출력** ENO : 동작 여부 확인

■ 기능

1. EN=1 이고, REQ=1 이면 전 출력을 Off 시킵니다.
2. EN=1 이고, REQ=0 이면 전 출력 Off 를 해제합니다.
3. 그 외의 경우에는 이전 상태가 유지됩니다.

■ 프로그램 예

1. LD



2. ST

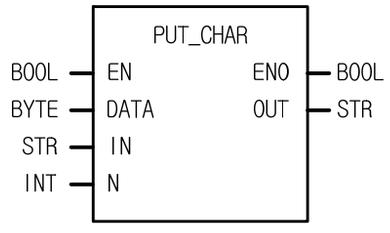
```
%QX0.0.0 := SW1;
OUTOFF(EN:=SW2, REQ:= Reg);
```

(1) 출력 모듈을 장착한 후 위의 예와 같이 프로그램을 구성합니다.

(2) SW1 을 On 시키면 출력(%QX0.0.0)이 SET 됩니다.

(3) SW2 를 On 시킨 후 Reg = 1 로 동작 시키면 OUTOFF 펑션이 실행이 되면서 모든 출력 모듈이 Off 됩니다.
프로그램 모니터 상으로는 셋(SET)되어있는 것 같이 보이나 실제 출력 모듈은 Off 되어 있습니다.

PUT_CHAR	적용 기종	발생플래그
문자열에 한 문자 써넣기	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 DATA : STRING 에 삽입할 BYTE 입력 IN : STRING 입력 N : STRING 내의 위치 지정 **출력** ENO : EN 값을 그대로 출력 OUT : STRING 출력

■ 기능

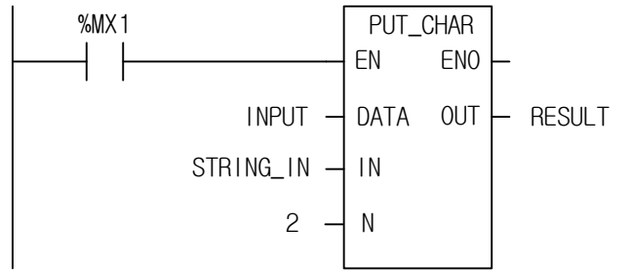
1. 하나의 바이트 입력값을 STRING 상의 지정된 위치(N 숫자)에 덮어쓰기(Overwrite)합니다.

■ 플래그

플래그	설명
_ERR	N값이 스트링의 바이트 개수를 넘는 경우 _ERR, _LER 플래그가 셋(SET)됩니다. 에러가 발생했을 경우 16#00이 출력됩니다.

■ 프로그램 예

1. LD



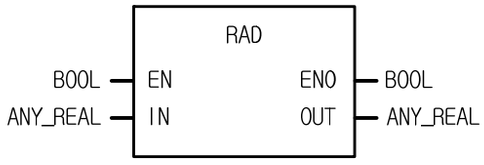
2. ST

```
RESULT := PUT_CHAR(EN:=%MX1, DATA:= INPUT, IN:= STRING_IN, N:=2);
```

(1) 실행조건(%MX1)이 On 되면 PUT_CHAR 펑션이 실행됩니다.

(2) 입력변수인 INPUT = 16#41(“A”)이고 STRING_IN = “TOKEN” 일 때 입력 변수 STRING_IN 의 2 번째 위치에 입력변수 INPUT 이 지닌 값을 덮어쓰기 하게 되면 출력 변수인 RESULT 는 “TAKEN” 이 됩니다.

RAD	적용 기종	발생플래그
각도(DEG)를 Radian 값으로 변환	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 각도 입력 **출력** ENO : EN값을 그대로 출력 OUT : 라디안 값 출력

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

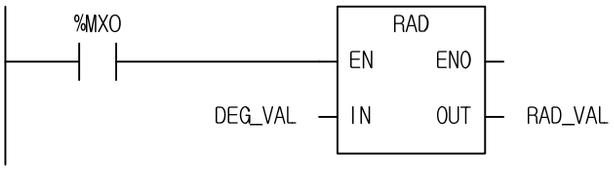
■ 기능

1. 각도의 단위를 도(°)에서 라디안(Radian) 값으로 출력 합니다.
2. 각도가 360°를 넘어서더라도 정상적으로 변환시켜 줍니다.
   (EX: 입력이 370°이면 출력은 360°를 뺀 10°에 해당하는 라디안 값을 출력합니다.)

펑션	입력 타입	출력 타입	동작 설명
RAD	REAL	REAL	각도의 단위를 도(°)에서 출력 데이터 타입에 해당하는 라디안 값으로 변환합니다.
RAD	LREAL	LREAL	

■ 프로그램 예

1. LD

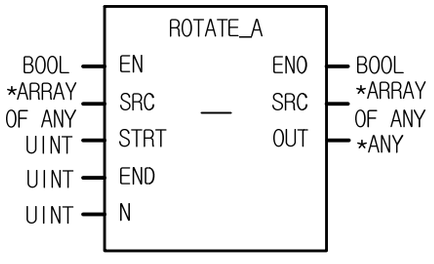


2. ST

```
RAD_VAL := RAD(EN:=%MX0, IN:= DEG_VAL);
```

(1) 실행조건(%MX0)이 On 하면, RAD_REAL 펑션이 실행됩니다.
(2) 펑션의 입력 변수로 선언된 DEG_VAL = 127(°)일 경우, 펑션의 출력변수 값은 RAD_VAL = 2.21656823 이 됩니다.

ROTATE_A	적용 기종	발생플래그
지정된 어레이 원소의 ROTATE	XGI, XGR, XEC	_ERR, _LER

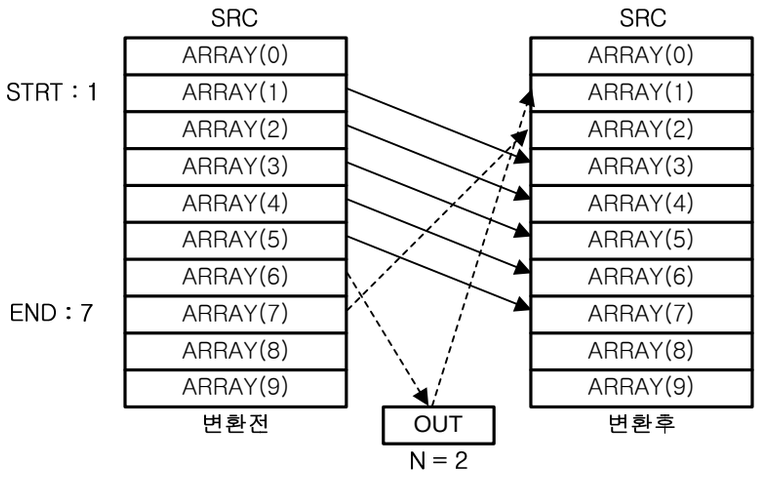
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 N : Rotate 시킬 개수 STRT : 어레이 블록 중 Rotate 할 원소의 시작위치 END : 어레이 블록 중 Rotate 할 원소의 종료위치 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : Rotate 하여 밀려나온 데이터를 출력 **입출력** SRC : Rotate 조작이 가해질 어레이 블록

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	SRC	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ANY: ANY 타입 중 STRING 제외

■ 기능

1. ROTATE_A 펑션은 어레이 블록 중 지정된 범위의 원소들을 지정된 방향으로 이동 시킵니다.
2. 동작의 지정:
   A. 범위지정: STRT 와 END 로 이동할 데이터 원소의 범위를 지정합니다.
   B. 이동방향 및 횟수: STRT 에서 END 방향으로 지정된 횟수(N)만큼 Rotate 됩니다.
   C. 입력 데이터 지정: Rotate 하여 비워지는 위치를 END 에서 밀려나온 데이터로 채웁니다.
   D. 출력: 데이터 조작의 결과는 SRC 에 지정된 ARRAY 에 저장되며, END 위치에서 Rotate 동작으로 STRT 로 돌아들어가는 데이터는 OUT 으로 출력됩니다.



펑션	입출력 어레이 타입	동작 설명
ROTATE_A	BOOL	각 타입 어레이의 지정된 범위의 원소들을 지정된 방향으로 ROTATE 시킵니다.
ROTATE_A	BYTE	
ROTATE_A	WORD	
ROTATE_A	DWORD	
ROTATE_A	LWORD	
ROTATE_A	SINT	
ROTATE_A	INT	
ROTATE_A	DINT	
ROTATE_A	LINT	
ROTATE_A	USINT	
ROTATE_A	UINT	
ROTATE_A	UDINT	
ROTATE_A	ULINT	
ROTATE_A	REAL	
ROTATE_A	LREAL	
ROTATE_A	TIME	
ROTATE_A	DATE	
ROTATE_A	TOD	
ROTATE_A	DT	

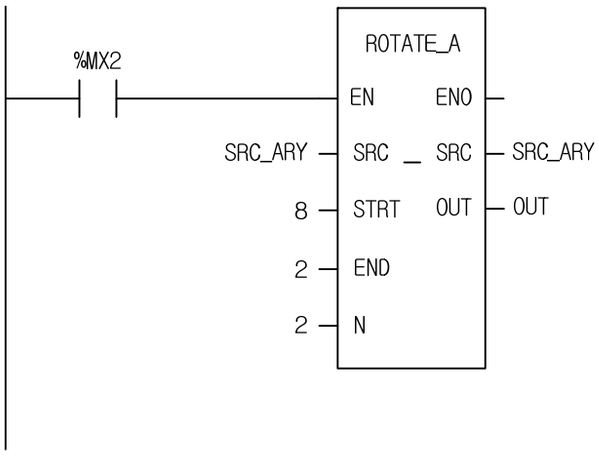
■ 플래그

플래그	설명
_ERR	STRT 또는 END 값이 SRC 어레이의 원소 개수 범위를 벗어나면 _ERR, _LER 플래그가 셋(Set)됩니다. 에러발생시 SRC 는 변하지 않고 출력은 각 변수타입의 초기화값(i.e. INT = 0, TIME = T#0S)이 출력됩니다.

☆ 출력단 Array 변수생략시 출력단 어레이의 개수를 0 으로 보아 _ERR, _LER 플래그가 발생합니다.

■ 프로그램 예

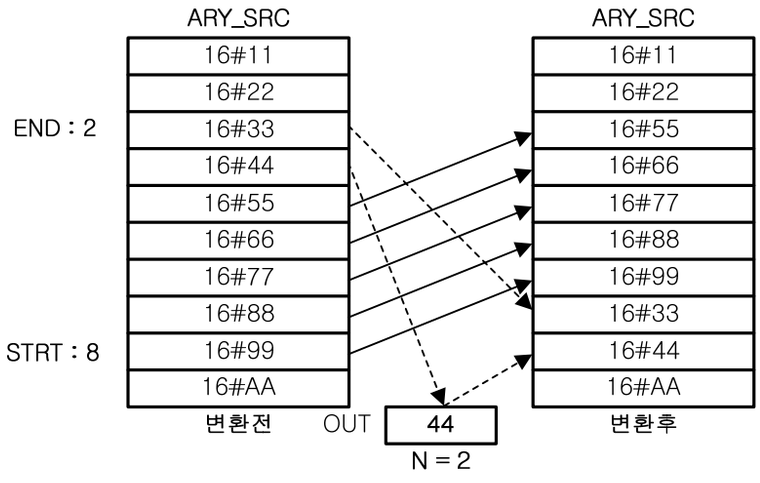
1. LD



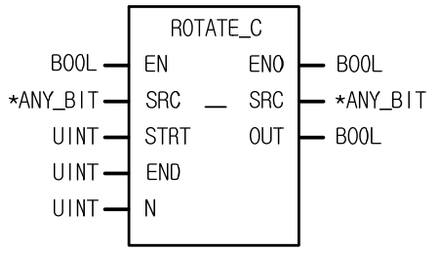
2. ST

```
OUT := ROTATE_A(EN:=%MX2, SRC:=SRC_ARY, STRT:=8, END:=2, N:=2);
```

(1) 입력 조건(%MX2)이 On 되면, ROTATE_A 펑션이 실행됩니다.
(2) 입출력 변수로 선언된 SRC_ARY 의 인덱스 8 의 원소부터 인덱스 2 의 원소의 방향으로 ROTATE 동작이 2 번 일어납니다.
(3) 출력값에는 캐리 출력에 해당하는 원소의 값 16#44 가 출력됩니다.


ARY_SRC
16#11
16#22
END : 2
16#33
16#44
16#55
16#66
16#77
16#88
STRT : 8
16#99
16#AA
변환전
OUT
44
N = 2
ARY_SRC
16#11
16#22
16#55
16#66
16#77
16#88
16#99
16#33
16#44
16#AA
변환후

ROTATE_C	적용 기종	발생플래그
Rotate with Carry	XGI, XGR, XEC	_ERR, _LER

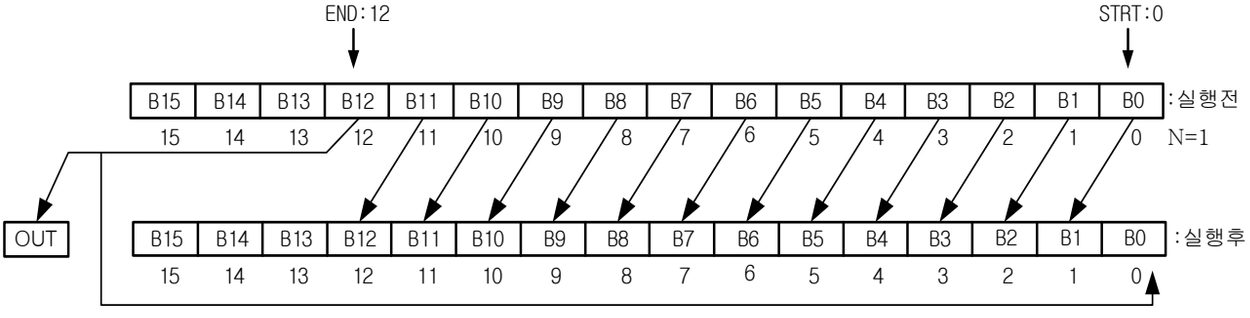
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 STRT: SRC의 비트열 중 회전할 범위의 시작bit 위치 END : SRC의 비트열 중 회전할 범위의 끝 bit 위치 N : Shift할 비트수 **출력** ENO : 에러 없이 수행되면 1을 출력 OUT : Carry 출력 **입출력** SRC : 회전할 변수

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	SRC		○	○	○	○															

*ANY_BIT: ANY_BIT 타입 중 BOOL 제외

■ 기능

1. SRC의 비트열 중 지정된 범위의 원소들을 지정된 방향으로 회전합니다.
2. 동작의 지정:
   A. 범위지정: STRT와 END로 이동할 비트의 범위를 지정합니다.
   B. 이동방향 및 횟수: STRT에서 END방향으로 지정된 횟수(N)만큼 회전됩니다.
   C. 출력: 데이터 조작의 결과는 SRC에 지정된 ANY_BIT에 바뀌어 저장되며, 회전동작으로 END위치에서 STRT로 돌아 들어가는 비트 데이터는 OUT으로도 출력됩니다.



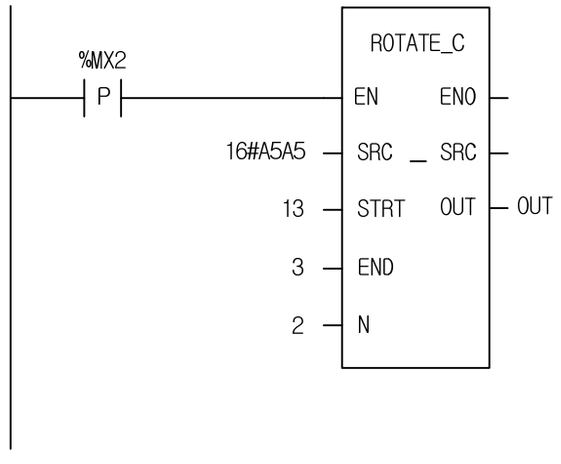
펑션	SRC 변수 타입	동작 설명
ROTATE_C	BYTE	입력값의 지정된 범위에 해당하는 비트들을 지정된 횟수만큼 Rotate 시킵니다.
ROTATE_C	WORD	
ROTATE_C	DWORD	
ROTATE_C	LWORD	

■ 플래그

플래그	설명
_ERR	STRT와 END값이 SRC 변수 타입의 비트 개수를 넘는 경우 SRC 값에는 변화가 없으며 _ERR, _LER 플래그가 셋(SET)됩니다.

■ 프로그램 예

1. LD



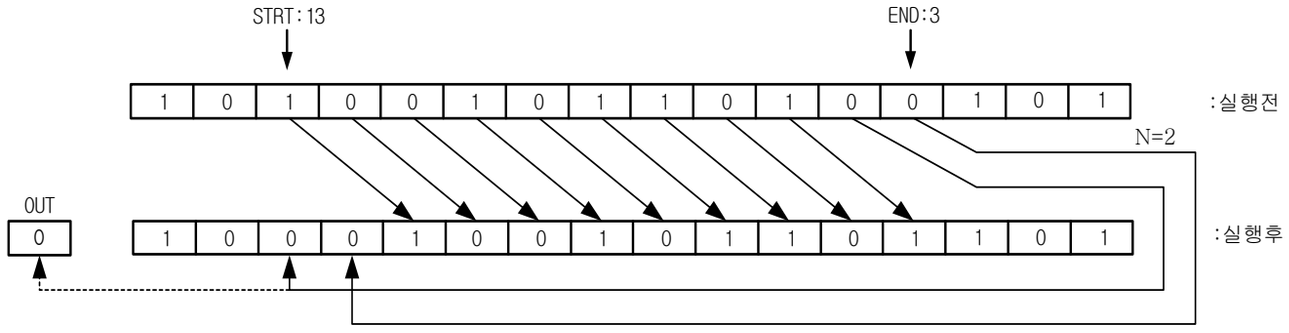
2. ST

```
OUT := ROTATE_C(EN:=%MX2, SRC:=16#A5A5, STRT:=13, END:=3, N:=2);
```

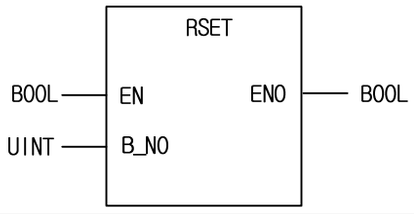
(1) 실행조건(%MX2)이 On 되면, ROTATE_C 펑션이 실행됩니다.

(2) 16#A5A5 값이 STRT(13)와 END(3)로 지정된 범위에서 STRT 부터 END 방향으로 2 번을 회전합니다.

(3) Rotate 된 후의 값이 SRC(16#896D)로 다시 출력되고 회전되면서 END 위치에서 STRT 로 되돌아가는 비트인 0 이 OUT 으로 출력됩니다.


STRT:13
END:3
1 0 1 0 0 1 0 1 1 0 1 0 0 1 0 1
:실행전
N=2
OUT
0
1 0 0 0 1 0 0 1 0 1 1 0 1 1 0 1
:실행후

RSET	적용 기종	발생플래그
설정된 블록 번호를 지정된 블록 번호로 전환	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 B_NO : 전환할 블록 NO(0~1). **출력** ENO : 에러 없이 수행되면 1을 출력

■ 기능

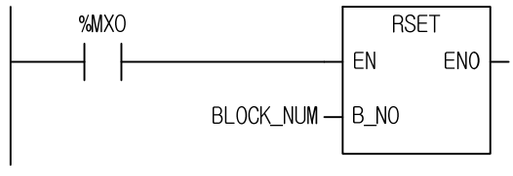
1. 설정된 블록 번호(_RBANK_NUM)를 지정된 블록 번호로 전환합니다.
2. STOP 상태에서 RUN 으로 전환할 경우, 블록 번호는 0 으로 초기화 됩니다.
3. S 값이 최대 블록번호를 넘어갈 경우 에러 플래그(_ERR)를 셋(Set)합니다.

■ 플래그

플래그	설명
_ERR	B_NO 값이 2이상일 경우 _ERR, _LER 플래그가 셋(SET)됩니다.

■ 프로그램 예

1. LD

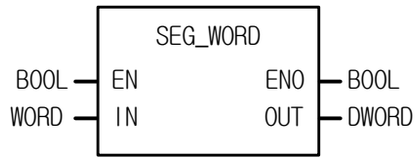


2. ST

```
RSET(EN:=%MX0, B_NO:=BLOCK_NUM);
```

(1) 실행조건(%MX0)이 On 되면 RSET 펑션이 실행됩니다.

(2) BLOCK_NUM(UINT 타입)은 0 과 1 의 둘 중 하나를 선택할 수 있으며 지정된 R 블록으로 전환시킵니다.

SEG_WORD	적용 기종	발생플래그
BCD 또는 HEX 값을 7 세그먼트 디스플레이 코드로 변환	XGI, XGR, XEC	-
펑 션	설 명	
	**입력** EN : 1 일 때 펑션 실행 IN : 7 세그먼트 코드로 변환할 입력 데이터 **출력** ENO : EN 값이 그대로 출력 OUT : 7 세그먼트 코드로 변환된 결과 데이터	

■ 기능

1. EN 이 1 이면, IN 의 BCD 또는 HEX(16 진) 숫자를 아래표와 같이 7 세그먼트 디스플레이를 위한 코드로 변환하여 OUT 으로 출력합니다.
2. BCD 입력의 경우 0000 ~ 9999 까지의 값이 4 개의 7 세그먼트에 표시 가능하며, HEX 입력의 경우 0000 ~ FFFF 의 값이 4 개의 7 세그먼트에 표시 가능합니다.

   표시 예

   1) 4 자리 BCD -> 4 자리 7 세그먼트 코드: 'SEG' 펑션 사용

   2) 4 자리 HEX -> 4 자리 7 세그먼트 코드: 'SEG' 펑션 사용

   3) 정수 -> 4 자리 BCD 형의 7 세그먼트 코드: 'INT_TO_BCD' 펑션을 거쳐서 'SEG' 펑션 사용

   4) 정수 -> 4 자리 HEX 형의 7 세그먼트 코드: 'INT_TO_WORD' 펑션을 거쳐서 'SEG' 펑션 사용

   5) 7 세그먼트의 자릿수가 4 자리를 넘을 때

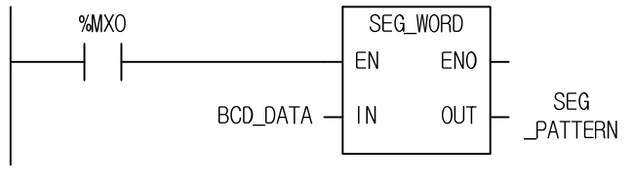
      가) BCD, HEX 는 4 자리씩 나누어 'SEG' 펑션을 사용

      나) 정수 -> 8 자리 BCD 형 7 세그먼트 코드:

      정수를 10,000 으로 나누어 몫과 나머지를 각각 'INT_TO_BCD' 펑션을 거쳐서 'SEG' 로 변환하여 상위 4 자리와 하위 4 자리의 7 세그먼트 코드를 생성

■ 프로그램 예

1. LD

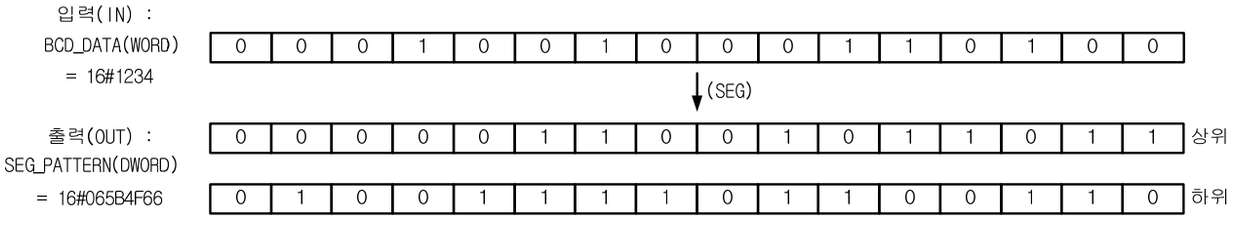


2. ST

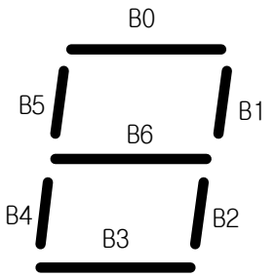
```
SEG_PATTERN := SEG_WORD(EN:=%MX0, IN:=BCD_DATA);
```

(1) 실행조건(%MX0)이 On 되면 SEG_WORD 펑션이 실행됩니다.

(2) 입력변수로 선언된 BCD_DATA(WORD 타입) = 16#1234 라면, 7 세그먼트 디스플레이에 '1234'가 표시 되는 코드 '2#0000_0110_0101_1011_0100_1111_0110_0110'이 출력되어 SEG_PATTERN(DWORD)에 저장됩니다.



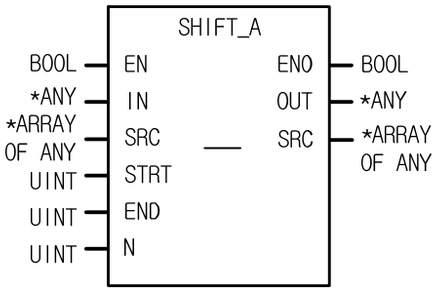
■ 7 세그먼트의 구성



■ 7 세그먼트 코드 변환표

입력 (BCD)	입력 (16 진수)	정수값	출력 B7 B6 B5 B4 B3 B2 B1 B0	표시 데이터
0	0	0	0 0 1 1 1 1 1 1	0
1	1	1	0 0 0 0 0 1 1 0	1
2	2	2	0 1 0 1 1 0 1 1	2
3	3	3	0 1 0 0 1 1 1 1	3
4	4	4	0 1 1 0 0 1 1 0	4
5	5	5	0 1 1 0 1 1 0 1	5
6	6	6	0 1 1 1 1 1 0 1	6
7	7	7	0 0 1 0 0 1 1 1	7
8	8	8	0 1 1 1 1 1 1 1	8
9	9	9	0 1 1 0 1 1 1 1	9
-	A	10	0 1 1 1 0 1 1 1	A
-	B	11	0 1 1 1 1 1 0 0	B
-	C	12	0 0 1 1 1 0 0 1	C
-	D	13	0 1 0 1 1 1 1 0	D
-	E	14	0 1 1 1 1 0 0 1	E
-	F	15	0 1 1 1 0 0 0 1	F

SHIFT_A	적용 기종	발생플래그
지정된 어레이 원소의 SHIFT	XGI, XGR, XEC	_ERR, _LER

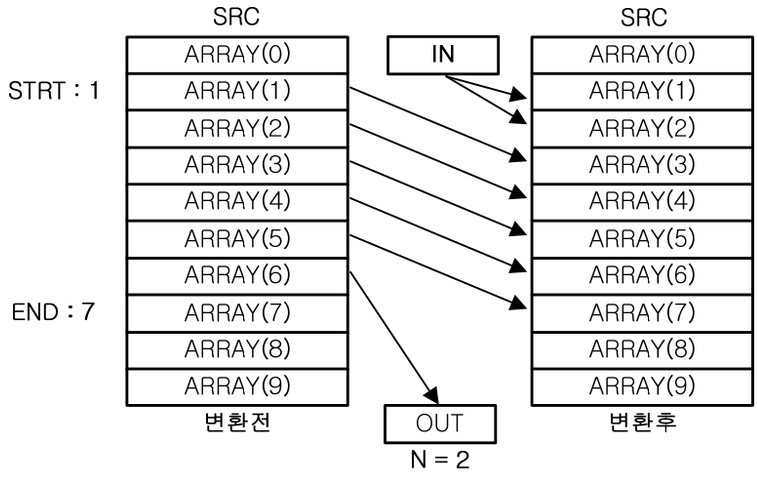
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : Shift된 후 비워진 원소들의 자리에 입력될 값 N : Shift 시킬 개수 STRT: 어레이 블록 중 Shift할 원소의 시작위치 END : 어레이 블록 중 Shift할 원소의 종료위치 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : Shift하여 밀려나온 데이터를 출력 **입출력** SRC : Shift 조작이 가해질 어레이 블록

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	IN2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ANY: ANY 타입 중 STRING 제외

■ 기능

1. SHIFT_A 펑션은 어레이 블록 중 지정된 범위의 원소들을 지정된 방향으로 이동시킵니다.
2. 동작의 지정:
   - 범위지정: STRT에서 END로 이동할 데이터 원소의 범위를 지정합니다.
   - 이동방향 및 횟수: STRT에서 END방향으로 지정된 횟수(N)만큼 Shift 됩니다.
   - 입력 데이터 지정: Shift하여 비워지는 위치를 입력 데이터(IN)로 채웁니다.
   - 출력: 데이터 조작의 결과는 SRC에 지정된 ARRAY에 저장되며, END 위치에서 Shift 동작으로 밀려 나온 데이터는 OUT으로 출력됩니다.



펑션	입출력 어레이 타입	동작 설명
SHIFT_A	BOOL	각 타입 어레이의 지정된 범위의 원소들을 지정된 방향으로 이동 시킵니다.
SHIFT_A	BYTE	
SHIFT_A	WORD	
SHIFT_A	DWORD	
SHIFT_A	LWORD	
SHIFT_A	SINT	
SHIFT_A	INT	
SHIFT_A	DINT	
SHIFT_A	LINT	
SHIFT_A	USINT	
SHIFT_A	UINT	
SHIFT_A	UDINT	
SHIFT_A	ULINT	
SHIFT_A	REAL	
SHIFT_A	LREAL	
SHIFT_A	TIME	
SHIFT_A	DATE	
SHIFT_A	TOD	
SHIFT_A	DT	

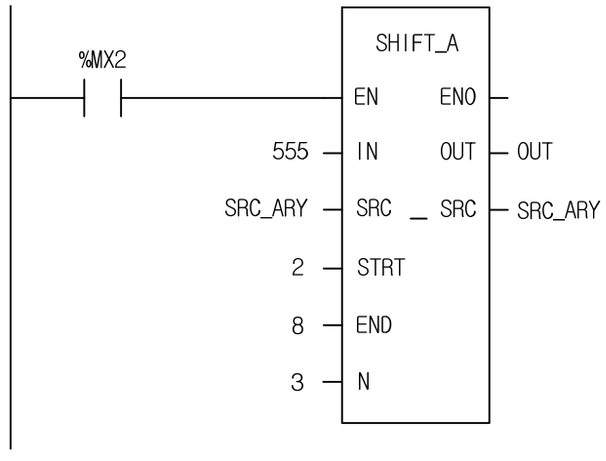
■ 플래그

플래그	설명
_ERR	STRT 또는 END 값이 SRC 어레이의 원소 개수 범위를 벗어나면 _ERR, _LER 플래그가 셋(Set)됩니다. 에러발생시 SRC 는 변하지 않고 출력은 각 변수타입의 초기화값(i.e. INT = 0, TIME = T#0S)이 출력됩니다.

☆ 출력단 Array 생략시 어레이의 개수를 0 으로 보아 _ERR, _LER 플래그가 발생합니다.

■ 프로그램 예

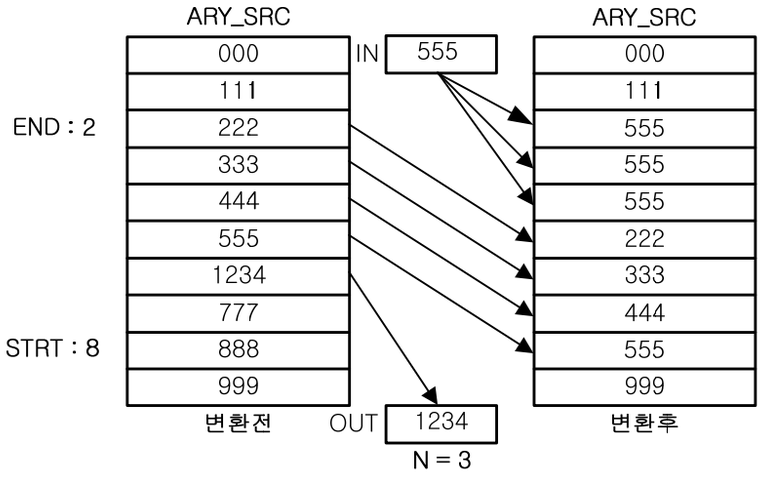
1. LD



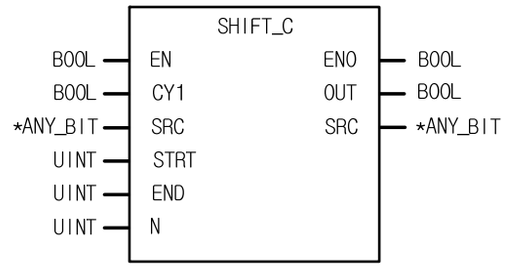
2. ST

```
OUT := SHIFT_A(EN:=%MX2, IN:=555, SRC:=SRC_ARY, STRT:=2, END:=8, N:=3);
```

(1) 입력 조건(%MX2)이 On 되면, SHIFT_A 펑션이 실행됩니다.
(2) 입출력 변수로 선언된 SRC_ARY 의 인덱스 2 의 원소부터 인덱스 8 의 원소까지 SHIFT 동작이 일어납니다.
(3) 지정된 영역의 원소들이 3 번 SHIFT 됩니다.
(4) SHIFT 된 후에 비워지는 원소들 즉, 어레이 인덱스 2 의 원소부터 3 개의 원소가 입력값 555 로 채워집니다.
(5) 출력 값에는 캐리(Carry) 출력에 해당하는 원소의 값 1234 가 출력됩니다.


ARY_SRC
000
111
222
333
444
555
1234
777
888
999
END : 2
STRT : 8
변환전
IN
555
OUT
1234
N = 3
ARY_SRC
000
111
555
555
555
222
333
444
555
999
변환후

SHIFT_C	적용 기종	발생플래그
Shift with Carry	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 CYI : Carry 입력 STRT : SRC의 비트열 중 Shift할 시작 bit 위치 END : SRC의 비트열 중 Shift할 끝 bit 위치 N : Shift할 비트수 **출력** ENO : 에러 없이 수행되면 1을 출력 OUT : Carry 출력 **입출력** SRC : Shift할 변수

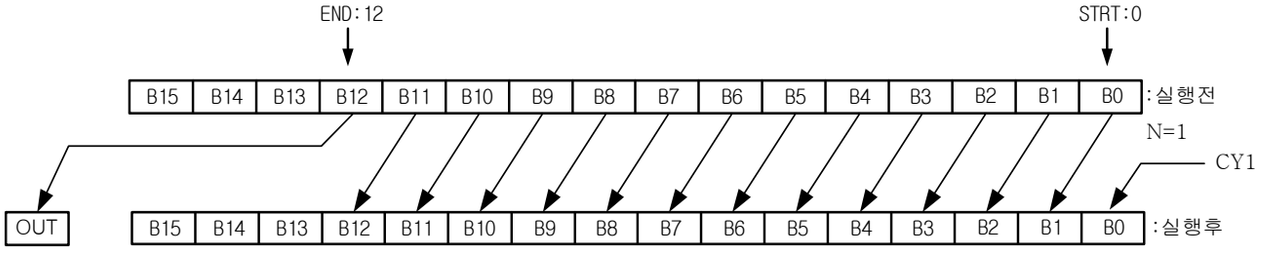
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT		○	○	○	○															

*ANY_BIT: ANY_BIT 타입 중 BOOL 제외

■ 기능

1. SRC의 비트열중 지정된 범위의 원소들을 지정된 방향으로 이동합니다.
2. 동작의 지정:
   - 범위지정: STRT에서 END로 이동할 비트의 범위를 지정합니다.
   - 이동방향 및 횟수: STRT에서 END방향으로 지정된 횟수(N)만큼 Shift 됩니다.
   - 입력 데이터 지정: Shift하여 비워지는 위치를 입력 데이터(CYI)로 채웁니다.
   - 출력: 데이터 조작의 결과는 SRC에 지정된 ANY_BIT에 바뀌어 저장되며, END위치에서 Shift 동작으로 밀려 나온 비트 데이터는 OUT으로 출력됩니다.

.



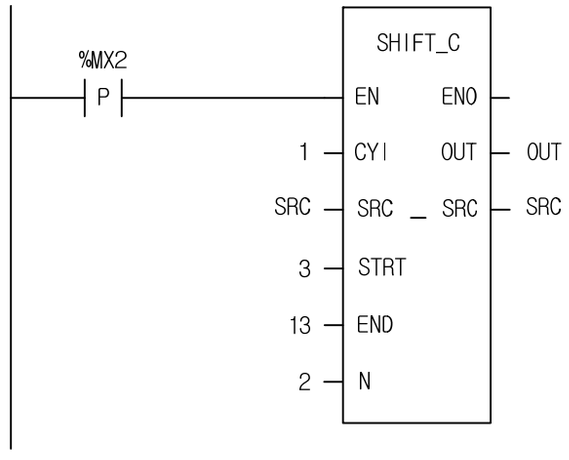
펑션	SRC 변수 타입	동작 설명
SHIFT_C	BYTE	입력값의 지정된 범위에 해당하는 비트들을 지정된 횟수만큼 Shift 시킵니다.
SHIFT_C	WORD	
SHIFT_C	DWORD	
SHIFT_C	LWORD	

■ 플래그

플래그	설명
_ERR	STRT와 END값이 SRC변수 타입의 비트 개수를 넘는 경우 SRS의 변화는 없으며 _ERR, _LER 플래그가 셋(SET)됩니다.

■ 프로그램 예

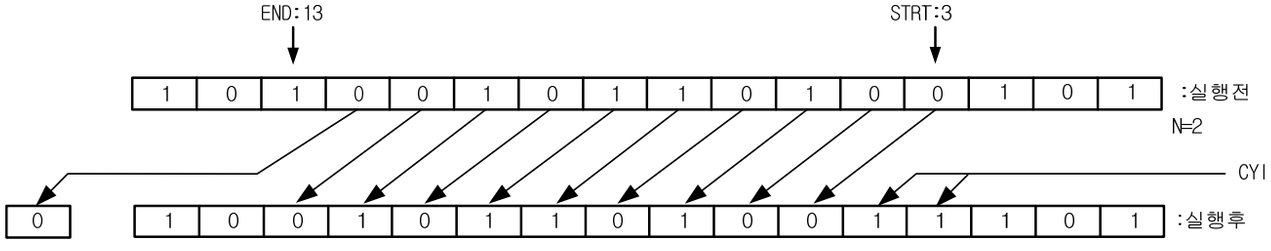
1. LD



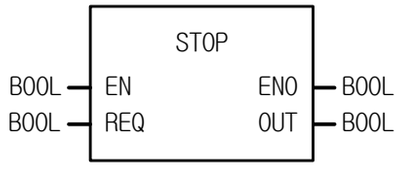
2. ST

```
OUT := SHIFT_C(EN:=%MX2, CYI:=1, SRC:=SRC, STRT:=3, END:=13, N:=2);
```

(1) 실행 조건(%MX2)이 On 되면, SHIFT_C 펑션이 실행됩니다.
(2) SRC(WORD 타입)에 16#A5A5 를 설정하고 STRT 에서 END 방향으로 2 비트 shift 하고 shift 후에 비워지는 비트는 CYI 값인 1 로 채워집니다.
(3) Shift 된 후의 값은 다시 SRC(16#969D)로 출력되고 2 비트 shift 되면서 밀려나온 값 0 이 OUT 으로 출력됩니다.



STOP	적용 기종	발생플래그
프로그램에 의한 운전정지	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 RE : 프로그램에 의한 운전정지 요구 **출력** ENO : EN 값이 그대로 출력 OUT : STOP 동작이 실행되면 1 출력

■ 기능

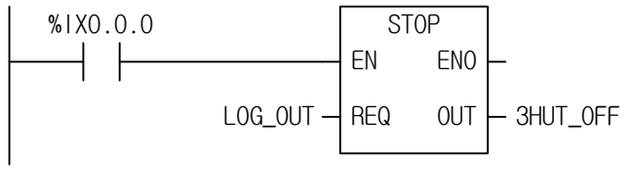
1. EN 이 1 이고 REQ 에 1 의 값이 들어오면 운전을 정지하고 STOP 모드로 됩니다.
2. 'STOP'펑션이 수행되면 수행 중인 스캔 프로그램의 수행을 완료한 후 정지합니다.
3. 전원을 재 투입하거나 운전모드를 STOP 으로 하였다가 RUN 으로 하면 재 기동이 됩니다.

■ 관련 플래그

플래그	설명
_USTOP_ON	STOP 명령어에 의해 STOP 된 경우 On 됩니다. 이 상태에서 다시 런 진입시 Off 됩니다.

■ 프로그램 예

1. LD

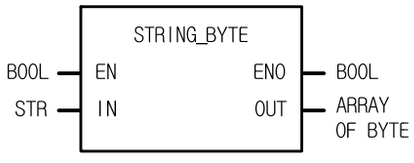


2. ST

```
3HUT_OFF := STOP(EN:=%IX0.0.0, REQ:=LOG_OUT);
```

(1) 실행조건(%IX0.0.0)이 On 하고 LOG_OUT 가 1 이 되면 실행중인 스캔 프로그램을 완료한 후 STOP 모드로 됩니다.

(2) 입력변수로 선언한 'STOP'펑션 수행 후 안정된 상태에서 PLC 의 전원을 끄는 것이 좋습니다.

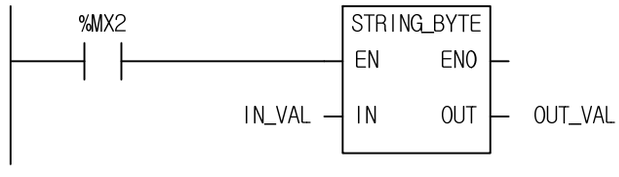
STRING_BYTE	적용 기종	발생플래그
문자열을 Byte Array 로 변환	XGI, XGR, XEC	-
펑 션	설 명	
	**입력** EN : 1 일 때 펑션 실행 IN1 : String 입력 **출력** ENO : EN 값을 그대로 출력 OUT : 변환된 Byte Array 출력	

■ 기능

1. 하나의 String 을 31 개의 Byte Array 로 변환합니다.

■ 프로그램 예

1. LD

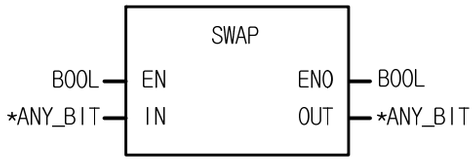


2. ST

```
OUT_VAL := STRING_BYTE(EN:=%MX2, IN:=IN_VAL);
```

(1) 실행조건(%MX2)이 On 되면 STRING_BYTE 펑션이 실행됩니다.
(2) IN_VAL = 'ABC' 이면 OUT_VAL[0] = 16#41, OUT_VAL[1] = 16#42, OUT_VAL[2] = 16#43 이 됩니다.

SWAP	적용 기종	발생플래그
데이터의 상위 하위 바꾸기	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 IN : 입력 **출력** ENO : EN 값을 그대로 출력 OUT : Swap 된 값

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*ANY_BIT: ANY_BIT 타입 중 BOOL 제외

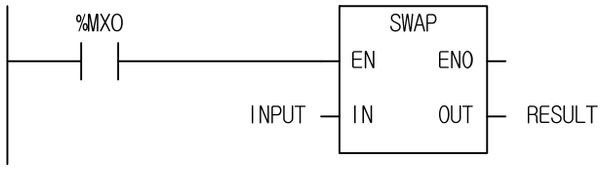
■ 기능

1. 입력된 변수를 2 개의 크기로 구분하여 상위와 하위를 서로 교환합니다.

펑션	입력 타입	동작 설명
SWAP	BYTE	BYTE의 상하위 니블(Nibble)을 서로 교환하여 출력합니다.
SWAP	WORD	WORD의 상하위 BYTE를 서로 교환하여 출력합니다.
SWAP	DWORD	DWORD의 상하위 WORD를 서로 교환하여 출력합니다.
SWAP	LWORD	LWORD의 상하위 DWORD를 서로 교환하여 출력합니다.

■ 프로그램 예

1. LD



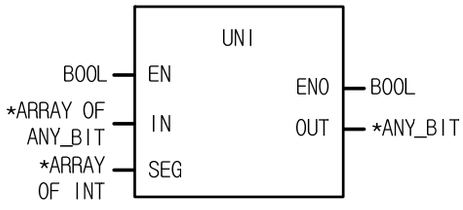
2. ST

```
RESULT := SWAP(EN:=%MX0, IN:=INPUT);
```

(1) 실행조건(%MX0)이 On 되면, SWAP 펑션이 실행됩니다.

(2) 펑션의 입력 변수 INPUT(BYTE 타입) = 16#5F 일 경우, 펑션의 출력 변수 RESULT(BYTE 타입) = 16#F5 가 됩니다.

UNI	적용 기종	발생플래그
데이터 결합(union)	XGI, XGR, XEC	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 입력 데이터 어레이 SEG : 데이터 결합 비트수 지정 어레이 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 결합된 데이터 출력

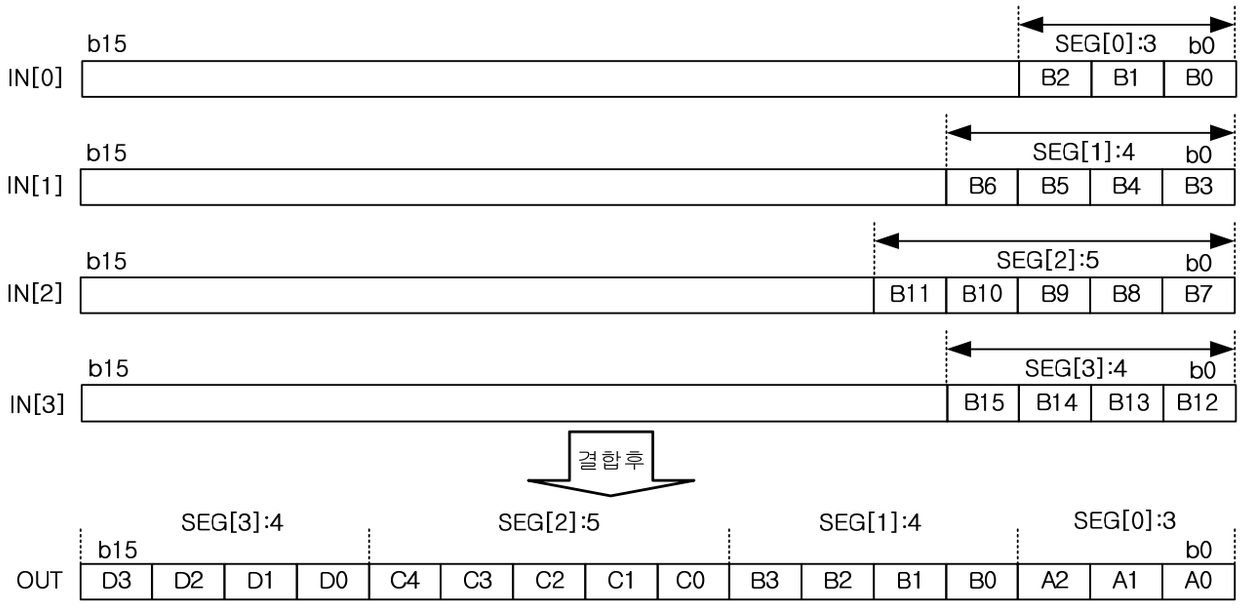
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*ANY_BIT: ANY_BIT 타입 중 BOOL 제외

■ 기능

1. 입력 데이터 어레이를 SEG 어레이에서 지정한 비트수 별로 하위 비트부터 지정된 비트 수만큼 결합하여 출력합니다.

펑션	입력 타입	출력 타입	동작 설명
UNI	BYTE	BYTE	입력 어레이를 SEG가 지정하는 비트의 개수만큼씩 자른 후, 입력 어레이 타입과 동일한 하나의 값으로 묶어 출력합니다.
UNI	WORD	WORD	
UNI	DWORD	DWORD	
UNI	LWORD	LWORD	



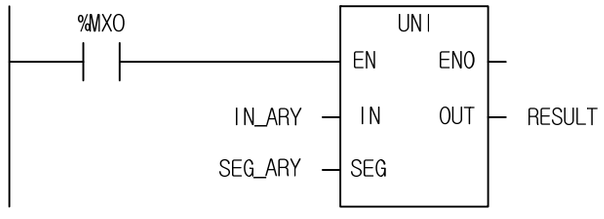
SEG로 지정된 값들의 합이 입력 변수 타입의 비트수를 초과할 경우 _ERR/_LER 플래그가 셋(Set)됩니다.

■ 플래그

플래그	설명
_ERR	SEG 로 지정된 값들의 합이 입력 변수 타입의 비트수와 일치하지 않는 경우 _ERR, _LER 플래그가 셋(SET)됩니다. IN 과 SEG 어레이의 개수가 서로 다를 경우, 0 이 출력되고 _ERR, _LER 플래그가 셋(SET)됩니다.

■ 프로그램 예

1. LD



2. ST

```
RESULT := UNI(EN:=%MX0, IN:=IN_ARY, SEG:=SEG_ARY);
```

(1) 실행조건(%MX0)이 On 되면, UNI 펑션이 실행됩니다.

(2) 입력 변수로 선언된 IN_ARY 와 SEG_ARY 의 값이 다음과 같을 경우

IN_ARY[0]	A3B5	SEG_ARY[0]	3
IN_ARY[1]	B4C6	SEG_ARY[1]	4
IN_ARY[2]	C5D7	SEG_ARY[2]	7
IN_ARY[3]	D6E8	SEG_ARY[3]	2

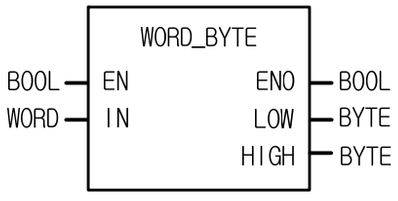
펑션이 실행된 후에 출력변수의 RESULT 의 값은 2#0010_10111_0110_0101 = 16#2BB5 로 출력됩니다.

IN_ARY[0]	2#1010 0011 1011 0101	SEG_ARY[0]	3
IN_ARY[1]	2#1011 0100 1100 0110	SEG_ARY[1]	4
IN_ARY[2]	2#1100 0101 1101 0111	SEG_ARY[2]	7
IN_ARY[3]	2#1101 0110 1110 1000	SEG_ARY[3]	2

↓

RESULT : 2#00 1010111 0110 101

WORD_BYTE	적용 기종	발생플래그
WORD 를 2 개의 BYTE 로 나눔	XGI, XGR, XEC	-

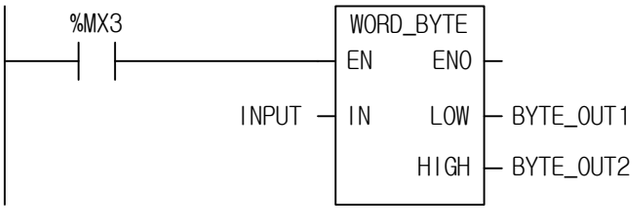
펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 IN : WORD 입력 **출력** ENO : EN 값을 그대로 출력 LOW : 하위 BYTE 출력 HIGH : 상위 BYTE 출력

■ 기능

1. 하나의 워드를 2 개의 바이트로 분산합니다.
   LOW: 하위 바이트 출력, HIGH: 상위 바이트 출력

■ 프로그램 예

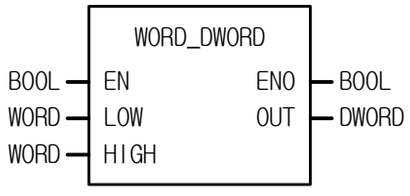
1. LD



2. ST

```
WORD_BYTE(EN:=%MX3, IN:=INPUT, LOW=>BYTE_OUT1, HIGH=>BYTE_OUT2);
```

(1) 실행조건(%MX3)이 On 되면 WORD_BYTE 펑션이 실행됩니다.
(2) 입력변수로 선언된 INPUT값이 16#ABCD일 때 출력 변수로 선언된 변수 BYTE_OUT1 = 16#CD, BYTE_OUT2 = 16#AB 값이 저장됩니다.

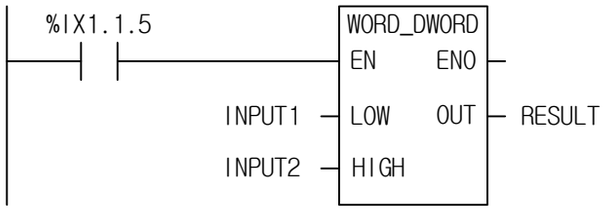
WORD_DWORD	적용 기종	발생플래그
2 개의 WORD 를 DWORD 로 모음	XGI, XGR, XEC	-
펑 션	설 명	
	**입력** EN : 1 일 때 펑션 실행 LOW : 하위 WORD 입력 HIGH : 상위 WORD 입력 **출력** ENO : EN 값을 그대로 출력 OUT : DWORD 출력	

■ 기능

1. 2 개의 WORD 를 하나의 DWORD 로 조합합니다.
   LOW: 하위 워드 입력, HIGH: 상위 워드 입력

■ 프로그램 예

1. LD

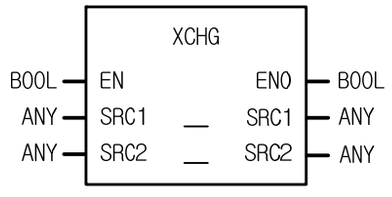


2. ST

```
RESULT := WORD_DWORD(EN:=%IX1.1.5, LOW:=INPUT1, HIGH:=INPUT2);
```

(1) 실행조건(%IX1.1.5)이 On 되면 WORD_DWORD 펑션이 실행됩니다.
(2) 입력변수로 선언된 INPUT1 = 16#1020이고 INPUT2 = 16#A0B0 일 때 출력변수로 선언된 RESULT = 16#A0B0_1020가 됩니다.

XCHG	적용 기종	발생플래그
2 개의 입력변수 값을 서로 교환	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 **출력** ENO : EN 값을 그대로 출력 **입출력** SRC1 : 입출력 1 SRC2 : 입출력 2

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	SRC1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	SRC2	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ 기능

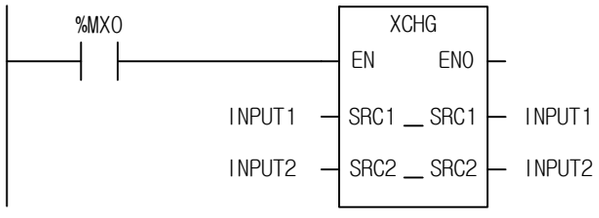
1. 입력 1 과 입력 2 의 데이터를 교환합니다.

펑 션	입출력타입	동작 설명
XCHG	BOOL	두 개의 BOOL 입력 값을 서로 교환합니다.
XCHG	BYTE	두 개의 BYTE 입력 값을 서로 교환합니다.
XCHG	WORD	두 개의 WORD 입력 값을 서로 교환합니다.
XCHG	DWORD	두 개의 DWORD 입력 값을 서로 교환합니다.
XCHG	LWORD	두 개의 LWORD 입력 값을 서로 교환합니다.
XCHG	SINT	두 개의 SINT 입력 값을 서로 교환합니다.
XCHG	INT	두 개의 INT 입력 값을 서로 교환합니다.
XCHG	DINT	두 개의 DINT 입력 값을 서로 교환합니다.
XCHG	LINT	두 개의 LINT 입력 값을 서로 교환합니다.
XCHG	USINT	두 개의 USINT 입력 값을 서로 교환합니다.
XCHG	UINT	두 개의 UINT 입력 값을 서로 교환합니다.
XCHG	UDINT	두 개의 UDINT 입력 값을 서로 교환합니다.
XCHG	ULINT	두 개의 ULINT 입력 값을 서로 교환합니다.
XCHG	REAL	두 개의 REAL 입력 값을 서로 교환합니다.

펑　　션	입출력타입	동작 설명
XCHG	LREAL	두 개의 LREAL 입력 값을 서로 교환합니다.
XCHG	TIME	두 개의 TIME 입력 값을 서로 교환합니다.
XCHG	DATE	두 개의 DATE 입력 값을 서로 교환합니다.
XCHG	TOD	두 개의 TOD 입력 값을 서로 교환합니다.
XCHG	DT	두 개의 DT 입력 값을 서로 교환합니다.
XCHG	STRING	두 개의 STRING 입력 값을 서로 교환합니다.

■ 프로그램 예

1. LD

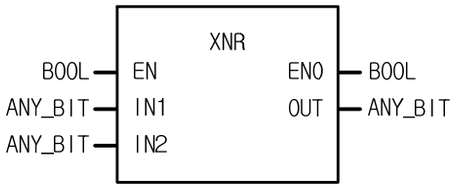


2. ST

```
XCHG(EN:=%MX0, SRC1:=INPUT1, SRC2:=INPUT2);
```

(1) 실행조건(%MX0)이 On 되면 XCHG 펑션이 실행됩니다.
(2) 펑션의 입력변수 INPUT1 = 0 이고, INPUT2 = 1이면 펑션이 실행된 후 2개의 입력값은 서로 바뀌어 출력됩니다.
따라서 펑션이 실행된 후 INPUT1 = 1이고, INPUT2 = 0이 됩니다.

XNR	적용 기종	발생플래그
배타적 논리곱	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 IN1 : XNR 될 값 IN2 : XNR 될 값 입력 8 개까지 확장 가능 **출력** ENO : EN 값이 그대로 출력 OUT : XNR 된 값 IN1, IN2, OUT 은 모두 같은 타입이어야 함.

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN1	○	○	○	○	○															
	IN2	○	○	○	○	○															
	OUT	○	○	○	○	○															

■ 기능

1. IN1 을 IN2 와 비트별로 XNR 해서 OUT 으로 출력시킵니다.

   IN1 1111 0000

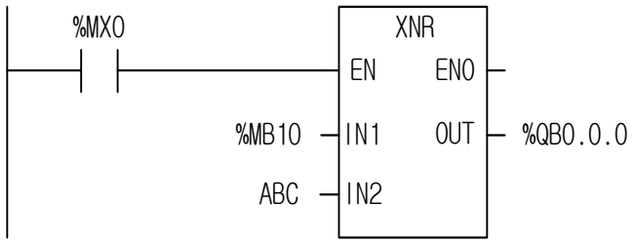
   XNR

   IN2 1010 1010

   OUT 1010 0101

■ 프로그램 예

1. LD



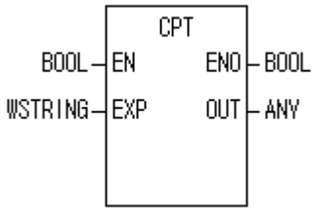
2. ST

```
%QB0.0.0 := XNR(EN:=%MX0, IN1:=%MB10, IN2:=ABC);
```

(1) 실행조건(%MX0)이 On 되면 XNR(배타적 논리곱) 펑션이 실행됩니다.

(2) %MB10 = 16#F0 = 2#1111_0000 이고, ABC(BYTE 타입) = 16#AA = 2#1010_1010 이면 XNR 펑션이 실행된 후의 출력값은 %QB0.0.0 = 16#A5 = 2#1010_0101 이 됩니다.

CPT	적용 기종	발생플래그
ST 표현식 연산	XGI, XGR, XEC	-

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 EXP : ST 언어의 표현식 **출력** ENO : EN값이 그대로 출력 OUT : 표현식의 결과 값

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	WSTRING
	IN																				○
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○					

■ 기능

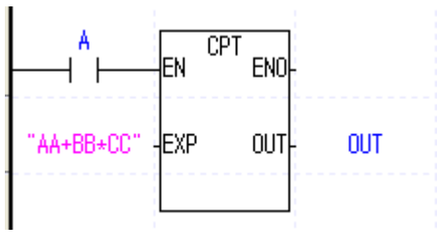
1. EN이 1이면, EXP의 표현식을 수행하여 OUT으로 출력합니다.
2. 입력 표현식의 최대 사이즈는 100 Byte 입니다. (영문: 100자, 한글 50자)
3. 표현식에 사용 가능한 펑션은 비교, 수치 연산, 각도 변환, 타입 변환 만 가능 합니다.
   A. 비교: EQ, GE, GT, LE, LT, NE
   B. 수치 연산: ABS, ACOS, ADD, ASIN, ATAN, COS, DIV, EXP, EXPT, LN, LOG, MOVE, MUL, SIN, SQRT, SUB, TAN, TRUNC (단, MOD은 사용 못함, 키워드로 동작함)
   C. 각도 변환: DEG, RAD
   D. 타입 변환: 특수 기호(***)가 들어 있지 않는 타입 변환 함수
4. ST 연산자는 ST 명령어 도움말 참조 하세요.

펑션	출력 타입	동작 설명
CPT	BOOL	입력 표현식에 대한 출력 값이 BOOL 타입 이어야 합니다.
CPT	BYTE	입력 표현식에 대한 출력 값이 BYTE 타입 이어야 합니다.
CPT	WORD	입력 표현식에 대한 출력 값이 WORD 타입 이어야 합니다.
CPT	DWORD	입력 표현식에 대한 출력 값이 DWORD 타입 이어야 합니다.
CPT	LWORD	입력 표현식에 대한 출력 값이 LWORD 타입 이어야 합니다.
CPT	SINT	입력 표현식에 대한 출력 값이 SINT 타입 이어야 합니다.
CPT	INT	입력 표현식에 대한 출력 값이 INT 타입 이어야 합니다.

펑션	출력 타입	동작 설명
CPT	DINT	입력 표현식에 대한 출력 값이 DINT 타입 이어야 합니다.
CPT	LINT	입력 표현식에 대한 출력 값이 LINT 타입 이어야 합니다.
CPT	USINT	입력 표현식에 대한 출력 값이 USINT 타입 이어야 합니다.
CPT	UINT	입력 표현식에 대한 출력 값이 UINT 타입 이어야 합니다.
CPT	UDINT	입력 표현식에 대한 출력 값이 UDINT 타입 이어야 합니다.
CPT	ULINT	입력 표현식에 대한 출력 값이 ULINT 타입 이어야 합니다.
CPT	REAL	입력 표현식에 대한 출력 값이 REAL 타입 이어야 합니다.
CPT	LREAL	입력 표현식에 대한 출력 값이 LREAL 타입 이어야 합니다.

■ 프로그램 예

1. LD



2. ST

- CPT 펑션은 사용 할 수 없습니다. 단, 표현식을 바로 사용 할 수 있습니다.

```
IF A THEN
  OUT := AA+BB *CC ;
END_IF;
```

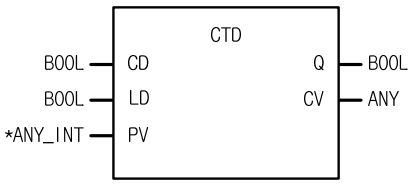
(1) 실행조건(A)이 On 되면, CPT 펑션이 실행됩니다.

(2) 입력 변수로 선언된 AA = 10, BB = 10, CC = 2 이라면, 연산이 실행된 후에는 OUT = 30 이 됩니다.

제 9 장 기본 펑션 블록

1. 기본 펑션 블록에 대한 설명입니다.
2. 기본 펑션 블록을 사용하시기 전에 3.4.2 의 펑션 블록에 대한 일반사항을 이해 하신 후 프로그램에 적용하시면, 프로그램 작성이 용이합니다.

CTD	적용 기종	발생플래그
감산 카운터 (펑션 블록)	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** CD : 다운_카운트(Down_Count) 펄스입력 LD : 설정값 입력(Load) PV : 설정값(Preset Value) **출력** Q : 카운트_다운(Count_Down) 출력 CV : 현재값(Current Value)

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	PV							○	○	○		○	○	○							
	CV							○	○	○		○	○	○							

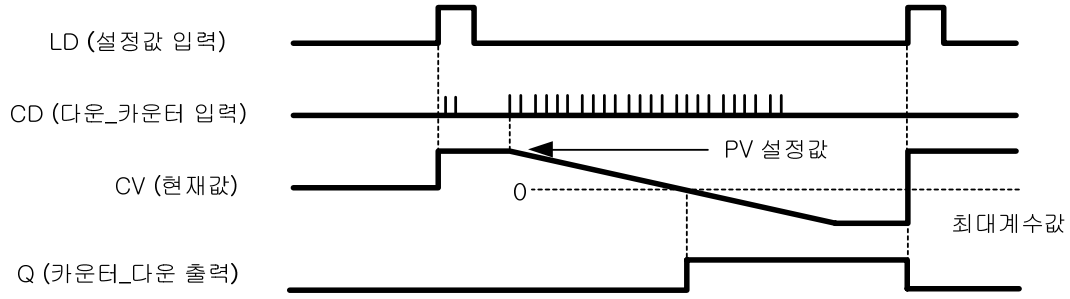
*ANY_INT: ANY_INT 타입 중 SINT, USINT 제외.

■ 기능

1. 감산 카운터 펑션 블록 CTD 는 다운 카운터 펄스입력 CD 가 0 에서 1 이 되면, 현재값 CV 가 이전값보다 1 만큼 감소하는 카운터 입니다.
2. 단, CV 는 PV 의 최소보다 클 때만 감소하고, 최소값이 되면 더이상 감소하지 않습니다.
3. 설정값 입력 LD 가 1 이 되면 현재값 CV 에는 설정값 PV 값이 로드 됩니다. (CV = PV)
4. 출력 Q 는 CV 가 0 이하일 때만 1 이 됩니다.

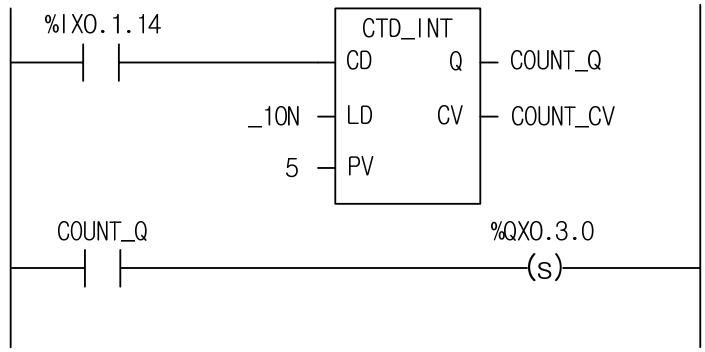
펑션 블록	PV	동작 설명
CTD_INT	INT	INT(설정값)의 최소값(-32,768)만큼 감소합니다.
CTD_DINT	DINT	DINT(설정값)의 최소값(-2,147,483,648)만큼 감소합니다.
CTD_LINT	LINT	LINT(설정값)의 최소값(-9,223,372,036,854,775,808)만큼 감소합니다.
CTD_UINT	UINT	UINT(설정값)의 최소값(0)만큼 감소합니다.
CTD_UDINT	UDINT	UDINT(설정값)의 최소값(0)만큼 감소합니다.
CTD_ULINT	ULINT	ULINT(설정값)의 최소값(0)만큼 감소합니다.

■ 타임차트



■ 프로그램 예

1. LD



2. ST

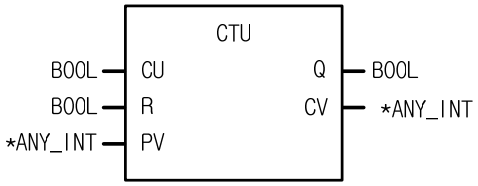
```
INST_CTD_INT(CD:=%IX0.1.14, LD:=_10N, PV:=5, Q=>COUNT_Q, CV=>COUNT_CV);
%QX0.3.0 := COUNT_Q;
```

입력접점 %IX0.1.14 에 5 번의 펄스가 들어오면, 출력접점 %QX0.3.0 을 Set 시키는 프로그램

(1) CTD 펑션 블록의 이름을 등록합니다. (COUNT_D)
(2) CD 에 펄스입력이 들어올 접점 %IX0.1.14 를 입력합니다.
(3) PV 를 CV 로 로드시킬 사용자 플래그_10N(첫스캔 On)을 입력합니다.
(4) PV 값은 INT 범위(-32,768~32,767) 내에서 5 를 입력합니다.
(5) CV 에는 임의의 출력변수(COUNT_CV)를 설정하여 입력합니다.
(6) Q 에는 임의의 출력변수(COUNT_Q)를 설정 입력합니다.
(7) 프로그램의 작성이 완료되면, 컴파일을 실행하고, PLC 로 쓰기를 합니다.
(8) 쓰기를 완료하면 모드전환(Stop → Run) 시킵니다.
(9) 프로그램이 Run 되면 PV 값 5 가 CV(Count_CV)로 로드됩니다.
(10) 입력펄스가 입력접점 %IX0.1.14 에 들어오면 현재값 CV(COUNT_CV)는 1 만큼 줄어듭니다.
(11) 입력접점에 5 번의 펄스가 들어오면 현재값 CV 는 0 가 되고, Q(COUNT_Q)는 1 이 됩니다.

(12) Q(COUNT_Q)가 1 이 되면 출력접점 %QX0.3.0 이 Set 됩니다.

CTU	적용 기종	발생플래그
가산 카운터 (펑션 블록)	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** CU : 업_카운트(Up_Count) 펄스입력 R : 리셋 입력(Reset) PV : 설정값 (Preset Value) **출력** Q : 업_카운트(Up_ Count) 출력 CV : 현재값(Current Value)

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	PV							○	○	○		○	○	○							
	CV							○	○	○		○	○	○							

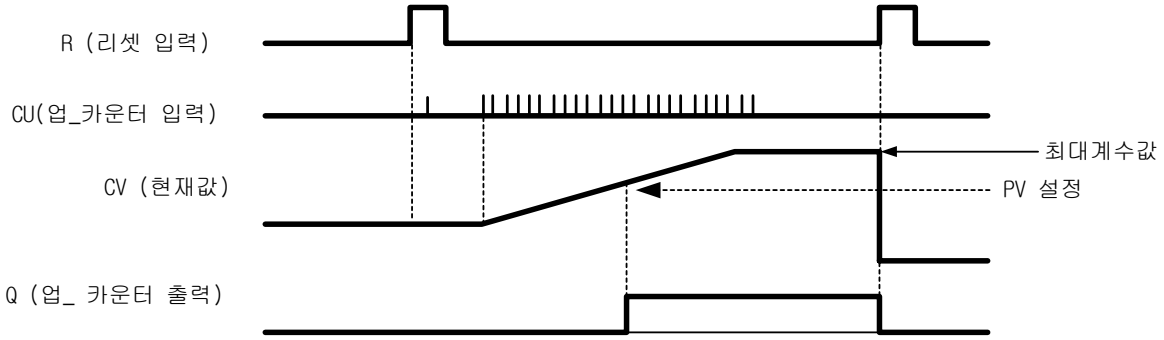
*ANY_INT: ANY_INT 타입 중 SINT, USINT 제외.

■ 기능

1. 가산 카운터 펑션 블록 CTU 는 업 카운터 펄스입력 CU 가 0 에서 1 이 되면 현재값 CV 가 이전값 보다 1 만큼 증가하는 카운터입니다.
2. 단, CV 가 PV 의 최대값 미만일 때만 증가하고, 최대값이 되면 더이상 증가하지 않습니다.
3. 리셋 입력 R 이 1 이 되면 현재값 CV 는 0 으로 클리어(Clear)됩니다.
4. 출력 Q 는 CV 가 PV 이상이 될 때만 1 이 됩니다.
5. PV 값은 CTU 펑션 블록을 수행 시 설정값을 새롭게 가져와 연산합니다.

펑션 블록	PV	동작 설명
CTU_INT	INT	INT(설정값)의 최대값(32767)만큼 증가합니다.
CTU_DINT	DINT	DINT(설정값)의 최대값(2147483647)만큼 증가합니다.
CTU_LINT	LINT	LINT(설정값)의 최대값(9223372036854775807)만큼 증가합니다.
CTU_UINT	UINT	UINT(설정값)의 최대값(0)만큼 증가합니다.
CTU_UDINT	UDINT	UDINT(설정값)의 최대값(0)만큼 증가합니다.
CTU_ULINT	ULINT	ULINT(설정값)의 최대값(0)만큼 증가합니다.

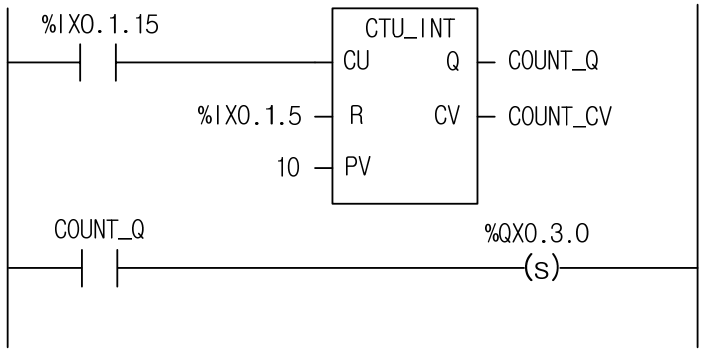
■ 타임차트



■ 프로그램 예

1. 입력접점 %IX0.1.15 에 10 번의 펄스가 들어오면, 출력접점 %QX0.3.0 을 Set 시키는 프로그램

1. LD



2. ST

```
INST_CTU_INT(CU:=%IX0.1.15, R:=%IX0.1.5, PV:=10, Q=>COUNT_Q, CV=>COUNT_CV);
%QX0.3.0 := COUNT_Q;
```

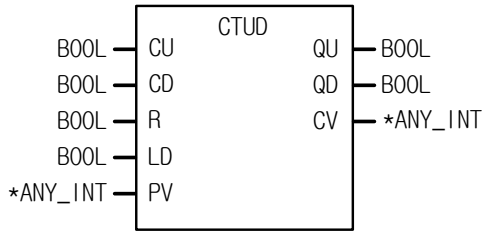
(1) CTU 펑션 블록의 이름을 등록합니다. (COUNT_U)
(2) CU 에 펄스 입력이 들어올 입력 접점 %IX0.1.15 를 입력합니다.
(3) PV 에 10 을 입력합니다.
(4) CV 를 초기화시키는 R 에는 임의의 입력 접점을 설정합니다.(%IX0.1.5)
(5) CV 에는 임의의 변수(COUNT_CV)를 설정하여 입력합니다.
(6) Q 에는 임의의 출력변수(COUNT_Q)를 설정 입력합니다.
(7) 프로그램의 작성이 완료되면, 컴파일을 실행하고, PLC 로 쓰기를 합니다.
(8) 쓰기를 완료하면 모드전환(Stop →Run) 시킵니다.

(9) 입력펄스가 입력접점 %IX0.1.15 에 들어오면 현재값 CV(COUNT_CV)는 1 만큼 증가합니다.

(10) 입력접점에 10 번의 펄스가 들어오면 현재값 CV 는 10 이 되고, 설정값과 같게 되므로 출력 Q(COUNT_Q)가 1 이 됩니다.

(11) Q(COUNT_Q)가 1 이 되면 출력접점 %QX0.3.0 이 셋(Set)됩니다.

CTUD	적용 기종	발생플래그
가감산 카운터 (펑션 블록)	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** CU : 업_카운트(Up_Count) 펄스입력 CD : 다운_카운트(Down_Count) 펄스입력 R : 리셋 입력(Reset) LD : 설정값 입력(Load) PV : 설정값(Preset Value) **출력** QU : 카운트_업(Count_UP) 출력 QD : 카운트_다운(Count_Down) 출력 CV : 현재값(Current Value)

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	PV							○	○	○		○	○	○							
	CV							○	○	○		○	○	○							

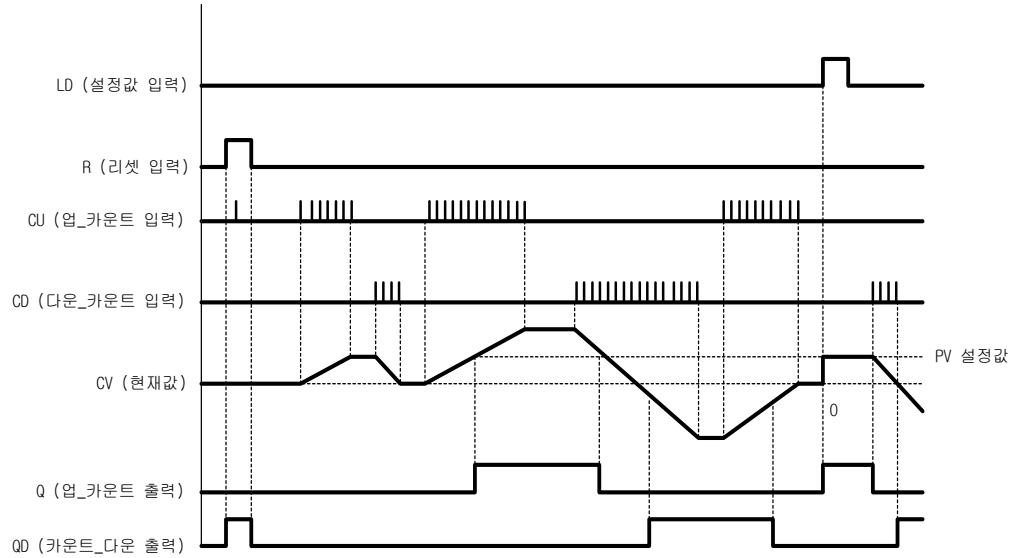
*ANY_INT: ANY_INT 타입 중 SINT, USINT 제외.

■ 기능

1. 가감산 카운터 펑션 블록 CTUD 는 업카운터 펄스 입력 CU 가 0 에서 1 이 되면 현재값 CV 가 이전값보다 1 만큼 증가하고, 다운 카운터 펄스입력 CD 가 0 에서 1 이 되면 현재값 CV 가 이전값보다 1 만큼 감소하는 카운터 입니다.
2. 단, CV 가 PV 의 최소값과 최대값 사이의 값을 가지며 최대, 최소값에 이르면 각각 더이상 증가, 감소하지 않습니다.
3. 설정값 입력 LD 가 1 이 되면 현재값 CV 에는 설정값 PV 값이 로드됩니다. (CV = PV)
4. 설정값 입력 R 이 1 이 되면 현재값 CV 는 0 으로 클리어(Clear) 됩니다. (CV = 0)
5. 출력 QU 는 CV 가 PV 보다 크면 1 이 되고, QD 는 CV 가 0 이하일 때 1 이 됩니다.
6. 각 입력신호에 대해서 R > LD > CU > CD 순으로 동작을 수행하며, 신호의 중복 발생시 우선 순위가 높은 동작 하나만 수행합니다.

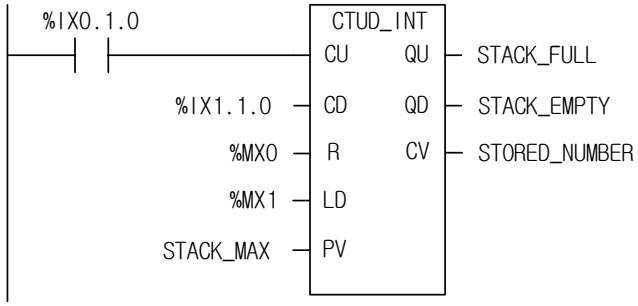
펑션 블록	PV	동작 설명
CTUD_INT	INT	INT(설정값)의 -32768 ~ 32767 만큼 증가, 감소합니다.
CTUD_DINT	DINT	DINT(설정값)의 0 ~ 2^{31}-1 만큼 증가, 감소합니다.
CTUD_LINT	LINT	LINT(설정값)의 0 ~ 2^{63}-1 만큼 증가, 감소합니다.
CTUD_UINT	UINT	UINT(설정값)의 0 ~ 65535 만큼 증가, 감소합니다.
CTUD_UDINT	UDINT	UDINT(설정값)의 0 ~ 2^{32}-1 만큼 증가, 감소합니다.
CTUD_ULINT	ULINT	ULINT(설정값)의 0 ~ 2^{63}-1 만큼 증가, 감소합니다.

■ 타임차트



■ 프로그램 예

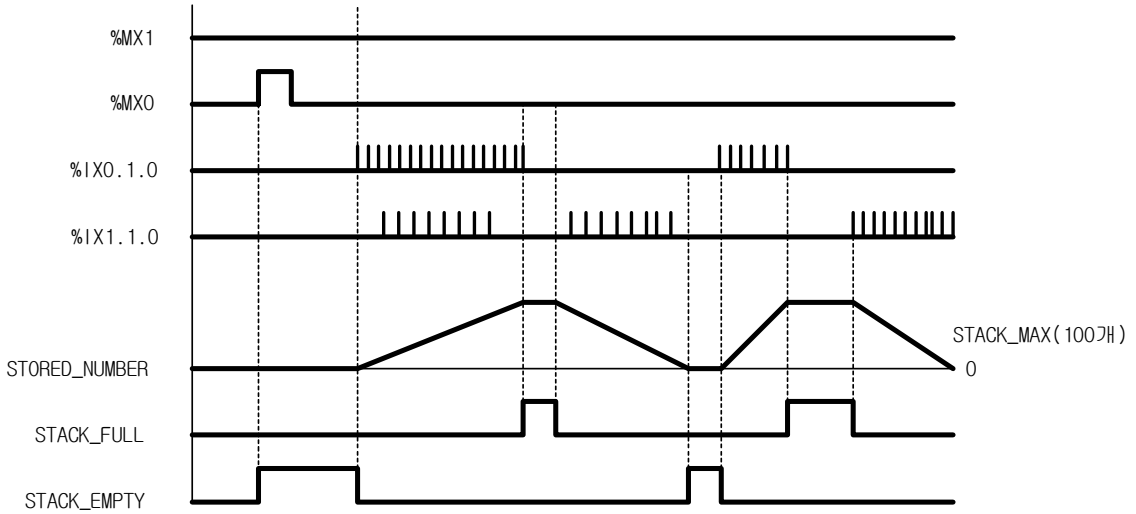
1. LD

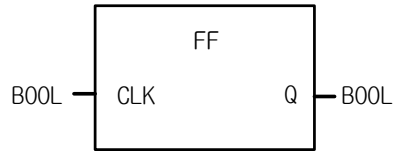


2. ST

```
INST_CTUD_INT(CU:=%IX0.1.0, CD:=%IX1.1.0, R:=%MX0, LD:=%MX1, PV:=STACK_MAX, QU=>STACK_FULL, QD=>STACK_EMPTY,
CV=>STORED_NUMBER);
```

공정라인에서 중간 적재 완충부에 임시 저장될 수 있는 용량 STACK_MAX 의 값이 100 입니다. 적재부에 자재가 투입될 때마다 IN 신호가 1 이 되고, 적재부에서 자재가 빠져 나갈 때마다 OUT 신호가 1 이 되도록 설정되어 있습니다. 운전을 시작하여 선 공정에서 완충부에 투입되는 속도가 다음 공정을 위해 빠져나가는 속도보다 빠를 때, 중간 적재부에 100 개가 쌓이면 카운터의 상한 출력 QU 가 1 이 되어 STACK_FULL 에 1 이 출력됩니다. 반대로 중간 적재부에 남아 있는 것이 없으면 하한 출력 QD 가 1 이 되어 STACK_EMPTY 에 1 이 출력됩니다. STORED_NUMBER 에는 현재 적재부에 남아 있는 자재의 숫자가 표시됩니다.

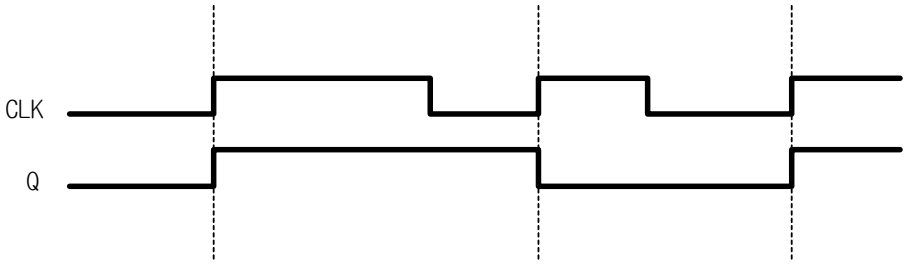

%MX1
%MX0
%IX0.1.0
%IX1.1.0
STACK_MAX(100개)
STORED_NUMBER
0
STACK_FULL
STACK_EMPTY

FF	적용 기종	발생플래그
출력 비트 반전	XGI, XGR, XEC	-
펑션 블록	설 명	
	**입력** CLK : 입력신호 **출력** Q : 상승에 지시 출력반전	

■ 기능

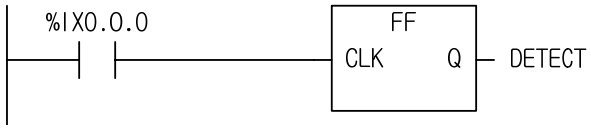
1. FF 는 CLK 에 연결된 입력의 상태가 0 에서 1 로 변할 때 출력 Q 를 반전 시킵니다.

■ 타임차트



■ 프로그램 예

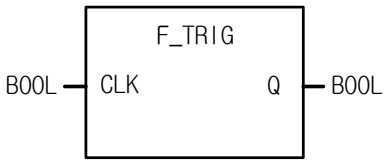
1. LD



2. ST

```
INST_FF(CLK:=%IX0.0.0, Q=>DETECT);
```

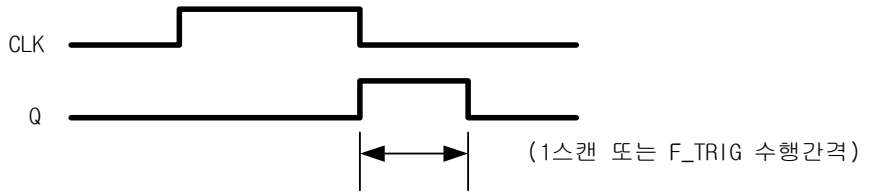
(1) 입력변수 %IX0.0.0 의 상태를 감시하여, 입력변수가 0 에서 1 으로 변할 때마다 출력변수 DETECT 의 출력을 반전시킵니다.

F_TRIG	적용 기종	발생플래그
하강 에지 검출 (펑션 블록)	XGI, XGR, XEC	-
펑션 블록	설 명	
	**입력** CLK : 입력신호 **출력** Q : 하강 에지 검출결과	

■ 기능

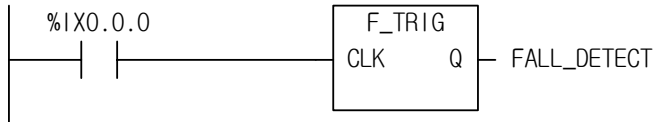
1. F_TRIG 는 CLK 에 연결된 입력의 상태가 1 에서 0 으로 변할 때 출력 Q 를 1 로 만들고 다음 수행에서 0 으로 만듭니다. 그 외는, 출력 Q 가 항상 0 이 됩니다.

■ 타임차트



■ 프로그램 예

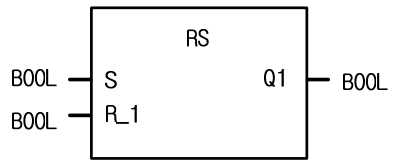
1. LD



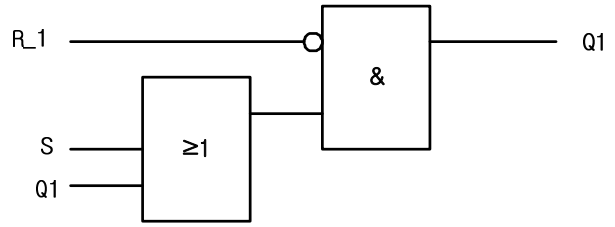
2. ST

```
INST_F_TRIG(CLK:=%IX0.0.0, Q=>FALL_DETECT);
```

(1) 입력변수 %IX0.0.0 의 상태를 감시하여 1 에서 0 으로 변할 때에 출력변수 FALL_DETECT 에 1 을 출력하고, 다음에 펑션 블록 수행시 FALL_DETECT 에 0 을 출력합니다.

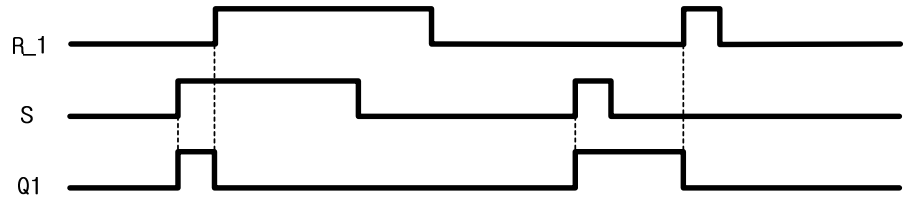
RS	적용 기종	발생플래그
Reset 우선 Bistable(펑션 블록)	XGI, XGR, XEC	-
펑션 블록	설 명	
	**입력** R_1 : Reset 조건 S : Set 조건 **출력** Q1 : 연산결과	

■ 기능



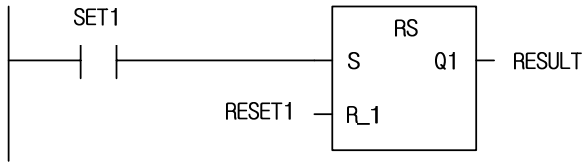
R_1 이 1 이면, S 와 관계없이 출력 Q1 은 항상 0 이 됩니다. 출력 Q1 은 이전 상태를 유지하며, R_1 이 0 이고 S 가 1 일 때 1 이 됩니다. Q1 의 초기상태는 0 입니다.

■ 타임차트



■ 프로그램 예

1. LD



2. ST

```
INST_RS(S:=SET1, R_1:=RESET1, Q=>RESULT);
```

RESET1 을 Reset 조건으로, SET1 을 Set 조건으로 하여 연산 결과를 RESULT 에 출력합니다.

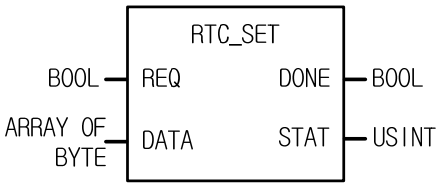
동작 조건은 위의 타임 차트에서 R_1 을 RESET1, S 를 SET1 으로, Q1 을 RESULT 로 대치하여 보십시오.

(1) 입력 변수로 선언된 SET1 이 On 되면 출력 변수 RESULT 는 1 이 됩니다.

(2) 입력 변수로 선언된 RESET1 이 On 되면 RESULT 로 선언된 출력 변수 값은 0 이 됩니다.

(3) 입력 변수로 선언된 SET1 과 RESET1 이 On 되면 출력 변수 RESULT 는 0 가 됩니다.

RTC_SET	적용 기종	발생플래그
시간 데이터 쓰기	XGI, XGR, XEC	_ERR, _LER

펑션 블록	설 명
	**입력** REQ : Rising Edge 시 펑션 블록 실행 DATA : 입력할 시간 데이터 **출력** DONE : RTC_SET 을 정상적으로 수행되면 1 을 출력 STAT : 에러 발생시 에러 코드 출력

■ 기능

1. REQ 변수의 Rising Edge 에서 아래와 같이 Setting 한 RTC 시간(DATA)을 PLC 의 Clock Device 에 저장합니다.

변수	내용	예	변수	내용	예
DATA[0]	년도	16#01	DATA[4]	분	16#30
DATA[1]	월	16#03	DATA[5]	초	16#45
DATA[2]	일	16#15	DATA[6]	체크 안함	-
DATA[3]	시	16#18	DATA[7]	년대	16#20

* 위 예는 2001 년 3 월 15 일 (목) 18 시 30 분 45 초인 경우입니다.
* 요일 데이터는 별도로 입력하지 않습니다. 날짜에 따른 요일이 자동으로 설정됩니다.

2. 위 DATA 변수는 반드시 어레이 BYTE 변수로 선언하고, 각각의 데이터는 BCD 값으로 Setting 합니다.

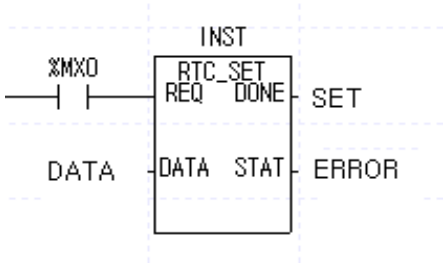
■ 플래그

플래그	설명
_ERR	CPU 가 RTC 를 지원하지 않는 경우나, RTC 데이터가 범위를 벗어난 경우 출력(DONE)은 '0' 이 STAT 에는 아래표와 같이 에러 코드를 출력합니다.

에러 코드	내용
00	No error
02	부적절한 RTC 데이터 입니다. 예) 14(월) 32(일) 25(시간) * RTC 데이터를 바르게 설정하여 주십시오.

■ 프로그램 예

1. LD



2. ST

```
INST_RTC_SET(REQ:=%MX0, DATA:=DATA, DONE=>SET, STAT=>ERROR);
```

RTC 데이터가 2006. 12. 05. 10:39:45, 화요일인 경우입니다.

(1) 설정스위치가 On 될 때 설정값으로 PLC 의 RTC 값이 입력 또는 변경됩니다.

(2) 아래는 변수(설정값) 설정 방법입니다.

변수	내용	예	변수	내용	예
DATA[0]	년도	16#06	DATA[4]	분	16#39
DATA[1]	월	16#12	DATA[5]	초	16#45
DATA[2]	일	16#05	DATA[6]	체크 안 함	-
DATA[3]	시	16#10	DATA[7]	년대	16#20

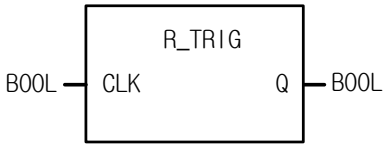
(3) 설정값은 DATA[] 변수에 초기값을 주어 설정하는 방법 외에 펑션 MOVE 를 사용하여 각각의 설정치를 DATA[]변수 저장하여 설정할 수 있습니다.

(4) RTC 값을 읽을 경우 아래의 플래그를 사용합니다.

예: 1998. 12. 22. 19:37:46, 화

키워드	Type	내용	설명	Data
_RTC_TOD	TOD	현재 시각	현재 시각 데이터	TOD#19:37:46
_RTC_WEEK	UINT	현재 요일	요일 데이터 *(0: 일, 1: 월, 2: 화, 3: 수, 4: 목, 5: 금, 6: 토)	2
_RTC_DATE	DATE	현재 날짜	현재 날짜 데이터(1984 년 1 월 1 일 ~2163 년 06 월 06 일)	D#1998-12-22
_HUND_WK	WORD	백년/요일	각각 BYTE 단위로 구분합니다.	16#1902
_TIME_DAY	WORD	시/일		16#1922

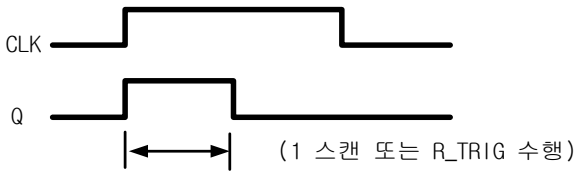
_MON_YEAR	WORD	월/년		16#1298
_SEC_MIN	WORD	초/분		16#4637

R_TRIG	적용 기종	발생플래그
상승에지 검출 (펑션 블록)	XGI, XGR, XEC	-
펑션 블록	설 명	
	**입력** CLK : 입력신호 **출력** Q : 상승 에지 검출결과	

■ 기능

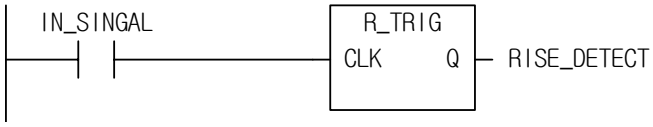
1. R_TRIG 는 CLK 에 연결된 입력의 상태가 0 에서 1 로 변할 때 출력 Q 를 1 로 만들고 R_TRIG 재 실행시 0 으로 만듭니다. 그 이외의 경우, 출력 Q 는 항상 0 이 됩니다.

■ 타임차트



■ 프로그램 예

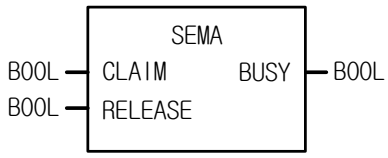
1. LD



2. ST

```
INST_R_TRIG(CLK:=IN_SIGNAL, Q=>RISE_DETECT);
```

입력변수로 선언된 IN_SIGNAL 의 상태를 감시하여 0 에서 1 로 변할 때, RISE_DETECT 에 1 을 출력하고 R_TRIG 펑션 블록 수행시 RISE_DETECT 에 0 을 출력합니다.

SEMA	적용 기종	발생플래그
시스템 자원 배분 (펑션 블록)	XGI, XGR, XEC	-
펑션 블록	설 명	
	**입력** CLAIM : 자원독점 요구신호 RELEASE : 해제신호 **출력** BUSY : 요구자원의 취득불가신호(대기)	

■ 기능

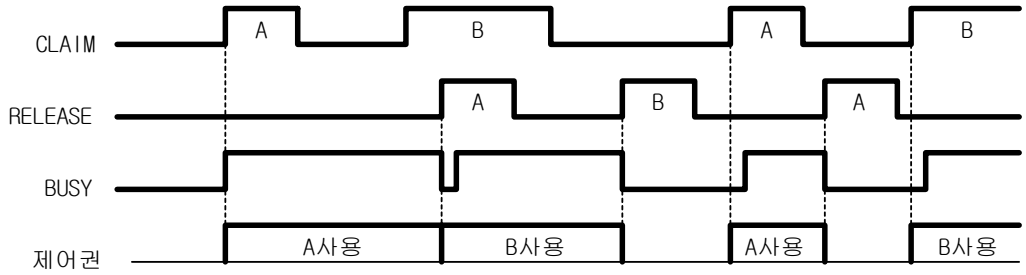
이 펑션 블록은 시스템 자원에 대한 독점적 제어권을 취득하는 데 사용합니다. SEMA 펑션 블록을 수행했을 때 (CLAIM = 1 또는 0, RELEASE = 0 값으로) 타 프로그램에서 자원을 사용 중이면 BUSY 가 1 이 됩니다. 자원의 제어권을 획득하고자 할 때에는 CLAIM = 1, RELEASE = 0 의 값으로 SEMA 를 계속 기동 하여 BUSY 가 0 이 될 때를 기다립니다. BUSY 가 0 이 되면 해당 자원에 대하여 제어를 하고 제어 동작이 끝난 후에는 CLAIM = 0, RELEASE = 1 의 값으로 SEMA 를 한 번 수행하여 제어권을 양도합니다.
(이때 CLAIM = 0, RELEASE = 1 의 값으로 SEMA 를 수행하는 제어 양도 동작은 반드시 현재 제어권을 가지고 있는 프로그램에서만 수행해야 합니다.)

1. SEMA 의 인스턴스는 자원을 요구하는 프로그램에서 공통으로 액세스할 수 있도록 글로벌 영역에 설정하여야 합니다.
2. 동일한 자원을 요구하는 각각의 프로그램은 우선 순위상 동일한 우선순위로 지정되어 있어야 합니다.
3. SEMA 펑션 블록의 내부 수행 구조

```
VAR  X : BOOL : = 0 ;  END_VAR
BUSY : = X ;
IF  CLAIM  THEN  X : = 1 ;
ELSIF  RELEASE  THEN  BUSY : = 0; X : = 0 ;
END_IF
```

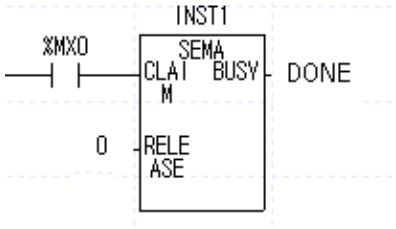
■ 타임차트

프로그램 블록 A 와 프로그램 블록 B 에서 동일한 자원에 대한 액세스권을 주고 받는 경우



■ 프로그램 예

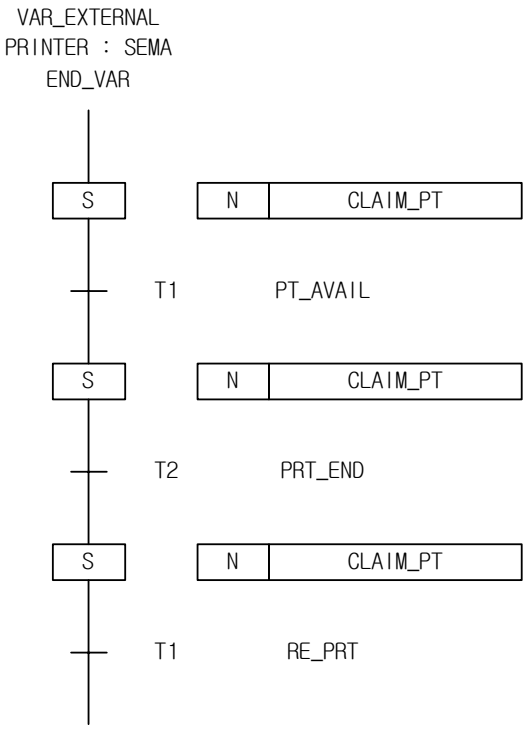
1. LD



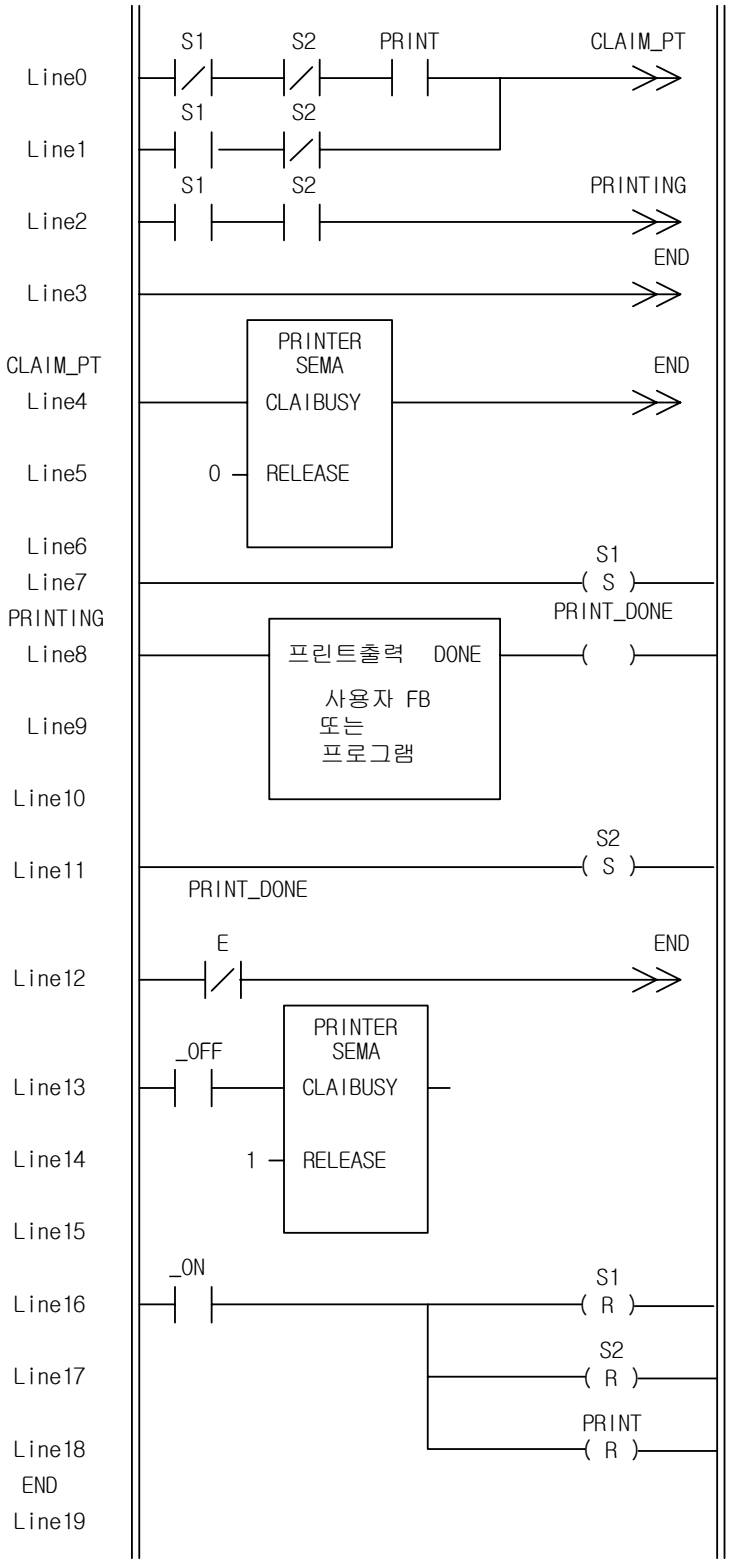
2. ST

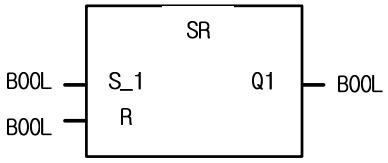
```
INST_SEMA(CLAIM:=%MX0, RELEASE:=0, BUSY=>DONE);
```

PLC 시스템에 부착되어 있는 프린터로 서로 다른 프로그램 블록에서 출력을 내고자 할 때 인스턴스 'PRINTER'를 글로벌로 선언하고 각각의 프로그램에서 'PRINTER'로 명명된 SEMA 펑션 블록을 이용하여 프린터의 제어권을 쉽게 제어할 수 있습니다. 프린터에 출력이 필요한 시점에서 START 는 1, END 는 0 인 상태에서 'PRINTER' SEMA 를 수행하여 프린터의 제어권을 요구했을 때 다른 프로그램 블록에서 이미 프린터를 사용하고 있다면 BUSY 신호가 1 이 되어 OT_AVAIL 에 1 이 출력됩니다. 만약 다른 블록에서 프린터를 사용하고 있지 않다면 BUSY 는 0 인 상태가 되어 그것을 신호로 프린터에 출력을 내는 프로그램을 기동시키면 됩니다. 프린트 동작이 모두 끝난 후에는 START 는 0, END 는 1 인 상태로 'PRINTER' SEMA 를 실행하여 타 부분에서 프린터의 제어권을 가져갈 수 있도록 해제하여 주어야 합니다.

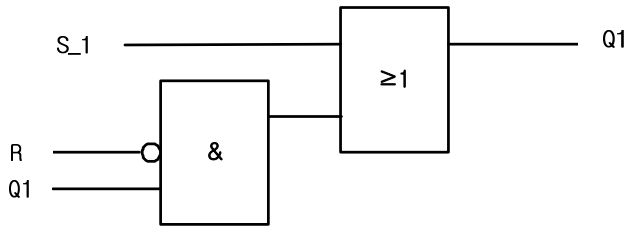


S	CLAIM_PT;프린터 제어권 요구
	CAL SEMA PRINTER CLAIM:= 1 RELEASE:= 0
T	PT_AVAIL;프린터 제어권 획득 체크
	LDN PRINTER.BUSY ST TRANS
S	PRINTING;프린터 출력
	프린터 제어 프로그램 프린터 완료시 PRINT_DONE:=1
T	PRT_END;프린터 완료 체크
	LD PRINTER_DONE ST TRANS
S	REL_PRT;프린터 제어권 양도
	CAL SEMA PRINTER CLAIM:= 0 RELEASE:= 1
T	RE_PRT;프린터 재요구
	LD PRT_REQ ST TRANS


Line0
S1
S2
PRINT
CLAIM_PT
Line1
S1
S2
Line2
S1
S2
PRINTING
Line3
END
CLAIM_PT
Line4
PRINTER
SEMA
CLAIBUSY
END
Line5
0
RELEASE
Line6
Line7
S1
(S)
PRINTING
Line8
PRINT_DONE
프린트출력 DONE
( )
Line9
사용자 FB
또는
프로그램
Line10
Line11
S2
(S)
PRINT_DONE
Line12
E
END
Line13
PRINTER
SEMA
_OFF
CLAIBUSY
Line14
1
RELEASE
Line15
Line16
_ON
S1
(R)
Line17
S2
(R)
Line18
PRINT
(R)
END
Line19

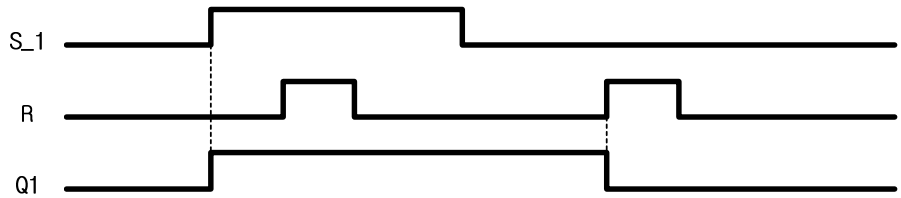
SR	적용 기종	발생플래그
Set 우선 Bistable (펑션 블록)	XGI, XGR, XEC	-
펑션 블록	설 명	
	**입력** S_1 : Set 조건 R : Reset 조건 **출력** Q1 : 연산결과	

■ 기능



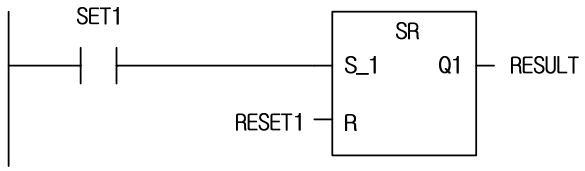
1. S_1이 1이 되면, R과 관계없이 출력 Q1은 항상 1이 됩니다.
2. 출력 Q1은 이전 상태를 유지하며, S_1이 0이고 R이 1일 때 0이 됩니다.
3. Q1의 초기상태는 0입니다.

■ 타임차트



■ 프로그램 예

1. LD

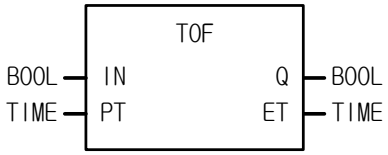


2. ST

```
INST_SR(S_1:=SET1, R:=RESET1, Q=>RESULT);
```

(1) 입력변수로 선언된 SET1 이 On 되면, 출력변수로 선언된 RESULT 같이 1 이 됩니다.

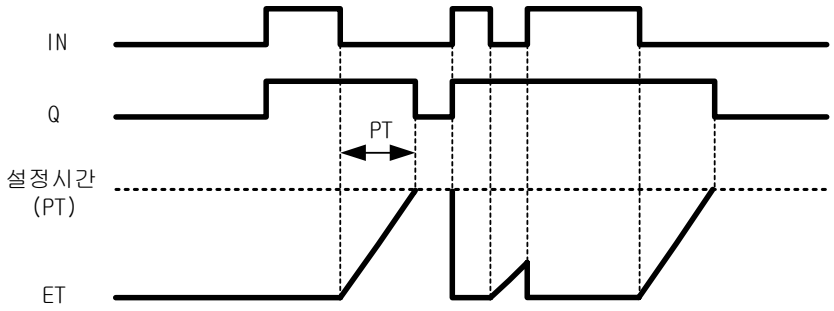
(2) 출력변수로 선언된 RESULT 값이 0 이 되도록 하려면, 입력변수로 선언된 SET1 이 Off 되고, RESET 이 On 되어야 합니다.

TOF	적용 기종	발생플래그
Off 딜레이 타이머 (펑션 블록)	XGI, XGR, XEC	-
펑션 블록	설 명	
	**입력** IN : 타이머 기동 조건 PT : 설정 시간(Preset Time) **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)	

■ 기능

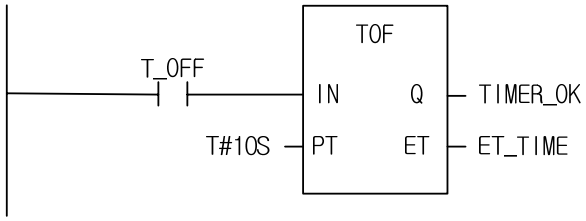
1. IN 이 1 이 되면, Q 가 1 이 되고, IN 이 0 이 된 후부터 PT 에 의해서 지정된 설정시간이 경과한 후 Q 가 0 이 됩니다.
2. IN 이 0 이 된 후 경과시간이 ET 로 출력됩니다.
3. 만일 경과시간 ET 가 설정시간에 도달하기 전에 IN 이 1 이 되면, 경과시간은 다시 0 으로 됩니다.

■ 타임차트



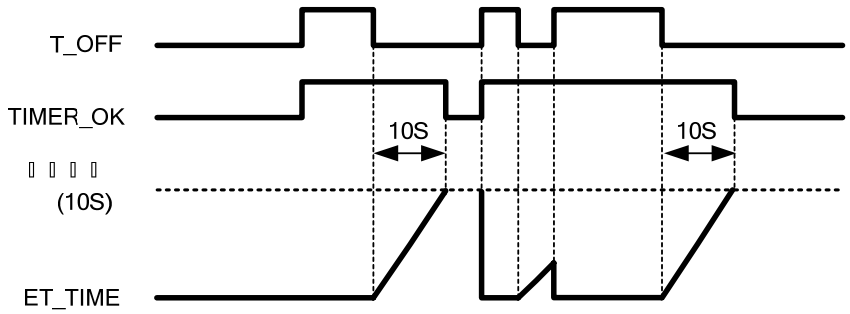
■ 프로그램 예

1. LD

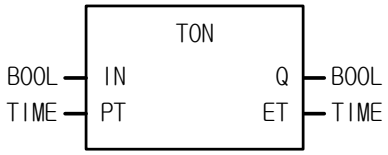


2. ST

```
INST_TOF(IN:=T_OFF, PT:=T#10S, Q=>TIMER_OK, ET=>ET_TIME);
```



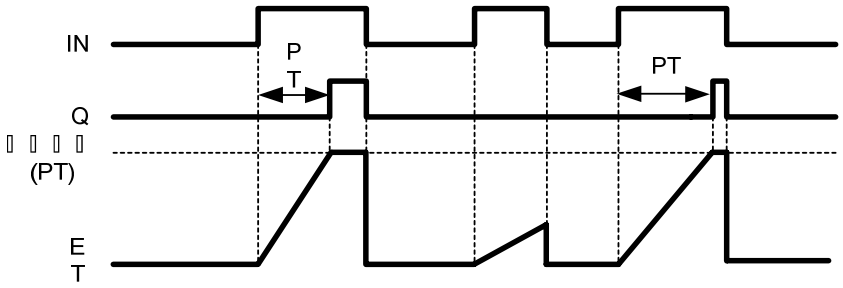
(1) 입력변수로 설정된 T_OFF 가 1 이 되면, 출력변수 TIMER_OK 에 1 이 출력되고 T_OFF 가 0 이 된 후 10 초 후에 TIMER_OK 가 0 이 됩니다.
(2) T_OFF 가 0 이 된 후 10 초 이내에 다시 1 이 되면 타이머는 다시 초기상태가 됩니다.
(3) 타이머의 시간 측정값은 ET_TIME 로 출력됩니다.

TON	적용 기종	발생플래그
On 딜레이 타이머 (펑션 블록)	XGI, XGR, XEC	-
펑션 블록	설 명	
	**입력** IN : 타이머 기동 조건 PT : 설정 시간 (Preset Time) **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)	

■ 기능

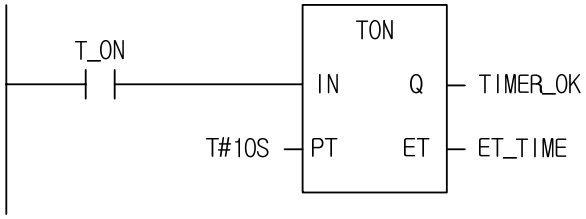
1. IN이 1이 된 후 경과 시간이 ET로 출력됩니다.
2. 만일 경과시간 ET가 설정 시간에 도달하기 전에 IN이 0이 되면, 경과 시간은 0으로 됩니다.
3. Q가 1이 된 후 IN이 0이 되면, Q는 0이 됩니다.

■ 타임차트



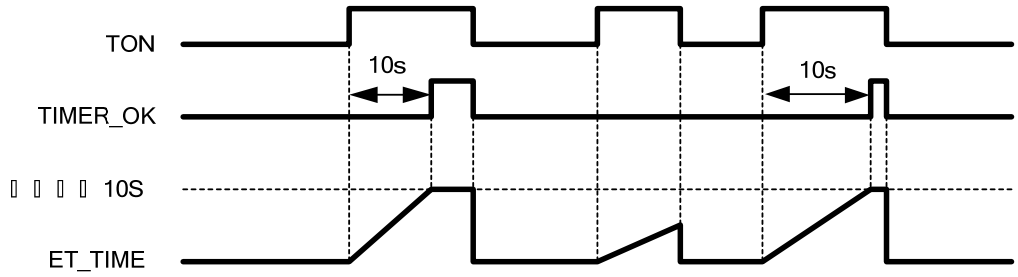
■ 프로그램 예

1. LD



2. ST

```
INST_TON(IN:=T_ON, PT:=T#10S, Q=>TIMER_OK, ET=>ET_TIME);
```

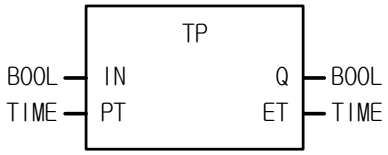


(1) 입력변수 T_ON 이 1 이 된 후 10 초가 경과한 후 출력 변수 TIMER_OK 가 1 이 됩니다.

(2) 입력변수 T_ON 이 1 이 된 후 경과 시간이 출력 변수 ET_TIME 로 출력됩니다.

(3) 만일 경과 시간 ET_TIME 이 설정 시간 10 초에 도달하기 전에 T_ON 이 0 이 되면, 경과 시간 ET_TIME 은 0 으로 됩니다.

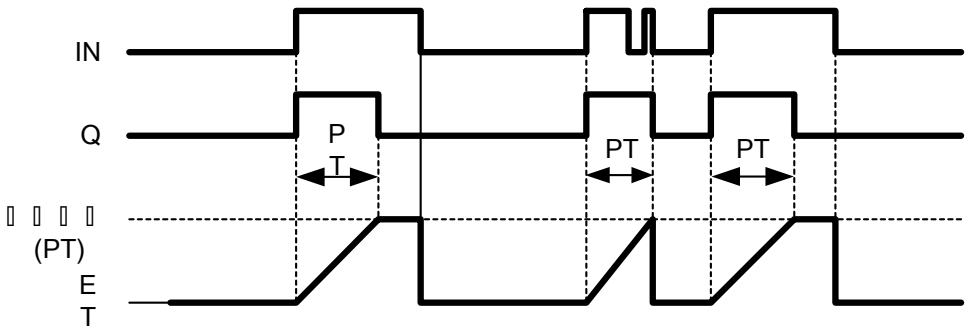
(4) TIMER_OK 가 1 이 된 후 T_ON 이 0 이 되면, TIMER_OK 는 0 이 되고 경과 시간 ET_TIME 도 0 으로 됩니다.

TP	적용 기종	발생플래그
펄스 타이머 (펑션 블록)	XGI, XGR, XEC	-
펑션 블록	설 명	
	**입력** IN : 타이머 기동조건 PT : 설정 시간 (Preset Time) **출력** Q : 타이머 출력 ET : 경과 시간 (Elapsed Time)	

■ 기능

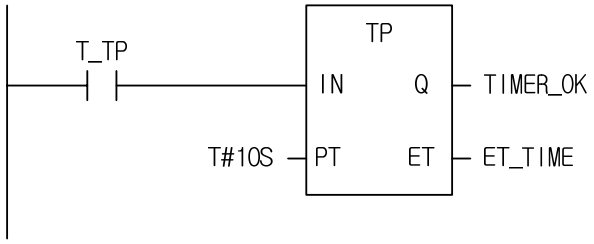
1. IN 이 1 이 되면 PT 에 의해서 지정된 설정 시간 동안만 Q 가 1 이 되고, ET 가 PT 에 도달하면 자동으로 0 이 됩니다.
2. 경과시간 ET 는 IN 이 1 이 되었을 때부터 증가하며 PT 에 이르면 값을 유지하다가 IN 이 0 이 될 때 0 의 값이 됩니다.
3. ET 가 증가할 동안은 IN 이 0 이 되거나 재차 1 이 되어도 영향이 없습니다.

■ 타임차트



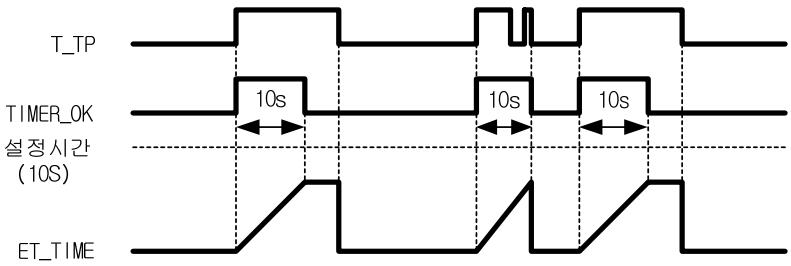
■ 프로그램 예

1. LD



2. ST

```
INST_TP(IN:=T_TP, PT:=T#10S, Q=>TIMER_OK, ET=>ET_TIME);
```



(1) 입력변수 T_TP 가 0 에서 1 이 된 후 10 초 동안 TIMER_OK 는 1 이 됩니다. 일단 타이머가 가동된 후에는 10 초 동안 T_TP 신호의 변화는 무시됩니다.

(2) ET_TIME 값은 증가 후 T#10S 에서 멈춥니다. 그리고 T_TP 가 0 이 될 때 0 으로 됩니다.

☆ 참고

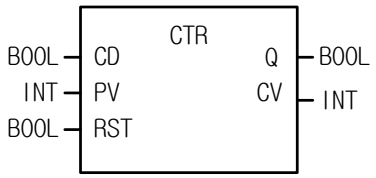
TP 펑션 블록은 접점이 On 된 후 Off 되어도 해당 동작이 마무리 될 때까지 펑션 블록은 동작합니다.

배열 인덱스를 이용한 변수를 사용하는 경우 배열 인덱스 에러는 접점이 On 이 되었을 때만 발생합니다.

그렇기 때문에 TP 펑션 블록에서는 펑션 블록이 동작하더라도 접점이 Off 되어 있다면 배열 인덱스 에러가 발생하지 않습니다.

제 10 장 응용 펑션 블록

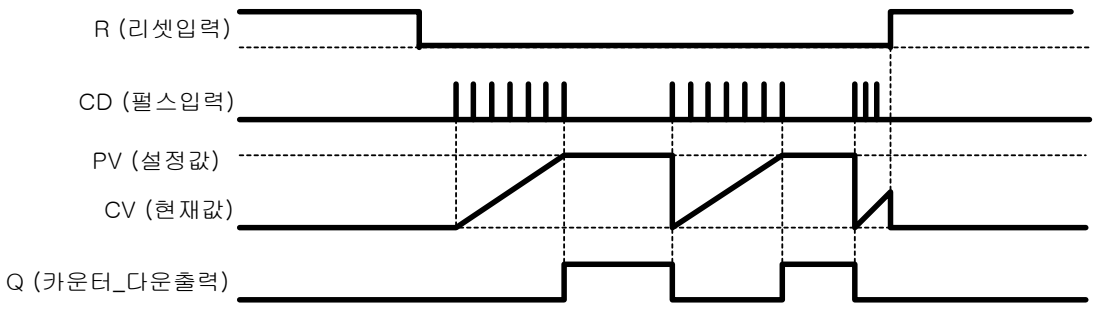
9 장에서 설명한 기본 펑션 블록과는 다른 응용 펑션 블록에 대하여 설명합니다.

CTR	적용 기종	발생플래그
Ring 카운터	XGI, XGR, XEC	-
펑션 블록	설 명	
	**입력** CD : 링 카운트(Ring Count) 펄스 입력 PV : 설정값(Preset Value) RST : 리셋 입력(Reset) **출력** Q : 링 카운트(Ring Count) 출력 CV : 현재값(Current Value)	

■ 기능

1. 링 카운터 CTR 펑션 블록은 펄스 입력 CD 가 0 에서 1 이 될 때마다 현재값 CV 를 +1 하고 현재값 CV 가 설정값 PV 에 도달한 후 CD 가 다시 0 에서 1 로 되면 현재값은 1 로 됩니다.
2. 현재값이 설정값에 도달하면 출력(Q)은 1 이 됩니다.
3. 현재치가 설정치 미만이거나 Reset 조건이 1 이 되면 출력(Q)은 0 이 됩니다

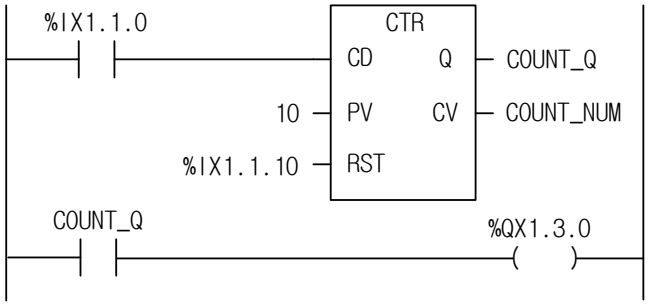
■ 타임 차트



■ 프로그램 예

입력 접점 %IX1.1.0 에 10 번의 펄스가 들어올 때마다, 출력접점 %QX1.3.1 을 On 시키는 프로그램

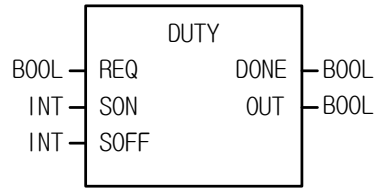
1. LD



2. ST

```
INST_CTR(CD:=%IX1.1.0, PV:=10, RST:=%IX1.1.10, Q=>COUNT_Q, CV=>COUNT_NUM);
%QX1.3.0 := COUNT_Q;
```

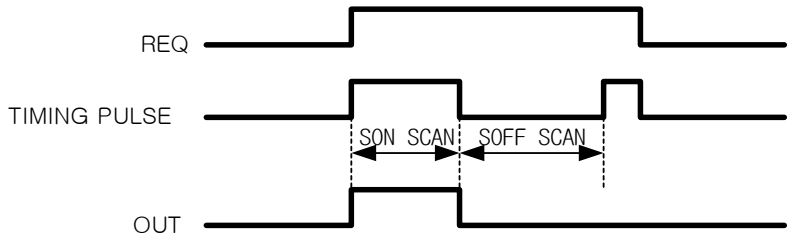
(1) CTR 펑션 블록의 이름을 등록합니다. (INS_CTR)
(2) CD 에 펄스 입력이 들어올 입력 접점 %IX1.1.0 을 입력합니다.
(3) PV 에 10 을 입력합니다.
(4) CV 를 초기화 시키는 RST 에는 임의의 입력 접점을 설정합니다. (%IX1.1.10)
(5) CV 에는 임의의 변수(COUNT_NUM)를 설정하여 입력합니다.
(6) Q 에는 임의의 출력변수(COUNT_Q)를 설정 입력합니다.
(7) 프로그램의 작성이 완료되면, 컴파일을 실행하고, PLC 로 쓰기를 합니다.
(8) 쓰기를 완료하면 모드전환(Stop → Run)시킵니다.
(9) 입력펄스가 입력접점 %IX1.1.0 에 들어오면 현재값 CV(COUNT_NUM)은 1 만큼 증가합니다.
(10) 입력접점에 10 번의 펄스가 들어오면 현재값 CV 는 10 이 되고, 설정값과 같게 되므로 출력 Q(COUNT_Q)가 1 이 됩니다.
(11) Q(COUNT_Q)가 1 이 되면 출력 접점 %QX1.3.0 은 On 됩니다.
(12) 다시 한번의 입력 펄스가 입력 접점 %IX1.1.0 에 들어오면 Q(COUNT_Q)는 0 이 되고 출력 접점인 %QX1.3.0 은 Off 됩니다.

DUTY	적용 기종	발생플래그
스캔 지정 On/Off	XGI, XGR. XEC	-
펑션 블록	설 명	
	**입력** REQ : 펑션 블록 실행 요구 SON : On 될 Scan 수 SOFF : Off 될 Scan 수 **출력** DONE : REQ 가 On 이고 SON, SOFF 두 입력변수가 0 보다 작지 않으면 1 을 출력 OUT : On Scan Time 동안 1 을 출력	

■ 기능

1. DUTY FB 는 REQ 가 On 된 시점부터 SON Scan 동안 On, SOFF Scan 동안 Off 하는 펄스를 발생시킵니다.
2. SON = 0 이면 출력은 항상 Off 가 됩니다.
3. SON > 0, SOFF = 0 이면 출력은 항상 On 이 됩니다.
4. REQ 가 Off 면 출력은 Off 됩니다.
5. SON < 0 이거나 SOFF < 0 이면, DONE 은 Off 되고 OUT = 0 이 됩니다.

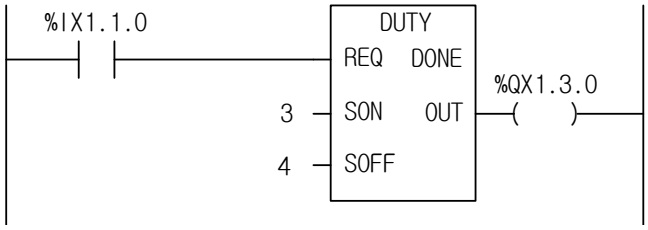
■ 타임 차트



■ 프로그램 예

입력접점 %IX1.1.0 이 SET 되어 있으면, 3 번의 스캔 타임 동안 출력 접점 %QX1.3.0 을 On 시키고, 4 번의 스캔타임 동안 출력 접점 %QX1.3.0 을 Off 시키는 프로그램

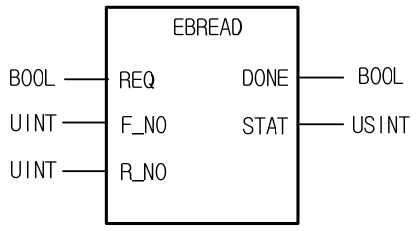
1. LD



2. ST

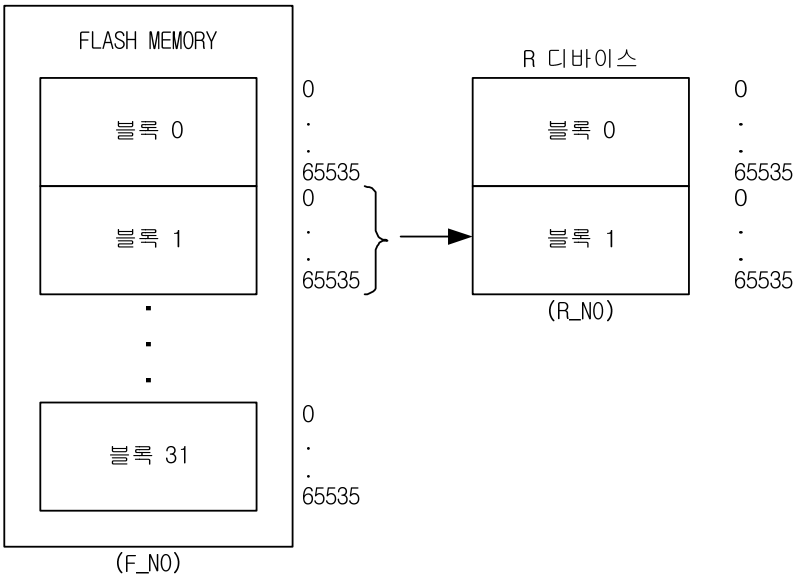
```
INST_DUTY(REQ:=%IX1.1.0, SON:=3, SOFF:=4, OUT=>%QX1.3.0);
```

(1) DUTY 펑션 블록의 이름을 등록합니다. (DUTY_C)
(2) REQ 에 펑션 블록을 실행시킬 입력 접점 %IX1.1.0 을 입력합니다.
(3) SON 에 3 을 입력합니다.
(4) SOFF 에 4 를 입력합니다.
(5) OUT 에 출력 접점 %QX1.3.0 을 입력합니다.
(6) 프로그램의 작성이 완료되면, 컴파일을 실행하고 PLC 로 쓰기를 수행합니다.
(7) 쓰기를 완료하면 모드전환(Stop → Run) 시킵니다.
(8) 프로그램이 실행되면 3 번의 스캔 타임 동안 출력접점 %QX1.3.0 이 On 되고, 4 번의 스캔 타임 동안에는 출력 접점 %QX1.3.0 이 Off 되는 동작을 반복합니다.

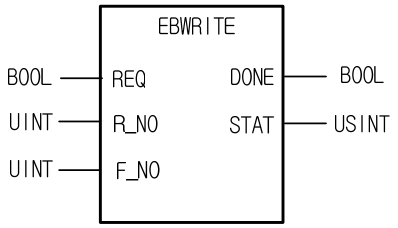
EBREAD	적용 기종	발생플래그
플래시 영역에 R 영역 데이터 읽기	XGI, XEC	-
펑션 블록	설 명	
	**입력** REQ : 펑션 블록 실행요구 F_NO : 데이터를 읽을 플래시 블록 번호 R_NO : R 영역의 블록 번호 **출력** DONE : 정상적으로 동작 후 1 을 유지 STAT : 에러시 정보 표시	

■ 기능

1. 플래시 영역의 1 개 블록(64Kbyte) 데이터를 지정한 R 영역의 블록으로 저장합니다. 정상적으로 수행 완료시 DONE 은 1 이 됩니다.

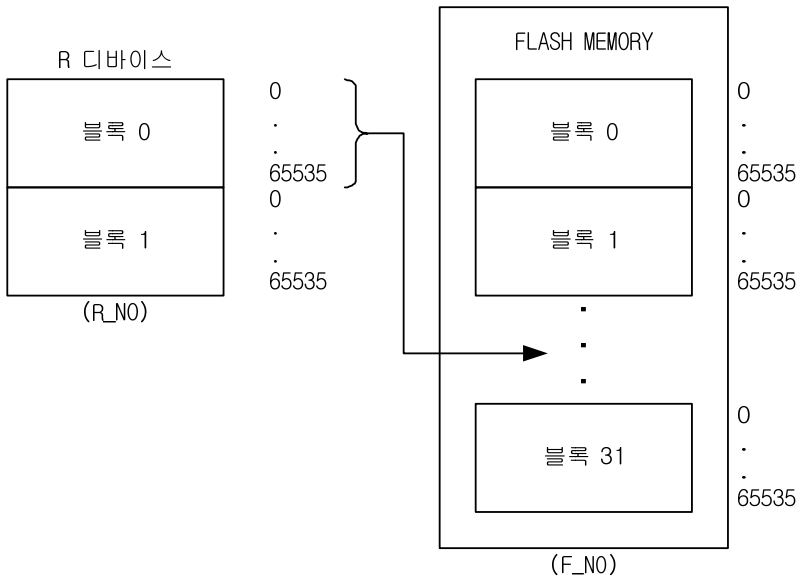


2. R_NO 의 값이 2 이상이면 STAT = 1, F_NO 의 값이 32 이상이면 STAT = 2 가 되며, _ERR, _LER 이 On 됩니다.
   또한 플래시로부터 읽기를 진행하고 있을 경우에는 DONE = 0 이고, STAT = 5 가 됩니다.
   이미 플래쉬 영역에 읽기/쓰기가 진행중인 경우에는 명령어 동작 시 DONE = 0 이고, STAT = 10 이 됩니다.
3. 명령어 수행 중에 _RBLOCK_RD_FLAG 의 F_NO 에 해당하는 비트가 On 됩니다.

EBWRITE	적용 기종	발생플래그
R 영역 데이터를 플래시 영역에 쓰기	XGI, XEC	-
펑션 블록	설 명	
	**입력** REQ : 펑션 블록 실행요구 R_NO : R 영역(내부램)의 블록번호 E_NO : 저장할 플래쉬 영역의 블록 번호 **출력** DONE : 정상적으로 동작 후 1 을 유지 STAT : ERR 정보	

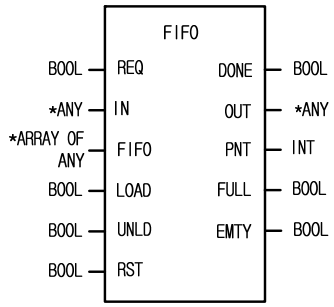
■ 기능

1. 지정된 R 영역의 1 개의 블록(64Kbyte) 내용을 저장할 플래쉬 영역의 블록으로 데이터를 전송합니다. 정상적으로 수행 완료시 DONE 이 1 이 됩니다.



2. R_NO 의 값이 2 이상이면 STAT = 1, F_NO 의 값이 32 이상이면 STAT = 2 가 되며, _ERR, _LER 이 On 됩니다.
   또한 플래시에 쓰기를 진행하고 있을 경우에는 DONE = 0 이고, STAT = 5 가 됩니다.
   이미 플래쉬 영역에 읽기/쓰기가 진행중인 경우에는 명령어 동작 시 DONE = 0 이고, STAT = 10 이 됩니다.
3. 명령어 수행 중에 _RBLOCK_WR_FLAG 의 F_NO 에 해당하는 비트가 On 됩니다.

FIFO	적용 기종	발생플래그
FIFO 스택에 값을 Load / Unload (선입 선출)	XGI, XGR, XEC	-

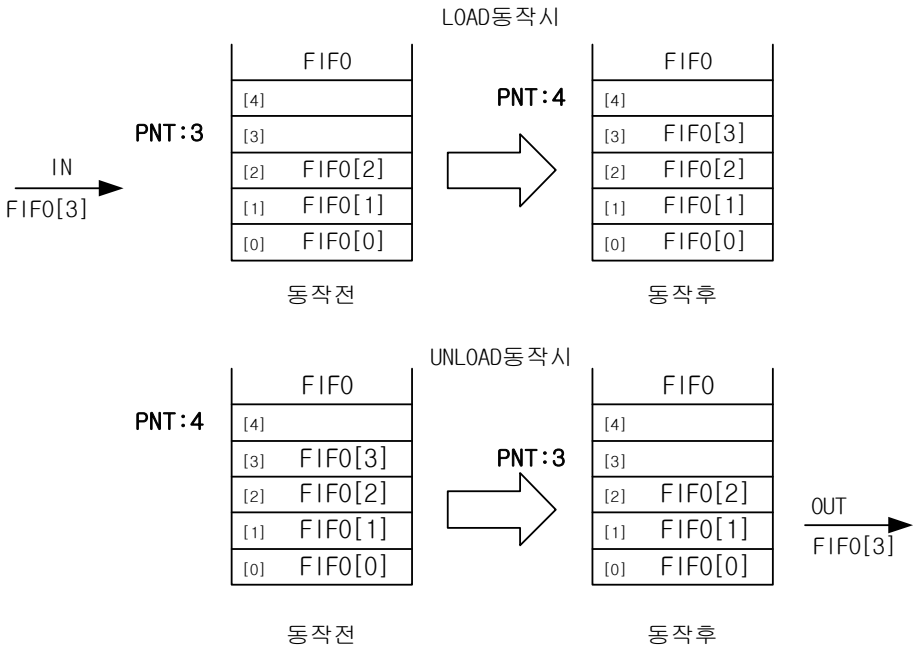
펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 IN : FIFO 스택에 저장할 변수 또는 상수값 LOAD : On 이면 입력 모드 UNLD : On 이면 출력 모드 RST : POINTER VALUE 리셋 FIFO : FIFO 스택으로 사용되는 어레이 **출력** DONE : 최초 동작 후 1 을 유지 OUT : 출력 모드일 경우 FIFO 스택으로부터 나온 값을 출력 PNT : FIFO 스택에 입력된 값에 대한 Pointer FULL : FIFO 스택이 가득차면 1 을 출력 EMTY : FIFO 스택에 아무런 값도 저장되어 있지 않으면 1 을 출력

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	FIFO	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ANY: ANY 타입 중 STRING 제외, *ARRAY OF ANY: ARRAY_ANY 타입 중 STRING 제외

■ 기능

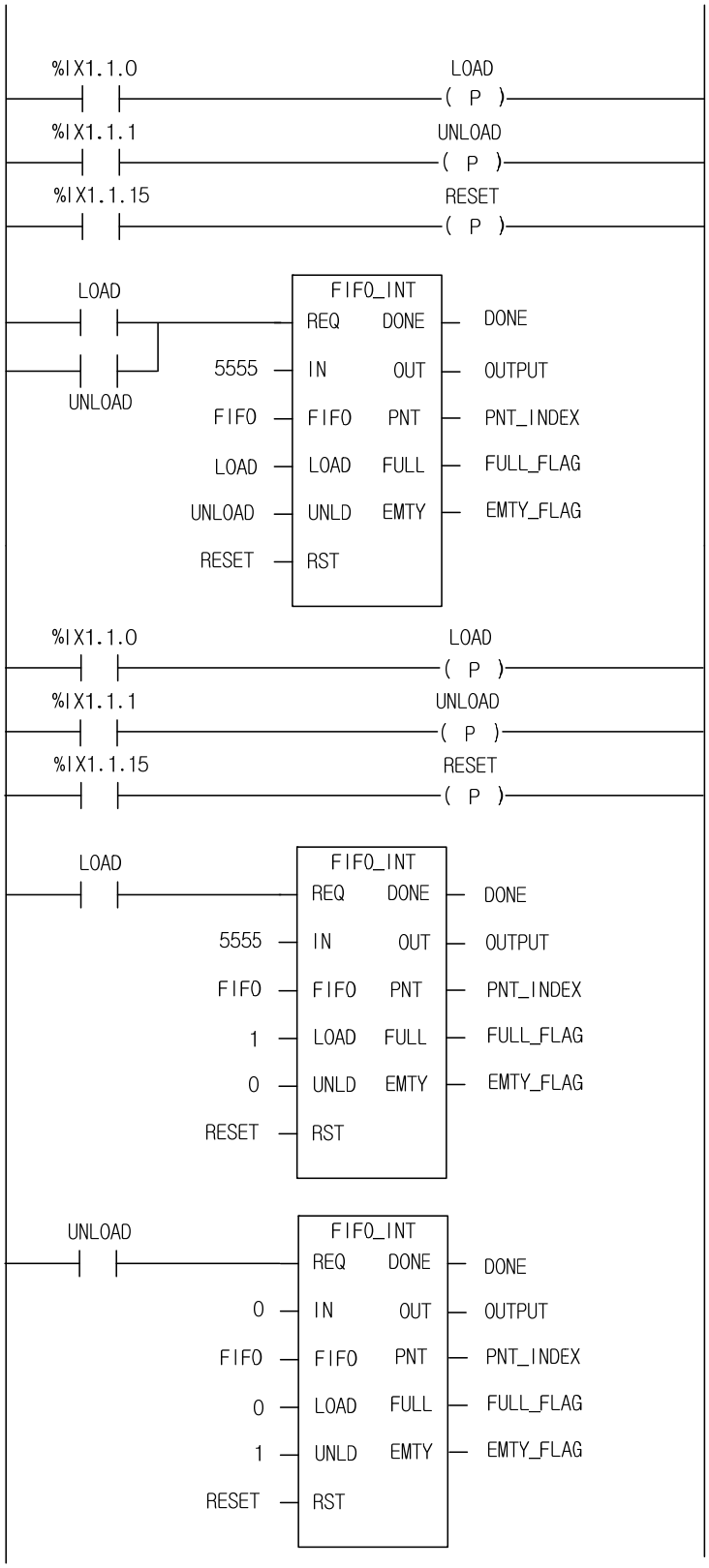
1. FIFO 펑션 블록은 IN 값을 FIFO 에 Load 또는 FIFO 로부터 값을 Unload 합니다.
2. 입력모드 지정과 출력모드 지정이 동시에 On 되면 입출력을 동시에 수행합니다.
3. FIFO 로부터 값을 Unload 하면 스택의 최하위 원소가 출력되고, 나머지 값들은 Shift 되며, PNT 값이 하나 줄고, PNT 가 위치한 곳은 0 으로 클리어(clear)됩니다.
4. RST 가 입력되면 PNT 는 0 으로 초기화되고, EMTY 가 On 되며, FIFO 스택의 모든 값은 0 으로 클리어(clear)됩니다.
5. 스택의 개수는 입출력 변수 FIFO 에 지령되는 입력시 어레이의 개수가 됩니다.
6. 정전시 또는 전원 Off 시에도 입력된 값들을 유지하기 위해서는 FIFO 어레이 변수와 FIFO 펑션 블록 인스턴스를 모두 RETAIN 으로 설정하여야 합니다.
7. 리셋 동작은 REQ 요구가 없어도 동작이 가능합니다.
8. PNT 는 다음번 Load 동작시 IN 값이 입력될 위치를 나타냅니다. 또는 Load 되어 있는 전체 개수를 나타낸다고 볼 수 있습니다.
9. 입력 모드일 경우 OUT 출력은 0 이 됩니다. 그러나 출력모드 동작 후 전환된 입력모드에서는 출력모드의 OUT 출력이 유지됩니다.



펑션 블록	FIFO 변수 타입	동작 설명
FIFO_BOOL	BOOL	BOOL 타입 DATA 에 대한 FIFO 동작을 수행합니다.
FIFO_BYTE	BYTE	BYTE 타입 DATA 에 대한 FIFO 동작을 수행합니다.
FIFO_WORD	WORD	WORD 타입 DATA 에 대한 FIFO 동작을 수행합니다.
FIFO_DWORD	DWORD	DWORD 타입 DATA 에 대한 FIFO 동작을 수행합니다.
FIFO_LWORD	LWORD	LWORD 타입 DATA 에 대한 FIFO 동작을 수행합니다.
FIFO_SINT	SINT	SINT 타입 DATA 에 대한 FIFO 동작을 수행합니다.
FIFO_INT	INT	INT 타입 DATA 에 대한 FIFO 동작을 수행합니다.
FIFO_DINT	DINT	DINT 타입 DATA 에 대한 FIFO 동작을 수행합니다.
FIFO_LINT	LINT	LINT 타입 DATA 에 대한 FIFO 동작을 수행합니다.
FIFO_USINT	USINT	USINT 타입 DATA 에 대한 FIFO 동작을 수행합니다.
FIFO_UINT	UINT	UINT 타입 DATA 에 대한 FIFO 동작을 수행합니다.
FIFO_UDINT	UDINT	UDINT 타입 DATA 에 대한 FIFO 동작을 수행합니다.
FIFO_ULINT	ULINT	ULINT 타입 DATA 에 대한 FIFO 동작을 수행합니다.
FIFO_REAL	REAL	REAL 타입 DATA 에 대한 FIFO 동작을 수행합니다.
FIFO_LREAL	LREAL	LREAL 타입 DATA 에 대한 FIFO 동작을 수행합니다.
FIFO_TIME	TIME	TIME 타입 DATA 에 대한 FIFO 동작을 수행합니다.
FIFO_DATE	DATE	DATE 타입 DATA 에 대한 FIFO 동작을 수행합니다.
FIFO_TOD	TOD	TOD 타입 DATA 에 대한 FIFO 동작을 수행합니다.
FIFO_DT	DT	DT 타입 DATA 에 대한 FIFO 동작을 수행합니다.

■ 프로그램 예

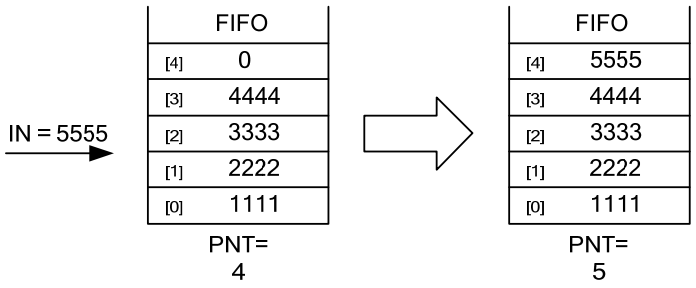
1. LD


%IX1.1.0
LOAD
(P)
%IX1.1.1
UNLOAD
(P)
%IX1.1.15
RESET
(P)
LOAD
UNLOAD
FIFO_INT
REQ
DONE
DONE
5555
IN
OUT
OUTPUT
FIFO
FIFO
PNT
PNT_INDEX
LOAD
LOAD
FULL
FULL_FLAG
UNLOAD
UNLD
EMTY
EMTY_FLAG
RESET
RST
%IX1.1.0
LOAD
(P)
%IX1.1.1
UNLOAD
(P)
%IX1.1.15
RESET
(P)
LOAD
FIFO_INT
REQ
DONE
DONE
5555
IN
OUT
OUTPUT
FIFO
FIFO
PNT
PNT_INDEX
1
LOAD
FULL
FULL_FLAG
0
UNLD
EMTY
EMTY_FLAG
RESET
RST
UNLOAD
FIFO_INT
REQ
DONE
DONE
0
IN
OUT
OUTPUT
FIFO
FIFO
PNT
PNT_INDEX
0
LOAD
FULL
FULL_FLAG
1
UNLD
EMTY
EMTY_FLAG
RESET
RST

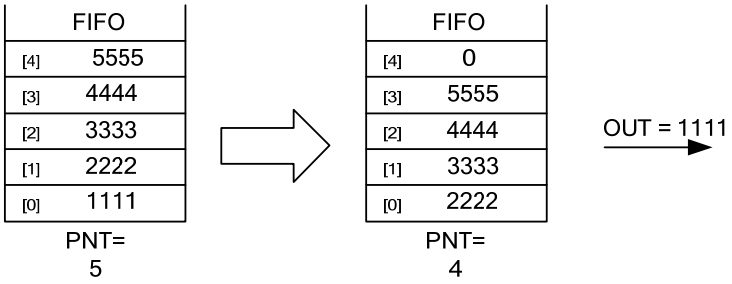
FIFO_INT 펑션 블록은 위의 그림과 같이 2 가지 방법으로 사용이 가능합니다. 위에서 예를 든 2 가지 방법은 동일한 동작을 수행합니다. 위의 펑션 블록은 하나의 펑션 블록 만을 사용하여 입력과 출력을 동시에 수행할 수 있도록 한 프로그램입니다. 아래의 펑션 블록은 입력동작을 위한 펑션 블록과 출력동작을 위한 펑션 블록을 따로 작성하여 입출력 동작을 다른 위치에서 동작하도록 작성한 프로그램입니다. 단, 이때 주의할 점은 인스턴스 이름이 같도록 설정해야 합니다.

(1) 입력조건(%IX1.1.0, %IX1.1.1, %IX1.1.15)이 성립되면 FIFO_INT 가 실행됩니다.
(2) 입력접점 %IX1.1.0 이 On 되면 Load 동작을 수행합니다. 5555 가 FIFO 스택에 입력되고 PNT_INDEX 가 1 증가합니다.
(3) 입력접점 %IX1.1.1 이 On 되면 Unload 동작을 수행합니다. FIFO 스택으로부터 1111 이 출력되고 PNT_INDEX 가 1 감소합니다.
(4) 입력접점 %IX1.1.15 가 On 되면 Reset 동작을 수행합니다. FIFO 스택의 모든 값이 0 으로 클리어(Clear)되고, PNT_INDEX 가 0 으로 초기화되며, EMTY_FLAG 가 On 됩니다.

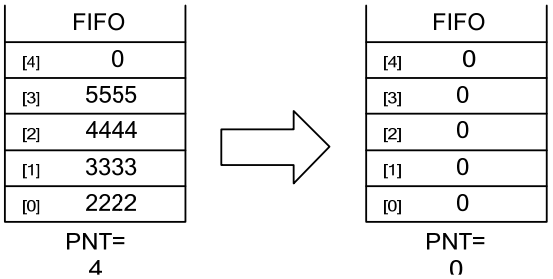
LOAD 동작 (%IX1.1.0 On 동작)



UNLOAD 동작 (%XI1.1.1 On 동작)



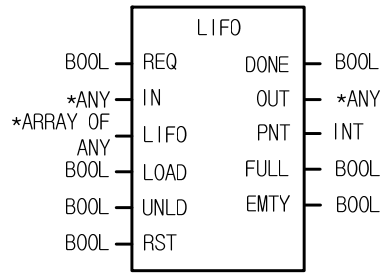
RESET 동작 (%IX1.1.15 On 동작)



2. ST

```
INST_FIFO_INT(REQ:=LOAD OR UNLOAD, IN:=5555, FIFO:=FIFO, LOAD:=LOAD, UNLD:=UNLOAD, RST:=RESET, DONE=>DONE, OUT=>OUTPUT, PNT=>PNT_INDEX, FULL=>FULL_FLAG, EMTY=>EMTY_FLAG);
```

LIFO	적용 기종	발생플래그
LIFO 스택에 값을 Load/Unload (후입 선출)	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 IN : LIFO 스택에 저장할 변수 또는 상수값 LOAD : On 이면 입력 모드 UNLD : On 이면 출력 모드 RST : POINTER VALUE 리셋 LIFO : LIFO 스택으로 사용되는 어레이 **출력** DONE : 최초 동작 후 1을 유지 OUT : 출력 모드일 경우 LIFO 스택으로부터 나온 값을 출력 PNT : LIFO 스택에 입력된 값에 대한 Pointer FULL : LIFO 스택이 가득차면 1을 출력 EMTY : LIFO 스택에 아무런 값도 저장되어 있지 않으면 1을 출력

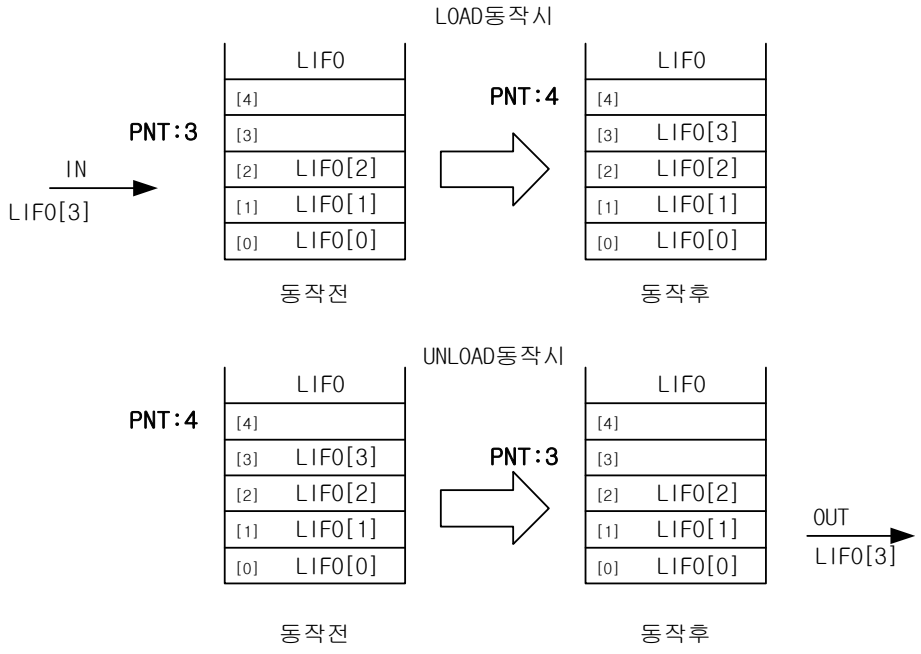
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	LIFO	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ANY: ANY 타입 중 STRING 제외, *ARRAY OF ANY: ARRAY OF ANY 타입 중 STRING 제외

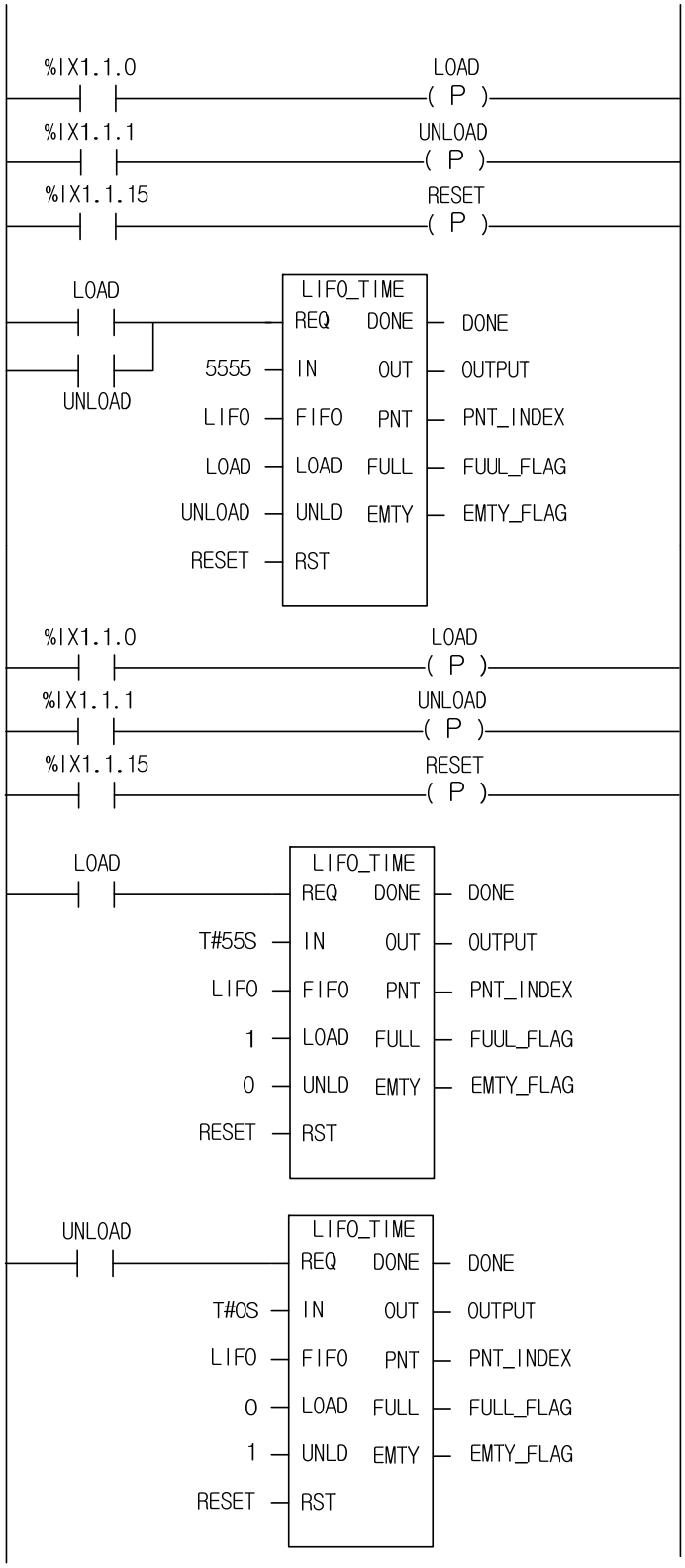
■ 기능

1. LIFO 펑션 블록은 IN 값을 LIFO 에 Load 또는 LIFO 로부터 값을 Unload 합니다.
2. 입력모드 지정과 출력모드 지정이 동시에 On 되면 입력된 값이 바로 출력됩니다.
3. LIFO 로부터 Unload 동작이 수행되면 Unload 된 값은 출력된 후 스택으로부터 삭제되고 0 으로 초기화 됩니다.
4. RST 가 입력되면 PNT 는 0 으로 초기화되고, EMTY 가 On 되며, LIFO 스택의 모든 값은 0 으로 클리어(clear)됩니다.
5. 스택의 개수는 LIFO 에 지정되는 어레이의 개수가 됩니다.
6. 정전시 또는 전원 Off 시에도 입력된 값들을 유지하기 위해서는 LIFO 어레이 변수와 LIFO 펑션 블록 Instance 를 모두 RETAIN 으로 설정하여야 합니다.
7. 리셋 동작은 REQ 요구가 없어도 동작이 가능합니다.
8. PNT 는 다음번 Load 동작시 IN 값이 입력될 위치를 나타냅니다. 또는 Load 되어 있는 전체 개수를 나타낸다고 볼 수 있습니다.
9. 입력모드일 경우 OUT 출력은 0 이 됩니다.
10. Load 및 Unload 신호가 동시에 입력되면 IN 값이 그대로 OUT 으로 출력됩니다.
11. 입력 모드일 경우 OUT 출력은 0 이 됩니다. 그러나 출력모드 동작 후 전환된 입력모드에서는 출력모드의 OUT 출력이 유지됩니다.

펑션 블록	FIFO 변수 타입	동작 설명
LIFO_BOOL	BOOL	BOOL 타입 DATA 에 대한 LIFO 동작을 수행합니다.
LIFO_BYTE	BYTE	BYTE 타입 DATA 에 대한 LIFO 동작을 수행합니다.
LIFO_WORD	WORD	WORD 타입 DATA 에 대한 LIFO 동작을 수행합니다.
LIFO_DWORD	DWORD	DWORD 타입 DATA 에 대한 LIFO 동작을 수행합니다.
LIFO_LWORD	LWORD	LWORD 타입 DATA 에 대한 LIFO 동작을 수행합니다.
LIFO_SINT	SINT	SINT 타입 DATA 에 대한 LIFO 동작을 수행합니다.
LIFO_INT	INT	INT 타입 DATA 에 대한 LIFO 동작을 수행합니다.
LIFO_DINT	DINT	DINT 타입 DATA 에 대한 LIFO 동작을 수행합니다.
LIFO_LINT	LINT	LINT 타입 DATA 에 대한 LIFO 동작을 수행합니다.
LIFO_USINT	USINT	USINT 타입 DATA 에 대한 LIFO 동작을 수행합니다.
LIFO_UINT	UINT	UINT 타입 DATA 에 대한 LIFO 동작을 수행합니다.
LIFO_UDINT	UDINT	UDINT 타입 DATA 에 대한 LIFO 동작을 수행합니다.
LIFO_ULINT	ULINT	ULINT 타입 DATA 에 대한 LIFO 동작을 수행합니다.
LIFO_REAL	REAL	REAL 타입 DATA 에 대한 LIFO 동작을 수행합니다.
LIFO_LREAL	LREAL	LREAL 타입 DATA 에 대한 LIFO 동작을 수행합니다.
LIFO_TIME	TIME	TIME 타입 DATA 에 대한 LIFO 동작을 수행합니다.
LIFO_DATE	DATE	DATE 타입 DATA 에 대한 LIFO 동작을 수행합니다.
LIFO_TOD	TOD	TOD 타입 DATA 에 대한 LIFO 동작을 수행합니다.
LIFO_DT	DT	DT 타입 DATA 에 대한 LIFO 동작을 수행합니다.



■ 프로그램 예


%IX1.1.0
LOAD
(P)
%IX1.1.1
UNLOAD
(P)
%IX1.1.15
RESET
(P)
LOAD
UNLOAD
LIFO_TIME
REQ
DONE
DONE
5555
IN
OUT
OUTPUT
LIFO
FIFO
PNT
PNT_INDEX
LOAD
LOAD
FULL
FUUL_FLAG
UNLOAD
UNLD
EMTY
EMTY_FLAG
RESET
RST
%IX1.1.0
LOAD
(P)
%IX1.1.1
UNLOAD
(P)
%IX1.1.15
RESET
(P)
LOAD
LIFO_TIME
REQ
DONE
DONE
T#55S
IN
OUT
OUTPUT
LIFO
FIFO
PNT
PNT_INDEX
1
LOAD
FULL
FUUL_FLAG
0
UNLD
EMTY
EMTY_FLAG
RESET
RST
UNLOAD
LIFO_TIME
REQ
DONE
DONE
T#0S
IN
OUT
OUTPUT
LIFO
FIFO
PNT
PNT_INDEX
0
LOAD
FULL
FULL_FLAG
1
UNLD
EMTY
EMTY_FLAG
RESET
RST

LIFO_TIME 펑션 블록은 위의 그림과 같이 2 가지 방법으로 사용이 가능합니다. 위에서 예를 든 2 가지 방법은 동일한 동작을 수행합니다. 위의 블록은 하나의 펑션 블록을 사용하여 입력과 출력을 동시에 수행할 수 있도록 한 프로그램이고, 아래의 블록은 입력을 위한 펑션 블록과 출력을 위한 펑션 블록을 따로 작성하여 입출력 동작을 다른 위치에서 동작하도록 작성한 프로그램입니다. 단 이때 주의할 점은 인스턴스 이름이 같도록 설정해야 합니다.

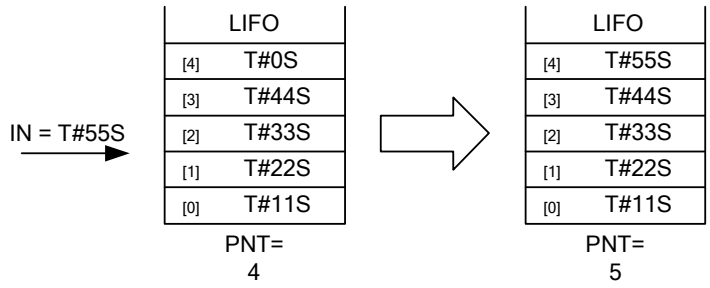
(1) 입력 조건(%IX1.1.0, %IX1.1.1, %IX1.1.15)이 성립되면 LIFO_TM 이 실행됩니다.

(2) 입력접점 %IX1.1.0 이 On 되면 Load 동작을 수행합니다. T#55S 가 LIFO 스택에 입력되고 PNT_INDEX 가 1 증가합니다.

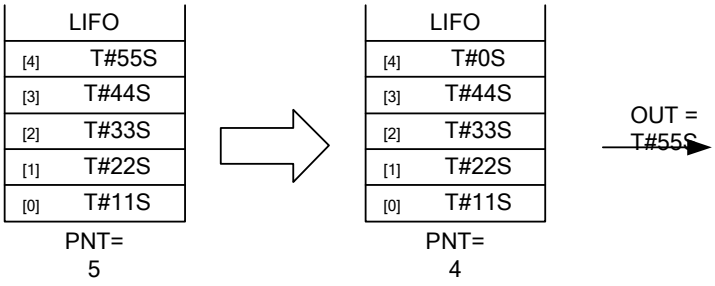
(3) 입력접점 %IX1.1.1 이 On 되면 Unload 동작을 수행합니다. LIFO 스택으로부터 T#55S 가 출력되고 PNT_INDEX 가 1 감소합니다.

(4) 입력접점 %IX1.1.15 가 On 되면 Reset 동작을 수행합니다. LIFO 스택의 모든 값이 T#0S 으로 클리어(Clear)되고, PNT_INDEX 가 0 으로 초기화되며, EMTY_FLAG 가 On 됩니다

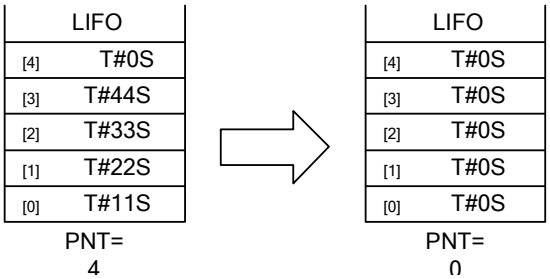
LOAD 동작 (%IX1.1.0 On 동작)



UNLOAD 동작 (%IX1.1.1 On 동작)



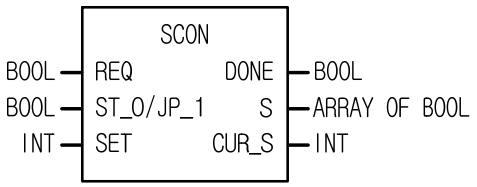
RESET 동작 (%IX1.1.15 On 동작)



2. ST

```
INST_LIFO_TIME(REQ:=LOAD OR UNLOAD, IN:=T#55S, LIFO:=LIFO, LOAD:=LOAD, UNLD:=UNLOAD, RST:=RST, DONE=>DONE,
OUT=>OUTPUT, PNT=>PNT_INDEX, FULL=>FULL_FLAG, EMTY=>EMTY_FLAG);
```

SCON	적용 기종	발생플래그
스텝 컨트롤러(순차스텝 및 스텝점프)	XGI, XGR, XEC	_ERR, _LER

펑션 블록	설 명
	**입력** REQ : 1일 때 펑션 블록 실행 ST_0/JP_1 : 0이면 SET 동작을 지정하고, 1이면 OUT 동작을 지정 SET : 스텝의 번호(0~99) **출력** DONE : 펑션 블록 실행이 에러 없이 종료된 경우 On 되며, 에러가 발생하거나 펑션 블록 실행 요구가 없으면 Off S : Set 된 bit array CUR_S : 현재 스텝 번호를 출력

■ 기능

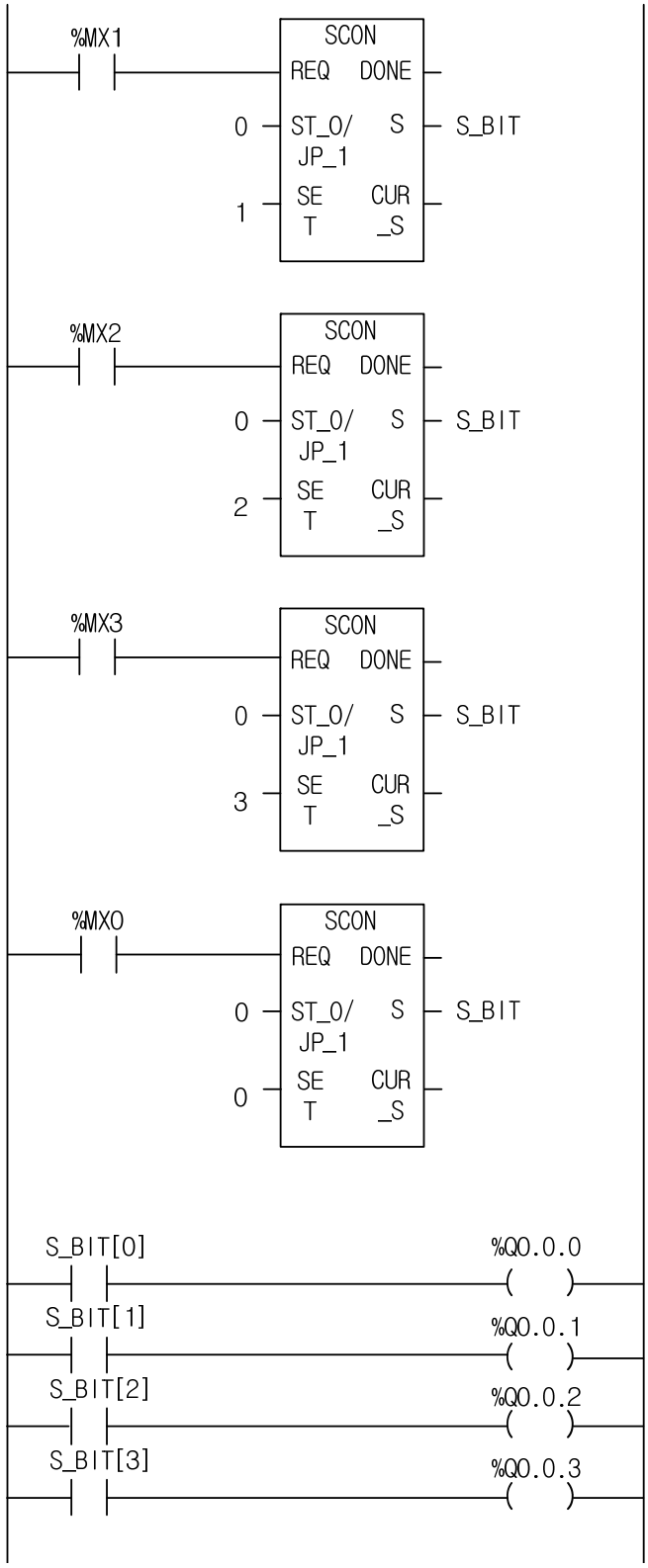
1. 순차작업 조의 설정
   펑션 블록의 인스턴스 이름이 하나의 순차작업 조의 이름이 됩니다.
   (펑션 블록 선언 예: S00, G01, 제조1, 스텝 접점 예: S00.S[1], G01.S[1], 제조1.S[1])
2. SET 동작일 경우(ST_0/JP_1 = 0)
   동일 조 내에서 바로 이전의 스텝 번호가 On 되었을 때 현재 스텝 번호가 On 됩니다.
   현재 스텝 번호가 On 되면 자기 유지되어 입력 접점이 Off 되어도 On 의 상태를 유지합니다.
   입력 조건 접점이 동시에 On 되어도 한 조 내에서는 한 스텝 번호만이 On 됩니다.
   Sxx.S[0]가 On 되면 모든 SET 출력이 Clear 됩니다.
3. JUMP 동작일 경우(ST_0/JP_1 = 1)
   동일 조 내에서 입력조건 접점이 다수가 On 하여도 한 개의 스텝 번호만 On 합니다.
   입력 조건이 동시에 On 하면 나중에 프로그램 된 것이 우선으로 출력 됩니다.
   현재 스텝번호가 On 되면 자기 유지되어 입력 조건이 Off 되어도 On 되어진 상태를 유지 합니다.
   Sxx.S[0]이 On 되면 초기 스텝으로 복귀합니다.

■ 플래그

플래그	설명
_ERR	스텝지정(SET)이 범위(0~99)를 벗어나면 에러가 발생합니다. 에러 발생시에는 DONE 이 Off 되고, 스텝출력은 이전 스텝을 유지합니다.

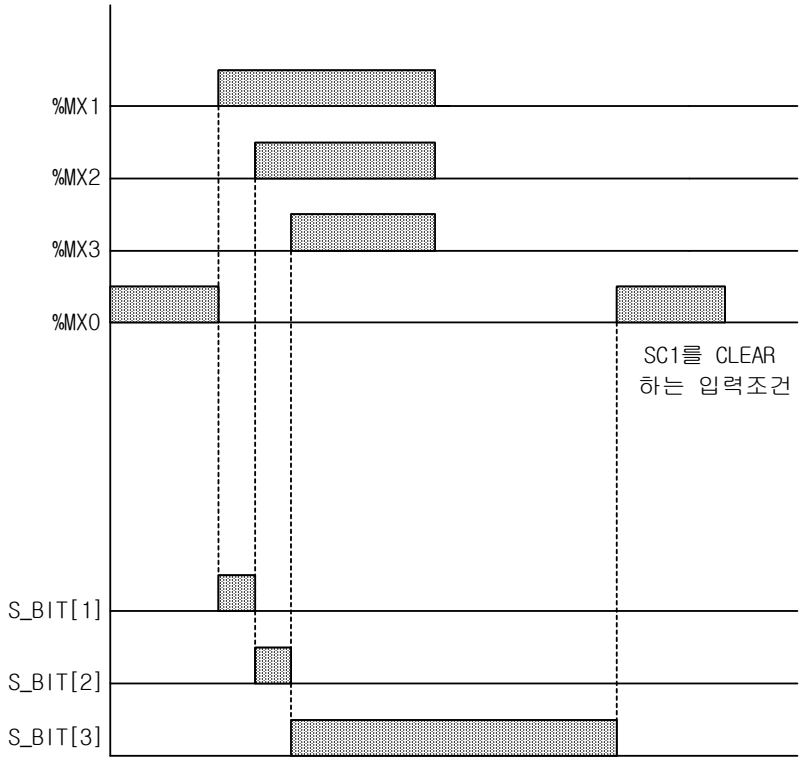
■ SET 동작일 경우(ST_0/JP_1 = 0)의 프로그램 설명 프로그램 SC1 조를 이용한 순차제어 프로그램

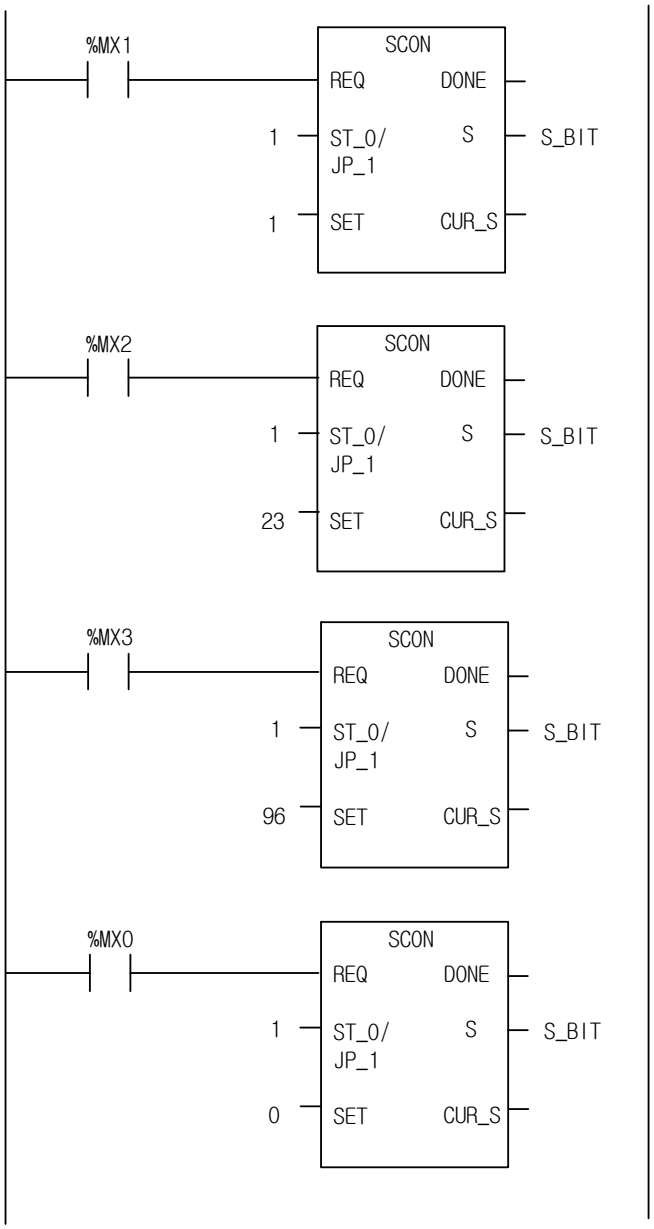
1. LD



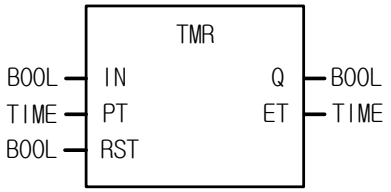
2. ST

```
INST_SCON(REQ:=%MX1, STO_JP1:=0, SET:=1, S=>S_BIT);
INST_SCON1(REQ:=%MX2, STO_JP1:=0, SET:=2, S=>S_BIT);
INST_SCON2(REQ:=%MX3, STO_JP1:=0, SET:=3, S=>S_BIT);
INST_SCON3(REQ:=%MX4, STO_JP1:=0, SET:=0, S=>S_BIT);

%QX0.0.0 := S_BIT[0];
%QX0.0.1 := S_BIT[1];
%QX0.0.2 := S_BIT[2];
%QX0.0.3 := S_BIT[3];
```





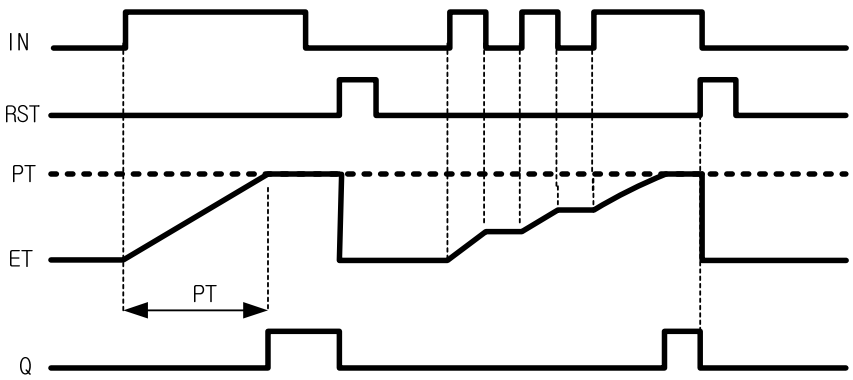
NO	%MX1	%MX2	%MX3	%MX4	S_0 [1]	S_0 [23]	S_0 [98]	S_0 [0]
1	On	Off	Off	Off	○			
2	On	On	Off	Off		○		
3	On	On	On	Off			○	
4	On	On	On	On				○

TMR	적용 기종	발생플래그
적산 타이머	XGI, XGR, XEC	-
펑션 블록	설 명	
	**입력** IN : 타이머 기동 조건 PT : 설정 시간(Preset Time) RST : 리셋 입력(Reset) **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)	

■ 기능

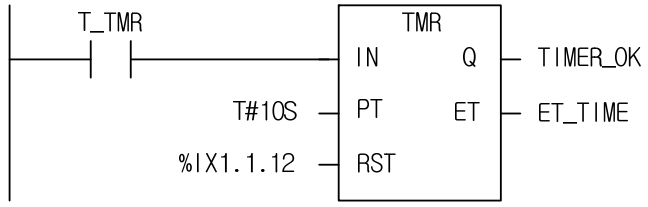
1. TMR 펑션 블록은 IN 이 1 이 된 후 경과 시간이 ET 로 출력됩니다.
2. 경과 시간 ET 가 설정시간에 도달하기 전에 IN 이 0 이 되어도 현재의 경과 시간을 유지하다가 IN 이 다시 1 이 되면 경과 시간을 다시 증가시킵니다.
3. 경과 시간이 설정 시간에 도달하면 Q 가 1 이 됩니다.
4. Reset 입력 조건이 성립되면 Q 는 0 이 되며 경과 시간도 0 이 됩니다.

■ 타임 차트



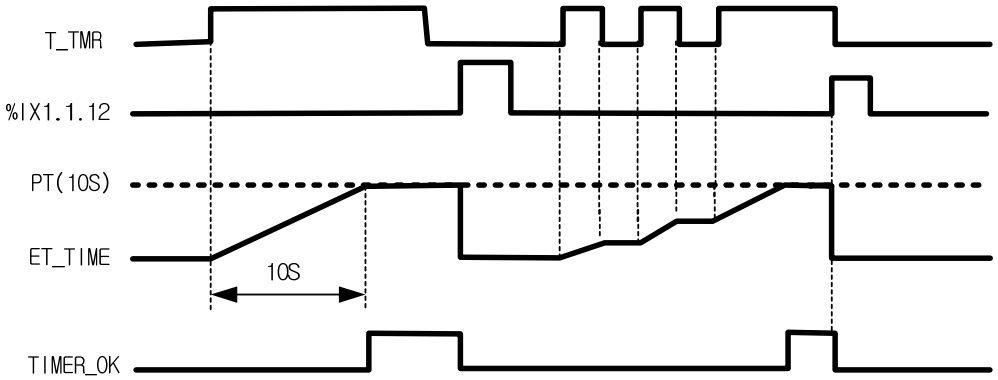
■ 프로그램 예

1. LD



2. ST

```
INST_TMR(IN:=T_TMR, PT:=T#10S, RST:=%IX1.1.12, Q=>TIMER_OK, ET=>ET_TIME);
```



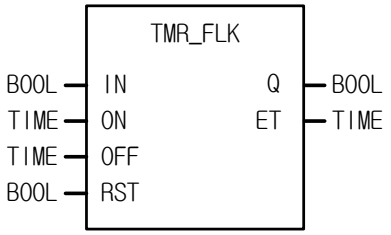
(1) 입력변수 T_TMR 이 1 이 된 후 10 초가 경과하면 출력 변수 TIMER_OK 가 1 이 됩니다.

(2) 입력변수 T_TMR 이 1 이 된 후 경과 시간이 출력변수 ET_TIME 으로 출력됩니다.

(3) 경과 시간 ET_TIME 이 설정시간 10 초에 도달하기 전에 T_TMR 이 0 이 되더라도 현재의 경과 시간을 유지합니다.

(4) 입력 변수 T_TMR 이 다시 1 이 되면 정지 이전의 경과 시간부터 다시 시작합니다.

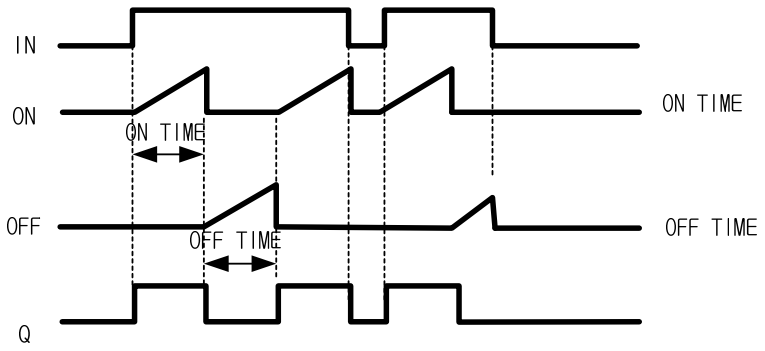
(5) 입력 접점 %IX1.1.12 가 1 이 되면 경과 시간 ET_TIME 및 출력 변수 TIMER_OK 모두 0 으로 클리어(Clear)됩니다.

TMR_FLK	적용 기종	발생플래그
점멸기능 타이머	XGI, XGR, XEC	-
펑션 블록	설 명	
	**입력** IN : 타이머 기동 조건 ON : On 타이머 설정 시간 OFF : Off 타이머 설정 시간 RST : 리셋 입력 **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)	

■ 기능

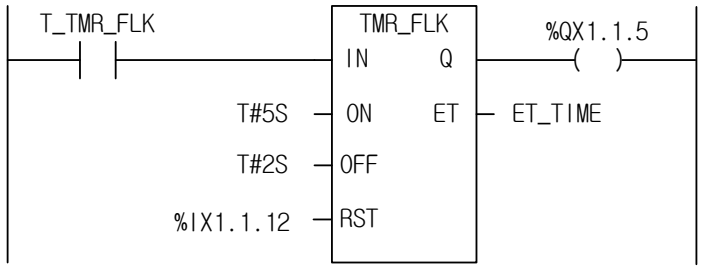
1. TMR_FLK 펑션 블록은 IN이 1이 되는 순간 Q는 1이 되고, ON에서 지정된 시간만큼 Q는 1을 유지합니다.
2. ON에서 지정된 시간이 경과하면 OFF에서 지정된 시간만큼 Q는 0이 됩니다.
3. IN이 0이 되면 On 또는 Off 동작을 수행을 중지하고, IN이 0인 동안 중지된 시간을 유지하다가 IN이 다시 1이 되면 정지된 시간부터 다시 타이머가 동작합니다.
4. IN이 0인 동안 출력 Q 는 0이 됩니다.
5. ON이 0이면 출력 Q는 항상 0이 됩니다.

■ 타임 차트



■ 프로그램 예

1. LD



2. ST

```
INST_TMR_FLK(IN:=T_TMR_FLK, ON:=T#5S, OFF:=T#2S, RST:=%IX1.1.12, Q=>%QX1.1.5, ET=>ET_TIME);
```

(1) 입력변수 T_TMR_FLK 가 0 에서 1 이 되면, TMR_FLK 펑션 블록이 동작을 시작합니다.

(2) 입력변수 T_TMR_FLK 가 1 이 된 후 ON 에서 지정된 5 초만큼 출력 접점 %QX1.1.5 는 1 이 됩니다.

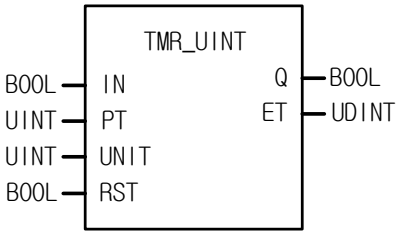
(3) 입력변수 T_TMK_FLK 가 1 이 된 후 ON 에서 지정된 시간이 경과하면 OFF 에서 지정된 2 초만큼 출력 접점 %QX1.1.5 는 0 이 됩니다.

(4) 입력변수 T_TMR_FLK 가 1 인 동안 출력 Q 가 1 인 동안의 경과 시간과 Q 가 0 인 동안의 경과 시간이 번갈아 ET_TIME 으로 출력됩니다.

(5) 입력변수 T_TMR_FLK 가 0 이 되면 동작 중인 시간을 유지하고 출력접점 %QX1.1.5 는 0 이 되며, T_TMR_FLK 가 다시 1 이 되면 정지되었던 시간부터 다시 동작합니다.

(6) 입력 접점 %IX1.1.12 가 1 이 되면 경과 시간 ET_TIME 및 출력접점 %QX1.1.5 모두 0 으로 클리어(Clear)됩니다.

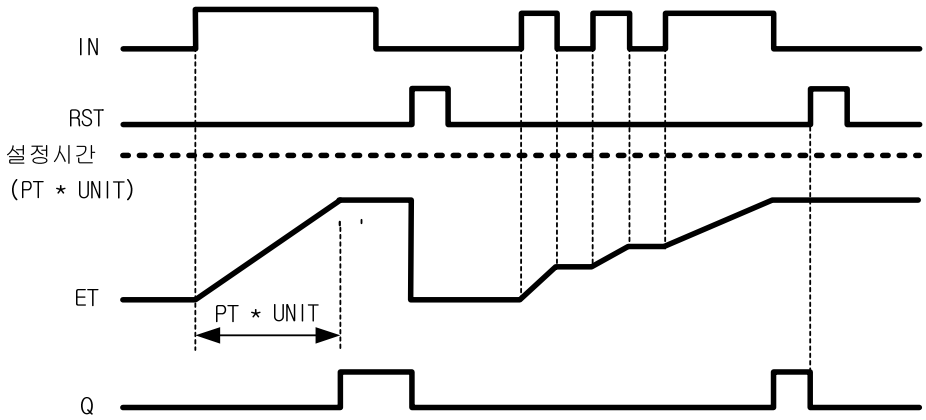
TMR_UINT	적용 기종	발생플래그
정수 설정 적산 타이머	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** IN : 타이머 기동 조건 PT : 설정 시간(Preset Time) UNIT : 설정 시간의 시간 단위(Unit) RST : 리셋 입력 **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)

■ 기능

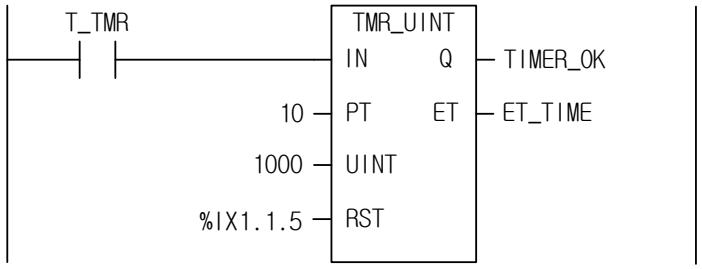
1. TMR_UINT 펑션 블록은 IN이 1이 된 후 경과 시간이 ET로 출력됩니다.
2. 경과 시간 ET가 설정 시간에 도달하기 전에 IN이 0이 되어도 현재의 경과 시간을 유지하다가 IN이 다시 1이 되면 경과 시간이 다시 증가됩니다.
3. 경과 시간이 설정 시간에 도달하면 Q가 1이 됩니다.
4. Reset 입력 조건이 성립되면 Q는 0이 되고 경과 시간도 0이 됩니다.
5. 설정시간은 PT * UNIT[ms]입니다.

■ 타임 차트



■ 프로그램 예

1. LD



2. ST

```
INST_TMR_UINT(IN:=T_TMR, PT:=10, UNIT:=1000, RST:=%IX1.1.5, Q=>TIMER_OK, ET=>ET_TIME);
```

(1) 설정 시간은 PT * UNIT[ms] = 10 * 1000[ms] = 10[s]가 됩니다.

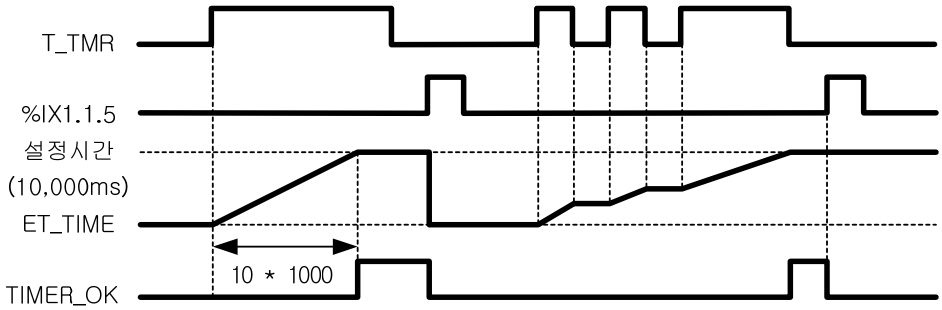
(2) 입력변수 T_TMR이 1이 된 후 10초가 경과하면 출력 변수 TIMER_OK가 1이 됩니다.

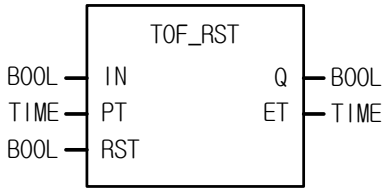
(3) 입력 변수 T_TMR이 1이 된 후 경과 시간이 출력변수 ET_TIME으로 출력됩니다.

(4) 경과 시간 ET_TIME이 설정시간 10초에 도달하기 전에 T_TMR이 0이 되더라도 현재의 경과 시간을 유지합니다.

(5) 입력 변수 T_TMR이 다시 1이 되면 멈추어 졌던 이전의 경과 시간부터 다시 시작합니다.

(6) 입력 접점 %IX1.1.5가 1이 되면 경과 시간 ET_TIME 및 출력 변수 TIMER_OK 모두 0으로 클리어(Clear)됩니다.

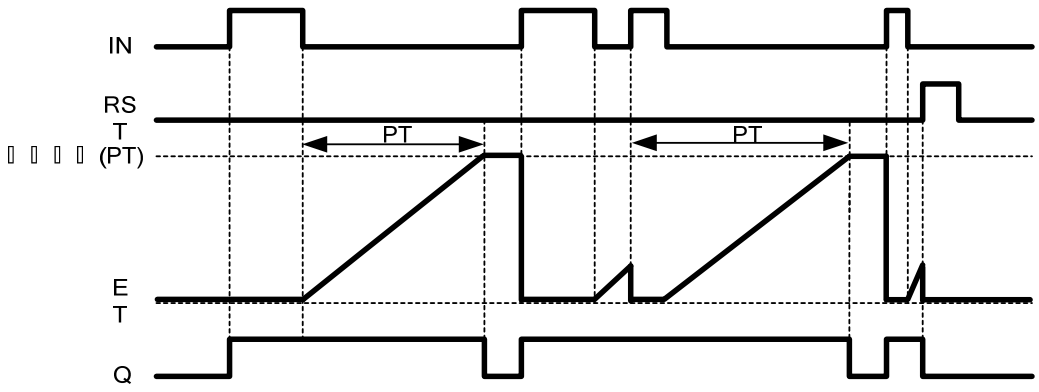


TOF_RST	적용 기종	발생플래그
동작 중 출력 Off 가 가능한 딜레이 타이머	XGI, XGR, XEC	-
펑션 블록	설 명	
	**입력** IN : 타이머 기동 조건 PT : 설정 시간(Preset Time) RST : 리셋 입력(Reset) **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)	

■ 기능

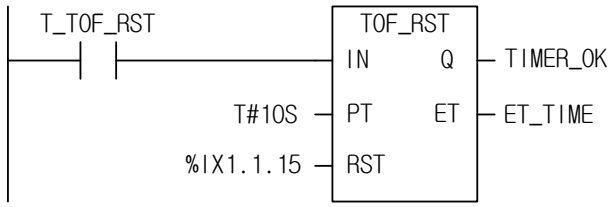
1. TOF_RST 펑션 블록은 기동 조건 IN 이 1 이 되는 순간 Q 는 1 이 되고, IN 이 0 이 된 후부터 PT 에 의하여 지정된 설정 시간이 경과한 후 Q 가 0 이 됩니다.
2. IN 이 0 이 된 후 경과 시간이 ET 로 출력됩니다.
3. 만일 경과시간 ET 가 설정 시간에 도달하기 전에 IN 이 1 이 되면, 경과 시간은 다시 0 으로 됩니다.
4. Reset 입력 조건이 성립하면 타이머 출력 Q 는 0 이 되고 경과 시간도 0 이 됩니다.

■ 타임 차트



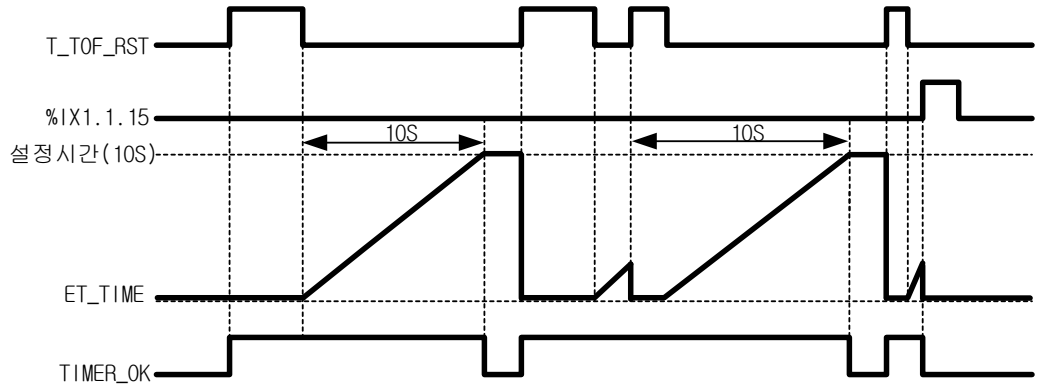
■ 프로그램 예

1. LD



2. ST

```
INST_TOF_RST(IN:=T_TOF_RST, PT:=T#10S, RST:=%IX1.1.15, Q=>TIMER_OK, ET=>ET_TIME);
```



(1) 입력변수로 설정된 T_TOF_RST 가 1 이 되면, 출력변수 TIMER_OK 에 1 이 출력되고 T_TOF_RST 가 0 이 된 후 10s 후에 TIMER_OK 가 0 이 됩니다.

(2) T_TOF_RST 가 0 이 된 후 10 초 이내에 다시 1 이 되면 타이머는 다시 초기 상태가 됩니다.

(3) 타이머의 시간 측정값은 ET_TIME 로 출력됩니다.

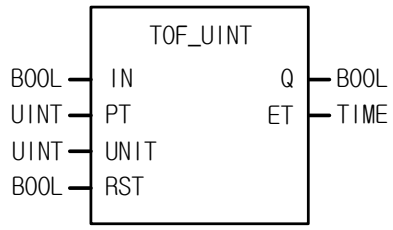
(4) 입력접점 %IX1.1.15 가 1 이 되면 TIMER_OK 와 ET_TIME 모두 0 으로 클리어(Clear)됩니다.

☆ 참고

TOF_RST 펑션 블록은 접점이 On 된 후 Off 되어도 해당 동작이 마무리 될 때까지 펑션 블록은 동작합니다.

배열인덱스를 이용한 변수를 사용했을 경우 배열인덱스 에러는 접점이 On 이 되었을 때만 발생합니다.

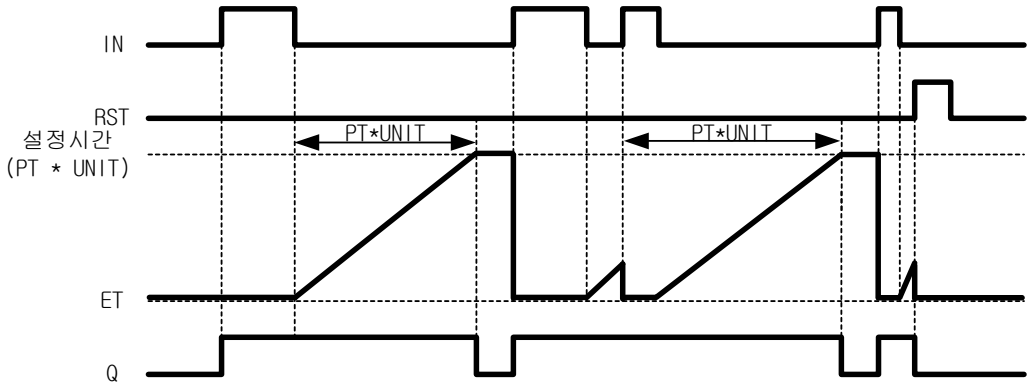
때문에 TOF_RST 펑션 블록에서는 펑션 블록이 동작하더라도 접점이 Off 되어 있다면 배열인덱스 에러가 발생하지 않습니다.

TOF_UINT	적용 기종	발생플래그
정수 설정 Off 타이머	XGI, XGR, XEC	-
펑션 블록	설 명	
	**입력** IN : 타이머 기동 조건 PT : 설정 시간(Preset Time) UNIT : 설정 시간의 시간 단위(Unit) RST : 리셋 입력 **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)	

■ 기능

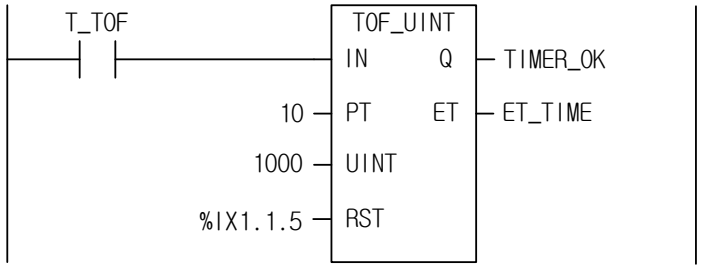
1. TOF_UINT 펑션 블록은 기동 조건 IN이 1이 되는 순간 Q는 1이 되고, IN이 0이 된 후부터 PT에 의하여 지정된 설정시간이 경과한 후 Q가 0이 됩니다.
2. IN이 0이 된 후 경과 시간이 ET로 출력됩니다.
3. 만일 경과시간 ET가 설정시간에 도달하기 전에 IN이 1이 되면, 경과 시간은 다시 0으로 됩니다.
4. Reset 입력 조건이 성립하면 타이머 출력 Q는 0이 되고 경과 시간도 0이 됩니다.
5. 설정시간은 PT * UNIT[ms]입니다.

■ 타임 차트



■ 프로그램 예

1. LD



2. ST

```
INST_TOF_UINT(IN:=T_TOF, PT:=10, UNIT:=1000, RST:=%IX1.1.5, Q=>TIMER_OK, ET=>ET_TIME);
```

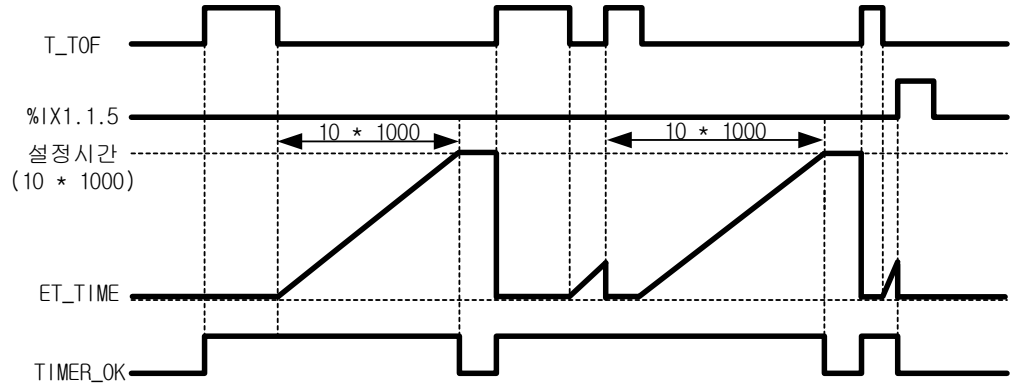
(1) 설정 시간은 PT * UNIT[ms] = 10 * 1000[ms] = 10[s]가 됩니다.

(2) 입력변수로 설정된 T_TOF 가 1 이 되면, 출력변수 TIMER_OK 에 1 이 출력되고 T_TOF 가 0 이 된 후 10 초 후에 TIMER_OK 가 0 이 됩니다.

(3) T_TOF 가 0 이 된 후 10 초 이내에 다시 1 이 되면 타이머는 다시 초기 상태가 됩니다.

(4) 타이머의 시간 측정값은 ET_TIME 로 출력됩니다.

(5) 입력접점 %IX1.1.5 가 1 이 되면 TIMER_OK 와 ET_TIME 모두 0 으로 클리어(Clear)됩니다.

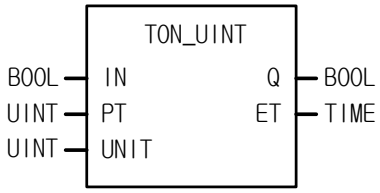


☆ 참고

TOF_UINT 펑션 블록은 접점이 On 된 후 Off 되어도 해당 동작이 마무리 될 때가지 펑션 블록은 동작합니다.

배열인덱스를 이용한 변수를 사용했을 경우 배열인덱스 에러는 접점이 On 이 되었을 때만 발생합니다.

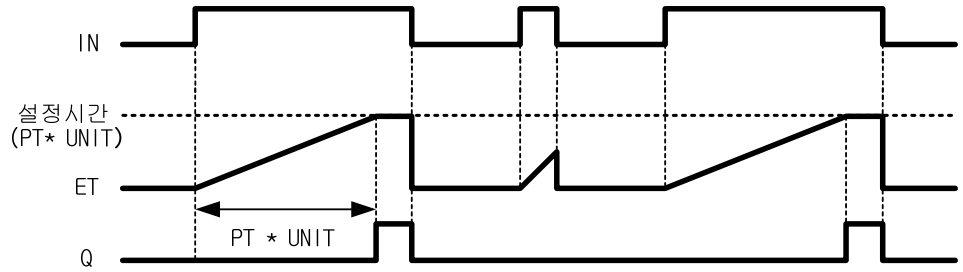
때문에 TOF_UINT 펑션 블록에서는 펑션 블록이 동작하더라도 접점이 Off 되어 있다면 배열인덱스 에러가 발생하지 않습니다.

TON_UINT	적용 기종	발생플래그
정수 설정 On 타이머	XGI, XGR, XEC	-
펑션 블록	설 명	
	**입력** IN : 타이머 기동 조건 PT : 설정 시간(Preset Time) UNIT : 설정 시간의 시간 단위(Unit) **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)	

■ 기능

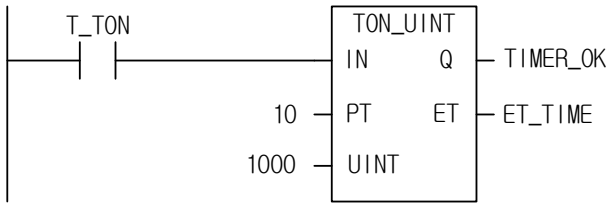
1. TON_UINT 펑션 블록은 IN이 1이 된 후 경과 시간이 ET로 출력됩니다.
2. 만일 경과시간 ET가 설정 시간에 도달하기 전에 IN이 0이 되면, 경과 시간 ET는 0으로 됩니다.
3. Q가 1이 된 후 IN이 0이 되면, Q는 0이 됩니다.
4. 설정 시간은 PT * UNIT[ms]입니다.

■ 타임 차트



■ 프로그램 예

1. LD



2. ST

```
INST_TON_UINT(IN:=T_TON, PT:=10, UNIT:=1000, Q=>TIMER_OK, ET=>ET_TIME);
```

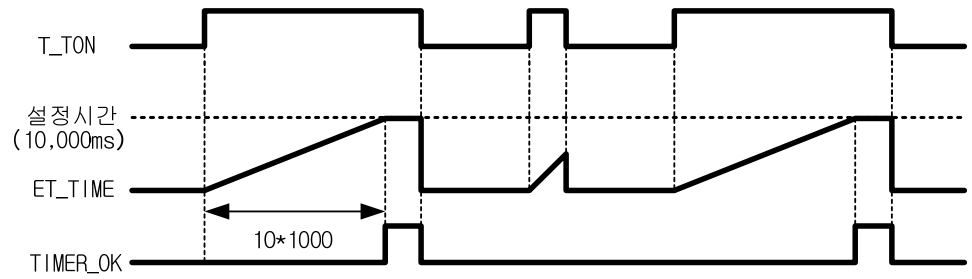
(1) 설정 시간은 PT * UNIT[ms] = 10 * 1000[ms] = 10[s]가 됩니다.

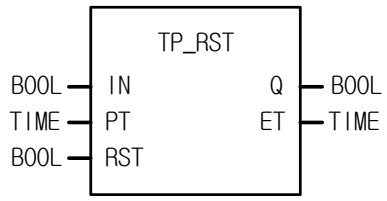
(2) 입력변수 T_TON 이 On 된 후, 10 초가 경과한 후에 출력 변수 TIMER_OK 가 1 이 됩니다.

(3) 입력변수 T_TON 이 On 된 후, 경과 시간이 출력 변수 ET_TIME 로 출력됩니다.

(4) 만일 경과 시간 ET_TIME 이 설정 시간 10 초에 도달하기 전에 T_TON 이 0 이 되면, 경과 시간 ET_TIME 은 0 으로 됩니다.

(5) TIMER_OK 가 1 이 된 후 T_TON 이 0 이 되면, TIMER_OK 는 0 이 되고 경과 시간 ET_TIME 도 0 이 됩니다.

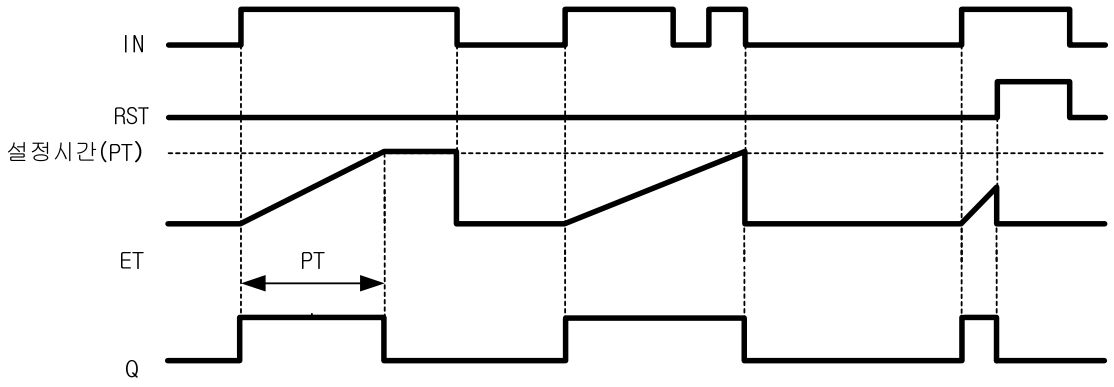


TP_RST	적용 기종	발생플래그
접점 출력 Off 가 가능한 펄스 타이머	XGI, XGR, XEC	-
펑션 블록	설 명	
	**입력** IN : 타이머 기동 조건 PT : 설정 시간(Preset Time) RST : 리셋 입력(Reset) **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)	

■ 기능

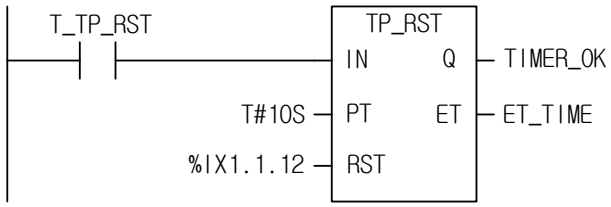
1. TP_RST 펑션 블록은 IN 이 1 이 되는 순간 Q 는 1 이 되고, 경과 시간이 설정 시간에 도달하면 타이머 출력 Q 는 0 이 됩니다.
2. 경과 시간 ET 는 IN 이 1 이 되었을 때부터 증가하며 PT 에 이르면 값을 유지하다가 IN 이 0 이 될 때 0 으로 클리어 (clear) 됩니다.
3. 타이머 출력 Q 가 1 인 동안(펄스 출력 중)에는 타이머 기동조건 IN 이 1, 0 변화를 하여도 무시합니다.
4. Reset 입력 조건이 성립하면 펄스 출력 중에도 타이머 출력 Q 는 0 이 되고 경과 시간도 0 이 됩니다.

■ 타임 차트



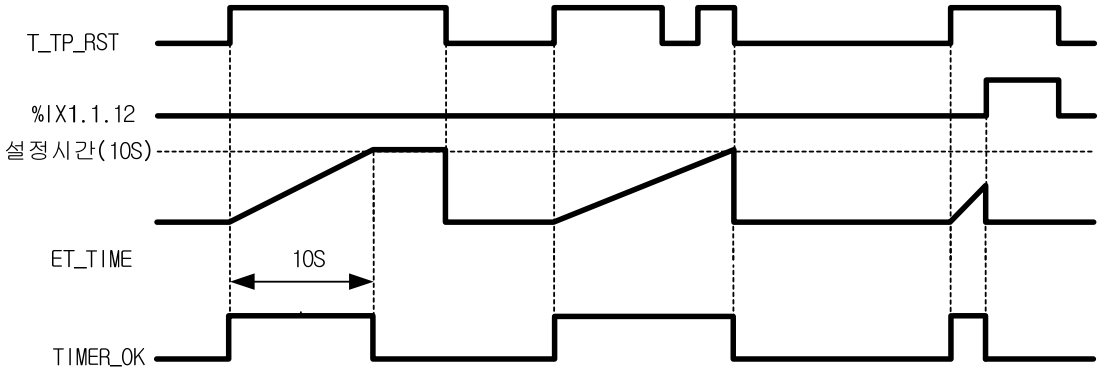
■ 프로그램 예

1. LD



2. ST

```
INST_TP_RST(IN:=T_TP_RST, PT:=T#10S, RST:=%IX1.1.12, Q=>TIMER_OK, ET=>ET_TIME);
```



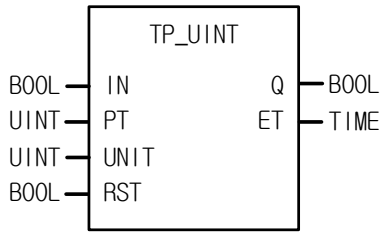
(1) 입력변수 T_TP_RST 가 1 이 되면 출력변수 TIMER_OK 가 1 이 되고 10 초가 경과하면 출력변수 TIMER_OK 는 0 이 됩니다. 일단 타이머가 가동된 후에는 10 초 동안 T_TP_RST 신호는 무시됩니다.

(2) ET_TIME 값은 증가 후 10S 에서 멈춥니다. 그리고 T_TP_RST 가 0 이 될 때 0 으로 됩니다.

(3) 입력 접점 %IX1.1.12 가 1 이 되면 TIIMER_OK 와 ET_TIME 모두 0 으로 클리어(Clear)됩니다.

☆ 참고

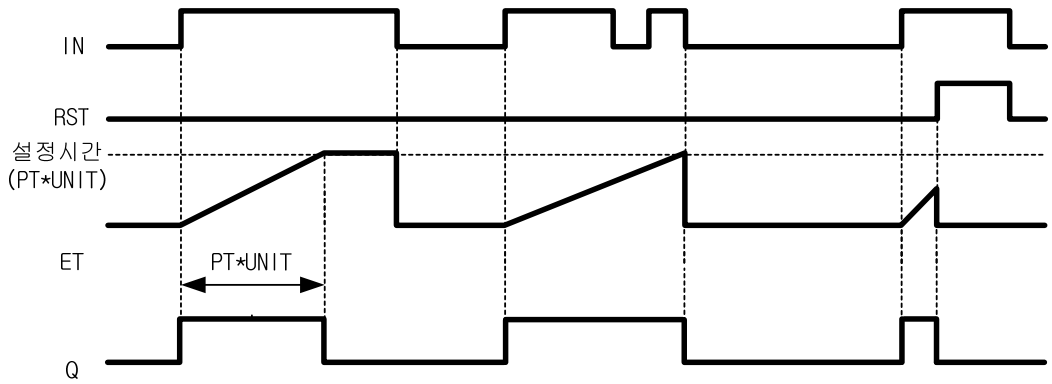
TP_RST 펑션 블록은 접점이 On 된 후 Off 되어도 해당 동작이 마무리 될 때가지 펑션 블록은 동작합니다.
배열인덱스를 이용한 변수를 사용했을 경우 배열인덱스 에러는 접점이 On 이 되었을 때만 발생합니다.
때문에 TP_RST 펑션 블록에서는 펑션 블록이 동작하더라도 접점이 Off 되어 있다면 배열인덱스 에러가 발생하지 않습니다.

TP_UINT	적용 기종	발생플래그
정수 설정 펄스 타이머	XGI, XGR, XEC	-
펑션 블록	설 명	
	**입력** IN : 타이머 기동 조건 PT : 설정 시간(Preset Time) UNIT : 설정 시간의 시간 단위(Unit) RST : 리셋 입력 **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)	

■ 기능

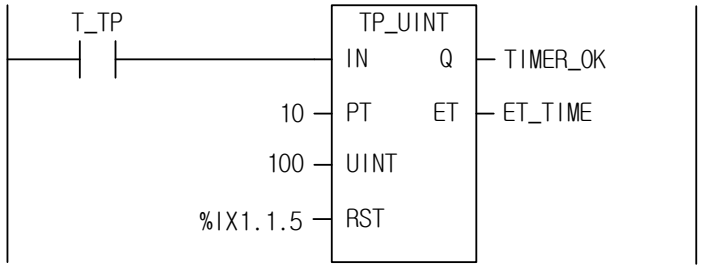
1. TP_UINT 펑션 블록은 IN이 1이 되는 순간 Q는 1이 되고, 경과 시간이 설정 시간에 도달하면 타이머 출력 Q는 0이 됩니다.
2. 경과 시간 ET는 IN이 1이 되었을 때부터 증가하며 PT에 이르면 값을 유지하다가 IN이 0이 될 때 0으로 클리어(Clear) 됩니다.
3. 타이머 출력 Q가 1인 동안(펄스 출력 중)에는 타이머 기동 조건 IN이 1,0 변화를 하여도 무시합니다.
4. Reset 입력 조건이 성립하면 펄스 출력 중에도 타이머 출력 Q는 0이 되고 경과 시간도 0이 됩니다.
5. 설정시간은 PT * UNIT[ms]입니다.

■ 타임 차트



■ 프로그램 예

1. LD



2. ST

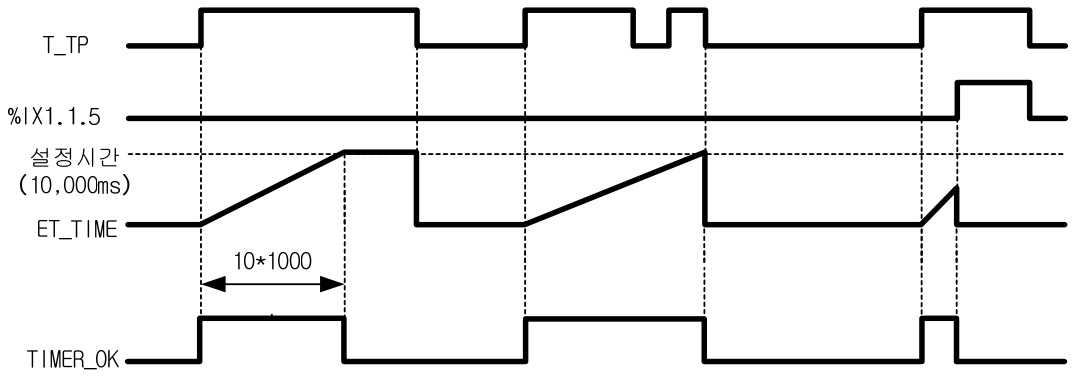
```
INST_TP_UINT(IN:=T_TP, PT:=10, UNIT:=100, RST:=%IX1.1.5, Q=>TIMER_OK, ET=>ET_TIME);
```

(1) 설정 시간은 PT * UNIT[ms] = 10 * 100[ms] = 1[s]가 됩니다.

(2) 입력변수 T_TP 가 1 이 되면 출력변수 TIMER_OK 가 1 이 되고 1 초가 경과하면 출력변수 TIMER_OK 는 0 이 됩니다. 일단 타이머가 가동된 후에는 1 초 동안 T_TP 신호는 무시됩니다.

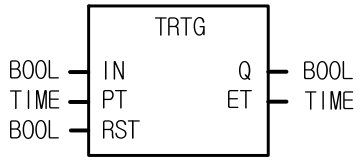
(3) ET_TIME 값은 증가 후 1,000 에서 멈춥니다. 그리고 T_TP 가 0 이 될 때 0 으로 됩니다.

(4) 입력 접점 %IX1.1.5 가 1 이 되면 TIMER_OK 와 ET_TIME 모두 0 으로 클리어(Clear) 됩니다.



☆ 참고

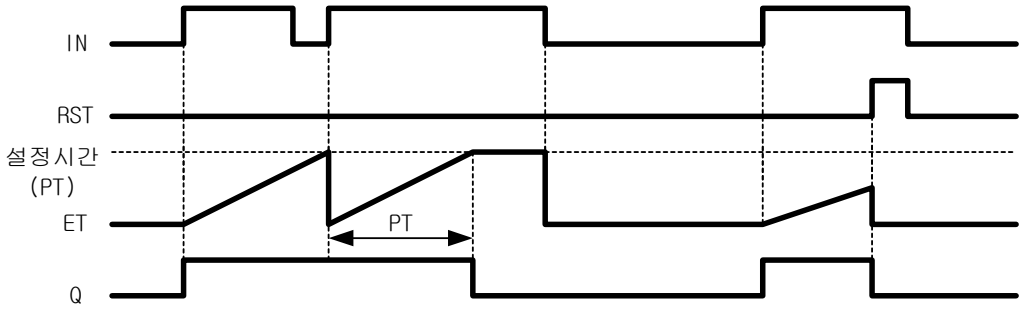
TP_UINT 펑션 블록은 접점이 On 된 후 Off 되어도 해당 동작이 마무리 될 때가지 펑션 블록은 동작합니다.
배열인덱스를 이용한 변수를 사용했을 경우 배열인덱스 에러는 접점이 On 이 되었을 때만 발생합니다.
때문에 TP_UINT 펑션 블록에서는 펑션 블록이 동작하더라도 접점이 Off 되어 있다면 배열인덱스 에러가 발생하지 않습니다.

TRTG	적용 기종	발생플래그
리트리거블 타이머	XGI, XGR, XEC	-
펑션 블록	설 명	
	**입력** IN : 타이머 기동 조건 PT : 설정 시간(Preset Time) RST : 리셋 입력(Reset) **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)	

■ 기능

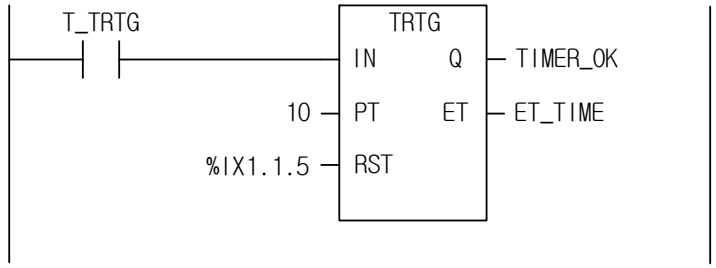
1. TRTG 펑션 블록은 기동 조건 IN이 1이 되는 순간 Q는 1이 되고, 경과 시간이 설정 시간에 도달하면 타이머 출력 Q는 0이 됩니다.
2. 타이머 경과 시간이 설정 시간이 되기 전에 IN이 또 다시 0에서 1로 되면 경과 시간은 0으로 재설정 되고 다시 증가하여 설정 시간 PT에 도달하면 Q는 0이 됩니다.
3. Reset 입력 조건이 성립하면 타이머 출력 Q는 0이 되고 경과 시간도 0이 됩니다.

■ 타임 차트



■ 프로그램 예

1. LD



2. ST

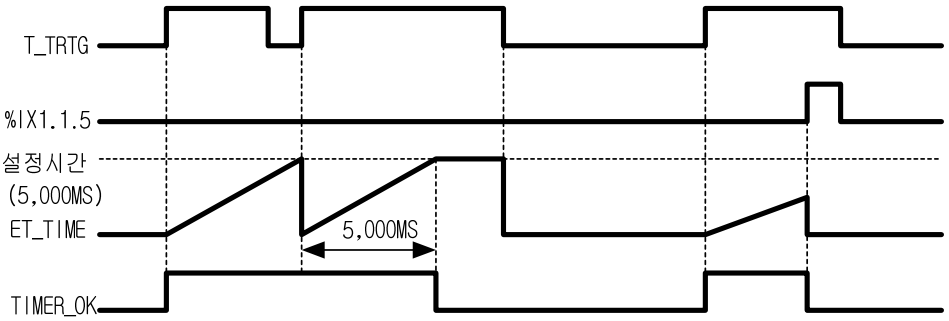
```
INST_TRTG(IN:=T_TRTG, PT:=10, RST:=%IX1.1.5, Q=>TIMER_OK, ET=>ET_TIME);
```

(1) 입력변수 T_TRTG 가 0 에서 1 이 된 후 10 초 동안 TIMER_OK 는 1 이 됩니다. 타이머가 가동된 후 T_TRTG 가 다시 0 에서 1 이 되면 ET_TIME 은 0 부터 다시 시작됩니다.

(2) T_TRTG 가 1 에서 0 이 되어도 10 초 동안 TIMER_OK 는 1 이 됩니다.

(3) ET_TIME 은 증가 후 T#10S 에서 멈춥니다. 그리고 T_TRTG 가 0 이 될 때 0 으로 됩니다.

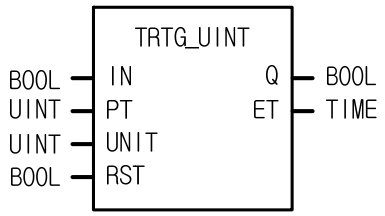
(4) 입력접점 %IX1.1.5 가 1 이 되면 TIMER_OK 와 ET_TIME 모두 0 으로 클리어(Clear) 됩니다.



☆ 참고

TRTG 펑션 블록은 접점이 On 된 후 Off 되어도 해당 동작이 마무리 될 때가지 펑션 블록은 동작합니다.
배열인덱스를 이용한 변수를 사용했을 경우 배열인덱스 에러는 접점이 On 이 되었을 때만 발생합니다.
때문에 TRTG 펑션 블록에서는 펑션 블록이 동작하더라도 접점이 Off 되어 있다면 배열인덱스 에러가 발생하지 않습니다.

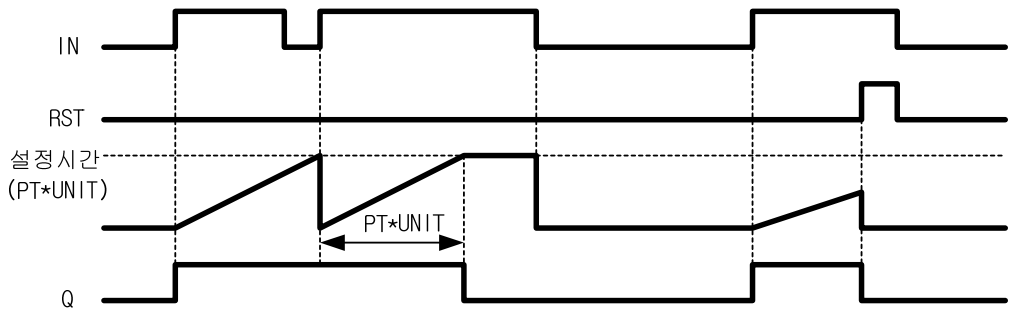
TRTG_UINT	적용 기종	발생플래그
정수 설정 리트리거블 타이머	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** IN : 타이머 기동 조건 PT : 설정 시간(Preset Time) UNIT : 설정 시간의 시간 단위(Unit) RST : 리셋 입력 **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)

■ 기능

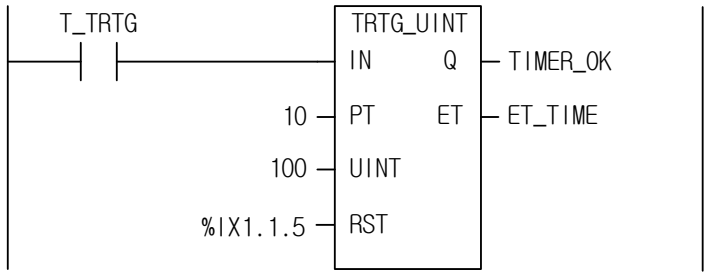
1. TRTG_UINT 펑션 블록은 기동 조건 IN이 1이 되는 순간 Q는 1이 되고, 경과 시간이 설정 시간에 도달하면 타이머 출력 Q는 0이 됩니다.
2. 타이머 경과 시간이 설정 시간이 되기 전에 IN이 또 다시 0에서 1로 되면 경과 시간은 0으로 재설정 되고 다시 증가하여 설정 시간 PT에 도달하면 Q는 0이 됩니다.
3. Reset 입력 조건이 성립하면 타이머 출력 Q는 0이 되고 경과 시간도 0이 됩니다.
4. 설정 시간은 PT * UNIT[ms]입니다.

■ 타임 차트



■ 프로그램 예

1. LD



2. ST

```
INST_TRTG_UINT(IN:=T_TRTG, PT:=10, UNIT:=100, RST:=%IX1.1.5, Q=>TIMER_OK, ET=>ET_TIME);
```

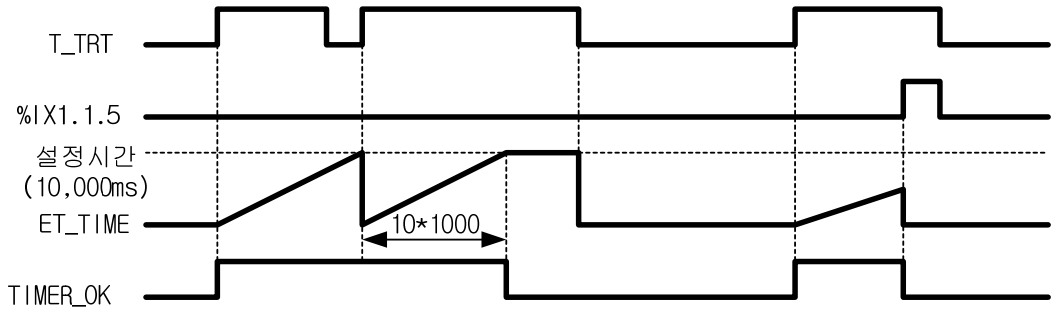
(1) 설정 시간은 PT * UNIT[ms] = 10 * 1000[ms] = 10[s]가 됩니다.

(2) 입력변수 T_TRTG 가 0 에서 1 이 된 후 10 초 동안 TIMER_OK 는 1 이 됩니다. 타이머가 가동된 후 T_TRTG 가 다시 0 에서 1 이 되면 ET_TIME 은 0 부터 다시 시작됩니다.

(3) T_TRTG 가 1 에서 0 이 되어도 10 초 동안 TIMER_OK 는 1 이 됩니다.

(4) ET_TIME 은 증가 후 10,000 에서 멈춥니다. 그리고 T_TRTG 가 0 이 될 때 0 으로 됩니다.

(5) 입력접점 %IX1.1.5 가 1 이 되면 TIMER_OK 와 ET_TIME 모두 0 으로 클리어(Clear) 됩니다.



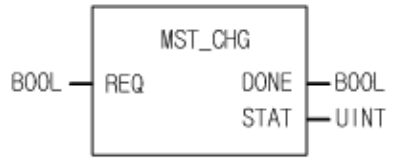
☆ 참고

TRTG_UINT 펑션 블록은 접점이 On 된 후 Off 되어도 해당 동작이 마무리 될 때가지 펑션 블록은 동작합니다.

배열인덱스를 이용한 변수를 사용했을 경우 배열인덱스 에러는 접점이 On 이 되었을 때만 발생합니다.

때문에 TRTG_UINT 펑션 블록에서는 펑션 블록이 동작하더라도 접점이 Off 되어 있다면 배열인덱스 에러가 발생하지 않습니다.

MST_CHG	적용 기종	발생플래그
프로그램에 의한 마스터 전환 명령	XGR	_MASTER_CHG

펑션 블록	설 명
	**입력** REQ : 프로그램에 의한 마스터 전환 요구 **출력** DONE : 마스터 전환이 된 후 On 을 유지 STAT : 수행 결과를 나타냄. 0 이면 에러 없음.

■ 기능

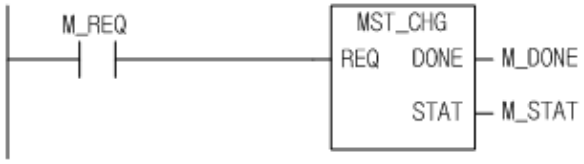
1. REQ 가 On 되면 실행 중인 스캔을 마친 후 마스터 CPU 를 전환합니다.
2. DONE 은 마스터 전환을 한 순간부터 On 되어 REQ 이 Off 될 때까지 On 상태를 유지합니다.
3. STAT 는 명령어 실행 후 에러 상태를 표현합니다.
   - 0 : 정상 동작
   - 1 : 스탠바이 CPU 전원이 Off 상태인 경우
   - 2 : 스탠바이 CPU 운전 모드가 STOP 상태인 경우
   - 3 : 스탠바이 CPU 가 에러 상태인 경우
   - 4 : 런 중 수정 쓰기 상태인 경우

■ 관련 플래그

플래그	설명
_MASTER_CHG	쓰기 가능한 비트 플래그 On 시 마스터 CPU 를 전환하고, 플래그는 Off 됩니다.

■ 프로그램 예

1. LD



2. ST

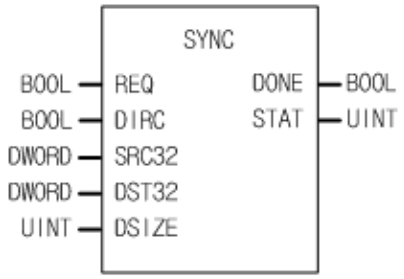
```
INST_MST_CHG(REQ:=M_REQ, DONE=>M_DONE, STAT=>M_STAT);
```

(1) M_REQ 값이 On 되면 마스터 CPU 를 전환합니다.

(2) 전환 후 M_DONE 은 On 됩니다.

에러가 발생한 경우는 M_STAT 에 에러번호를 출력합니다. 정상 동작일 경우 0 이 출력됩니다.

SYNC	적용 기종	발생플래그
마스터 CPU와 스탠바이 CPU 데이터 동기	XGR	-

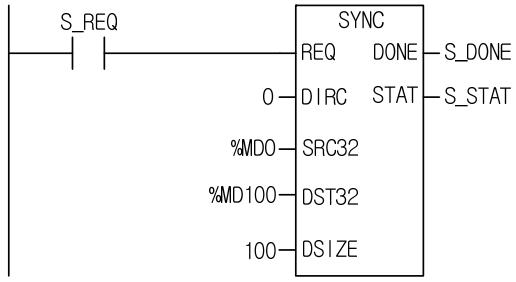
펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 DIRC : 0:마스터 CPU 데이터를 스탠바이 CPU로 동기 1:스탠바이 CPU 데이터를 마스터 CPU로 동기 SRC32 : 데이터를 보낼 직접 변수. DWORD 타입 DST32 : 데이터를 받을 직접 변수. DWORD 타입 DSIZE : 동기시킬 DWORD 데이터 개수 **출력** DONE : 정상 수행시 On STAT : 수행 결과를 나타냄. 0이면 에러 없음.

■ 기능

1. 마스터 CPU와 스탠바이 CPU간에 직접 변수를 동기화하는 명령어입니다.
2. DIRC 변수가 Off 인 경우
   (1) 방향: 마스터 CPU에서 스탠바이 CPU로 데이터가 이동합니다.
   (2) 데이터: SRC32에 설정된 직접 변수에서 DST32에 설정된 직접 변수로 DSIZE만큼의 DWORD 데이터를 동기화합니다.
3. DIRC 변수가 On 인 경우
   (1) 방향: 스탠바이 CPU에서 마스터 CPU로 데이터가 이동합니다.
   (2) 데이터: SRC32로 설정된 직접 변수에서 DST32에 설정된 직접 변수로 DSIZE만큼의 DWORD 데이터를 동기화합니다.
4. SRC32와 DST32 위치에는 직접 변수만 선언이 가능합니다.
5. 스탠바이 CPU가 STOP 모드이거나 에러 상태인 경우에도 직접 변수 영역 동기화를 합니다.
6. STAT는 명령어 실행 후 에러 상태를 표현합니다.
   - 0 : 정상 동작
   - 1 : SRC32 또는 DST32 에 설정된 직접 변수부터 DSIZE 만큼의 범위가 설정한 직접 변수 전체 범위를 초과할 경우
   - 2 : 스탠바이 CPU가 존재하지 않거나 SYNC 명령을 수행할 수 없는 상태일 경우

■ 프로그램 예

1. LD



2. ST

```
INST_SYNC(REQ:=S_REQ, DIRC:=0, SRC32:=%MD0, DST32:=%MD100, DSIZE:=100, DONE=>S_DONE, STAT=>S_STAT);
```

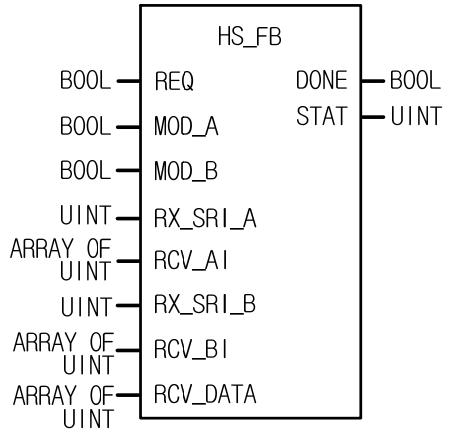
(1) S_REQ 가 On 되면 마스터 CPU 와 스탠바이 CPU 의 직접 변수 데이터를 동기화합니다.

(2) 마스터 CPU 의 %MD0 에서 스탠바이 CPU %MD100 위치에 100 DWORD 만큼 데이터를 복사합니다.

(3) 데이터 동기 후 S_DONE 은 On 됩니다.

(4) 에러가 발생한 경우는 S_STAT 에 에러번호를 출력하고, 정상 동작일 경우 0 을 출력합니다.

HS_FB	적용 기종	발생플래그
이중화 고속링크 서비스 수행을 위한 FB	XGI, XGR	_HSn_STATEm [n:1~12, m:0~127]

펑션 블록	설 명
	**입력** REQ : 1일 때 HS_FB 펑션 블록 실행 MOD_A: A 사이드의 고속링크 플래그(_HSn_STATEm) MOD_B: B 사이드의 고속링크 플래그(_HSn_STATEm) RX_SRI_A: A 사이드 SEQ 번호 RCV_AI: A 사이드 데이터를 저장할 어레이 변수 RX_SRI_B: B 사이드 SEQ 번호 RCV_BI: B 사이드 데이터를 저장할 어레이 변수 RCV_DATA: 입력 데이터를 저장할 어레이 변수 **출력** DONE : 정상 수행 시 On STAT : 수행 결과를 나타냄. 0 이면 에러 없음.

■ 기능

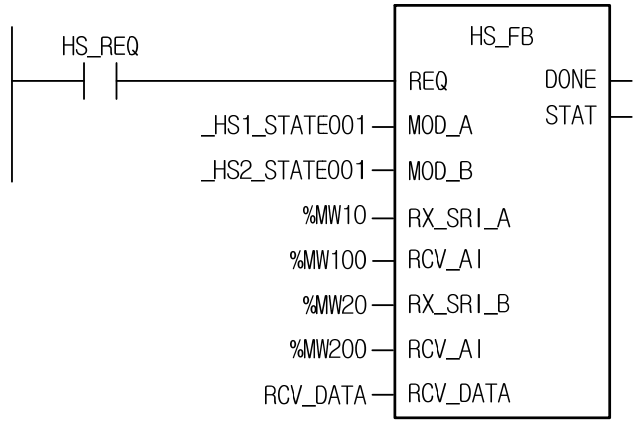
1. 펑션 블록의 요구 신호 REQ 가 On 되면 명령어가 동작합니다.
2. REQ 가 On 되면 DONE 이 On 됩니다. DONE 은 REQ 가 Off 될 때까지 On 을 유지합니다.
3. XG-PD 에서 설정한 고속링크의 파라미터 번호와 블록 인덱스에 맞게 MOD_A, MOD_B 에 고속링크 플래그 (_HSn_STATEm: 종합적 상태 표시 플래그)를 입력합니다.
4. SEQ 번호는 차례대로 증가하는 값으로 매 스캔마다 1 씩 증가하도록 송신측에서 설정합니다.
5. XG-PD 에서 설정한 SEQ 번호 저장영역을 RX_SRI_A, RX_SRI_B 에 입력합니다. (SEQ 번호는 1 WORD 를 사용합니다.)
6. XG-PD 에서 설정한 데이터 저장영역을 RCV_AI, RCV_BI 에 입력합니다.
7. RCV_AI, RCV_BI 에 배열 타입(ARRAY OF UINT)과 개수(XG-PD 에서 설정한 데이터 개수)에 맞추어 데이터 저장 영역을 입력합니다.
8. STAT 는 명령어 실행 후 에러 상태를 표현합니다.
   (1) 0 : 정상 동작
   (2) 1 : 입력 배열 측의 어레이 변수 개수가 다른 경우 (RCV_AI, RCV_BI, RCV_DATA)
   (3) 2 : A, B 사이드 고속링크가 모두 에러 상황인 경우

■ 관련 플래그

플래그	설명
_HSn_STATEm [n:1~12, m:0~127]	고속링크 n 번 m 블록의 종합적 상태 표시

■ 프로그램 예

1. LD



2. ST

```
INST_HS_FB(REQ:=HS_REQ, MOD_A:=_HS1_STATE001, MOD_B:=_HS2_STATE001, RX_SRI_A:=%MW10, RCV_AI:=%MW100,
RX_SRI_B:=%MW20, RCV_BI:=%MW200, RCV_DATA:=RCV_DATA);
```

(1) HS_REQ 가 On 이 되면 HS_FB 가 동작합니다.

(2) A 사이드 SEQ 번호는 %MW10 으로, B 사이드의 SEQ 번호는 %MW20 으로 수신합니다. (XG-PD 설정)

(3) A 사이드 데이터는 %MW100 으로, B 사이드의 데이터는 %MW200 으로 수신합니다. (XG-PD 설정)

(4) 이중화 시스템의 A 사이드 측 통신 모듈의 오류가 발생한 경우, B 사이드의 데이터를 RCV_DATA 에 저장합니다.

(5) 이중화 시스템의 B 사이드 측 통신 모듈의 오류가 발생한 경우, A 사이드의 데이터를 RCV_DATA 에 저장합니다.

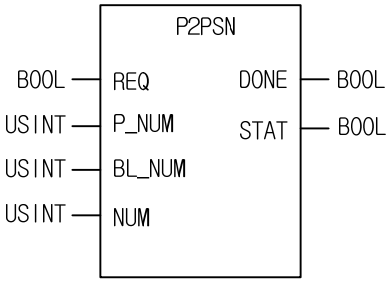
제11장 통신 및 특수 펑션 블록

이번 장에서는 통신 펑션 블록, 특수 펑션 블록, 모션 제어 펑션 블록, 위치결정 펑션 블록에 대한 설명을 합니다.
통신 펑션 블록에 대한 자세한 사용 방법은 각 통신 모듈의 사용설명서를 참고하시기 바랍니다.
특수 펑션 블록, 모션 제어 펑션 블록, 위치결정 펑션 블록에 대한 자세한 사용 방법은 각 특수 모듈, 모션 제어 모듈, 위치결정 모듈의 사용설명서를 참고하시기 바랍니다.

11.1. 통신 펑션 블록

1. 각각의 통신 펑션 블록에 대한 설명입니다.

P2PSN	적용 기종	발생플래그
국번 설정	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 P_NUM : P2P 번호 BL_NUM : 블록 번호 NUM : 국번 **출력** DONE : 최초 동작 후 1을 유지 STAT : 완료 및 ERR 정보

■ 기능

1. P2PSN명령어를 이용하여 런 중 P2P서비스 상대의 국번을 변경할 수 있습니다.
2. P_NUM번 P2P의 BL_NUM번 블록 리모트 국번을 NUM으로 변경합니다.
   해당 통신 모듈: FDEnet, Cnet.

■ 에러

1. 에러 발생 시 STAT에 해당 에러 번호를 표시합니다.

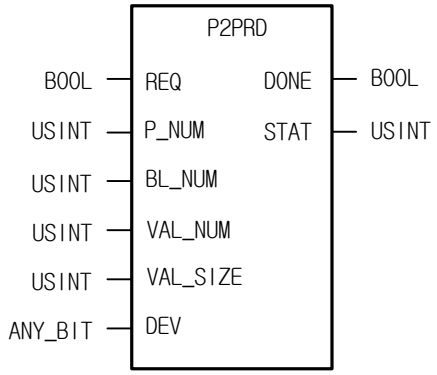
STAT_NUM	내용	세부 설명
1	P2P번호 설정	P_NUM(1~8)이외의 값 설정 시 발생
2	블록 번호 설정	BL_NUM(0~63)이외의 값 설정 시 발생 <단, CNET인 경우 0~31>
4	슬롯 존재 안 함	-
5	모듈 불일치	통신 모듈 아님
6	모듈 불일치	해당 명령어에 사용할 수 없는 통신모듈
7	국번 설정 오류	NUM(0~63)이외의 값 설정 시 발생 단, Cnet인 경우 (0~31)

■ 프로그램 예

1. ST

```
INST_P2PSN(REQ:=REQ_BOOL, P_NUM:=P_NUM_USINT, BL_NUM:=BL_NUM_USINT, NUM:=NUM_USINT, DONE=>DONE_BOOL,
STAT=>STAT_USINT);
```

P2PRD	적용 기종	발생플래그
읽기 영역 지정	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 P_NUM : P2P 번호 BL_NUM : 블록 번호 VAL_NUM : 변수 번호 VAL_SIZE : 변수 크기 DEV : 디바이스(직접변수만 입력 가능) 출력 DONE : 최초 동작 후 1을 유지 STAT : 완료 및 ERR 정보

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	DEV	○	○	○	○	○															

■ 기능

1. P2PRD명령어는 해당 P2P 파라미터 블록의 변수 크기와 READ디바이스 영역을 변경합니다.
   (개별/연속 읽기 모두 변경가능 합니다.)
2. P_NUM, BL_NUM, VAL_NUM을 이용하여 해당 P2P파라미터, 블록, 변수를 지정한 후 변수 크기와 디바이스를 각각 VAL_SIZE(연속일 경우에 VAL_SIZE는 variable size를 의미하며, 개별일 경우에는 변수 type별 크기입니다.), DEV로 변경합니다. 여기서 DEV은 직접 변수만 입력 가능합니다. (예, %MW100)
   해당 통신 모듈: FEnet, FDEnet, Cnet.

■ 에러

1. PD 에서 설정된 P2P 파라미터의 허용 범위를 벗어난 설정을 하는 경우 해당 에러 번호가 발생합니다.

STAT	내용	세부 설명
1	P2P번호 설정 에러	P_NUM(1~8)이외의 값 설정 시 발생
2	블록 번호 설정 에러	BL_NUM(0~63)이외의 값 설정 시 발생 <단, Cnet 인 경우 0~31>
3	변수 번호 설정 에러	PD에서 설정된 P2P 파라미터에 허용되지 않는 변수 번호 입력 시 발생
4	슬롯 존재 안함	-

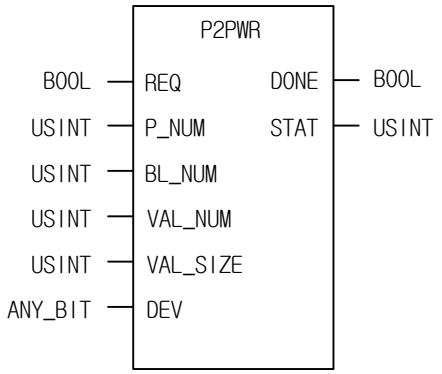
STAT	내용	세부 설명
5	모듈 불일치	통신 모듈 아님
6	모듈 불일치	해당 명령어에 사용할 수 없는 통신모듈
10	모드버스 설정 에러	모드버스의 옵셋은 입력 불가능.(예, 0x10000) DEV은 직접 변수만 입력 가능하기 때문입니다.
11	변수 사이즈 설정 에러	PD 에서 설정된 P2P 파라미터에 허용되지 않는 변수 사이즈 입력 시 발생
12	데이터 타입 설정 에러	PD 에서 설정된 P2P 파라미터에 허용되지 않는 변수 타입 입력 시 발생

■ 프로그램 예

1. ST

```
INST_P2PRD_BOOL(REQ:=REQ_BOOL, P_NUM:=P_USINT, BL_NUM:=BL_USINT, VAL:=VAL_USINT, VAL_SIZE:=SIZE_UINT,
DEV_NUM:=DEV_BOOL, DONE=>DONE_BOOL, STAT=>STAT_USINT);
```

P2PWR	적용 기종	발생플래그
쓰기 영역 지정	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 P_NUM : P2P 번호 BL_NUM : 블록 번호 VAL_NUM : 변수 번호 VAL_SIZE : 변수 크기 DEV : 디바이스(직접변수만 입력 가능) **출력** DONE : 최초 동작 후 1을 유지 STAT : 완료 및 ERR 정보

ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	DEV	○	○	○	○	○															

■ 기능

1. P2PRD 명령어는 해당 P2P 파라미터 블록의 변수 크기와 WRITE 디바이스 영역을 변경합니다.
   (개별/연속 읽기 모두 변경가능 합니다.)
2. P_NUM,BL_NUM,VAL_NUM을 이용하여 해당 P2P 파라미터, 블록, 변수를 지정한 후 변수 크기와 디바이스를 각각 VAL_SIZE(연속일 경우 VAL_SIZE는 variable size를 의미하며, 개별일 경우에는 변수 type별 크기입니다.), DEV로 변경합니다. 여기서 DEV은 직접 변수만 입력 가능합니다. (예, %MW100)
   해당 통신 모듈: FEnet, FDEnet, Cnet.

■ 에러

PD 에서 설정된 P2P 파라미터의 허용 범위를 벗어난 설정을 하는 경우 해당 에러 번호가 발생합니다.

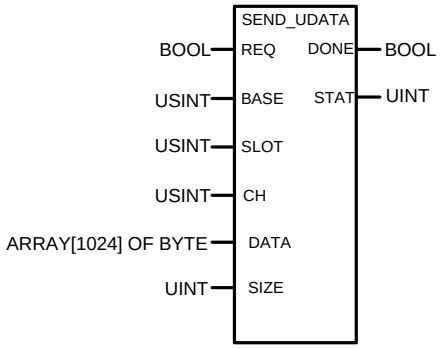
STAT	내용	세부 설명
1	P2P번호 설정 에러	P_NUM(1~8)이외의 값 설정 시 발생
2	블록 번호 설정 에러	BL_NUM(0~63)이외의 값 설정 시 발생 <단, Cnet 인 경우 0~31>
3	변수 번호 설정 에러	PD에서 설정된 P2P 파라미터에 허용되지 않는 변수 번호 입력 시 발생
4	슬롯 존재 안함	-

STAT	내용	세부 설명
5	모듈 불일치	통신 모듈 아님
6	모듈 불일치	해당 명령어에 사용할 수 없는 통신모듈
10	모드버스 설정 에러	모드버스의 옵셋은 입력 불가능. (예: 0x10000) DEV은 직접 변수만 입력 가능하기 때문입니다.
11	변수 사이즈 설정 에러	PD 에서 설정된 P2P 파라미터에 허용되지 않는 변수 사이즈 입력 시 발생
12	데이터 타입 설정 에러	PD에서 설정된 P2P 파라미터에 허용되지 않는 변수 타입 입력 시 발생

■ **프로그램 예**

1. ST

```
INST_P2PWR_BOOL(REQ:=REQ_BOOL, P_NUM:=P_USINT, BL_NUM:=BL_USINT, VAL:=VAL_USINT, VAL_SIZE:=SIZE_UINT,
DEV_NUM:=DEV_BOOL, DONE=>DONE_BOOL, STAT=>STAT_USINT);
```

SEND_UDATA	적용 기종	발생 플래그
사용자 정의 데이터를 송신	XGI, XGR	
펑션	설명	
	입력 REQ : 0->1일 때 펑션블록 실행(펄스 동작) BASE : 베이스 SLOT : 슬롯 CH : 채널(1 또는 2) DATA : 보낼 데이터 영역 SIZE : 보낼 데이터 사이즈 출력 DONE : 에러 없이 실행되면 1을 출력 STAT : 상태 정보	

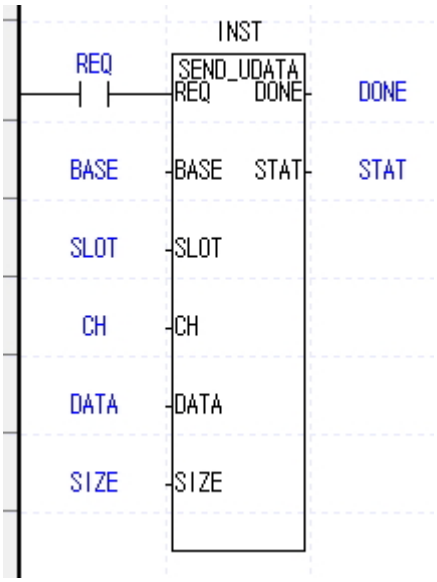
►기능

1. 이 명령은 사용자 정의 데이터(이하:UDATA)를 송신하는 명령 입니다.
2. BASE와 SLOT은 현재 CNET모듈이 장착된 베이스와 슬롯 번호를 입력하면 됩니다.
3. CH는 채널 번호를 의미하고, 1 또는 2만 설정해야 합니다.
4. DATA는 UDATA가 저장된 배열을 나타내며 반드시 ARRAY OF BYTE 타입으로 선언하여야 합니다.
5. SIZE로 선언되는 배열의 사이즈는 1~1024 입니다.(단위 : Byte)
6. DATA[0]부터 SIZE 개수만큼 데이터를 송신 버퍼에 저장한다. (한 번에 보낼 수 있는 데이터 크기는 1024개로 제한)
7. 정상적으로 수행 되면 DONE 및 STAT에 1이 출력되고 에러가 발생되면 STAT에 상태정보가 표시됩니다.

►에러

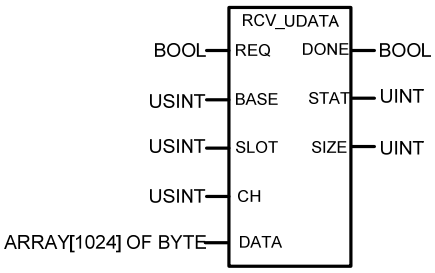
STAT	내용	세부 설명
0	초기상태	명령어 수행 전 초기 상태
1	에러 없음	정상 동작
2	모듈 설정 에러	해당 베이스 슬롯에 모듈이 장착되지 않았거나 CNET 모듈이 아닌 경우 발생
3	채널 설정 에러	입력범위(1, 2)를 초과한 경우
4	배열 사이즈 이상	송신 데이터의 사이즈가 1024 를 넘은 경우 발생
5	통신파라미터 설정에러	Cnet 모듈의 통신파라미터를 사용자정의로 설정하지 않았을 경우 및 링크인에이블을 하지 않은 경우
6	명령어 타임아웃 에러	모듈로부터 응답이 없거나 최대 스캔시간(10 스캔)을 초과한 경우 발생
7	버전 호환 에러	XGI CPU 버전이 V3.9 미만, XGR CPU 버전이 V2.6 미만 혹은 Cnet 버전이 V3.2 미만인 경우 발생

►프로그램 예



BASE, SLOT 에 장착된 CNET 모듈을 이용하여 최대 1024 Byte 를 전송하는 명령

RCV_UDATA	적용 기종	발생 플래그
사용자 정의 데이터를 수신	XGI, XGR	

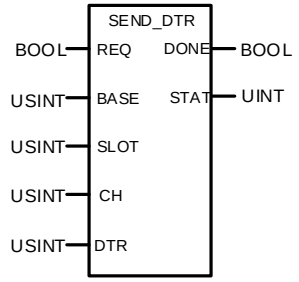
펑션	설명
	입력 REQ : 0->1일 때 펑션블록 실행(펄스 동작) BASE : 베이스 번호 SLOT : 슬롯 번호 CH : 채널 번호(1 또는 2) DATA : 수신 데이터 출력 DONE : 에러 없이 실행되면 1을 출력 STAT : 상태 정보 SIZE : 수신 데이터 크기

►기능

1. 이 명령은 CNET 모듈을 통해 수신된 해당 프레임의 데이터를 저장하는 명령입니다.
2. BASE와 SLOT은 현재 CNET모듈이 장착된 베이스와 슬롯 번호를 입력하면 됩니다.
3. CH는 채널 번호를 의미하고, 1 또는 2만 설정해야 합니다.
4. 출력 DATA는 UDATA를 저장할 배열을 나타내며, 반드시 ARRAY OF BYTE 타입으로 선언하여야 합니다.
5. 출력 SIZE는 수신된 데이터의 크기는 나타냅니다.
6. 정상적으로 수행 되면 DONE 및 STAT에 1이 출력되고 에러가 발생하면 STAT에 상태정보가 표시됩니다.

►에러

STAT	내용	세부 설명
0	초기상태	명령어 수행 전 초기 상태
1	에러 없음	정상 동작
2	모듈 설정 에러	해당 베이스 슬롯에 모듈이 장착되지 않았거나 CNET 모듈이 아닌 경우 발생
3	채널 설정 에러	입력범위(1, 2)를 초과한 경우
4	수신 데이터 없음	수신 된 데이터가 없는 경우 발생
5	통신파라미터 설정에러	Cnet 모듈의 통신파라미터를 사용자정의로 설정하지 않았을 경우 및 링크인에이블을 하지 않은 경우
6	명령어 타임아웃 에러	모듈로부터 응답이 없거나 최대 스캔시간(10 스캔)을 초과한 경우 발생
7	버전 호환 에러	XGI CPU 버전이 V3.9 미만, XGR CPU 버전이 V2.6 미만 혹은 Cnet 버전이 V3.2 미만인 경우 발생

SEND_DTR	적용 기종	발생 플래그
DTR 신호를 전송	XGI, XGR	
펑션	설명	
	입력 REQ : 0->1일 때 펑션블록 실행(펄스 동작) BASE : 베이스 번호 SLOT : 슬롯 번호 CH : 채널 번호(1 또는 2) DTR : (0 또는 1) 출력 DONE : 에러 없이 실행되면 1을 출력 STAT : 상태 정보	

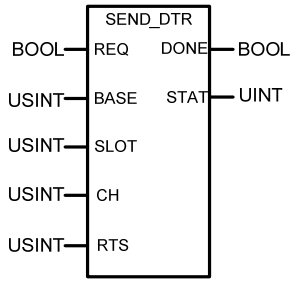
►기능

1. 이 명령은 통신준비가 완료 되었다는 것을 전달하는 DTR(Data Terminal Ready) 신호를 내보내는 명령입니다.
2. 정상적으로 수행 되면 DONE 및 /STAT에 1이 출력되고 에러가 발생하면 STAT에 상태정보가 표시됩니다.

►에러

STAT	내용	세부 설명
0	초기상태	명령어 수행 전 초기상태
1	에러 없음	정상 동작
2	모듈 설정 에러	해당 베이스 슬롯에 모듈이 장착되지 않았거나 CNET 모듈이 아닌 경우 발생
3	채널 설정 에러	입력범위(1, 2)를 초과한 경우
4	DTR 설정 에러	입력범위(0, 1)를 초과한 경우
5	통신파라미터 설정 에러	Cnet 모듈의 통신파라미터를 사용자정의로 설정하지 않았을 경우 및 링크인에이블을 하지 않은 경우
6	명령어 타임아웃 에러	모듈로부터 응답이 없거나 최대 스캔시간(10 스캔)을 초과한 경우 발생
7	버전 호환 에러	XGI CPU 버전이 V3.9 미만, XGR CPU 버전이 V2.6 미만 혹은 Cnet 버전이 V3.2 미만인 경우 발생

SEND_RTS	적용 기종	발생 플래그
RTS 신호를 전송	XGI, XGR	

펑션	설명
	입력 REQ : 0->1일 때 펑션블록 실행(펄스 동작) BASE : 베이스 번호 SLOT : 슬롯 번호 CH : 채널 번호(1 또는 2) RTS : (0 또는 1) 출력 DONE : 에러 없이 실행되면 1을 출력 STAT : 상태 정보

►기능

1. 이 명령은 자신의 수신버퍼 상태를 알려주는 신호인 RTS(Request To Send)를 내보내는 명령입니다.
2. 정상적으로 수행 되면 DONE 및 STAT에 1이 출력 되고 에러가 발생하면 STAT에 상태정보가 표시됩니다.

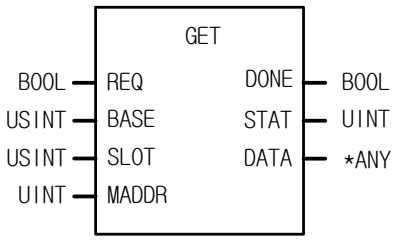
►에러

STAT	내용	세부 설명
0	초기상태	명령어 수행 전 초기상태
1	에러 없음	정상 동작
2	모듈 설정 에러	해당 베이스 슬롯에 모듈이 장착되지 않았거나 CNET 모듈이 아닌 경우 발생
3	채널 설정 에러	입력범위(1, 2)를 초과한 경우
4	RTS 설정 에러	입력범위(0, 1)를 초과한 경우
5	통신파라미터 설정에러	Cnet 모듈의 통신파라미터를 사용자정의로 설정하지 않았을 경우 및 링크인에이블을 하지 않은 경우
6	명령어 타임아웃 에러	모듈로부터 응답이 없거나 최대 스캔시간(10 스캔)을 초과한 경우 발생
7	버전 호환 에러	XGI CPU 버전이 V3.9 미만, XGR CPU 버전이 V2.6 미만 혹은 Cnet 버전이 V3.2 미만인 경우 발생

11.2. 특수 펑션 블록

1. 각각의 특수 펑션 블록에 대한 설명입니다.

GET	적용 기종	발생플래그
특수 모듈 데이터 읽기	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 1일 때 펑션 실행 BASE : 베이스 위치 지정 SLOT : 슬롯 위치 지정 MADDR : 모듈 어드레스 512(0x200) ~ 1023(0x3FF) **출력** DONE : 정상 수행시 1 출력 STAT : 에러 정보 DATA : 모듈로부터 읽어온 데이터

*ANY: ANY 타입 중 WORD, DWORD, INT, UINT, DINT, UDINT 타입 가능

■ 기능

1. 지정한 특수 모듈로부터 데이터를 읽어 옵니다.

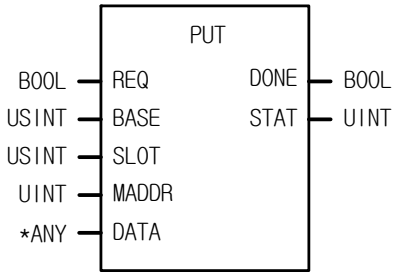
펑션 블록	출력(ANY) 타입	동작 설명
GET_WORD	WORD	지정한 모듈 어드레스(MADDR)부터 WORD 만큼 데이터를 읽어옵니다.
GET_DWORD	DWORD	지정한 모듈 어드레스(MADDR)부터 DWORD 만큼 데이터를 읽어옵니다.
GET_INT	INT	지정한 모듈 어드레스(MADDR)부터 INT 만큼 데이터를 읽어옵니다.
GET_UINT	UINT	지정한 모듈 어드레스(MADDR)부터 UINT 만큼 데이터를 읽어옵니다.
GET_DINT	DINT	지정한 모듈 어드레스(MADDR)부터 DINT 만큼 데이터를 읽어옵니다.
GET_UDINT	UDINT	지정한 모듈 어드레스(MADDR)부터 UDINT 만큼 데이터를 읽어옵니다.

■ 프로그램 예

1. ST

```
INST_GET_WORD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, MADDR:=MADDR_UINT, DONE=>DONE_BOOL,
STAT=>STAT_UINT, DATA=>DATA_WORD);
```

PUT	적용 기종	발생플래그
특수 모듈에 데이터 쓰기	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 1일 때 펑션 실행 BASE : 베이스 위치 지정 SLOT : 슬롯 위치 지정 MADDR : 모듈 어드레스 DATA : 모듈에 저장할 데이터 **출력** DONE : 정상 수행시 1 출력 STAT : 에러 정보

*ANY: ANY 타입 중 WORD, DWORD, INT, USINT, DINT, UDINT 타입 가능

■ 기능

1. 지정한 특수 모듈로부터 데이터를 읽어 옵니다.

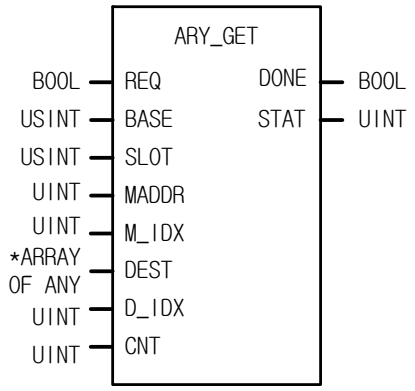
펑션 블록	입력(ANY) 타입	동작 설명
PUT_WORD	WORD	지정한 모듈 어드레스(MADDR)에 WORD 데이터를 저장합니다.
PUT_DWORD	DWORD	지정한 모듈 어드레스(MADDR)에 DWORD 데이터를 저장합니다.
PUT_INT	INT	지정한 모듈 어드레스(MADDR)에 INT 데이터를 저장합니다.
PUT_UINT	UINT	지정한 모듈 어드레스(MADDR)에 UINT 데이터를 저장합니다.
PUT_DINT	DINT	지정한 모듈 어드레스(MADDR)에 DINT 데이터를 저장합니다.
PUT_UDINT	UDINT	지정한 모듈 어드레스(MADDR)에 UDINT 데이터를 저장합니다.

■ 프로그램 예

1. ST

```
INST_PUT_WORD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, MADDR:=MADDR_UINT, DATA:=DATA_WORD,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

ARY_GET	적용 기종	발생플래그
특수 모듈 데이터 읽기(어레이)	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 1일 때 펑션 실행 BASE : 베이스 위치 지정 SLOT : 슬롯 위치 지정 MADDR : 모듈 어드레스 M_IDX : MADDR 로부터 떨어진 거리 DEST : 읽은 데이터를 저장할 어레이 변수 D_IDX : DEST 변수의 시작 인덱스 CNT : 읽을 데이터 개수 **출력** DONE : 정상 수행시 1 출력 STAT : 에러 정보

*ARRAY OF ANY: ANY 타입 중 WORD, DWORD, INT, UINT, DINT, UDINT 타입 가능

■ 기능

1. 지정한 특수 모듈로부터 데이터를 읽어 옵니다.

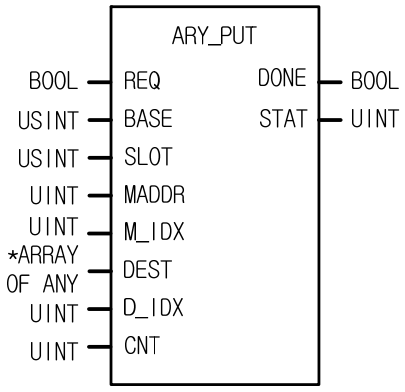
펑션 블록	출력(DEST) 타입	동작 설명
ARY_GET_WORD	WORD	지정한 모듈 어드레스(MADDR)부터 WORD 단위로 CNT 만큼 데이터를 읽어옵니다.
ARY_GET_DWORD	DWORD	지정한 모듈 어드레스(MADDR)부터 DWORD 단위로 CNT 만큼 데이터를 읽어옵니다.
ARY_GET_INT	INT	지정한 모듈 어드레스(MADDR)부터 INT 단위로 CNT 만큼 데이터를 읽어옵니다.
ARY_GET_UINT	UINT	지정한 모듈 어드레스(MADDR)부터 UINT 단위로 CNT 만큼 데이터를 읽어옵니다.
ARY_GET_DINT	DINT	지정한 모듈 어드레스(MADDR)부터 DINT 단위로 CNT 만큼 데이터를 읽어옵니다.
ARY_GET_UDINT	UDINT	지정한 모듈 어드레스(MADDR)부터 UDINT 단위로 CNT 만큼 데이터를 읽어옵니다.

■ 프로그램 예

1. ST

```
INST_ARY_GET_WORD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, MADDR:=MADDR_UINT, M_IDX:=M_UINT,
DEST:=ARY_DEST, D_IDX:=D_UINT, CNT:=CNT_UINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

ARY_PUT	적용 기종	발생플래그
특수 모듈 데이터 쓰기(어레이)	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 1일 때 펑션 실행 BASE : 베이스 위치 지정 SLOT : 슬롯 위치 지정 MADDR : 모듈 어드레스 M_IDX : MADDR 로부터 떨어진 거리 DEST : 저장할 데이터 어레이 변수 D_IDX : DEST 변수의 시작 인덱스 CNT : 읽을 데이터 개수 **출력** DONE : 정상 수행시 1 출력 STAT : 에러 정보

*ARRAY OF ANY: ANY 타입 중 WORD, DWORD, INT, UINT, DINT, UDINT 타입 가능

■ 기능

1. 지정한 특수 모듈로부터 데이터를 읽어 옵니다.

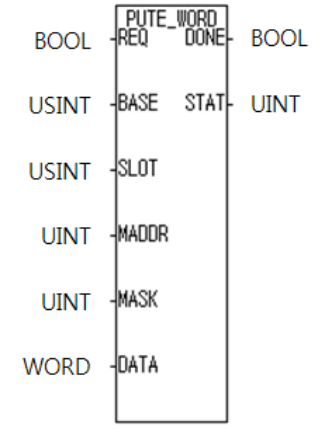
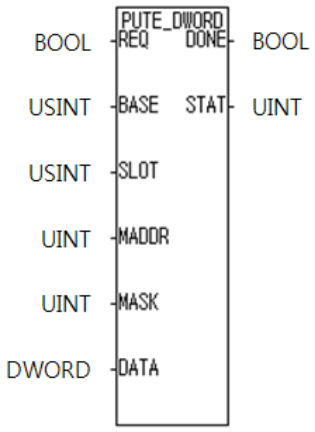
펑션 블록	입력(DEST) 타입	동작 설명
ARY_PUT_WORD	WORD	지정한 모듈 어드레스(MADDR)에 WORD 단위로 CNT 만큼 데이터를 저장합니다.
ARY_PUT_DWORD	DWORD	지정한 모듈 어드레스(MADDR)에 DWORD 단위로 CNT 만큼 데이터를 저장합니다.
ARY_PUT_INT	INT	지정한 모듈 어드레스(MADDR)에 INT 단위로 CNT 만큼 데이터를 저장합니다.
ARY_PUT_UINT	UINT	지정한 모듈 어드레스(MADDR)에 UINT 단위로 CNT 만큼 데이터를 저장합니다.
ARY_PUT_DINT	DINT	지정한 모듈 어드레스(MADDR)에 DINT 단위로 CNT 만큼 데이터를 저장합니다.
ARY_PUT_UDINT	UDINT	지정한 모듈 어드레스(MADDR)에 UDINT 단위로 CNT 만큼 데이터를 저장합니다.

■ 프로그램 예

1. ST

```
INST_ARY_PUT_WORD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, MADDR:=MADDR_UINT, M_IDX:=M_UINT,
DEST:=ARY_DEST, D_IDX:=D_UINT, CNT:=CNT_UINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

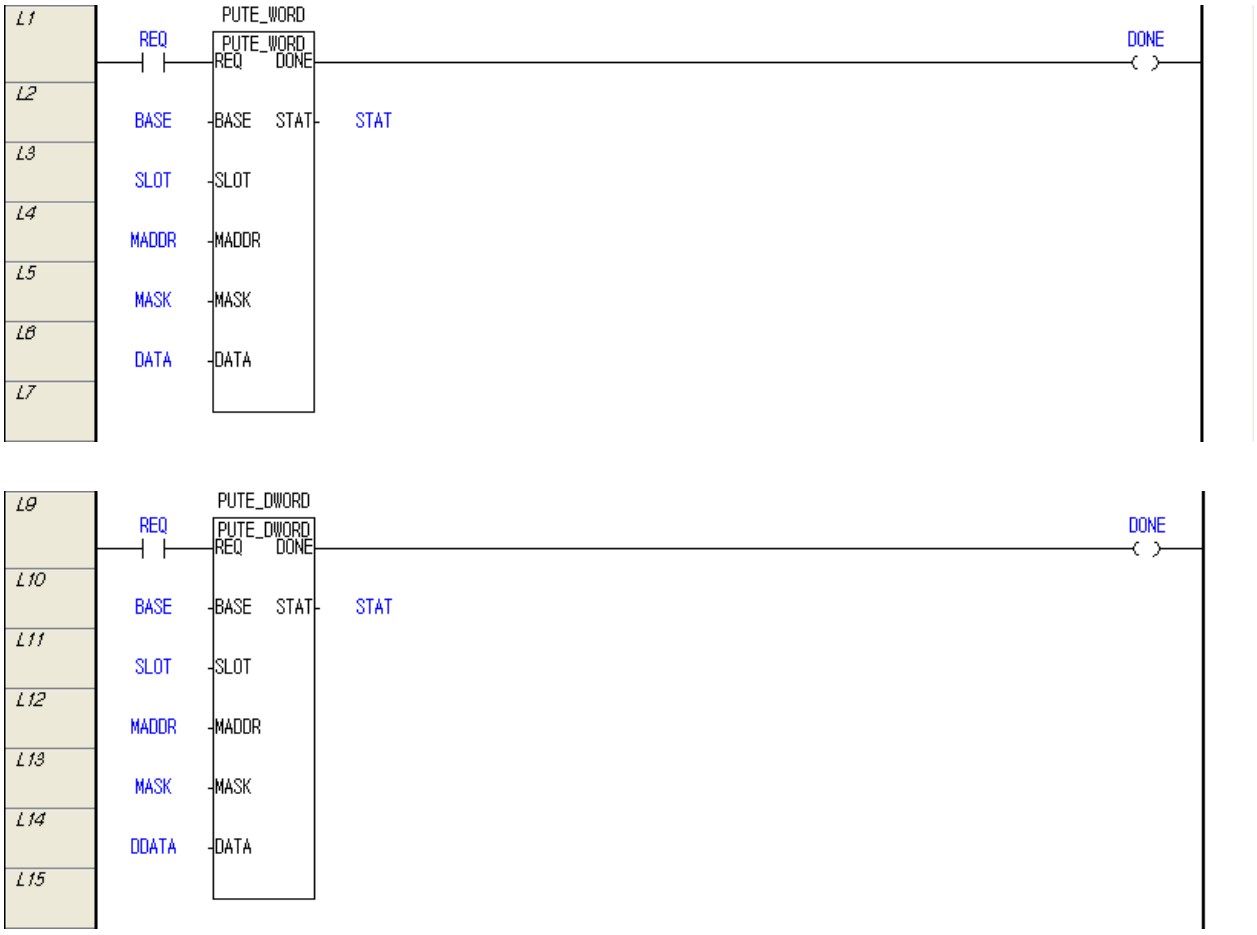
PUTE_WORD/_DWORD	적용 기종	발생플래그
특수 모듈에 데이터 쓰기 (워드선택형)	XGI, XGR	-

펑션 블록	설 명
 	**입력** REQ : 1 일 때 펑션 블록 실행 BASE : 모듈이 장착된 베이스 베이스 위치 SLOT : 모듈이 장착된 슬롯 위치 MADDR : 모듈 어드레스 0 ~ 1023 MASK : 워드 위치 선택 0(하위 워드), 1(상위 워드) DATA : 모듈에 저장할 데이터 (WORD 형, DWORD 형) **출력** DONE : 정상 수행시 1 출력 STAT : 에러 정보

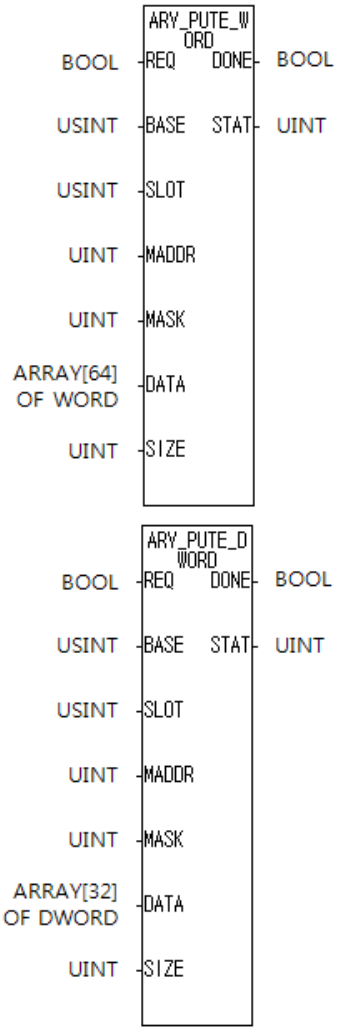
■ 기능

1) 지정한 특수 모듈에 데이터를 써줍니다.
2) 데이터 타입에 따라 WORD 와 DWORD 용을 구분하여 사용합니다.
3) MASK 에 설정한 상태에 따라 데이터를 저장할 위치가 선택됩니다.
   0 => MADDR 에 지정한 모듈 주소의 하위워드
   1 => MADDR 에 지정한 모듈 주소의 상위워드

■ 프로그램 예


L1
L2
L3
L4
L5
L6
L7
PUTE_WORD
REQ
PUTE_WORD
REQ DONE
BASE
BASE STAT
SLOT
SLOT
MADDR
MADDR
MASK
MASK
DATA
DATA
STAT
DONE
L9
L10
L11
L12
L13
L14
L15
PUTE_DWORD
REQ
PUTE_DWORD
REQ DONE
BASE
BASE STAT
SLOT
SLOT
MADDR
MADDR
MASK
MASK
DDATA
DATA
STAT
DONE

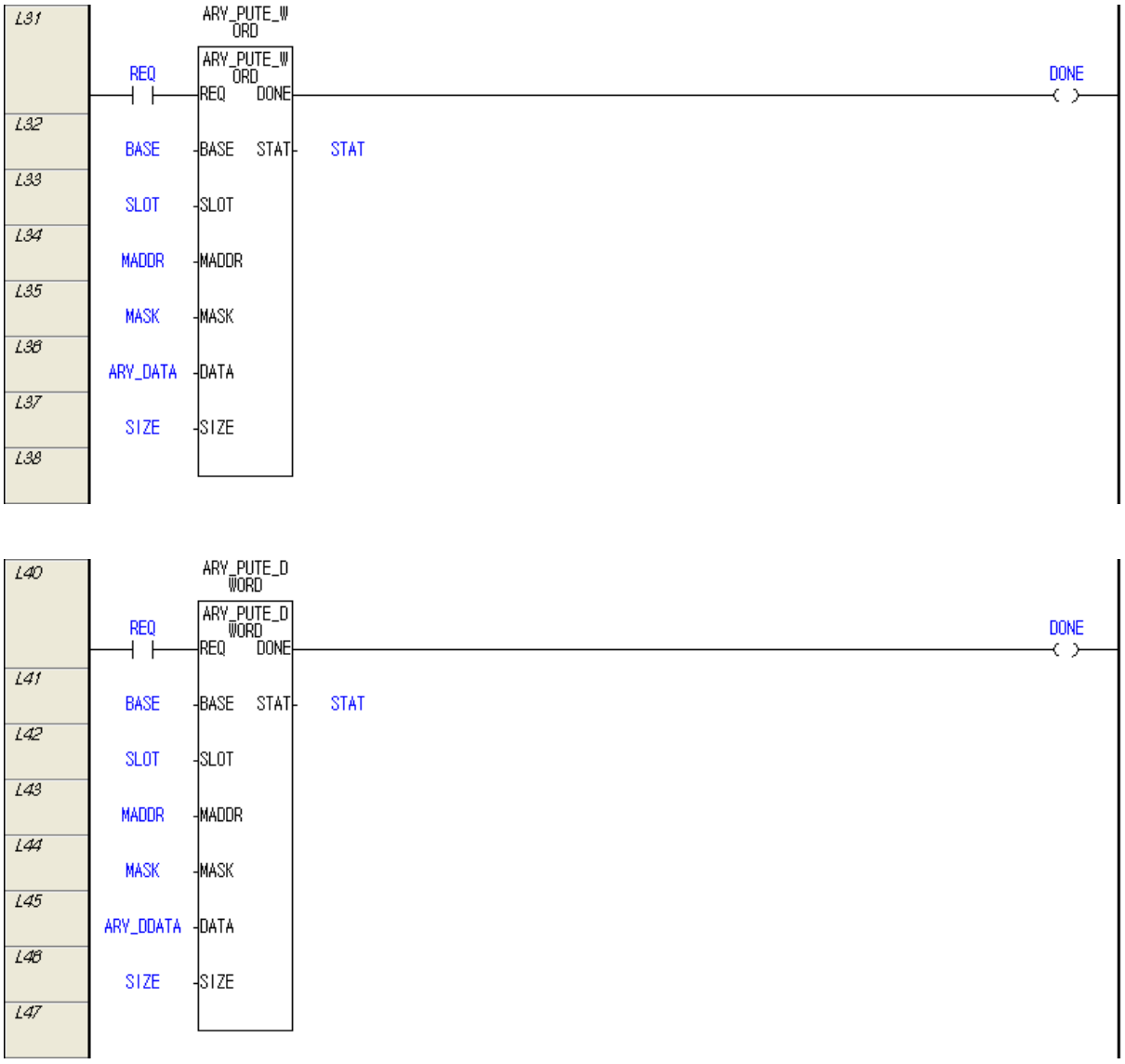
ARY_PUTE_WORD/_DWORD	적용 기종	발생플래그
특수 모듈에 데이터 쓰기 (어레이 워드선택형)	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 1 일 때 펑션 블록 실행 BASE : 모듈이 장착된 베이스 베이스 위치 SLOT : 모듈이 장착된 슬롯 위치 MADDR : 모듈 어드레스 0 ~ 1023 MASK : 워드 위치 선택 0(하위 워드), 1(상위 워드) DATA : 모듈에 저장할 데이터 (어레이) (WORD 형, DWORD 형) SIZE : 쓸 데이터 개수 (1~64[WORD 형], 1~32[DWORD 형]) **출력** DONE : 정상 수행시 1 출력 STAT : 에러 정보

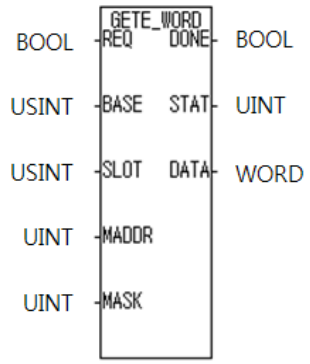
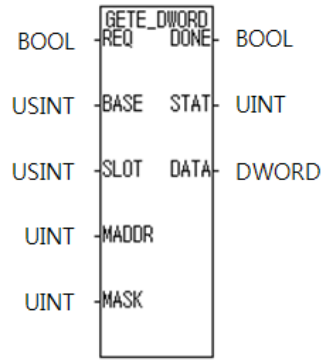
■ 기능

1) 지정한 특수 모듈에 설정한 개수의 데이터를 써줍니다.
2) 데이터 타입에 따라 WORD 와 DWORD 용을 구분하여 사용합니다.(어레이)
3) MASK 에 설정한 상태에 따라 데이터를 저장할 위치가 선택됩니다.
   0 => MADDR 에 지정한 모듈 주소의 하위워드
   1 => MADDR 에 지정한 모듈 주소의 상위워드

■ 프로그램 예


L31
L32
L33
L34
L35
L36
L37
L38
ARY_PUTE_WORD
ARY_PUTE_WORD
REQ
DONE
BASE
STAT
SLOT
MADDR
MASK
ARY_DATA
DATA
SIZE
REQ
BASE
STAT
DONE
L40
L41
L42
L43
L44
L45
L46
L47
ARY_PUTE_DWORD
ARY_PUTE_DWORD
REQ
DONE
BASE
STAT
SLOT
MADDR
MASK
ARY_DDATA
DATA
SIZE
REQ
BASE
STAT
DONE

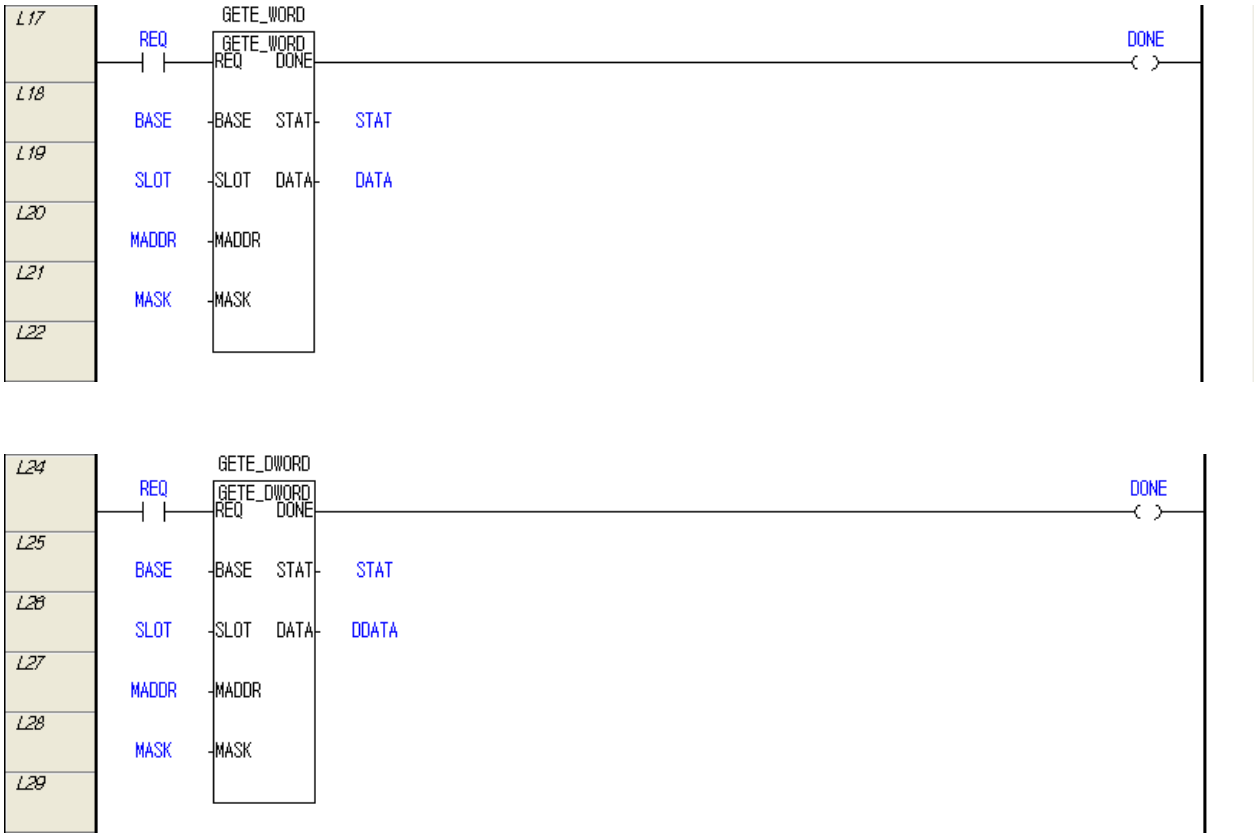
GETE_WORD/_DWORD	적용 기종	발생플래그
특수 모듈 데이터 읽기 (워드선택형)	XGI, XGR	-

펑션 블록	설 명
 	**입력** REQ : 1 일 때 펑션 블록 실행 BASE : 모듈이 장착된 베이스 베이스 위치 SLOT : 모듈이 장착된 슬롯 위치 MADDR : 모듈 어드레스 0 ~ 1023 MASK : 워드 위치 선택 0(하위 워드), 1(상위 워드) **출력** DONE : 정상 수행시 1 출력 STAT : 에러 정보 DATA : 모듈로부터 읽어온 데이터 (WORD 형, DWORD 형)

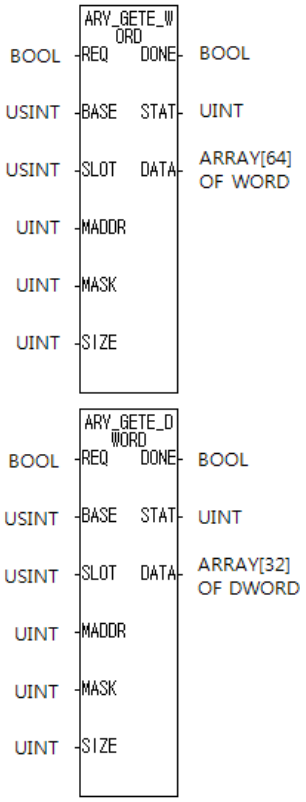
■ 기능

1) 지정한 특수 모듈로부터 데이터를 읽어옵니다.
2) 데이터 타입에 따라 WORD 와 DWORD 용을 구분하여 사용합니다.
3) MASK 에 설정한 상태에 따라 읽을 데이터의 위치가 선택됩니다.
   0 => MADDR 에 지정한 모듈 주소의 하위워드
   1 => MADDR 에 지정한 모듈 주소의 상위워드

■ 프로그램 예


L17
L18
L19
L20
L21
L22
GETE_WORD
REQ
GETE_WORD
REQ
DONE
DONE
BASE
BASE
STAT
STAT
SLOT
SLOT
DATA
DATA
MADDR
MADDR
MASK
MASK
L24
L25
L26
L27
L28
L29
GETE_DWORD
REQ
GETE_DWORD
REQ
DONE
DONE
BASE
BASE
STAT
STAT
SLOT
SLOT
DATA
DDATA
MADDR
MADDR
MASK
MASK

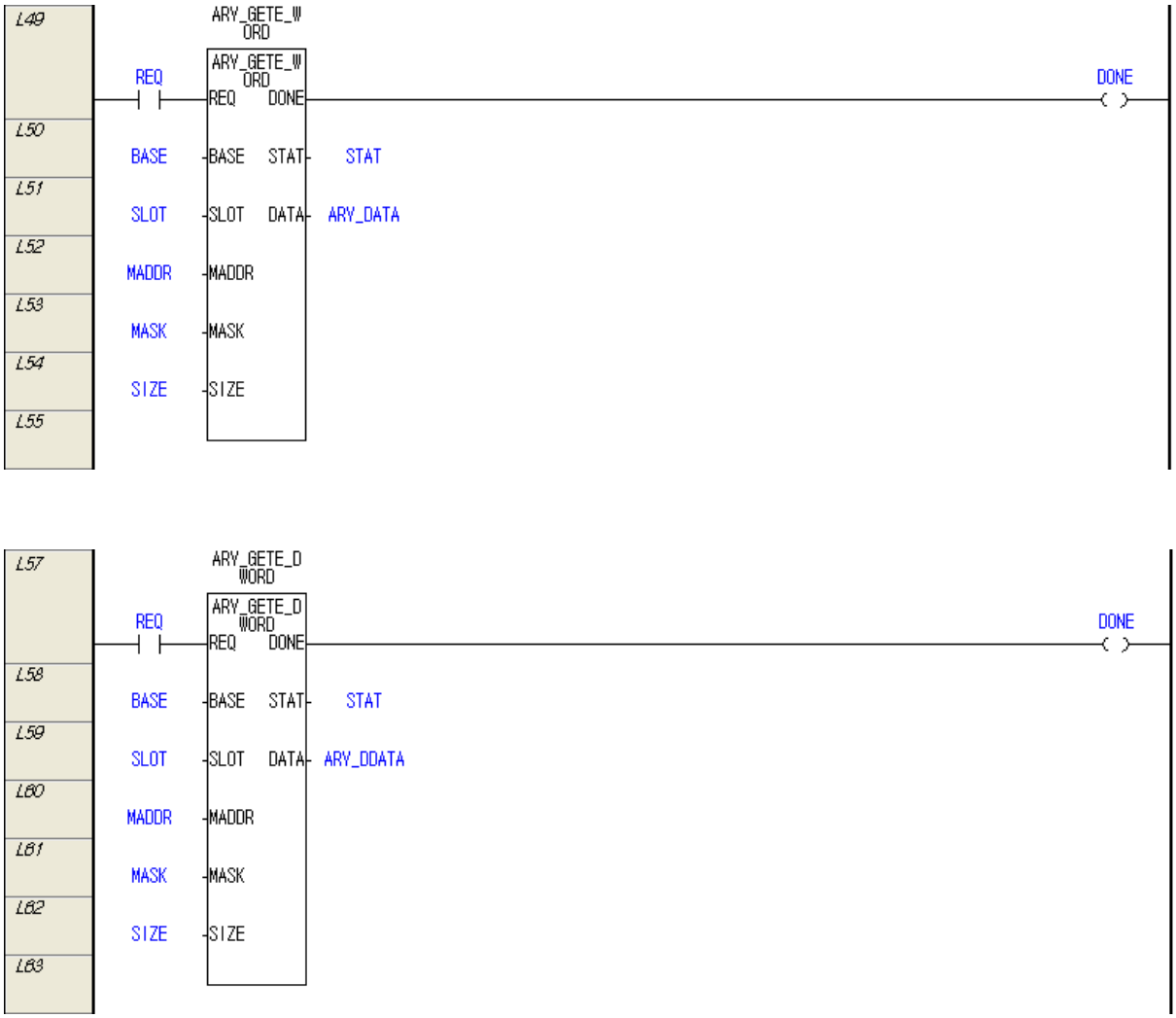
ARY_GETE_WORD/_DWORD	적용 기종	발생플래그
특수 모듈 데이터 읽기 (어레이 워드선택형)	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 1 일 때 펑션블록 실행 BASE : 모듈이 장착된 베이스 위치 SLOT : 모듈이 장착된 슬롯 위치 MADDR : 모듈 어드레스 0 ~ 1023 MASK : 워드위치선택 0(하위워드), 1(상위워드) SIZE : 읽을 데이터 개수(1~64[WORD 형], 1~32[DWORD 형]) **출력** DONE : 정상 수행시 1 출력 STAT : 에러정보 DATA : 모듈로부터 읽어온 데이터(어레이) (WORD 형, DWORD 형)

■ 기능

1) 지정한 특수 모듈로부터 설정한 개수만큼 데이터를 읽어옵니다.
2) 데이터 타입에 따라 WORD 와 DWORD 용을 구분하여 사용합니다.(어레이)
3) MASK 에 설정한 상태에 따라 읽을 데이터의 위치가 선택됩니다.
   0 => MADDR 에 지정한 모듈 주소의 하위워드
   1 => MADDR 에 지정한 모듈 주소의 상위워드

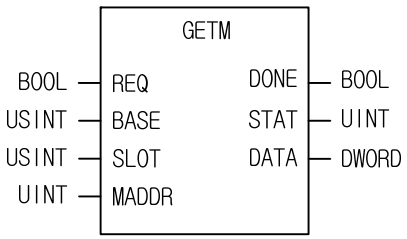
■ 프로그램 예


L49
L50
L51
L52
L53
L54
L55
ARY_GETE_W ORD
ARY_GETE_W ORD
REQ
DONE
REQ
BASE
BASE
STAT
STAT
SLOT
SLOT
DATA
ARY_DATA
MADDR
MADDR
MASK
MASK
SIZE
SIZE
L57
L58
L59
L60
L61
L62
L63
ARY_GETE_D WORD
ARY_GETE_D WORD
REQ
DONE
REQ
BASE
BASE
STAT
STAT
SLOT
SLOT
DATA
ARY_DDATA
MADDR
MADDR
MASK
MASK
SIZE
SIZE

11.3. 모션 제어 펑션 블록

1. 모션 제어 펑션 블록에 대한 설명입니다.

GETM	적용 기종	발생플래그
모션 제어 모듈 데이터 읽기	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 1일 때 펑션 실행 BASE : 베이스 위치 지정 SLOT : 슬롯 위치 지정 MADDR : 모듈 어드레스 512(0x200) ~ 1023(0x3FF) **출력** DONE : 정상 수행시 1 출력 STAT : 에러 정보 DATA : 모듈로부터 읽어온 데이터

■ 기능

1. 지정한 모션 제어 모듈의 읽기 공유 메모리 주소 MADDR 에서 데이터를 읽어옵니다.

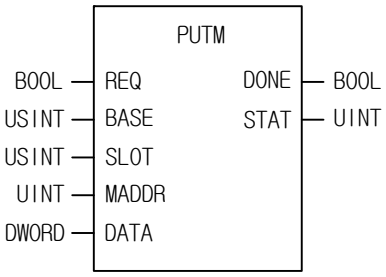
펑션 블록	출력(DATA) 타입	동작 설명
GETM_DWORD	DWORD	지정한 모듈 어드레스(MADDR)부터 DWORD 만큼 데이터를 읽어옵니다.

■ 프로그램 예

1. ST

```
INST_GETM(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, MADDR:=MADDR_UINT, DONE=>DONE_BOOL,
STAT=>STAT_UINT, DATA=>DATA_DWORD);
```

PUTM	적용 기종	발생플래그
특수 모듈(모션모듈)에 데이터 쓰기	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 1일 때 펑션 실행 BASE : 베이스 위치 지정 SLOT : 슬롯 위치 지정 MADDR : 모듈 어드레스 0(0x00) ~ 511(0x1FF) DATA : 모듈에 저장할 데이터 **출력** DONE : 정상 수행시 1 출력 STAT : 에러 정보

■ 기능

1. 지정한 모션 제어 모듈의 쓰기 공유 메모리 주소 MADDR에 데이터를 저장합니다.

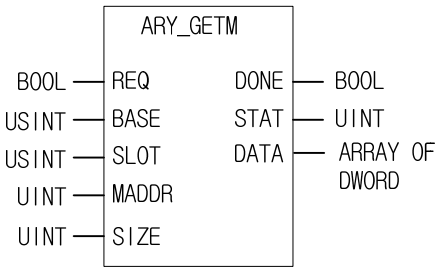
펑션 블록	DATA 타입	동작 설명
PUT_DWORD	DWORD	지정한 모듈 어드레스(MADDR)에 DWORD 데이터를 저장합니다.

■ 프로그램 예

1. ST

```
INST_PUTM(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, MADDR:=MADDR_UINT, DATA:=DATA_DWORD,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

ARY_GETM	적용 기종	발생플래그
모션 제어 모듈 데이터 읽기(어레이)	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 1일 때 펑션 실행 BASE : 베이스 위치 지정 SLOT : 슬롯 위치 지정 MADDR : 읽기 시작주소 512(0x200) ~ 1023(0x3FF) SIZE : 읽을 데이터 개수 (1 ~ 512) **출력** DONE : 정상 수행시 1 출력 STAT : 에러 정보 DATA : 읽은 데이터를 저장할 어레이 변수 (ARRAY of DWORD)

■ 기능

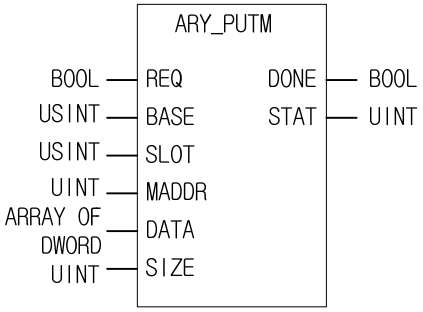
1. 지정한 모션 제어 모듈의 읽기 공유 메모리 MADDR 에서 SIZE 크기만큼 데이터를 읽어옵니다.

■ 프로그램 예

1. ST

```
INST_ARY_GETM(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, MADDR:=MADDR_UINT, SIZE:=SIZE_UINT,
DONE=>DNOE_BOOL, STAT=>STAT_UINT, DATA=>ARY_DATA);
```

ARY_PUTM	적용 기종	발생플래그
모션 제어 모듈 데이터 쓰기(어레이)	XGI	-

펑션 블록	설 명
	**입력** REQ : 1일 때 펑션 실행 BASE : 베이스 위치 지정 SLOT : 슬롯 위치 지정 MADDR : 쓰기 시작주소 0(0x0) ~ 511(0x1FF) DATA : 저장할 데이터 어레이 변수 (ARRAY OF DWORD) SIZE : 쓰기 데이터 개수 (1 ~ 512) **출력** DONE : 정상 수행시 1 출력 STAT : 에러 정보

■ 기능

1. 지정한 모션 제어 모듈의 쓰기 공유 메모리 주소 MADDR 에 SIZE 크기만큼 데이터를 저장합니다.

■ 프로그램 예

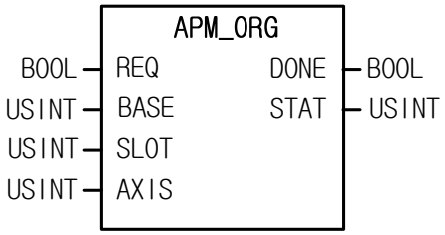
1. ST

```
INST_ARY_PUTM(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, MADDR:=MADDR_UINT,
DATA:=ARY_DATA, SIZE:=SIZE_UINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

11.4. 위치결정 펑션 블록 (APM)

1. 각각의 특수 펑션 블록에 대한 설명입니다.

APM_ORG	적용 기종	발생플래그
원점 복귀 기동	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 이 명령은 위치결정 모듈에 원점 복귀 기동을 실행하는 명령입니다.
2. 각 축의 원점 복귀 파라미터에 설정된 방향, 보정량, 속도(고속, 저속), 어드레스 및 드웰 시간 등으로 원점을 찾는 운전 지령입니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 원점 복귀 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.

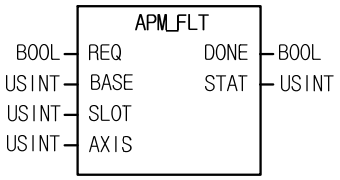
   0: X 축, 1: Y 축, 2: Z 축(XEC 의 경우 Z 축 지원안함)

■ 프로그램 예

1. ST

```
INST_APM_ORG(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, DONE=>DNOE_BOOL, STAT=>STAT_UINT);
```

APM_FLT	적용 기종	발생플래그
부동 원점 설정	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

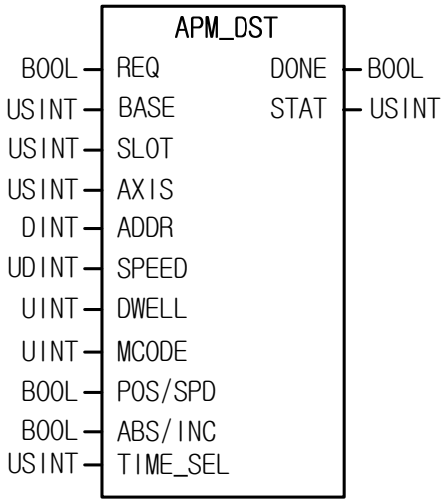
1. 이 명령은 위치결정 모듈에 부동 원점 설정을 실행하는 명령입니다.
2. 기계의 원점 복귀 동작을 수행하지 않고 현재 위치를 강제로 원점으로 설정하고자 할 때 사용하는 지령으로 원점 복귀 어드레스에 지정된 어드레스 값이 현재의 위치가 됩니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 부동 원점 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러6” 이 발생합니다.
   0: X 축, 1: Y 축, 2: Z 축(XEC 의 경우 Z 축 지원안함)

■ 프로그램 예

1. ST

```
INST_APM_FLT(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_DST	적용 기종	발생플래그
직접 기동	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 ADDR : 목표위치 어드레스 설정 -2,147,483,648 ~ +2,147,483,647 SPEED : 목표속도 설정 Open Collector : 1 ~ 200,000[pps] Line Driver : 1 ~ 1,000,000[pps] DWELL : 드웰 시간 0 ~ 50000[ms] MCODE : M Code 값 설정 POS/SPD : 위치제어/속도제어 설정 0 : 위치제어, 1 : 속도제어 ABS/INC : 절대좌표/상대좌표 설정 0 : 절대좌표, 1 : 상대좌표 TIME_SEL: 가감속 시간 번호 설정 0 : 가감속 시간 1 1 : 가감속 시간 2 2 : 가감속 시간 3 3 : 가감속 시간 4 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

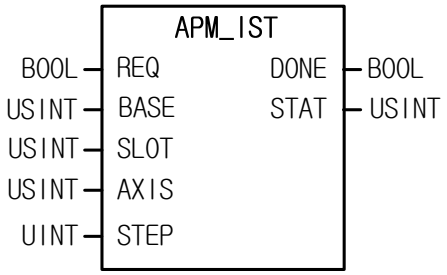
1. 이 명령은 위치결정 모듈에 직접 기동 실행하는 명령입니다.
2. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS로 지정된 축에 직접기동 지령을 내립니다.
3. AXIS에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러6"이 발생합니다.
   0: X축, 1: Y축, 2: Z축(XEC의 경우 Z축 지원안함)
4. SPEED, DWELL, TIME_SEL에 설정된 값이 설정범위를 넘을 경우 STAT에 "에러11"이 발생합니다.

■ 프로그램 예

1. ST

INST_APM_DST(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, ADDR:=ADDR_DINT, SPEED:=SPEED_UDINT, DWELL:=DWELL_UINT, MCODE:=MCODE_UINT, POS_SPD:=POS_BOOL, ABS_INC:=ABS_BOOL, TIME_SEL:=TIME_USINT, DONE=>DNOE_BOOL, STAT=>STAT_UINT);

APM_IST	적용 기종	발생플래그
간접 기동	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 STEP : 운전할 스텝번호 0 ~ 400 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

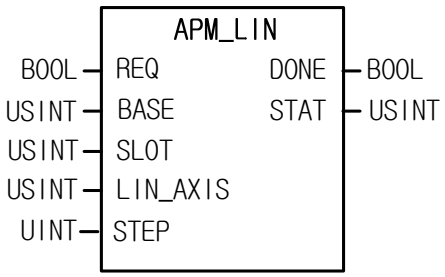
1. 이 명령은 위치결정 모듈에 간접 기동을 실행하는 명령입니다.
2. 운전 데이터로 설정된 축의 운전 스텝 번호를 지정하여 운전하고자 할 때 사용합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS로 지정된 축에 간접기동 지령을 내립니다.
4. AXIS에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러6"이 발생합니다.
   0: X축, 1: Y축, 2: Z축 (XEC의 경우 Z축 지원안함)
5. STEP에 설정된 값이 설정 범위(0 ~ 400 (XEC의 경우 0 ~ 80))를 넘을 경우 STAT에 "에러11"이 발생합니다.
6. STEP에 0을 설정하면 현재 스텝을 운전합니다.

■ 프로그램 예

1. ST

```
INST_APM_IST(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, STEP:=STEP_UINT,
DONE=>DNOE_BOOL, STAT=>STAT_UINT);
```

APM_LIN	적용 기종	발생플래그
직선 보간 기동	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 LIN_AXIS : 보간 운전축 설정 3 : X/Y 축 5 : X/Z 축 6 : Y/Z 축 7 : X/Y/Z 축 STEP : 운전할 스텝번호 0 ~ 400 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 이 명령은 위치결정 모듈에 직선보간 기동 지령을 내리는 명령입니다.
2. 2축 또는 3축용 위치결정 모듈에서 직선 보간 운전을 위한 지령입니다.
3. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 LIN_AXIS에 설정된 축에 직선보간 기동 지령을 내립니다.
4. LIN_AXIS에 설정값 이외의 값을 설정하면 "에러6"이 발생합니다. 설정은 아래와 같이 각 bit를 셋(Set)하여 설정합니다.

15 ~ 4	2	1	0
-	Z축(XEC의 경우 Z축 지원안함)	Y축	X축

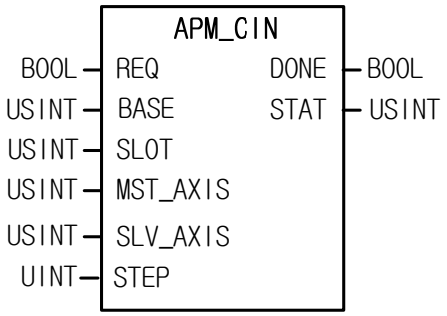
5. STEP에 설정된 값이 설정 범위(0 ~ 400(XEC의 경우 0 ~ 80))를 넘을 경우 STAT에 "에러11"이 발생합니다.
6. STEP에 0을 설정하면 현재 스텝을 운전합니다.

■ 프로그램 예

1. ST

```
INST_APM_LIN(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, LIN_AXIS:=LIN_USINT, STEP:=STEP_UINT,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_CIN	적용 기종	발생플래그
원호 보간 기동	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 MST_AXIS : 원호보간 주축 설정 0:X 축, 1:Y 축, 2:Z 축 SLV_AXIS : 직선보간 종축 설정 0:X 축, 1:Y 축, 2:Z 축 STEP : 운전할 스텝번호 0 ~ 400 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

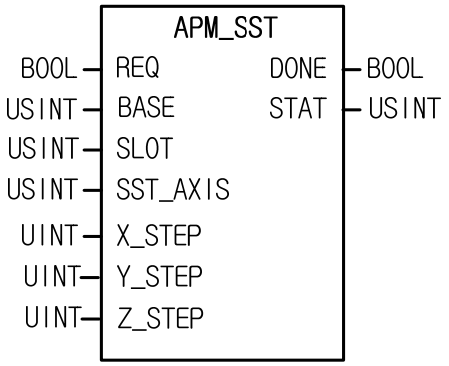
1. 이 명령은 위치결정 모듈에 원호 보간 기동 지령을 내리는 명령입니다.
2. 2축 또는 3축용 위치결정 모듈에서 원호 보간 운전을 위한 지령입니다.
3. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 MST_AXIS로 지정된 축에 원호보간 기동 지령을 내립니다.
4. MST_AXIS에는 원호보간 동작의 주축을 설정하며 다음과 같은 값을 설정할 수 있습니다.
   0: X축, 1: Y축, 2: Z축
5. SLV_AXIS에는 원호보간 동작의 종축을 설정하며 다음과 같은 값을 설정할 수 있습니다.
   0: X축, 1: Y축, 2: Z축
6. MST_AXIS와 SLV_AXIS에 설정값 이상의 값을 설정하면 "에러6"이 발생합니다.
7. STEP에 설정된 값이 설정범위(0 ~ 400)를 넘을 경우 STAT에 "에러11"이 발생합니다.
8. STEP에 0을 설정하면 현재스텝을 운전합니다.

■ 프로그램 예

1. ST

```
INST_APM_CIN(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, MST_AXIS:=MST_USINT, SLV_AXIS:=SLV_USINT,
STEP:=STEP_UINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SST	적용 기종	발생플래그
동시기동	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 SST_AXIS : 동시기동 축 설정 3 : X/Y 축 5 : X/Z 축 6 : Y/Z 축 7 : X/Y/Z 축 X_STEP : X 축의 동시 기동운전 스텝 번호 설정 0 ~ 400 Y_STEP : Y 축의 동시 기동운전 스텝 번호 설정 0 ~ 400 Z_STEP : Z 축의 동시 기동운전 스텝 번호 설정 0 ~ 400 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 이 명령은 위치결정 모듈에 동시 기동 지령을 내리는 명령입니다.
2. 2축 또는 3축의 운전을 동시에 시작하고자 할 때 사용합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 SST_AXIS로 지정된 축에 동시기동 지령을 내립니다.
4. SST_AXIS에 설정값 이외의 값을 설정하면 “에러6”이 발생합니다. 설정은 아래와 같이 각 bit를 set하여 설정합니다.

15 ~ 4	2	1	0
-	Z축(XEC의 경우 Z축 지원안함)	Y축	X축

5. X_STEP, Y_STEP, Z_STEP에는 동시 기동할 축 중 각각 X축, Y축, Z축이 운전할 스텝 번호를 설정합니다.
6. X_STEP, Y_STEP, Z_STEP에 설정된 값이 설정범위(0 ~ 400(XEC의 경우 0 ~ 80))를 넘을 경우 STAT에 “에러11”이 발생합니다.
7. X_STEP, Y_STEP, Z_STEP에 0을 설정하면 각 축의 현재 스텝을 운전합니다.

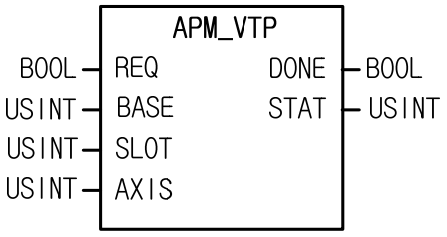
■ 프로그램 예

1. ST

```
INST_APM_SST(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, SST_AXIS:=SST_USINT, X_STEP:=X_UINT,
```

```
Y_STEP:=Y_UINT, Z_STEP:=Z_UINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_VTP	적용 기종	발생플래그
속도/위치 제어 변경	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

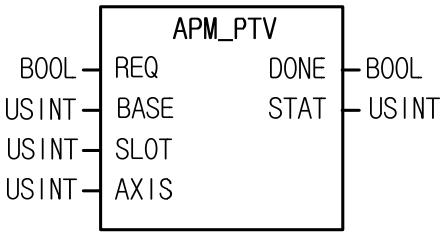
1. 이 명령은 위치결정 모듈에 속도/위치 제어 전환을 실행하는 명령입니다.
2. 지정된 축이 속도 제어 운전을 하다가 속도/위치제어 변경 지령을 받으면 속도 제어에서 위치 제어로 전환 되어 운전합니다.
3. 이 지령이 실행되면 실행된 순간에 원점이 결정되고 이전 속도 제어 기동시 설정한 목표 위치까지 이동한 후 위치결정을 완료합니다.
4. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS로 지정된 축에 속도/위치 제어 변경 지령을 내립니다.
5. AXIS에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러6"이 발생합니다.
   0: X축, 1: Y축, 2: Z축(XEC의 경우 Z축 지원안함)

■ 프로그램 예

1. ST

```
INST_APM_VTP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, DONE=>DONE_BOOL,
STAT=>STAT_UINT)
```

APM_PTV	적용 기종	발생플래그
위치/속도 제어 전환	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

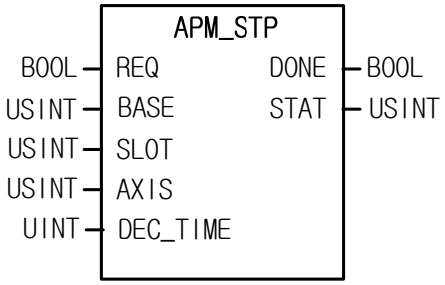
1. 이 명령은 위치결정 모듈에 위치/속도 제어 전환을 실행하는 명령입니다.
2. 지정된 축이 지정된 이동량으로 위치 제어 운전을 하다가 위치/속도 제어 변경 지령을 받으면 위치 제어에서 속도 제어로 전환 되어 운전합니다.
3. 이 지령이 실행되면 실행된 순간에 원점이 미결정되면서 속도제어 운전을 합니다.
4. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS로 지정된 축에 위치/속도 제어 변경 지령을 내립니다.
5. AXIS에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러16"이 발생합니다.
   0: X축, 1: Y축, 2: Z축(XEC의 경우 Z축 지원안함)

■ 프로그램 예

1. ST

```
INST_APM_PTV(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, DONE=>DONE_BOOL,
STAT=>STAT_UINT);
```

APM_STP	적용 기종	발생플래그
감속 정지	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 DEC_TIME : 감속정지 시간 0:운전 시작시 적용된 가감속 시간. 1 ~ 65,535 : 1 ~ 65,535ms **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

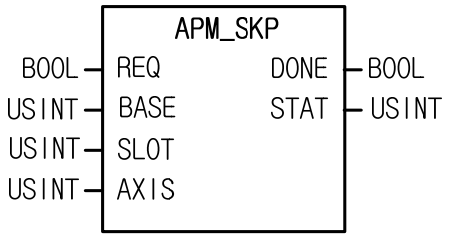
■ 기능

1. 이 명령은 위치결정 모듈에 감속 정지를 실행하는 명령입니다.
2. 운전 데이터에 의한 운전 중에 정지 지령을 만나면 감속 정지한 후 기동 명령에 의해 다시 운전을 시행합니다.
3. 속도동기, 위치동기에서는 각 속도동기나 위치동기 상태에서 빠져나올 때 사용합니다.
4. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS로 지정된 축에 감속 정지 지령을 내립니다.
5. AXIS에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러6"이 발생합니다.
   0: X축, 1: Y축, 2: Z축(XEC의 경우 Z축 지원안함)

■ 프로그램 예

1. ST

```
INST_APM_STP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, DEC_TIME:=DEC_UINT,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SKP	적용 기종	발생플래그
스킵 운전	XGI, XGR	-
펑션 블록	설 명	
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력	

■ 기능

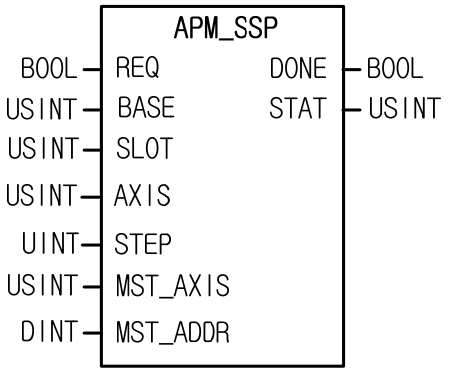
1. 이 명령은 위치결정 모듈에 스킵 운전을 실행하는 명령입니다.
2. 운전 스텝을 운전하지 않고 그 다음 스텝으로 이동하여 실행할 때 사용합니다.
3. 한번 실행할 때마다 현재의 운전 스텝을 건너 뛰어 그 다음의 운전 스텝을 운전합니다.
4. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS로 지정된 축에 스킵 운전 지령을 내립니다.
5. AXIS에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러6"이 발생합니다.
   0: X축, 1: Y축, 2: Z축

■ 프로그램 예

1. ST

```
INST_APM_SKP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SSP	적용 기종	발생플래그
위치 동기	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 STEP : 운전할 스텝번호 0 ~ 400 MST_AXIS : 위치동기 주축 설정 0:X 축, 1:Y 축, 2:Z 축 MST_ADDR : 위치동기를 실행할 주축의 위치 설정 -2,147,483,648 ~ 2,147,483,647 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

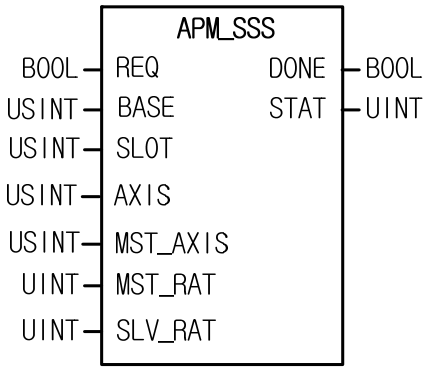
1. 이 명령은 위치결정 모듈에 위치동기 지령을 내리는 명령입니다.
2. 지령을 내린 축을 종축으로 하고 주축으로 설정된 축이 설정된 동기 위치에 도달하면 지령축이 설정한 운전 스텝을 운전 합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS로 지정된 축에 위치동기 지령을 내립니다.
4. AXIS에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러6"이 발생합니다.
   0: X축, 1: Y축, 2: Z축(XEC의 경우 Z축 지원안함)
5. MST_AXIS에는 위치동기의 주축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러6"이 발생합니다.
   0: X축, 1: Y축, 2: Z축(XEC의 경우 Z축 지원안함)

■ 프로그램 예

1. ST

```
INST_APM_SSP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, STEP:=STEP_UINT,
MST_AXIS:=AXIS_USINT, MST_ADDR:=ADDR_DINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SSS	적용 기종	발생플래그
속도 동기	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 MST_AXIS : 속도동기 주축 설정 0:X 축, 1:Y 축, 2:Z 축, 3:Encoder MST_RAT : 주축의 속도비 설정 1 ~ 65,535 SLV_RAT : 종축의 속도비 설정 1 ~ 65,535 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

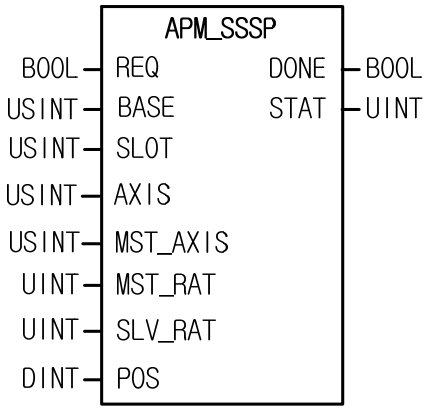
1. 이 명령은 위치결정 모듈에 속도 동기를 실행하는 명령입니다.
2. 두 축간에 운전 속도를 설정한 비율로 제어 하고자 할 때 사용합니다.
3. 속도 동기 운전 사용시 “종축의 속도비/주축의 속도비 ≤ 1” 가 되도록 설정해야 합니다.
4. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 속도 동기 지령을 내립니다.
5. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   0: X 축, 1: Y 축, 2: Z 축
6. MST_AXIS 에는 속도동기의 주축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   0: X 축, 1: Y 축, 2: Z 축, 3: Encoder

■ 프로그램 예

1. ST

```
INST_APM_SSS(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, MST_AXIS:=AXIS_USINT,
MST_RAT:=MST_UINT, SLV_RAT:=SLV_UINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SSSP	적용 기종	발생플래그
위치지정 속도 동기	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 MST_AXIS : 속도동기 주축 설정 0:X 축, 1:Y 축, 2:Z 축, 3:Encoder MST_RAT : 주축의 속도비 설정 1 ~ 65,535 SLV_RAT : 종축의 속도비 설정 1 ~ 65,535 POS : 속도동기 목표위치 -2,147,483,648 ~ 2,147,483,647 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 이 명령은 위치결정 모듈에 주축과 종축의 운전 속도를 설정한 비율로 운전하고자 할 때 실행하는 명령입니다.
2. 속도 동기 운전 사용시 “종축의 속도비/주축의 속도비 ≤ 1” 가 되도록 설정해야 합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS로 지정된 축에 속도 동기 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   0: X 축, 1: Y 축, 2: Z 축
5. MST_AXIS 에는 속도동기의 주축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   0: X 축, 1: Y 축, 2: Z 축, 3: Encoder
6. 주축의 운전을 실행하면 펑션 블록에서 설정한 속도 비율로 운전이 실행되고 종축이 속도동기 목표위치에 도달하면 정지합니다

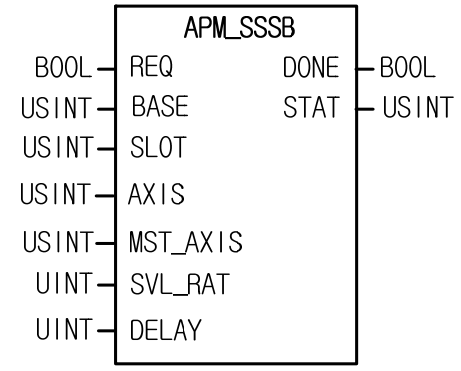
■ 프로그램 예

1. ST

```
INST_APM_SSSP(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), MST_AXIS:=(*USINT*),
MST_RAT:=(*UINT*), SLV_RAT:=(*UINT*), POS:=(*DINT*), DONE=>(*BOOL*), STAT=>(*UINT*))
```

APM_SSSB	적용 기종	발생플래그
속도 동기	XEC	-

펑션 블록



설 명

입력 REQ : 펑션 블록 실행 요구
BASE : 모듈이 장착된 베이스의 번호를 설정
SLOT : 모듈이 장착된 슬롯의 번호를 설정
AXIS : 지령을 내릴 축 지정
0:X 축, 1:Y 축
MST_AXIS : 속도동기 주축 설정

설정값	주축 설정	설정값	주축 설정
0	X축	5	고속카운터 Ch3
1	Y축	6	고속카운터 Ch4
2	고속카운터 Ch0	7	고속카운터 Ch5
3	고속카운터 Ch1	8	고속카운터 Ch6
4	고속카운터 Ch2	9	고속카운터 Ch7

SLV_RAT : 종축의 속도비 설정
1 ~ 10,000 (0.01 ~ 100.00%)
DELAY : 종축 지연시간 설정
1 ~ 10 (1 ~ 10ms)

출력 DONE : 최초 동작 후 1을 유지
STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 이 명령은 XGB 내장 위치결정에 속도 동기 기동을 실행시키는 명령입니다.
2. 입력조건의 상승 에지에서 AXIS 로 지정된 축이 종축, MST_AXIS 에 지정된 축이 주축이 되어 속도 동기 기동 명령이 실행됩니다.
3. 명령이 실행되면 종축은 실제 펄스를 출력하지 않고 있다가(이 때 종축의 운전중 상태 플래그(X 축:%KX6720, Y 축:%KX6880)은 On 됩니다.) 주축인 MST_AXIS 축이 기동되면 AXIS 에서 설정한 속도 동기 비율에 의해 기동됩니다.
4. SLV_RAT 에 설정 가능한 동기 비율은 0.01% ~ 100.00%(설정값 1 ~ 10,000)입니다. 만약 설정한 동기 속도 비율이 이 범위를 벗어나면 에러코드 356 이 발생됩니다.
5. DELAY 의 지연 시간은 종축의 속도가 현재의 주축 속도에 도달하기까지 걸리는 지연 시간을 의미합니다. XGB 내장 위치결정에서는 속도 동기 제어시 매 500 ㎲ 마다 현재의 주축의 속도를 검출하여 종축의 속도를 조정하는데 이 때 지연 시간 없이 종축의 속도를 현재의 주축 속도에 동기 시켜 즉시 변경하게 되면 종축 속도의 급변으로 모터의 무리와 충격음이 발생할 수 있습니다.
   예를 들어 동기속도 비율이 100.00%이고 지연시간이 5[ms]로 설정된 경우를 가정하면 현재 주축의 속도가 10,000[pps]인 경우 XGB 내장 위치 결정은 설정된 지연시간인 5[ms]후에 종축의 속도가 10,000[pps]에 도달하도록 현재의 속도를 조정하는 방법으로 매 500[㎲]마다 현재 주축의 속도에 따라 종축의 속도를 조정하게 됩니다.

지연시간이 클수록 주축과 종축간의 지연 시간은 커지지만 출력 펄스는 안정되게 출력됩니다. 모터의 탈조 등이 우려될 경우 지연 시간을 크게 하여 주시기 바랍니다.

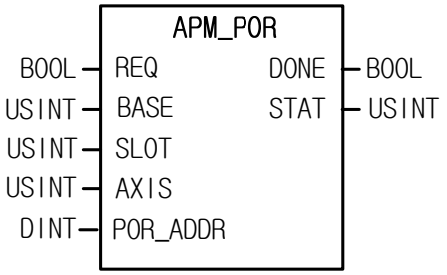
6. DELAY n2 에 설정 가능한 지연 시간은 1 ~ 10[ms]입니다. 설정 가능 범위를 벗어나는 경우는 에러코드 357 이 발생합니다.
7. MST_AXIS 의 주축 설정은 아래와 같이 0 ~ 9 까지 설정 가능합니다. 설정 가능 범위를 벗어나는 경우는 에러코드 355 가 발생합니다.
8. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
0: X 축, 1: Y 축
9. MST_AXIS 에는 속도동기의 주축을 설정하며 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.

■ 프로그램 예

1. ST

```
INST_APM_SSSP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, MST_AXIS:=AXIS_USINT,
MST_RAT:=MST_UINT, SLV_RAT:=SLV_UINT, POS:=POS_DINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_POR	적용 기종	발생플래그
위치 오버라이드	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 POR_ADDR : 새로운 목표위치 설정 -2,147,483,648 ~ 2,147,483,647 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

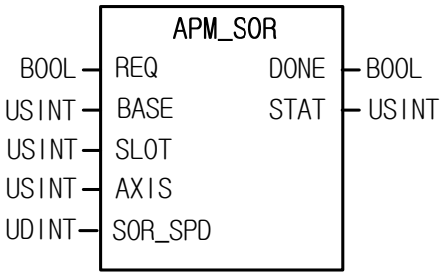
1. 이 명령은 위치결정 모듈에 위치 오버라이드 실행하는 명령입니다.
2. 지령 축이 운전 중인 상태에서 목표 위치를 변경하고자 할 때 사용합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 위치 오버라이드 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   0: X 축, 1: Y 축, 2: Z 축(XEC 의 경우 Z 축 지원안함)
5. POR_ADDR 에는 변경할 목표 위치를 설정합니다.

■ 프로그램 예

1. ST

```
INST_APM_POR(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, POR_ADDR:=POR_DINT,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SOR	적용 기종	발생플래그
속도 오버라이드	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 SOR_SPD : 새로운 운전 속도값 설정 Open Collector : 0 ~ 200,000[pps] Line Driver : 0 ~ 1,000,000[pps] **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

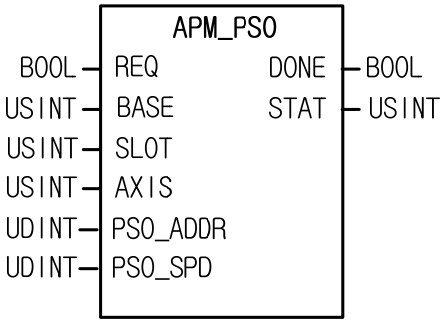
1. 이 명령은 위치결정 모듈에 속도 오버라이드를 실행하는 명령입니다.
2. 지령 축이 운전 중인 상태에서 운전 속도를 변경하여 사용하고자 할 때 사용합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 속도 오버라이드 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   0: X 축, 1: Y 축, 2: Z 축(XEC 의 경우 Z 축 지원안함)
5. SOR_SPD 에는 변경할 목표 속도를 설정합니다. 설정 범위는 아래와 같으며 설정 범위를 벗어나는 값을 설정했을 경우 “에러 11” 이 발생합니다.
   Open Collector: 0 ~ 200,000[pps] (XEC 의 경우 0 ~ 100,000[pps])
   Line Driver: 0 ~ 1,000,000[pps]

■ 프로그램 예

1. ST

```
INST_APM_SOR(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, SOR_SPD:=SOR_UDINT,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_PSO	적용 기종	발생플래그
위치 지정 속도 오버라이드	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 PSO_ADDR : 속도 변경을 수행할 위치 -2,147,483,648 ~ 2,147,483,647 PSO_SPD : 새로운 운전 속도값 설정 Open Collector : 0 ~ 200,000[pps] Line Driver : 0 ~ 1,000,000[pps] **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

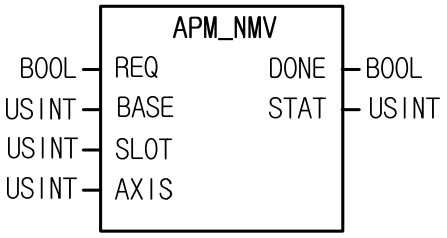
1. 이 명령은 위치결정 모듈에 위치 지정 속도 오버라이드 지령을 내리는 명령입니다.
2. 지령 축이 운전 중인 상태에서 일정 위치에 도달한 후 운전 속도를 변경하여 사용하고자 할 때 사용합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 속도 오버라이드 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   0: X 축, 1: Y 축, 2: Z 축(XEC 의 경우 Z 축 지원안함)
5. PSO_SPD 에는 변경할 목표 속도를 설정합니다. 설정 범위는 아래와 같으며 설정 범위를 벗어나는 값을 설정했을 경우 “에러 11” 이 발생합니다.
   Open Collector: 0 ~ 200,000[pps] (XEC 의 경우 0 ~ 100,000[pps])
   Line Driver: 0 ~ 1,000,000[pps]

■ 프로그램 예

1. ST

```
INST_APM_PSO(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, PSO_ADDR:=ADDR_UDINT,
PSO_SPD:=SPD_UDINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_NMV	적용 기종	발생플래그
연속 운전	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

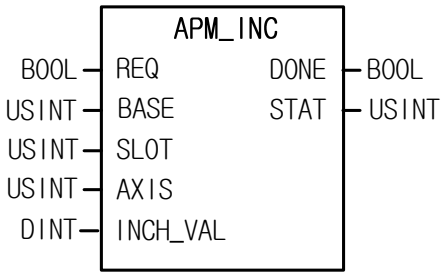
1. 이 명령은 위치결정 모듈에 연속 운전 지령을 실행하는 명령입니다.
2. 지령축이 현재 운전중인 스텝에서 정지하지 않고 다음 스텝으로 운전을 변경하고자 할 때 사용합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 연속 운전 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   0: X 축, 1: Y 축, 2: Z 축

■ 프로그램 예

1. ST

```
INST_APM_NMV(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, DONE=>DONE_BOOL,
STAT=>STAT_UINT);
```

APM_INC	적용 기종	발생플래그
인칭 운전	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 INCH_VAL: 인칭운전으로 이동하고자 하는 이동량 설정 -2,147,483,648 ~ 2,147,483,647 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

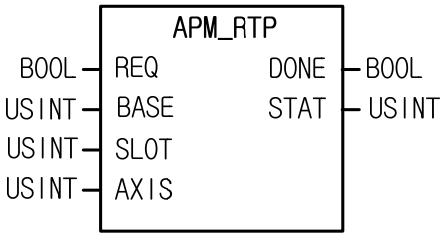
1. 이 명령은 위치결정 모듈에 인칭 운전을 실행하는 명령입니다.
2. 인칭 운전은 수동 운전의 일종으로 미세한 움직임을 정량적인 운전으로 처리할 할 경우 사용합니다.
3. 인칭 운전의 속도는 수동 운전 파라미터에서 설정합니다.
4. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 인칭 운전 지령을 내립니다.
5. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   0: X 축, 1: Y 축, 2: Z 축(XEC 의 경우 Z 축 지원안함)

■ 프로그램 예

1. ST

```
INST_APM_INC(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, INCH_VAL:=INCH_DINT,
DONE=>DNOE_BOOL, STAT=>STAT_UINT);
```

APM_RTP	적용 기종	발생플래그
수동 운전 이전 위치로 복귀	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

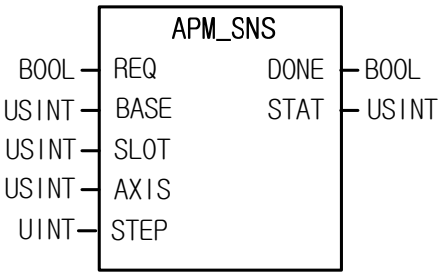
1. 이 명령은 위치결정 모듈에 수동 운전을 하기 이전 위치로 복귀를 실행하는 명령입니다.
2. 위치 결정 후 수동 운전에 의해 위치가 변경되었을 때 수동 운전 이전의 위치로 되돌리고자 할 때 사용하는 지령입니다.
3. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 수동운전 이전위치로 복귀 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   0: X 축, 1: Y 축, 2: Z 축

■ 프로그램 예

1. ST

```
INST_APM_RTP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, DONE=>DONE_BOOL,
STAT=>STAT_UINT);
```

APM_SNS	적용 기종	발생플래그
기동 스텝 번호 변경	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 STEP : 운전하고자 하는 운전스텝번호 설정 1 ~ 400 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

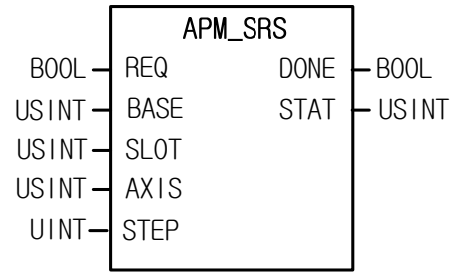
1. 이 명령은 위치결정 모듈에 기동 스텝 변경을 실행하는 명령입니다.
2. 지령 축의 운전 스텝을 변경하고자 할 때 사용합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 기동 스텝 변경 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   0: X축, 1: Y축, 2: Z축(XEC의 경우 Z축 지원안함)
5. STEP 에는 운전하고자 하는 스텝 번호를 설정합니다. 설정값의 범위는 1 ~ 400 이며 설정값 이외의 값을 설정했을 경우 “에러 11” 이 발생합니다.

■ 프로그램 예

1. ST

```
INST_APM_SNS(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, STEP:=STEP_UINT,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SRS	적용 기종	발생플래그
반복 스텝 번호 변경	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 STEP : 변경하고자 하는 반복스텝번호 설정 1 ~ 400 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

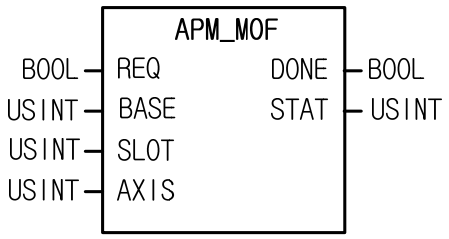
1. 이 명령은 위치결정 모듈에 반복 스텝 변경 지령을 내리는 명령입니다.
2. 운전 데이터로 운전하던 중 반복 운전을 만나면 반복 운전 스텝으로 되돌아 가는 반복 운전시 반복 운전의 시작 스텝 번호를 지정하여 특정 운전 스텝에서 운전을 시작하고자 할 때 사용합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 반복 스텝 변경 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   0: X축, 1: Y축, 2: Z축
5. STEP 에는 반복 운전을 시작하고자 하는 스텝 번호를 설정합니다. 설정값의 범위는 1 ~ 400 이며 설정값 이외의 값을 설정했을 경우 “에러 11” 이 발생합니다.

■ 프로그램 예

1. ST

```
INST_APM_SRS(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, STEP:=STEP_UINT,
DONE=>DONE_BOOL,
STAT=>STAT_UINT);
```

APM_MOF	적용 기종	발생플래그
M 코드 해제	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

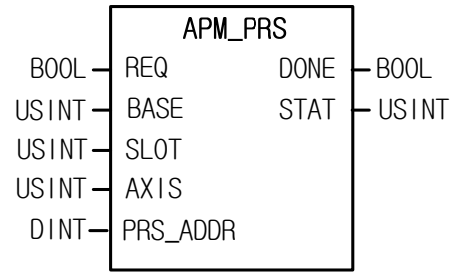
1. 이 명령은 위치결정 모듈에 M 코드 해제를 실행하는 명령입니다.
2. 각 축의 파라미터에서 M 코드를 With 나 After 모드로 설정한 경우, 지령축의 M 코드 신호가 On 되었을 때 이 신호를 Off 하고자 할 때 사용할 수 있습니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 M 코드 해제 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   0: X축, 1: Y축, 2: Z축(XEC의 경우 Z축 지원안함)

■ 프로그램 예

1. ST

```
INST_APM_MOF(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, DONE=>DONE_BOOL,
STAT=>STAT_UINT);
```

APM_PRS	적용 기종	발생플래그
현재 위치 프리셋	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 PRS_ADDR : 변경하고자 하는 현재 위치값 설정. -2,147,483,648 ~ 2,147,483,647 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

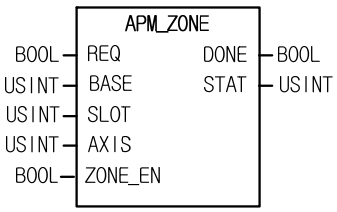
1. 이 명령은 위치결정 모듈에 현재 위치 프리셋을 실행하는 명령입니다.
2. 지령축의 현재 위치를 임의의 위치로 변경하고자 할 때 사용되는 지령으로 이를 실행하면 원점이 재결정 됩니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 현재 위치 프리셋 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   0: X축, 1: Y축, 2: Z축(XEC의 경우 Z축 지원안함)

■ 프로그램 예

1. ST

```
INST_APM_PRS(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, PRS_ADDR:=ADDR_DINT,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_ZONE	적용 기종	발생플래그
Zone 출력 허용/금지	XGI, XGR	-
펑션 블록	설 명	



입력 REQ : 펑션 블록 실행 요구
BASE : 모듈이 장착된 베이스의 번호를 설정
SLOT : 모듈이 장착된 슬롯의 번호를 설정
AXIS : 지령을 내릴 축 지정
0:X 축, 1:Y 축, 2:Z 축
ZONE_EN : Zone 출력 허가/금지 설정
0:금지, 1:허가

출력 DONE : 최초 동작 후 1을 유지
STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 이 명령은 위치결정 모듈에 Zone 출력 허용/금지 지령을 실행하는 명령입니다.
2. 공통 파라미터에서 설정한 Zone 에 대해 지령축의 위치 데이터와 Zone1, Zone2, Zone3 에서 설정한 위치 데이터 값을 이용하여 Zone 출력을 허가 또는 금지하는 지령입니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 Zone 출력 허용/금지 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.

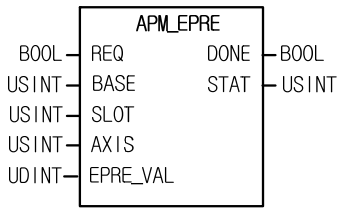
0: X축, 1: Y축, 2: Z축

■ 프로그램 예

1. ST

```
INST_APM_ZONE(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, ZONE_EN:=ZONE_BOOL,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_EPRE	적용 기종	발생플래그
엔코더 프리셋	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 EPRE_VAL : 엔코더 프리셋 값 설정 0 ~ 4,294,967,295 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 이 명령은 위치결정 모듈에 엔코더 프리셋을 실행하는 명령입니다.
2. EPRE_VAL 에 설정된 값으로 엔코더의 현재값을 프리셋하는 지령입니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 엔코더 프리셋 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.

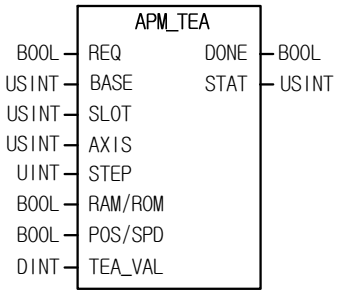
0: X축, 1: Y축, 2: Z축

■ 프로그램 예

1. ST

```
INST_APM_EPRE(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, EPRE_VAL:=EPRE_UDINT,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_TEA	적용 기종	발생플래그
단독 티칭	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 STEP : 티칭할 스텝번호 설정 0 ~ 400 RAM/ROM : RAM 티칭과 ROM 티칭 종류 선택 0 : RAM 티칭, 1 : ROM 티칭 POS/SPD : 위치 티칭과 속도 티칭 종류 선택 0 : 위치 티칭, 1 : 속도 티칭 TEA_VAL : 티칭값 설정 위치 티칭 : -2,147,483,648 ~ 2,147,483,647 속도 티칭 : Open Collector 0 ~ 200,000[pps] Line Driver 0 ~ 1,000,000[pps] **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

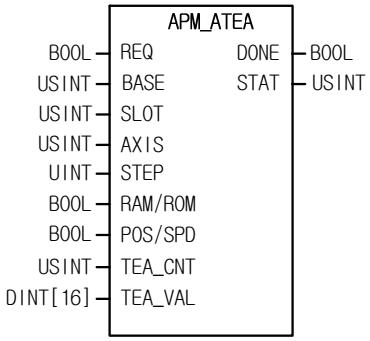
1. 이 명령은 위치결정 모듈에 단독 티칭을 실행하는 명령입니다.
2. 속도 티칭은 사용자가 임의의 속도 값을 특정 스텝의 운전 데이터에 사용하고자 할 때, 위치 티칭은 사용자가 임의의 위치값을 특정 운전 스텝의 운전 데이터에 설정하고자 할 때 사용할 수 있습니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 단독 티칭 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러 6" 이 발생합니다.
   0: X 축, 1: Y 축, 2: Z 축
5. STEP 에는 티칭할 운전 데이터의 스텝번호를 설정하며 0 ~ 400 사이의 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러 11" 이 발생합니다.
6. TEA_VAL 에는 위치 티칭일 경우 티칭할 위치값, 속도 티칭일 경우 티칭할 속도값을 설정하며 설정범위는 다음과 같습니다. 설정값 이외의 값을 설정했을 경우 "에러 11" 이 발생합니다.
   - 위치 티칭 범위: -2,147,483,648 ~ 2,147,483,647
   - 속도 티칭 범위: Open Collector 출력 -> 0 ~ 200,000 [pps]
     Line Driver 출력 -> 0 ~ 1,000,000 [pps]

■ 프로그램 예

1. ST

```
INST_APM_TEA(REQ:=REQ_BOOL,  BASE:=BASE_USINT,  SLOT:=SLOT_USINT,  AXIS:=AXIS_USINT,  STEP:=STEP_UINT,
RAM_ROM:=RAM_BOOL, POS_SPD:=SPD_BOOL);
```

APM_ATEA	적용 기종	발생플래그
복수 티칭	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0 : X 축, 1:Y 축, 2:Z 축 STEP : 티칭할 스텝번호 설정, 0 ~ 400 RAM/ROM : RAM 티칭과 ROM 티칭 종류 선택 0 : RAM 티칭, 1 : ROM 티칭 POS/SPD : 위치 티칭과 속도 티칭 종류 선택 0 : 위치 티칭, 1 : 속도 티칭 TEA_CNT : 티칭할 데이터의 개수 설정, 1 ~ 16 TEA_VAL : 티칭값 설정 위치 티칭 : -2,147,483,648 ~ 2,147,483,647 속도 티칭 : Open Collector 0 ~ 200,000[pps] Line Driver 0 ~ 1,000,000[pps] **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

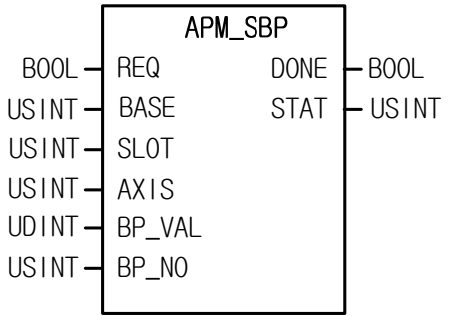
1. 이 명령은 위치결정 모듈에 복수 티칭을 실행하는 명령입니다.
2. 속도 티칭은 사용자가 임의의 속도 값을 특정 스텝의 운전데이터에 사용하고자 할 때, 위치 티칭은 사용자가 임의의 위치값을 특정 운전 스텝의 운전 데이터에 설정하고자 할 때 사용할 수 있습니다.
3. 티칭 복수형 펑션 블록을 이용하여 한번에 최대 16 개까지의 목표 위치와 속도값을 변경할 때 사용합니다.
4. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 복수 티칭 지령을 내립니다.
5. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다. 0: X 축, 1: Y 축, 2: Z 축
6. STEP 에는 티칭을 할 운전 데이터의 스텝 번호를 설정하며 0 ~ 400 사이의 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 11” 이 발생합니다.
7. TEA_CNT 에는 티칭할 데이터 개수를 설정하며 최대 16 개까지 티칭을 할 수 있습니다. 설정 범위 이외의 값을 설정했을 경우 “에러 11” 이 발생합니다.
8. TEA_VAL 에는 위치 티칭일 경우 티칭할 위치값, 속도 티칭일 경우 티칭할 속도값을 설정하며 설정범위는 다음과 같습니다. 설정값 이외의 값을 설정했을 경우 “에러 11” 이 발생합니다.
   - 위치 티칭 범위: -2,147,483,648 ~ 2,147,483,647
   - 속도 티칭 범위: Open Collector 출력 -> 0 ~ 200,000 [pps], Line Driver 출력 -> 0 ~ 1,000,000 [pps]

■ 프로그램 예

1. ST

```
INST_APM_ATEA1(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, STEP:=STEP_UINT,
RAM_ROM:=RAM_BOOL, POS_SPD:=SPD_BOOL, TEA_CNT:=CNT_USINT, ATEA_VAL:=ARY_ATEA, DONE=>DONE_BOOL,
  STAT=>STAT_UINT);
```

APM_SBP	적용 기종	발생플래그
기본 파라미터 설정	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 BP_VAL: 변경할 기본 파라미터 값 BP_NO : 변경할 기본 파라미터 항목번호 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

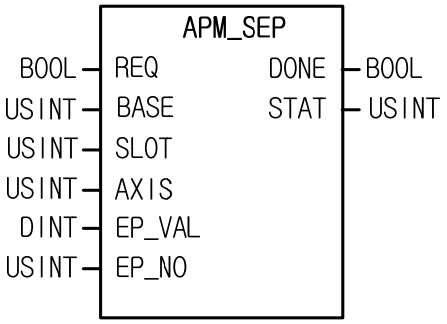
1. 이 명령은 위치결정 모듈에 기본 파라미터 티칭을 실행하는 명령입니다.
2. 기본 파라미터 설정 명령으로 수정한 파라미터 값은 전원이 켜져 있는 동안에만 유효합니다. 기본 파라미터 설정 명령으로 수정한 파라미터 값을 ROM 에 저장하려면 기본 파라미터 설정 후 파라미터/운전데이터 저장 명령(WRT)을 사용하여 수정한 파라미터 값을 ROM 에 저장하여야 합니다.
3. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 기본 파라미터 설정 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   0: X 축, 1: Y 축, 2: Z 축
5. 기본 파라미터 항목 번호에 설정할 값은 아래와 같습니다.
   1: 속도 제한치
   2: 바이어스 속도
   3: 가감속 시간 1
   4: 가감속 시간 2
   5: 가감속 시간 3
   6: 가감속 시간 4
   7: 1 회전당 펄스 수
   8: 1 회전당 이송거리
   9: 펄스 출력 모드
   10: 단위
   11: 단위 배정도

■ 프로그램 예

1. ST

```
INST_APM_SBP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, BP_NO:=EP_USINT*),
BP_VAL:=BP_UDINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SEP	적용 기종	발생플래그
확장 파라미터 티칭	XGI, XGR	-
펑션 블록	설 명	



입력 REQ : 펑션 블록 실행 요구
BASE : 모듈이 장착된 베이스의 번호를 설정
SLOT : 모듈이 장착된 슬롯의 번호를 설정
AXIS : 지령을 내릴 축 지정
0:X 축, 1:Y 축, 2:Z 축
EP_VAL : 변경할 확장 파라미터 값
EP_NO : 변경할 확장 파라미터 항목번호

출력 DONE : 최초 동작 후 1을 유지
STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 이 명령은 위치결정 모듈에 확장 파라미터 티칭을 실행하는 명령입니다.
2. 확장 파라미터 설정 명령으로 수정한 파라미터 값은 전원이 켜져 있는 동안에만 유효합니다. 확장 파라미터 설정 명령으로 수정한 파라미터 값을 ROM 에 저장하려면 확장 파라미터 설정 후 파라미터/운전데이터 저장 명령(WRT)을 사용하여 수정한 파라미터 값을 ROM 에 저장하여야 합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 확장 파라미터 설정 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다. 0: X 축, 1: Y 축, 2: Z 축
5. 확장 파라미터 항목 번호에 설정할 값은 아래와 같습니다.

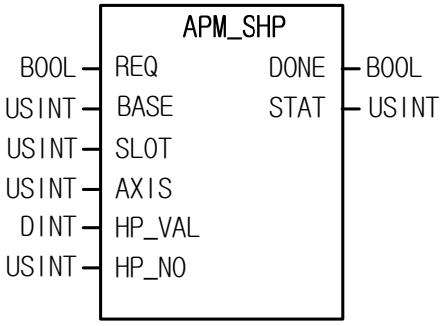
1: 소프트웨어 상한	2: 소프트웨어 하한
3: 백래쉬 보정량	4: 위치결정 완료 출력 시간
5: S-Curve 비율	6: 외부 명령 선택
7: 펄스 출력 방향	8: 가감속 패턴
9: M 코드 번호	10: 등속 운전 중 위치표시
11: 등속 운전 중 상하한 표시	12: 외부 속도/위치 제어 전환 허용
13: 외부 명령 허용	14: 외부 정지 허용
15: 외부 동시 기동 허용	16: 위치결정 완료 조건
17: 드라이버 레디/인포지션	

■ 프로그램 예

1. ST

```
INST_APM_SEP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, EP_NO:=NO_USINT,
EP_VAL:=EP_DINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SHP	적용 기종	발생플래그
원점 복귀 파라미터 설정	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 HP_VAL : 변경할 원점복귀 파라미터 값 HP_NO : 변경할 원점복귀 파라미터 항목번호 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

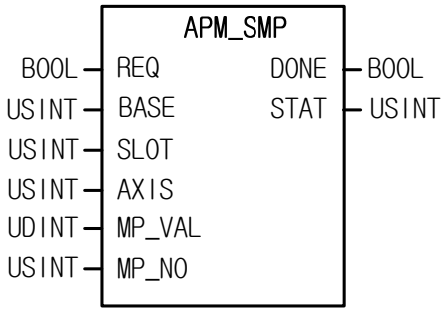
1. 이 명령은 위치결정 모듈에 원점 복귀 파라미터 티칭을 실행하는 명령입니다.
2. 원점 복귀 파라미터 설정 명령으로 수정한 파라미터 값은 전원이 켜져 있는 동안에만 유효합니다. 원점 복귀 파라미터 티칭 명령으로 수정한 파라미터 값을 ROM 에 저장하려면 원점 복귀 파라미터 설정 후 파라미터/운전 데이터 저장 명령(WRT)을 사용하여 수정한 파라미터 값을 ROM 에 저장하여야 합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 원점 복귀 파라미터 티칭 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   0: X 축, 1: Y 축, 2: Z 축
5. 원점 복귀 파라미터 항목 번호에 설정할 값은 아래와 같습니다.
   1: 원점 어드레스
   2: 원점 복귀 고속 속도
   3: 원점 복귀 저속 속도
   4: 원점 복귀 가감속 시간
   5: 원점 복귀 드웰 시간
   6: 원점 보정량
   7: 원점 복귀 재기동 시간
   8: 원점 복귀 방법
   9: 원점 복귀 방향

■ 프로그램 예

1. ST

```
INST_APM_SHP(REQ:=REQ_BOOL,  BASE:=BASE_USINT,  SLOT:=SLOT_USINT,  AXIS:=AXIS_USINT,  HP_NO:=NO_USINT,
HP_VAL:=HP_DINT,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SMP	적용 기종	발생플래그
수동 운전 파라미터 티칭	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 MP_VAL: 변경할 수동운전 파라미터 값 MP_NO : 변경할 수동운전 파라미터 항목번호 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

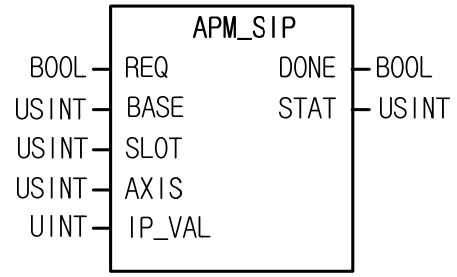
1. 이 명령은 위치결정 모듈에 수동 운전 파라미터 티칭을 실행하는 명령입니다.
2. 수동 운전 파라미터 설정 명령으로 수정한 파라미터 값은 전원이 켜져 있는 동안에만 유효합니다. 수동 운전 파라미터 티칭 명령으로 수정한 파라미터 값을 ROM 에 저장하려면 수동 운전 파라미터 설정 후 파라미터/운전 데이터 저장 명령(WRT)을 사용하여 수정한 파라미터 값을 ROM 에 저장하여야 합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 수동운전 파라미터 티칭 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   0: X 축, 1: Y 축, 2: Z 축
5. 수동 운전 파라미터 항목번호에 설정할 값은 아래와 같습니다.
   1: 조그 고속 속도
   2: 조그 저속 속도
   3: 조그 가감속 시간
   4: 인칭 속도

■ 프로그램 예

1. ST

```
INST_APM_SMP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, MP_NO:=NO_USINT,
MP_VAL:=MP_UDINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SIP	적용 기종	발생플래그
입력 신호 파라미터 티칭	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 IP_VAL : 변경할 외부신호 파라미터 값 각 Bit 별로 할당된 신호를 설정함. **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 이 명령은 위치결정 모듈에 입력 신호 파라미터 티칭을 실행하는 명령입니다.
2. 입력 신호 파라미터 티칭 명령으로 수정한 파라미터 값은 전원이 켜져 있는 동안에만 유효합니다. 입력 신호 파라미터 설정 명령으로 수정한 파라미터 값을 ROM 에 저장하려면 외부신호 파라미터 설정 후 파라미터/운전 데이터 저장 명령(WRT)을 사용하여 수정한 파라미터 값을 ROM 에 저장하여야 합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 외부신호 파라미터 티칭 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   0: X 축, 1: Y 축, 2: Z 축
   각 입력 신호 설정 영역의 설정값은 아래와 같은 의미를 가진다.
   0: A접점, 1: B접점
   변경할 입력 신호 파라미터 값의 각 Bit에 할당된 신호는 아래와 같습니다.

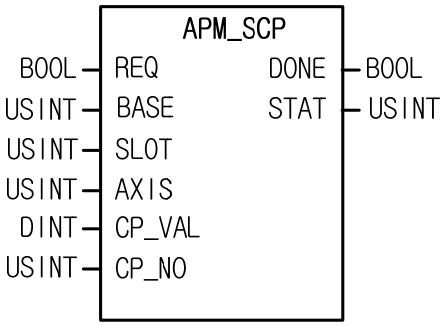
Bit	입력 신호	Bit	입력 신호
0	상한 신호	6	명령 신호
1	하한 신호	7	보조명령 신호
2	근사 원점 신호	8	속도/위치 전환 신호
3	원점 신호	9	드라이버 레디/인포지션 신호
4	비상 정지 신호	10	외부 동시 기동 신호
5	감속 정지 신호	15 ~ 11	-

■ 프로그램 예

1. ST

```
INST_APM_SIP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, IP_VAL:=IP_WORD,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SCP	적용 기종	발생플래그
공통 파라미터 티칭	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 CP_VAL : 변경할 공통 파라미터 값 CP_NO : 변경할 공통 파라미터 항목번호 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 이 명령은 위치결정 모듈에 공통 파라미터 티칭을 실행하는 명령입니다.
2. 공통 파라미터 설정 명령으로 수정한 파라미터 값은 전원이 켜져 있는 동안에만 유효합니다. 공통 파라미터 설정 명령으로 수정한 파라미터 값을 ROM 에 저장하려면 공통 파라미터 티칭 후 파라미터/운전데이터 저장 명령(WRT)을 사용하여 수정한 파라미터 값을 ROM 에 저장하여야 합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 공통 파라미터 설정 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다. 0: X 축, 1: Y 축, 2: Z 축

공통 파라미터 항목 번호에 설정할 값은 아래와 같습니다.

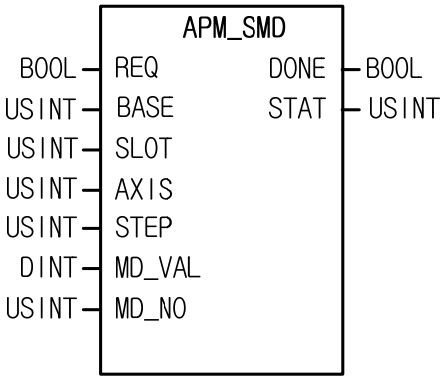
1: 펄스 출력 레벨	2: 원호 보간 방식
3: 엔코더 입력모드	4: 엔코더 Auto Reload값
5: ZONE 출력 모드	6: ZONE1 축 설정
7: ZONE2 축 설정	8: ZONE3 축 설정
9: ZONE1 On영역	10: ZONE1 Off영역
11: ZONE2 On영역	12: ZONE2 Off영역
13: ZONE3 On영역	14: ZONE3 Off영역

■ 프로그램 예

1. ST

```
INST_APM_SCP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, CP_NO:=NO_USINT,
CP_VAL:=CP_DINT, ENC_LD:=ENC_UDINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_SMD	적용 기종	발생플래그
운전 데이터 티칭	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 STEP : 변경할 운전스텝 번호 0 ~ 400 MD_VAL : 변경할 운전데이터 값 MD_NO : 변경할 운전데이터 항목번호 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 이 명령은 위치결정 모듈에 운전 데이터 티칭을 실행하는 명령입니다.
2. 운전 데이터 티칭 명령으로 수정한 파라미터 값은 전원이 켜져 있는 동안에만 유효합니다. 운전데이터 설정 명령으로 수정한 파라미터 값을 ROM 에 저장하려면 운전데이터 티칭 후 파라미터/운전 데이터 저장 명령(WRT)을 사용하여 수정한 파라미터 값을 ROM 에 저장하여야 합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 운전 데이터 티칭 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다. 0: X 축, 1: Y 축, 2: Z 축

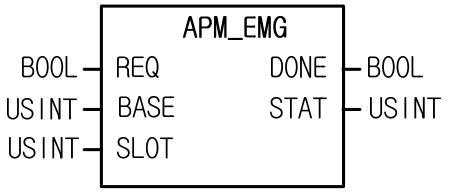
운전 데이터 항목번호에 설정할 값은 아래와 같습니다.

1: 목표 위치	2: 원호 보간 보조점
3: 목표 속도	4: 드웰 시간
5: M 코드	6: 제어 방식
7: 운전 방식	8: 운전 패턴
9: 좌표	10: 가감속 번호
11: 원호 보간 방향	

■ 프로그램 예

1. ST

```
INST_APM_SMD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, STEP:=STEP_UINT,
MD_NO:=NO_USINT, MD_VAL:=MD_DINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_EMG	적용 기종	발생플래그
비상 정지	XGI, XGR, XEC	-
펑션 블록	설 명	
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력	

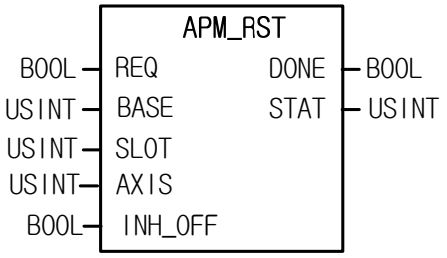
■ 기능

1. 이 명령은 위치결정 모듈에 비상 정지를 실행하는 명령입니다.
2. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈에 비상정지 지령을 내립니다.
3. 긴급 상황으로 운전을 즉시 정지시키고자 할 때 사용되며, 이 지령이 실행되는 모든 축은 정지상태가 됩니다.
4. 다시 기동 시키고자 할 때는 출력 금지 및 원점 미결정 상태로 전환 되었으므로 출력 금지를 해제하고 원점을 재결정하여 기동해야 합니다.

■ 프로그램 예

1. ST

```
INST_APM_EMG(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_RST	적용 기종	발생플래그
에러 리셋/출력 금지 해제	XGI, XGR, XEC	-
펑션 블록	설 명	
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 INH_OFF : 출력금지 해제 0: 에러 리셋, 1: 에러 리셋/출력금지해제 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력	

■ 기능

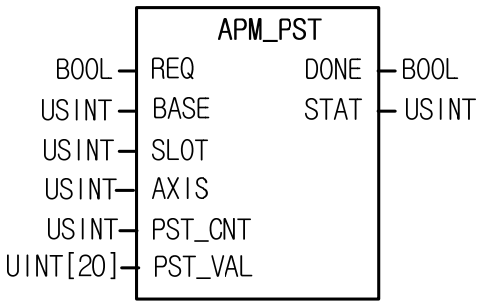
1. 이 위치결정 모듈에 에러 리셋/출력 금지 해제를 실행하는 명령입니다.
2. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 에 지정된 축에 에러 리셋/출력 금지 해제 지령을 내립니다.
3. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   0: X축, 1: Y축, 2: Z축(XEC의 경우 Z축 지원안함)
4. 외부 비상 정지, 상/하한 검출 등에 의해 펄스 출력이 금지되어 있는 상태를 해제하거나 또는 파라미터의 설정 범위 초과나 운전 중 에러가 발생하였을 때 발생한 에러를 리셋(Reset)하는데 사용합니다.

■ 프로그램 예

1. ST

```
INST_APM_RST(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, INH_OFF:=INH_BOOL,
DONE=>DOONE_BOOL, STAT=>STAT_UINT);
```

APM_PST	적용 기종	발생플래그
포인트 운전	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 PST_CMT : 포인트 운전 스텝 수 설정 0 ~ 19 PST_VAL : 포인트 운전 스텝 번호 설정 0 ~ 400 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

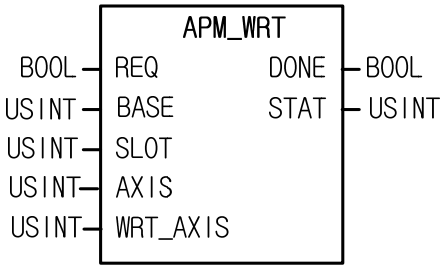
1. 이 명령은 위치결정 모듈에 포인트 운전을 실행하는 명령입니다.
2. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 에 지정된 축에 포인트 운전 기동 지령을 내립니다.
3. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   0: X축, 1: Y축, 2: Z축
4. PTP(Point to Point) 운전시 최대 20 개의 운전 스텝을 설정하여 한번의 지령으로 정지 없이 연속으로 운전할 때 사용합니다.
5. PST_CNT 나 PST_VAL 에 설정값 이외의 값을 설정하면 “에러 6” 이 발생합니다.

■ 프로그램 예

1. ST

```
INST_APM_PST(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, PST_CNT:=CNT_USINT,
PST_VAL:=ARY_PST, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_WRT	적용 기종	발생플래그
파라미터/운전 데이터 저장	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 WRT_AXIS : 저장축 설정(각 bit 를 set 하여 설정) 0bit:X 축, 1bit:Y 축, 2bit:Z 축 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

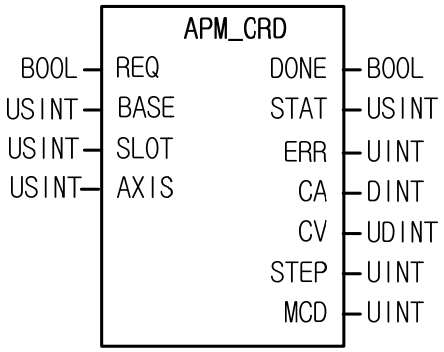
1. 이 명령은 위치결정 모듈에 파라미터/운전 데이터 저장을 실행하는 명령입니다.
2. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 에 지정된 축에 파라미터/운전데이터 저장 지령을 내립니다.
3. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   0: X축, 1: Y축, 2: Z축(XEC의 경우 Z축 지원안함)
4. 지령축에 WRT_AXIS 에 설정된 축의 현재 운전 중인 파라미터와 운전 데이터를 Flash ROM 에 저장하는 지령을 내립니다.

■ 프로그램 예

1. ST

```
INST_APM_WRT(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, WRT_AXIS:=WRT_USINT,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_CRD	적용 기종	발생플래그
운전 정보 읽기	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력 ERR : 동작 중 에러 표시 CA : 현재 위치 어드레스 표시 CV : 현재 운전 속도 표시 STEP : 현재 운전데이터 스텝번호 표시 MCD : 현재 MCode 값 표시

■ 기능

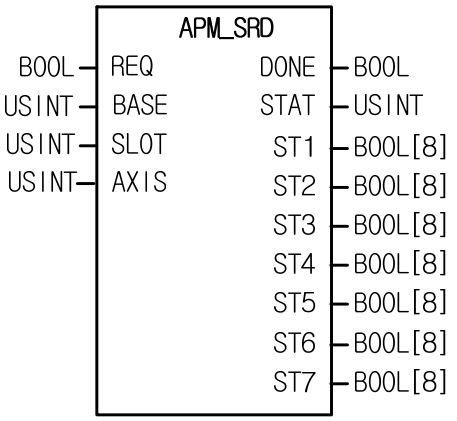
1. 이 명령은 위치결정 모듈의 현재 운전 정보 읽기를 실행하는 명령입니다.
2. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 에 지정된 축에 현재 운전 정보 읽기 지령을 내립니다.
3. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   0: X축, 1: Y축, 2: Z축
4. 설정된 축의 현재 위치 어드레스, 운전 속도, 운전 데이터 번호, MCode 값을 읽어 내어 모니터링 하거나 사용자 프로그램에서 조건으로 이용할 수 있습니다.

■ 프로그램 예

1. ST

```
NST_APM_CRD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, DONE=>DONE_BOOL,
STAT=>STAT_UINT, ERR=>ERR_UINT, CA=>CA_DINT, CV=>CV_UDINT, STEP=>STEP_UINT, MCD=>MCD_UINT);
```

APM_SRD	적용 기종	발생플래그
운전 상태 읽기	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력 ST1 : 상태 1 ST2 : 상태 2 ST3 : 상태 3 ST4 : 상태 4 ST5 : 상태 5 ST6 : 상태 6 ST7 : 상태 7

■ 기능

1. 이 명령은 위치결정 모듈의 현재 운전 상태 읽기를 실행하는 명령입니다.
2. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 에 지정된 축에 현재 운전 상태 읽기 지령을 내립니다.
3. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러 6" 이 발생합니다.
   0: X 축, 1: Y 축, 2: Z 축
4. 현재 운전 상태 비트 읽기 펑션 블록의 출력 변수 ST1 ~ ST7 의 내용은 프로그램에서 반드시 응용해야 하는 중요한 정보들입니다.
5. ST1 ~ ST4 의 각 Bit 가 의미하는 것은 아래와 같습니다.

	Bit	설명	Bit	설명
ST1	[0]	운전중(0: 정지, 1: BUSY)	[4]	원점 결정 상태 (0: 미결정, 1: 완료)
	[1]	Error 상태	[5]	펄스 출력 금지상태 (0: 가능, 1: 금지)
	[2]	위치결정 완료	[6]	정지 상태
	[3]	MCode On신호 (0: Off, 1: On)	[7]	-
ST2	[0]	상한 검출	[4]	가속 중

	Bit	설명	Bit	설명
	[1]	하한 검출	[5]	정속 중
	[2]	비상 정지 상태	[6]	감속 중
	[3]	방향 (0: 정방향, 1: 역방향)	[7]	드웰 중
ST3	[0]	1축 위치 제어 중	[4]	2축 원호 보간 중
	[1]	1축 속도 제어 중	[5]	원점 복귀 운전 중
	[2]	2축 직선 보간 중	[6]	위치 동기 운전 중
	[3]	3축 직선 보간 중	[7]	속도 동기 운전 중
ST4	[0]	조그 저속 운전 중	[4]	수동 운전 이전 위치로 복귀 중
	[1]	조그 고속 운전 중	[5]	-
	[2]	인칭 운전 중	[6]	-
	[3]	MPG 운전 중	[7]	-

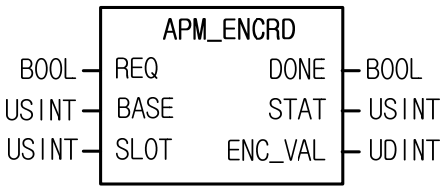
6. ST5 ~ ST7 의 각 Bit 가 의미하는 것은 아래와 같습니다.

	Bit	설명	Bit	설명
ST5	[0]	축상태(0: 종축, 1: 주축)	[4]	주축정보[Encoder]
	[1]	주축 정보(X축)	[5]	-
	[2]	주축 정보(Y축)	[6]	-
	[3]	주축 정보(Z축)	[7]	-
ST6	[0]	비상 정지 신호	[4]	상한 신호
	[1]	외부 정지 신호	[5]	하한 신호
	[2]	외부 지령 신호	[6]	원점 신호
	[3]	조그 고속 역방향 신호	[7]	근사 원점 신호
ST7	[0]	속도/위치 제어 전환 신호	[4]	-
	[1]	드라이버 레디/인포지션 신호	[5]	-
	[2]	외부 동시 기동 신호	[6]	-
	[3]	-	[7]	-

■ 프로그램 예

1. ST

```
INST_APM_SRD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, DONE=>DONE_BOOL, STAT=>STAT_UINT, ST1=>ARY_ST1, ST2=>ARY_ST2, ST3=> ARY_ST3, ST4=> ARY_ST4, ST5=> ARY_ST5, ST6=> ARY_ST6, ST7=> ARY_ST7);
```

APM_ENCRD	적용 기종	발생플래그
엔코더값 읽기	XGI, XGR	-
펑션 블록	설 명	
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력 ENC_VAL : 엔코더의 현재값	

■ 기능

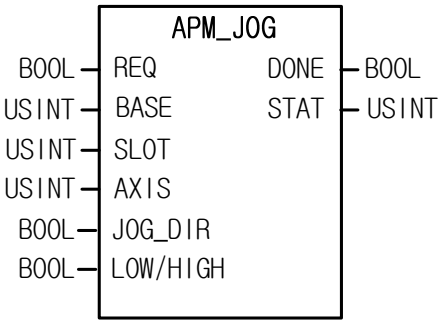
1. 이 명령은 위치결정 모듈에 엔코더값 읽기를 실행하는 명령입니다.
2. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈에 엔코더 값 읽기 지령을 내립니다.

■ 프로그램 예

1. ST

```
INST_APM_ENCRD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, DONE=>DONE_BOOL, STAT=>STAT_UINT,
ENC_VAL=>ENC_UDINT);
```

APM_JOG	적용 기종	발생플래그
조그 운전	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 JOG_DIR : 조그 운전시 회전 방향을 설정 0:정방향, 1:역방향 LOW/HIGH : 조그 운전시 조그 속도를 설정 0:조그 저속 운전, 1:조그 고속 운전 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

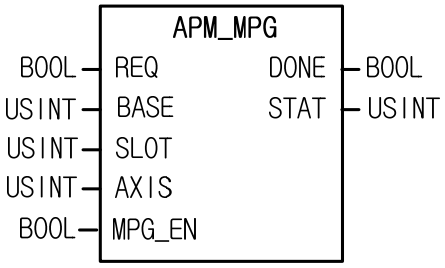
1. 이 명령은 위치결정 모듈에 조그 운전을 실행하는 명령입니다.
2. 테스트를 위한 수동 운전 기능으로 시스템의 동작, 배선상태 검사 및 티칭을 위한 위치 어드레스 확인용으로 사용되며 속도를 고속과 저속으로 구분하여 사용할 수 있습니다.
3. 입력 변수 REQ의 접속 조건이 On일 때 설정된 값에 의해 펄스가 출력되고 Off일 때 정지 됩니다.
4. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS로 지정된 축에 조그 운전 지령을 내립니다.
5. AXIS에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러6"이 발생합니다.
   0: X 축, 1: Y 축, 2: Z 축

■ 프로그램 예

1. ST

```
INST_APM_JOG(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, JOG_DIR:=JOG_BOOL,
LOW_HIGH:=LOW_HIGH_BOOL, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_MPG	적용 기종	발생플래그
수동 펄스 발생기(MPG) 운전	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 MPG_EN : MPG(수동펄스 발생기)운전의 허용/금지 설정 0:금지, 1:허용 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

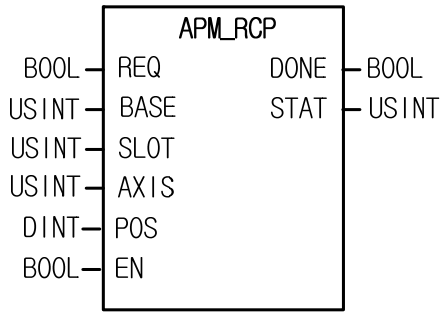
1. 이 명령은 위치결정 모듈에 수동 펄스 발생기(MPG) 운전을 실행하는 명령입니다.
2. 외부에 장착된 수동 펄스 발생기(MPG)를 이용하여 운전하고자 할 때 위치 결정 모듈에게 운전 준비 상태가 되도록 지령을 내립니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS로 지정된 축에 MPG 운전 지령을 내립니다.
4. AXIS에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러6"이 발생합니다.
   0: X 축, 1: Y 축, 2: Z 축

■ 프로그램 예

1. ST

```
INST_APM_MPG(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, MPG_EN:=MPG_BOOL,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

APM_RCP	적용 기종	발생플래그
현재 위치 구간 반복	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 POS : 반복 위치(어드레스) 설정 -2,147,483,648 ~ 2,147,483,647 EN : 현재 위치 구간 반복 허용 0: 현재 위치 구간 반복 금지, 1: 현재 위치 구간 반복 허용 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

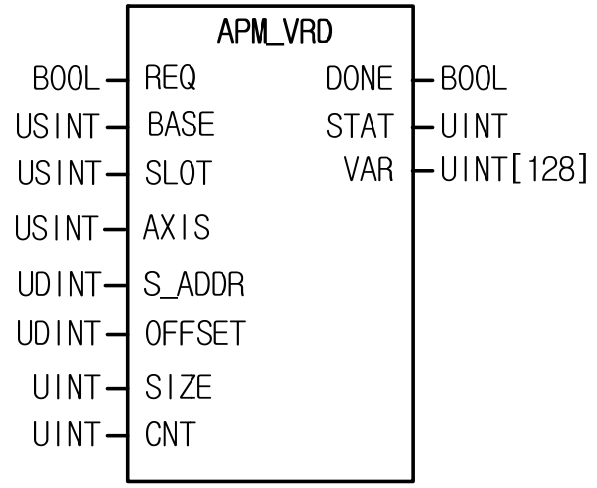
1. 이 명령은 위치결정 모듈의 현재 위치 구간을 설정 또는 금지하는 명령입니다.
2. 직접 기동에서만 동작합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS로 지정된 축에 현재 위치 구간 반복 운전 지령을 내립니다.
4. AXIS에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러6"이 발생합니다.
   0: X 축, 1: Y 축, 2: Z 축

■ 프로그램 예

1. ST

```
INST_APM_RCP(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), POS:=(*DINT*), DONE=>(*BOOL*),
STAT=>(*UINT*))
```

APM_VRD	적용 기종	발생플래그
가변 데이터 읽기	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 명령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 S_ADDR : 읽을 데이터의 모듈 내부 메모리 선두 번지 0 ~ 12147 OFFSET : 읽을 데이터 블록 간 옵셋:0 ~ 53329 SIZE : 읽을 데이터 블록의 크기:1 ~ 128 CNT : 읽을 데이터 블록의 개수:1 ~ 128 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력 VAR : 읽은 데이터가 저장되는 PLC 디바이스

■ 기능

(1) 이 명령은 위치결정 모듈에 파라미터, 운전데이터 직접 읽기 지령을 내리는 명령입니다.

(2) 파라미터와 운전데이터의 모듈 내부 메모리 번지를 직접 지정하여 원하는 데이터를 읽어오는 데 사용할 수 있습니다.

(3) BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 파라미터, 운전데이터 중 위치결정 모듈 내부 메모리의 "S_ADDR" 설정된 위치에서부터 "SIZE"만큼의 데이터를 WORD 단위로 "VAR"에 지정된 디바이스로 읽어오는 명령입니다. "CNT"가 2 이상인 경우 "S_ADDR"위치에서부터 "OFFSET"만큼 떨어져 있는 블록들을 "CNT"-1 횟수만큼 차례로 읽어와서 "VAR"에 지정된 디바이스에 저장합니다.

(4) 한 명령으로 읽을 수 있는 최대 데이터 크기 (SIZE x CNT)는 128 WORD 입니다.

(5) "가변 데이터 읽기" 명령은 운전 중에도 실행할 수 있습니다.

(6) AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정 값 이외의 값을 설정했을 경우 에러 "6"이 발생합니다.

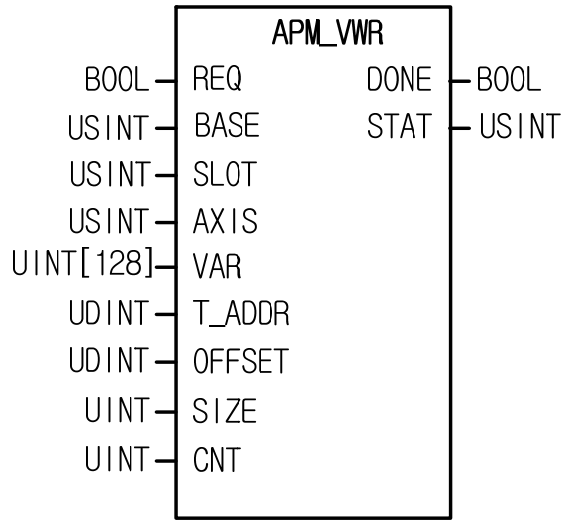
(7) 읽을 데이터 크기(SIZE x CNT)가 0 이거나 128 WORD 를 초과한 경우 STAT 에 에러 "11"이 발생합니다.

■ 프로그램 예

1. ST

```
INST_APM_VRD(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), S_ADDR:=(*UINT*), OFFSET:=(*UINT*),
SIZE:=(*UINT*), CNT:=(*UINT*), DONE=>(*BOOL*), STAT=>(*UINT*), VAR=>(*ARRAY[0..127]_OF_UINT*))
```

APM_VWR	적용 기종	발생플래그
가변 데이터 쓰기	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 명령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 VAR : 쓸 데이터가 저장되어있는 PLC 디바이스 T_ADDR : 데이터가 쓰여지는 모듈 내부 메모리 선두 번지 0 ~ 12147 OFFSET : 쓸 데이터 블록 간 옵셋 0 ~ 53329 SIZE : 쓸 데이터 블록의 크기 1 ~ 128 CNT : 쓸 데이터 블록의 개수 1 ~ 128 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

(1) 이 명령은 위치결정 모듈에 파라미터, 운전데이터 직접 쓰기 지령을 내리는 명령입니다.

(2) 파라미터와 운전데이터의 모듈 내부 메모리 번지를 직접 지정하여 원하는 데이터를 쓰는데 사용할 수 있습니다.

(3) BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈 내부 메모리의 파라미터, 운전데이터 중 "T_ADDR" 위치에서부터 "SIZE" 만큼의 데이터를 PLC 프로그램에서 "VAR" 에 지정한 데이터들로 WORD 단위 쓰기를 실행하는 명령입니다. 블록 개수 "CNT" 가 2 이상인 경우 "T_ADDR" 에 위치한 블록부터 "OFFSET" 만큼 떨어져 있는 블록들에 나머지 데이터들을 "CNT" -1 횟수만큼 차례로 데이터 쓰기를 실행합니다.

(4) 한 명령으로 쓸 수 있는 최대 데이터 크기 (SIZE x CNT)는 128 WORD 입니다.

(5) "가변 데이터 쓰기" 명령은 운전 중에는 실행할 수 없습니다.

(6) AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정 값 이외의 값을 설정했을 경우 에러 "6" 이 발생합니다.

(7) 읽을 데이터 크기(SIZE x CNT)가 0 이거나 128 WORD 를 초과한 경우 STAT 에 에러 "11" 이 발생합니다.

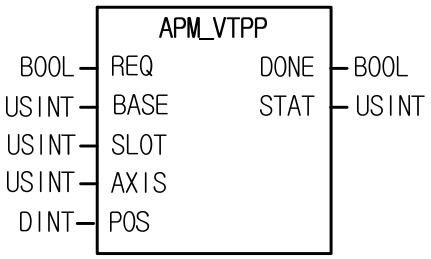
(8) 블록 개수 (CNT)가 2 이상이고, 블록 옵셋(OFFSET)이 블록 크기(CNT) 보다 작은 경우 데이터를 쓸 모듈 내부 메모리 블록이 서로 중복되므로 STAT 에 에러 "11" 이 발생합니다.

■ 프로그램 예

1. ST

```
INST_APM_VWR(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*),
VAR:=(*ARRAY[0..127]_OF_UINT*),  T_ADDR:=(*UINT*),  OFFSET:=(*UINT*),  SIZE:=(*UINT*),  CNT:=(*UINT*),
DONE=>(*BOOL*), STAT=>(*UINT*))
```

APM_VTPP	적용 기종	발생플래그
위치지정 속도/위치 전환 제어	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 0:X 축, 1:Y 축, 2:Z 축 POS : 목표위치 설정 1 ~ 2,147,483,647 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 이 명령은 속도 제어 운전이 실행 중인 축에 대하여 펑션 블록을 실행하면, 위치 제어 운전으로 전환되어 운전을 실행하는 명령입니다.
2. 펑션 블록이 실행되는 순간에 원점은 결정 상태가 되고, 펑션 블록에 설정된 목표위치로 운전한 후에 위치결정을 완료합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS로 지정된 축에 MPG 운전 지령을 내립니다.
4. AXIS에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러6"이 발생합니다.
   0: X 축, 1: Y 축, 2: Z 축

■ 프로그램 예

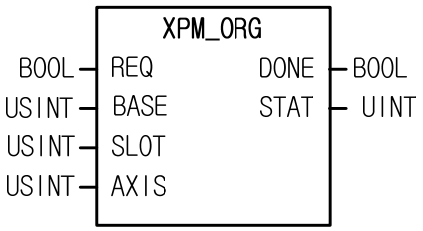
1. ST

```
INST_APM_VTPP(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), POS:=(*DINT*), DONE=>(*BOOL*),
STAT=>(*UINT*));
```

11.5 위치결정 펑션 블록 (XPM)

1. 각각의 특수 펑션 블록에 대한 설명입니다.

XPM_ORG	적용 기종	발생플래그
원점 복귀 기동	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

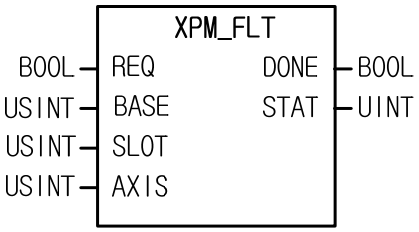
1. 이 명령은 위치결정 모듈에 원점복귀 지령을 내리는 명령입니다.
2. 각 축의 원점 복귀용 파라미터에 설정된 방향, 보정량, 속도(고속, 저속), 어드레스 및 드웰 시간으로 원점 복귀 처리 방식에 따라 기계의 원점을 찾도록 하는 운전 지령입니다.
3. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 원점복귀 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러6” 이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)

■ 프로그램 예

1. ST

```
INST_XPM_ORG(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*),
STAT=>(*UINT*))
```

XPM_FLT	적용 기종	발생플래그
부동 원점 기동	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

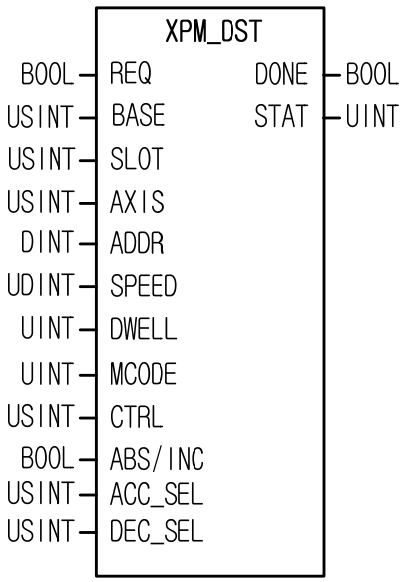
1. 이 명령은 위치결정 모듈에 부동원점 설정 지령을 내리는 명령입니다.
2. 기계의 원점복귀 동작을 수행하지 않고 현재위치를 강제로 원점으로 설정하고자 할 때 사용하는 지령으로 원점 복귀 어드레스에 지정된 어드레스 값이 현재의 위치가 됩니다.
3. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 부동원점 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러6” 이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)

■ 프로그램 예

1. ST

```
INST_XPM_FLT(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_DST	적용 기종	발생플래그
직접 기동	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) ADDR : 목표위치 어드레스 설정 -2147483648 ~ +2147483647 SPEED : 목표속도 설정 DWELL : 드웰타임 설정 0 ~ 65535[ms] MCODE : MCode 값 설정 CTRL : 제어방법 설정 0: 위치제어, 1: 속도제어 2: Feed 제어 ABS/INC: 절대좌표/상대좌표 설정 0: 절대좌표, 1: 상대좌표 ACC_SEL: 가속시간 번호 설정 0: 가속시간 1, 1: 가속시간 2 2: 가속시간 3, 3: 가속시간 4 DCC_SEL: 감속시간 번호 설정 0: 감속시간 1, 1: 감속시간 2 2: 감속시간 3, 3: 감속시간 4 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

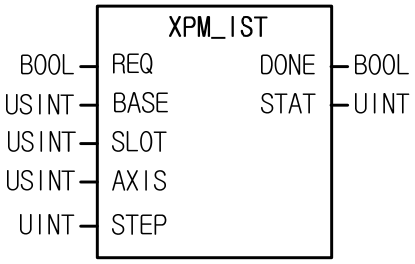
1. 이 명령은 위치결정 모듈에 직접기동 지령을 내리는 명령입니다.
2. 운전데이터에 의하지 않고 목표위치어드레스, 운전 속도, 드웰시간, MCode번호, 제어방식, 좌표설정 및 가감속 시간의 번호를 설정하여 직접운전하고자 할 때 사용합니다.
3. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS로 지정된 축에 직접기동 지령을 내립니다.
4. AXIS에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러6"이
   발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)
5. SPEED, CTRL, TIME_SEL에 설정된 값이 설정범위를 넘을 경우 STAT에 "에러11"이 발생합니다.

■ 프로그램 예

1. ST

```
INST_XPM_DST(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), ADDR:=(*DINT*),
SPEED:=(*UDINT*), DWELL:=(*UINT*), MCODE:=(*UINT*), CTRL:=(*USINT*), ABS_INC:=(*BOOL*),
ACC_SEL:=(*USINT*), DEC_SEL:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_IST	적용 기종	발생플래그
간접 기동	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1 축 ~ 4 축) XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축) STEP : 운전할 스텝번호 0 ~ 400 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

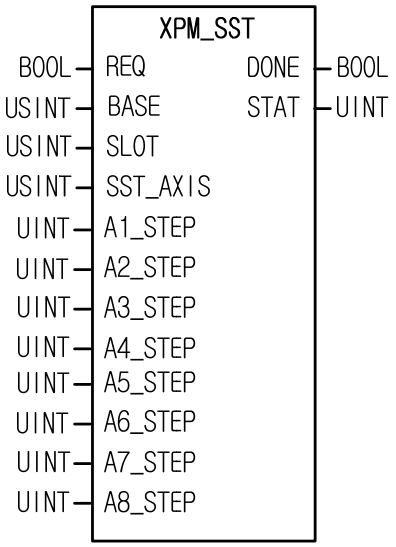
1. 이 명령은 위치결정 모듈에 간접기동 지령을 내리는 명령입니다.
2. 운전 데이터로 설정된 축의 운전 스텝 번호를 지정하여 운전하고자 할 때 사용합니다.
3. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS로 지정된 축에 간접기동 지령을 내립니다.
4. AXIS에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러6" 이
   발생합니다.
   XPM: 1 ~ 4 (1 축 ~ 4 축), XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축)
5. STEP에 설정된 값이 설정범위(0 ~ 400)를 넘을 경우 STAT에 "에러11"이 발생합니다.
6. STEP에 0을 설정하면 현재스텝을 운전합니다.

■ 프로그램 예

1. ST

```
INST_APM_IST(REQ:=(*BOOL*),  BASE:=(*USINT*),  SLOT:=(*USINT*),  AXIS:=(*USINT*),  STEP:=(*UINT*),
DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_SST	적용 기종	발생플래그
동시 기동	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 SST_AXIS : 동시기동 축 설정 XPM: 0bit ~ 3bit (1 축 ~ 4 축) XGF-PN8A/B: 0bit ~ 7bit (1 축 ~ 8 축) 각축에 해당하는 bit 를 Set 하여 선택 A1_STEP : 기동할 1 축 스텝번호 A2_STEP : 기동할 2 축 스텝번호 A3_STEP : 기동할 3 축 스텝번호 A4_STEP : 기동할 4 축 스텝번호 A5_STEP : 기동할 5 축 스텝번호 A6_STEP : 기동할 6 축 스텝번호 A7_STEP : 기동할 7 축 스텝번호 A8_STEP : 기동할 8 축 스텝번호 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 이 명령은 위치결정 모듈에 동시 기동 지령을 내리는 명령입니다.
2. XPM의 경우 두축 ~ 네축, XPM-PM8A의 경우 두축 ~ 여덟축의 운전을 동시에 시작하고자 할 때 사용합니다.
3. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 SST_AXIS로 지정된 축에 동시기동 지령을 내립니다.
4. SST_AXIS에 설정값 이외의 값을 설정하면 "에러6"이 발생합니다. 설정은 아래와 같이 각 bit를 set하여 설정합니다.

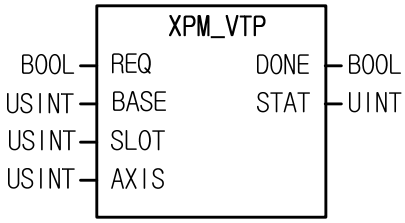
7bit	6bit	5bit	4bit	3bit	2bit	1bit	0bit
8 축	7 축	6 축	5 축	4 축	3 축	2 축	1 축

■ 프로그램 예

1. ST

```
INST_XPM_SST1(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), SST_AXIS:=(*USINT*), A1_STEP:=(*UINT*),
A2_STEP:=(*UINT*), A3_STEP:=(*UINT*), A4_STEP:=(*UINT*), A5_STEP:=(*UINT*), A6_STEP:=(*UINT*),
A7_STEP:=(*UINT*), A8_STEP:=(*UINT*), DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_VTP	적용 기종	발생플래그
속도/위치 전환 제어	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

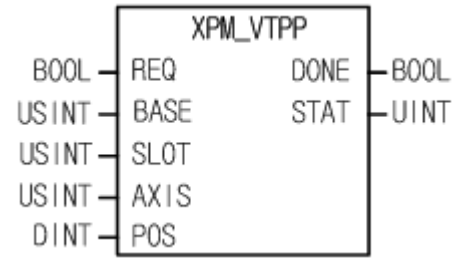
1. 이 명령은 위치결정 모듈에 속도/위치제어 변경 지령을 내리는 명령입니다.
2. 지정된 축이 속도제어 운전을 하다가 속도/위치제어 변경 지령을 받으면 속도제어에서 위치제어로 전환 되어 운전합니다
3. 이 지령이 실행되면 실행된 순간에 원점이 결정되고 이전 속도제어 기동시 설정한 목표위치까지 이동한 후 위치결정을 완료합니다.
4. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS로 지정된 축에 속도/위치 제어 변경 지령을 내립니다.
5. AXIS에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러6"이
   발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)

■ 프로그램 예

1. ST

```
INST_XPM_VTP(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*),
STAT=>(*UINT*))
```

XPM_VTPP	적용 기종	발생플래그
위치지정 속도/위치 전환 제어	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1 축 ~ 4 축) XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축) POS : 위치 이동량 -2,147,483,648 ~ 2,147,483,647 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

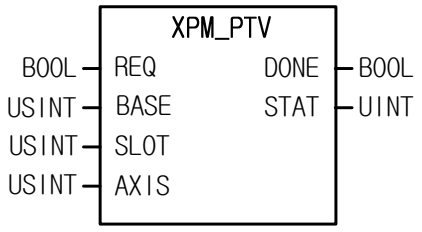
1. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS로 지정된 축에 위치지정 속도/위치 전환 제어 명령을 내립니다.
2. 지정된 축이 속도제어 운전을 하다가 위치지정 속도/위치제어 변경 지령을 받으면 속도제어에서 위치제어로 전환 되어 POS에 지정된 위치 이동량 만큼 위치결정 운전을 합니다
3. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러 6"이 발생합니다.
   XPM: 1 ~ 4 (1 축 ~ 4 축), XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축)

■ 프로그램 예

1. ST

```
INST_XPM_VTPP(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), POS:=(*DINT*),
DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_PTV	적용 기종	발생플래그
위치/속도 전환 제어	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1 축 ~ 4 축) XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축) **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

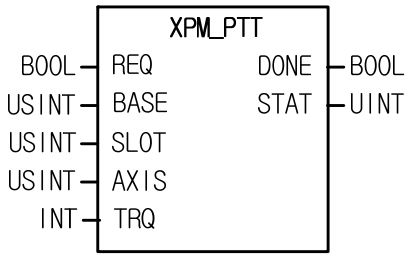
1. 이 명령은 위치결정 모듈에 위치/속도 제어 변경 지령을 내리는 명령입니다.
2. 지정된 축이 지정된 이동량으로 위치제어 운전을 하다가 위치/속도 제어 변경 지령을 받으면 위치제어에서 속도제어로 전환 되어 운전합니다.
3. 이 지령이 실행되면 실행된 순간에 원점이 미결정되면서 속도제어 운전을 합니다.
4. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS로 지정된 축에 위치/속도 제어 변경 지령을 내립니다.
5. AXIS에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러6" 이
   발생합니다.
   XPM: 1 ~ 4 (1 축 ~ 4 축), XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축)

■ 프로그램 예

1. ST

```
INST_XPM_PTV(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*),
STAT=>(*UINT*))
```

XPM_PTT	적용 기종	발생플래그
위치/토크 전환 제어	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 1 ~ 8: 1축 ~ 8축 TRQ : 토크값 -300 ~ 300 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

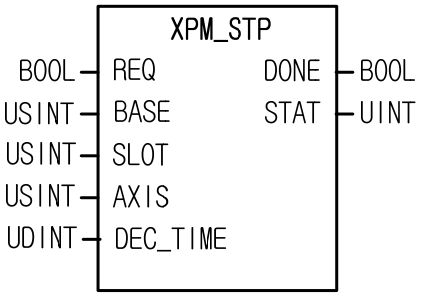
1. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS로 지정된 축에 위치/토크 제어 변경 명령을 내립니다.
2. 지정된 축이 위치제어 운전을 하다가 위치/토크제어 변경 명령을 받으면 위치제어에서 토크제어로 전환 되어 TRQ에 설정된 토크값으로 운전하며 감속정지등 정지 요인이 발생할 때까지 토크제어를 유지합니다.
3. 토크값의 범위는 -300 ~ 300까지 이며 단위는 [%]입니다.
4. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러 6" 이 발생합니다.
   1 ~ 8: 1축 ~ 8축
5. 이 명령어는 XGF-PN8A/B 모듈 전용입니다.

■ 프로그램 예

1. ST

```
INST_XPM_PTT(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), TQR:=(*INT*),
DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_STP	적용 기종	발생플래그
감속 정지	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) DEC_TIME : 감속정지 시간 0:운전시작시 적용된 가감속 시간. 1 ~ 2147483647: 1 ~ 2147483647ms **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

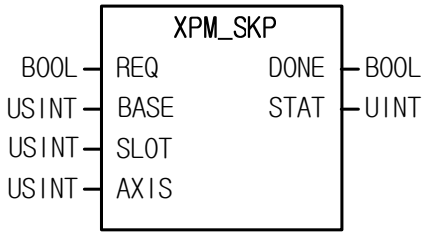
1. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 감속정지 명령을 내립니다.
2. 운전데이터에 의한 운전 중에 정지 명령을 만나면 감속 정지한 후 기동 명령에 의해 다시 운전을 시행합니다.
3. 속도동기나 위치동기, CAM 운전 에서 감속 정지 명령이 실행되면 현재 운전 제어 상태에 따라 속도동기나 위치동기, CAM운전을 끝내게 됩니다.
4. 감속 정지는 가속 및 정속 구간 뿐 아니라 감속 구간에서도 명령이 실행됩니다.
5. 감속 시간은 감속 시작부터 정지까지의 시간을 뜻하며 0 ~ 2,147,483,647ms 까지 설정할 수 있습니다. 단, 0 으로 설정했을 때는 운전을 시작할 때 설정된 감속 시간에 의해서 정지됩니다.
6. 감속 시간은 운전 축 기본 파라미터의 속도 제한치부터 정지까지 소요되는 시간을 뜻합니다.
7. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)

■ 프로그램 예

1. ST

```
INST_XPM_STP(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DEC_TIME:=(*UDINT*),
DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_SKP	적용 기종	발생플래그
스킵 운전	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1 축 ~ 4 축) XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축) **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

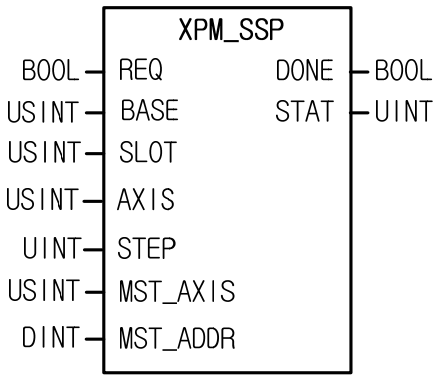
1. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 스킵운전 명령을 내립니다.
2. 운전 스텝을 운전하지 않고 그 다음 스텝으로 이동하여 실행할 때 사용합니다. 즉, 현재 운전 중인 스텝의 운전을 정지하여 종료하고 다음 스텝으로 운전을 계속합니다.
3. 한번 실행할 때마다 현재의 운전 스텝을 건너 뛰어 그 다음의 운전 스텝을 운전합니다.
4. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1 축 ~ 4 축), XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축)

■ 프로그램 예

1. ST

```
INST_XPM_SKP(REQ:=(*BOOL*),  BASE:=(*USINT*),  SLOT:=(*USINT*),  AXIS:=(*USINT*),  DONE=>(*BOOL*),
STAT=>(*UINT*))
```

XPM_SSP	적용 기종	발생플래그
위치 동기	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) STEP : 운전할 스텝번호 0 ~ 400 MST_AXIS : 위치동기 주축 설정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) 9: 엔코더 MST_ADDR : 위치동기를 실행할 주축의 위치 설정 -2,147,483,648 ~ 2,147,483,647 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

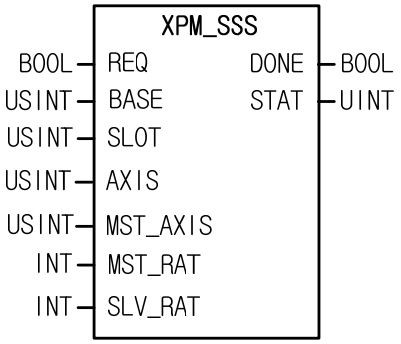
1. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 위치동기 명령을 내립니다.
2. 명령을 내린 축을 종축으로 하고 주축으로 설정된 축이 설정된 동기위치에 도달하면 명령축이 설정한 운전 스텝을 운전 합니다.
3. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)
4. MST_AXIS 에는 위치동기의 주축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축), 9: 엔코더

■ 프로그램 예

1. ST

```
INST_XPM_SSP(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), STEP:=(*UINT*),
MST_AXIS:=(*USINT*), MST_ADDR:=(*DINT*), DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_SSS	적용 기종	발생플래그
속도 동기	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) MST_AXIS : 속도동기 주축 설정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) 9: 엔코더 MST_RAT : 주축의 속도비 설정 1 ~ 32767, -32768 ~ -1 SLV_RAT : 종축의 속도비 설정 1 ~ 32767, -32768 ~ -1 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

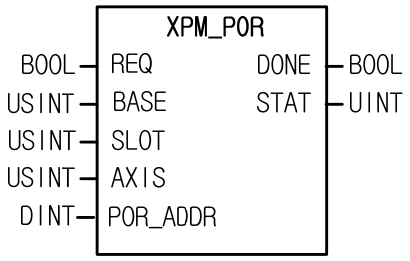
■ 기능

1. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 속도동기 명령을 내립니다.
2. 두 축간에 운전 속도를 설정한 비율로 제어 하고자 할 때 사용합니다.
3. 주축의 속도비와 종축의 속도비 사이에 크기에 대한 규칙은 없습니다. 즉, 주축의 속도비가 종축의 속도비보다 크면 주축이 종축보다 빠르게 움직이며, 종축의 속도비가 주축의 속도비보다 크면 종축이 주축보다 빠르게 움직입니다.
4. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러 6" 이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)
5. MST_AXIS 에는 속도동기의 주축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러 6" 이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축), 9: 엔코더
6. 종축의 운전방향은 속도 동기 비율($\frac{\text{종축비}}{\text{주축비}}$) 이 양수이면 주축의 운전방향으로 운전하고, 음수이면 주축의 반대방향으로 운전합니다.

■ 프로그램 예

1. ST

```
INST_XPM_SSS(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), MST_AXIS:=(*USINT*),
MST_RAT:=(*INT*), SLV_RAT:=(*INT*), DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_POR	적용 기종	발생플래그
위치 오버라이드	XGI, XGR	-
펑션 블록	설 명	
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1 축 ~ 4 축) XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축) POR_ADDR : 새로운 목표위치 설정 -2,147,483,648 ~ 2,147,483,647 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력	

■ 기능

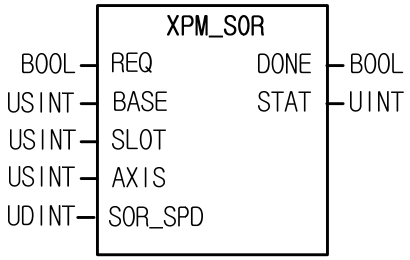
1. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 위치오버라이드 명령을 내립니다.
2. 명령 축이 운전 중인 상태에서 목표 위치를 변경하고자 할 때 사용합니다.
3. 위치 오버라이드를 실행하고자 하는 위치를 지나간 후에 위치 오버라이드를 실행하면 현재 위치에서 정지한 후 방향을 전환하여 POR_ADDR 에 설정된 위치로 이동합니다.
4. POR_ADDR 에는 변경할 목표위치를 설정합니다.
5. 위치 오버라이드 값에 설정된 오버라이드 위치는 절대좌표 위치입니다.
6. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1 축 ~ 4 축), XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축)

■ 프로그램 예

1. ST

```
INST_XPM_POR(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), POR_ADDR:=(*DINT*),
DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_SOR	적용 기종	발생플래그
속도 오버라이드	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) SOR_SPD : 새로운 운전속도값 설정 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

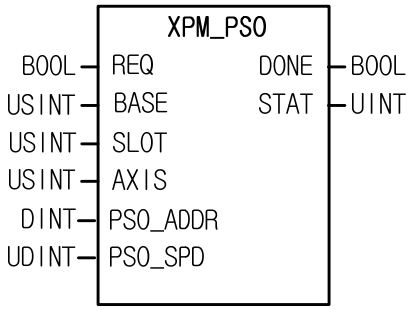
1. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 속도 오버라이드 명령을 내립니다.
2. 명령 축이 운전 중인 상태에서 운전속도를 변경하여 사용하고자 할 때 사용합니다.
3. SOR_SPD 는 공통 파라미터의 “속도 오버라이드”에 설정된 값에 따라 “%”나 “속도 값(단위/시간)” 으로 설정할 수 있습니다.
4. 속도 오버라이드 값의 단위가 %일 경우 설정 영역은 1 ~ 65,535 이며 이는 0.01% ~ 655.35%를 의미합니다.
5. 속도 오버라이드 값의 단위가 속도 값일 경우의 설정 영역은 1 ~ 속도제한치 이며, 이때 속도제한치는 기본 파라미터의 “속도제한치” 항목에 설정된 값입니다. 그리고, 속도 오버라이드 값의 단위는 축의 단위를 따릅니다.
6. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6”이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)

■ 프로그램 예

1. ST

```
INST_XPM_SOR(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), SOR_SPD:=(*UDINT*),
DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_PSO	적용 기종	발생플래그
위치지정 속도 오버라이드	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1 축 ~ 4 축) XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축) PSO_ADDR : 속도 변경을 수행할 위치 -2,147,483,648 ~ 2,147,483,647 PSO_SPD : 새로운 운전속도값 설정 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

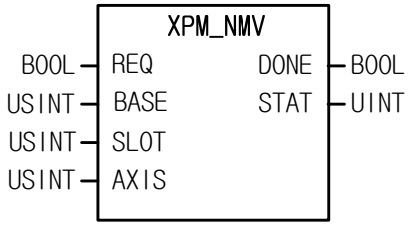
1. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 속도 오버라이드 명령을 내립니다.
2. 명령 축이 운전 중인 상태에서 일정 위치에 도달한 후 운전속도를 변경하여 사용하고자 할 때 사용합니다.
3. PSO_SPD 에 설정하는 속도 값은 공통 파라미터의 “속도 오버라이드”에 설정된 값에 따라 “% 지정”이나 “속도 값 지정”이 됩니다.
4. 속도 값의 단위가 %일 경우 설정 영역은 1 ~ 65,535 이며 이는 0.01% ~ 655.35%를 의미합니다.
5. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6”이 발생합니다.
   XPM: 1 ~ 4 (1 축 ~ 4 축), XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축)

■ 프로그램 예

1. ST

```
INST_XPM_PSO(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), PSO_ADDR:=(*DINT*),
PSO_SPD:=(*UDINT*), DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_NMV	적용 기종	발생플래그
연속 운전	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

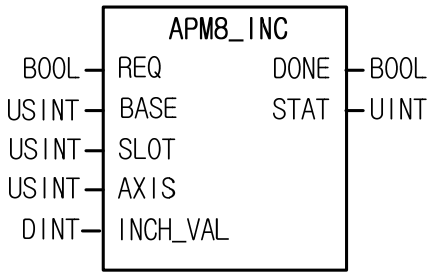
1. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 연속운전 명령을 내립니다.
2. 명령축이 현재운전중인 스텝에서 정지하지 않고 다음스텝으로 운전을 변경하고자 할 때 사용합니다.
3. 연속 운전 명령을 실행하면 현재 운전중인 스텝 번호가 다음 스텝 번호로 변경되면서 다음 스텝의 속도와 목표 위치로 위치결정 운전을 합니다. 다음 스텝과의 연결은 연속운전 패턴으로 수행됩니다.
4. 연속 운전 명령은 현재 실행중인 스텝의 현재 운전 패턴만을 변경하고 운전 데이터를 변경하지는 않습니다.
5. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러 6" 이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)

■ 프로그램 예

1. ST

```
INST_XPM_NMV(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*),
STAT=>(*UINT*))
```

XPM_INC	적용 기종	발생플래그
인칭 운전	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) INCH_VAL : 인칭운전으로 이동하고자 하는 이동량 설정. -2,147,483,648 ~ 2,147,483,647 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

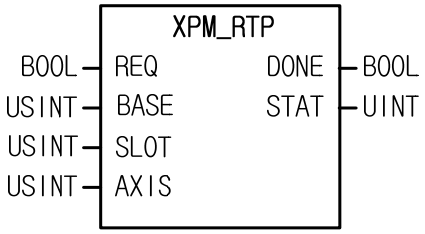
1. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 인칭운전 명령을 내립니다.
2. 인칭 운전은 수동운전의 일종으로 미세한 움직임을 정량적인 운전으로 처리할 할 경우 사용합니다.
3. 인칭 운전의 속도는 수동운전 파라미터에서 설정합니다.
4. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)

■ 프로그램 예

1. ST

```
INST_XPM_INC(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), INCH_VAL:=(*DINT*),
DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_RTP	적용 기종	발생플래그
수동 운전 이전 위치로 복귀	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

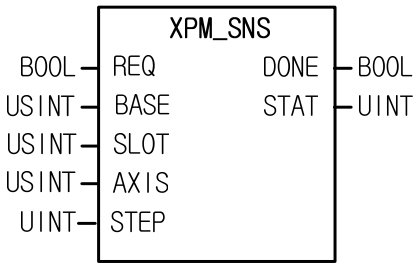
1. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 수동운전 이전위치로 복귀 명령을 내립니다.
2. 위치 결정 후 수동운전에 의해 위치가 변경되었을 때 수동 운전 이전의 위치로 되돌리고자 할 때 사용하는 명령입니다.
3. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)

■ 프로그램 예

1. ST

```
INST_XPM_RTP(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*),
STAT=>(*UINT*))
```

XPM_SNS	적용 기종	발생플래그
기동 스텝번호 변경	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) STEP : 운전하고자 하는 운전스텝번호 설정 1 ~ 400 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

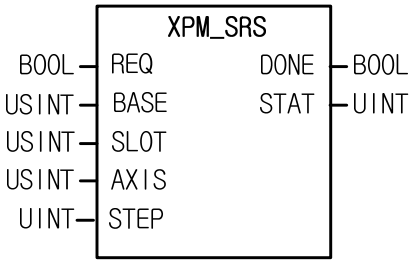
1. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 기동 스텝변경 명령을 내립니다.
2. 명령 축의 운전 스텝을 변경하고자 할 때 사용합니다.
3. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6”이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)
4. STEP 에는 운전하고자 하는 스텝번호를 설정합니다. 설정값의 범위는 1 ~ 400 이며 설정값 이외의 값을 설정했을 경우 “에러 11”이 발생합니다.

■ 프로그램 예

1. ST

```
INST_XPM_SNS(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), STEP:=(*UINT*),
DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_SRS	적용 기종	발생플래그
반복 스텝 번호 변경	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) STEP : 변경하고자 하는 반복스텝번호 설정 1 ~ 400 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

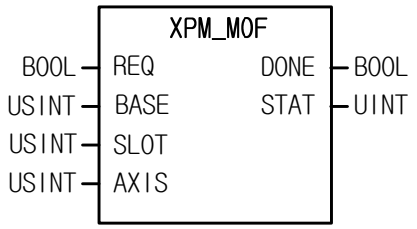
■ 기능

1. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 반복스텝변경 명령을 내립니다.
2. 운전 데이터로 운전하던 중 반복운전을 만나면 반복 운전 스텝으로 되돌아 가는 반복운전 시 반복운전의 시작 스텝번호를 지정하여 특정 운전 스텝에서 운전을 시작하고자 할 때 사용합니다.
3. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)
4. STEP 에는 반복운전을 시작하고자 하는 스텝번호를 설정합니다. 설정값의 범위는 1 ~ 400 이며 설정값 이외의 값을 설정했을 경우 “에러 11” 이 발생합니다.

■ 프로그램 예

1. ST

```
INST_XPM_SRS(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), STEP:=(*UINT*),
DONE=>(*BOOL*),
STAT=>(*UINT*))
```

XPM_MOF	적용 기종	발생플래그
M 코드 해제	XGI, XGR	-
펑션 블록	설 명	
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1 축 ~ 4 축) XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축) **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력	

■ 기능

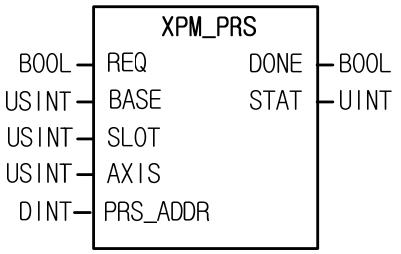
1. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 M 코드 해제 명령을 내립니다.
2. 각 축의 파라미터에서 M 코드를 With 나 After 모드로 설정한 경우, 명령축의 M 코드신호가 On 되었을 때 이 신호를 Off 하고자 할 때 사용할 수 있습니다. 즉, M 코드 신호는 Off, M 코드 번호는 0 으로 변경합니다.
3. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1 축 ~ 4 축), XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축)

■ 프로그램 예

1. ST

```
INST_XPM_MOF(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*),
STAT=>(*UINT*))
```

XPM_PRS	적용 기종	발생플래그
현재 위치 프리셋	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) PRS_ADDR : 변경하고자 하는 현재위치값 설정. -2,147,483,648 ~ 2,147,483,647 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

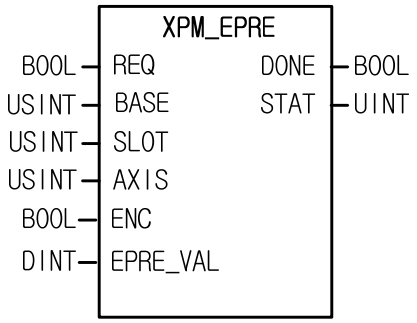
1. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 현재위치 프리셋 명령을 내립니다.
2. 명령축의 현재 위치를 임의의 위치로 변경하고자 할 때 사용되는 명령으로 원점 미결정 상태에서 실행하면 원점결정 신호(비트)가 On 되고 현재 위치가 설정값(PRS_ADDR)으로 변경됩니다..
3. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)

■ 프로그램 예

1. ST

```
INST_XPM_PRS(REQ:=(*BOOL*),  BASE:=(*USINT*),  SLOT:=(*USINT*),  AXIS:=(*USINT*),  PRS_ADDR:=(*DINT*),
DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_EPRE	적용 기종	발생플래그
엔코더 현재값 프리셋	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1 축 ~ 4 축) XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축) ENC : 엔코더 번호(항상 0 으로 설정) 0: 엔코더 EPRE_VAL : 엔코더 프리셋 값 설정 -2,147,483,648 ~ 2,147,483,647 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

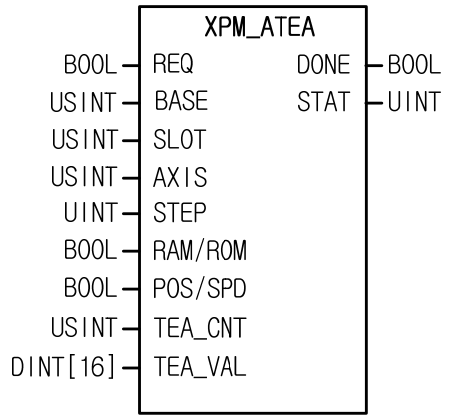
1. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 엔코더 프리셋 명령을 내립니다.
2. EPRE_VAL 에 설정된 값으로 엔코더의 현재 값을 변경하는 명령입니다.
3. ENC 에는 프리셋 할 엔코더를 설정하며 XPM 위치결정 모듈에서는 항상 0 으로 설정하십시오.
4. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러 6" 이 발생합니다.
   XPM: 1 ~ 4 (1 축 ~ 4 축), XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축)

■ 프로그램 예

1. ST

```
INST_XPM_EPRE(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), ENC:=(*BOOL*),
EPRE_VAL:=(*DINT*), DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_ATEA	적용 기종	발생플래그
복수 티칭	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1 축 ~ 4 축) XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축) STEP : 티칭할 스텝번호 설정 0 ~ 400 RAM/ROM : RAM 티칭과 ROM 티칭 종류 선택 0 : RAM 티칭, 1 : ROM 티칭 POS/SPD : 위치티팅과 속도티칭 종류 선택 0 : 위치티칭, 1 : 속도티칭 TEA_CNT : 티칭할 데이터의 개수 설정 1 ~ 16 TEA_VAL : 티칭값 설정 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

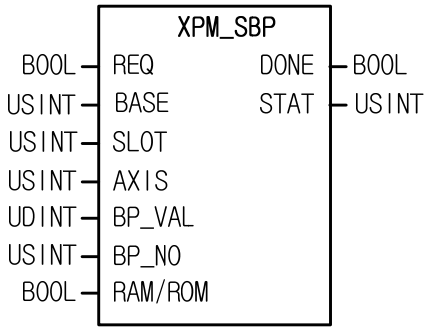
1. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 복수 티칭 명령을 내립니다.
2. 속도티칭은 사용자가 임의의 속도 값을 특정 스텝의 운전 데이터에 사용하고자 할 때, 위치티칭은 사용자가 임의의 위치 값을 특정 운전 스텝의 운전 데이터에 설정하고자 할 때 사용할 수 있습니다.
3. 복수형 티칭 펑션 블록을 이용하여 한번에 최대 16 개까지의 목표위치나 속도 값을 변경하고자 할 때 사용합니다.
4. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러 6" 이 발생합니다.
   XPM: 1 ~ 4 (1 축 ~ 4 축), XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축)
5. STEP 에는 티칭을 할 운전데이터의 스텝번호를 설정하며 0 ~ 400 사이의 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러 11" 이 발생합니다.
6. TEA_CNT 에는 티칭 할 데이터 개수를 설정하며 최대 16 개까지 티칭을 할 수 있습니다. 설정범위 이외의 값을 설정했을 경우 "에러 11" 이 발생합니다.
7. RAM/ROM 에 "0" 을 설정하고 티칭 명령으로 수정한 운전 데이터 값은 전원이 켜져 있는 동안에만 유효합니다. 티칭 명령으로 수정한 운전 데이터 값을 전원이 꺼진 후에도 유지하려면 RAM/ROM 을 "1" 로 설정하고 티칭 명령을 수행하거나 티칭 후 파라미터/운전데이터 저장 명령(XPM_WRT)을 사용하여 수정한 파라미터 값을 FRAM 에 저장하여야 합니다.

■ 프로그램 예

1. ST

```
INST_XPM_ATEA(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), STEP:=(*UINT*),
RAM_ROM:=(*BOOL*), POS_SPD:=(*BOOL*), TEA_CNT:=(*USINT*), TEA_VAL:=(*ARRAY[0..15]_OF_DINT*), DONE=>(*BOOL*),
STAT=>(*UINT*))
```

XPM_SBP	적용 기종	발생플래그
기본 파라미터 티칭	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1 축 ~ 4 축) XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축) BP_VAL: 변경할 기본 파라미터 값 BP_NO : 변경할 기본 파라미터 항목번호 RAM/ROM: 파라미터 저장 방법 0: RAM 에 저장, 1:ROM 에 저장 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 이 명령은 위치결정 모듈에 기본 파라미터 티칭을 실행하는 명령입니다.
2. 기본 파라미터 설정 명령으로 수정한 파라미터 값을 ROM 에 저장하려면 RAM/ROM 에 1 을 설정하여 기본 파라미터를 티칭 하거나 기본 파라미터 티칭 후 파라미터/운전데이터 저장 명령(WRT)을 사용하여 수정한 파라미터 값을 ROM 에 저장하여야 합니다.
3. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 기본 파라미터 티칭 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1 축 ~ 4 축), XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축)
   기본 파라미터 항목 번호에 설정할 값은 아래와 같습니다.

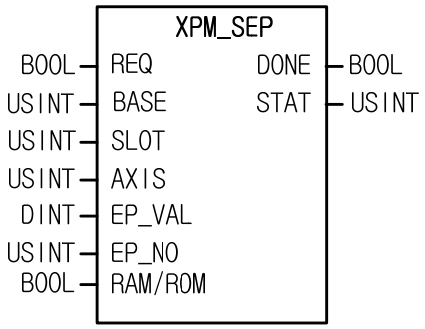
1: 속도제한치	10: 급정지 감속시간
2: 가속시간 0	11: 1 회전당 펄스수
3: 가속시간 1	12: 1 회전당 이송거리
4: 가속시간 2	13: 단위
5: 가속시간 3	14: 단위 배정도
6: 감속시간 0	15: 속도명령단위
7: 감속시간 1	16: 바이어스속도
8: 감속시간 2	17: 펄스출력방법
9: 감속시간 3	

■ 프로그램 예

1. ST

```
INST_XPM_SBP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, BP_VAL:=BP_UDINT,
BP_NO:=BP_USINT, RAM_ROM:=RAM_ROM_BOOL, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

XPM_SEP	적용 기종	발생플래그
확장 파라미터 티칭	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1 축 ~ 4 축) XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축) EP_VAL : 변경할 확장 파라미터 값 EP_NO : 변경할 확장 파라미터 항목번호 RAM/ROM: 파라미터 저장 방법 0: RAM 에 저장, 1: ROM 에 저장 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 이 명령은 위치결정 모듈에 확장 파라미터 티칭을 실행하는 명령입니다.
2. 확장 파라미터 티칭 명령으로 수정한 파라미터 값을 ROM 에 저장하려면 확장 파라미터 티칭 명령 시 RAM_ROM 을 1 로 설정하거나 확장 파라미터 티칭 후 파라미터/운전데이터 저장 명령(WRT)을 사용하여 수정한 파라미터 값을 ROM 에 저장하여야 합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 확장 파라미터 티칭 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1 축 ~ 4 축), XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축)
5. 확장 파라미터 항목 번호에 설정할 값은 아래와 같습니다.

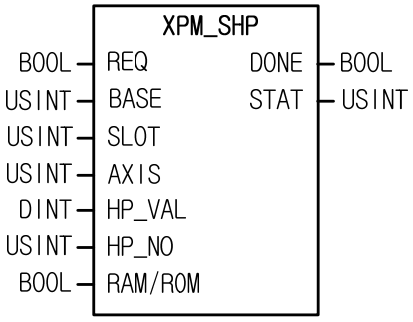
1: 소프트웨어 상한	9: 등속운전중 상하한 검출
2: 소프트웨어 하한	10: 위치결정완료 조건
3: 백래쉬 보정량	11: 직선보간 위치결정 방식
4: 위치결정완료 출력시간	12: 2 축 직선보간 연속운전 원호삽입
5: S-Curve 비율	13: 속도위치제어전환 허용.금지
6: 2 축 직선보간 연속운전 원호삽입 길이	14: 비상정지/정지 선택
7: 가감속 패턴	15: 위치지정 속도 오버라이드 좌표
8: M 코드 모드	16: 펄스 출력 방향

■ 프로그램 예

1. ST

```
INST_XPM_SEP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, EP_VAL:=EP_DINT,
EP_NO:=NO_USINT, RAM_ROM:=RAM_ROM_BOOL, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

XPM_SHP	적용 기종	발생플래그
원점 복귀 파라미터 설정	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) HP_VAL : 변경할 원점복귀 파라미터 값 HP_NO : 변경할 원점복귀 파라미터 항목번호 RAM/ROM: 파라미터 저장 방법 0: RAM에 저장, 1: ROM에 저장 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 이 명령은 위치결정 모듈에 원점 복귀 파라미터 티칭을 실행하는 명령입니다.
2. 원점 복귀 파라미터 티칭 명령으로 수정한 파라미터 값을 ROM 에 저장하려면 원점 복귀 파라미터 티칭 명령 시 RAM_ROM 을 1 로 설정하거나 원점 복귀 파라미터 티칭 후 파라미터/운전데이터 저장 명령(WRT)을 사용하여 수정한 파라미터 값을 ROM 에 저장하여야 합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 원점 복귀 파라미터 티칭 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)
5. 원점 복귀 파라미터 항목 번호에 설정할 값은 아래와 같습니다.

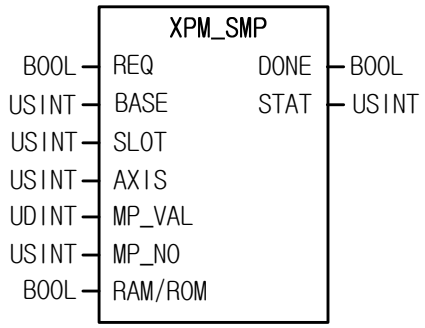
1: 원점 어드레스	6: 원점복귀 드웰타임
2: 원점복귀 고속 속도	7: 원점보정량
3: 원점복귀 저속 속도	8: 원점복귀 재기동 시간
4: 원점복귀 가속 시간	9: 원점복귀 방법
5: 원점복귀 감속 시간	10: 원점복귀 방향

■ 프로그램 예

1. ST

```
INST_XPM_SHP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, HP_VAL:=HP_DINT,
HP_NO:=NO_USINT, ㄱㄱRAM_ROM:=RAM_ROM_BOOL, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

XPM_SMP	적용 기종	발생플래그
수동 운전 파라미터 티칭	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) MP_VAL: 변경할 수동운전 파라미터 값 MP_NO : 변경할 수동운전 파라미터 항목번호 RAM/ROM: 파라미터 저장 방법 0: RAM에 저장, 1: ROM에 저장 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 이 명령은 위치결정 모듈에 수동 운전 파라미터 티칭을 실행하는 명령입니다.
2. 수동 운전 파라미터 티칭 명령으로 수정한 파라미터 값을 ROM 에 저장하려면 수동 운전 파라미터 티칭 명령 시 RAM_ROM 을 1 로 설정하거나 수동 운전 파라미터 티칭 후 파라미터/운전데이터 저장 명령(WRT)을 사용하여 수정한 파라미터 값을 ROM 에 저장하여야 합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 수동운전 파라미터 티칭 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러 6" 이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)
5. 수동 운전 파라미터 항목번호에 설정할 값은 아래와 같습니다.

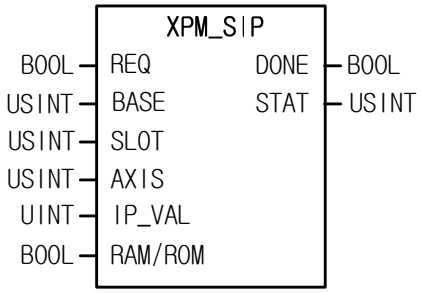
1: 조그 고속 속도
2: 조그 저속 속도
3: 조그 가속 시간
4: 조그 감속 시간
5: 인칭속도

■ 프로그램 예

1. ST

```
INST_XPM_SMP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, MP_VAL:=MP_UDINT,
MP_NO:=NO_USINT, RAM_ROM:=RAM_ROM_BOOL, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

XPM_SIP	적용 기종	발생플래그
외부 신호 파라미터 티칭	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1 축 ~ 4 축) XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축) IP_VAL : 변경할 외부신호 파라미터 값 각 Bit 별로 할당된 신호를 설정함. RAM/ROM: 파라미터 저장 방법 0: RAM 에 저장, 1: ROM 에 저장 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 이 명령은 위치결정 모듈에 외부 신호 파라미터 티칭을 실행하는 명령입니다.
2. 입력 신호 파라미터 티칭 명령으로 수정한 파라미터 값을 ROM 에 저장하려면 외부 신호 파라미터 티칭 명령 시 RAM_ROM 을 1 로 설정하거나 외부 신호 파라미터 티칭 후 파라미터/운전데이터 저장 명령(WRT)을 사용하여 수정한 파라미터 값을 ROM 에 저장하여야 합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 외부신호 파라미터 티칭 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1 축 ~ 4 축), XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축)
   각 입력 신호 설정 영역의 설정값은 아래와 같은 의미를 가진다.
   0: A접점, 1: B접점
   변경할 입력 신호 파라미터 값의 각 Bit에 할당된 신호는 아래와 같습니다.

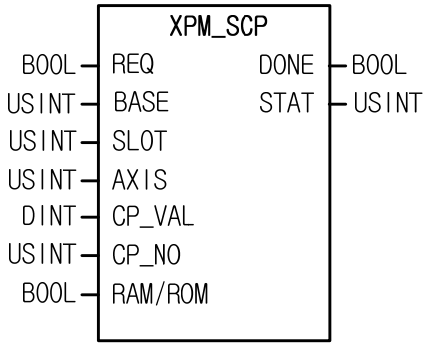
Bit	입력 신호	Bit	입력 신호
0	상한 신호	6	드라이브 레디 신호
1	하한 신호	7	인포지션 신호
2	근사 원점 신호	8	편차카운터 클리어 신호
3	원점 신호	9 ~ 15	-
4	비상 정지/정지 신호		
5	속도/위치 전환 신호		

■ 프로그램 예

1. ST

```
INST_XPM_SIP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, IP_VAL:=IP_WORD,
RAM_ROM:=RAM_ROM_BOOL, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

XPM_SCP	적용 기종	발생플래그
공통 파라미터 티칭	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1 축 ~ 4 축) XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축) CP_VAL : 변경할 공통 파라미터 값 CP_NO : 변경할 공통 파라미터 항목번호 RAM/ROM: 파라미터 저장 방법 0: RAM 에 저장, 1: ROM 에 저장 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 이 명령은 위치결정 모듈에 공통 파라미터 티칭을 실행하는 명령입니다.
2. 공통 파라미터 티칭 명령으로 수정한 파라미터 값을 ROM 에 저장하려면 공통 파라미터 티칭 명령 시 RAM_ROM 을 1 로 설정하거나 공통 파라미터 티칭 후 파라미터/운전데이터 저장 명령(WRT)을 사용하여 수정한 파라미터 값을 ROM 에 저장하여야 합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 공통 파라미터 티칭 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1 축 ~ 4 축), XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축)
   공통 파라미터 항목 번호에 설정할 값은 아래와 같습니다.

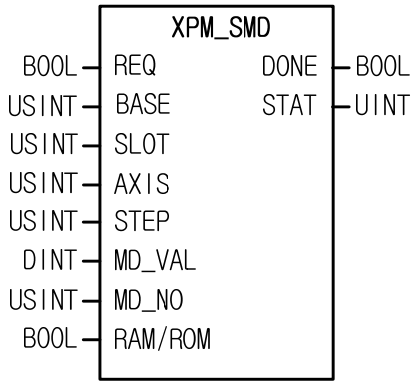
1: 속도오버라이드 방식
2: 엔코더 펄스 입력모드
3: 엔코더 최대값
4: 엔코더 최소값
5: 엔코더 Z 상 Clear
6: 펄스출력 레벨

■ 프로그램 예

1. ST

```
INST_XPM_SCP(REQ:=REQ_BOOL,  BASE:=BASE_USINT,  SLOT:=SLOT_USINT,  AXIS:=AXIS_USINT,  CP_VAL:=CP_DINT,
CP_NO:=NO_USINT,
RAM_ROM:=RAM_ROM_BOOL, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

XPM_SMD	적용 기종	발생플래그
운전 데이터 티칭	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) STEP : 변경할 운전스텝 번호 0 ~ 400 MD_VAL : 변경할 운전데이터 값 MD_NO : 변경할 운전데이터 항목번호 RAM/RPM: 파라미터 저장 방법 0: RAM에 저장, 1: ROM에 저장 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 이 명령은 위치결정 모듈에 운전 데이터 티칭을 실행하는 명령입니다.
2. 운전데이터 설정 명령으로 수정한 파라미터 값을 ROM 에 저장하려면 운전 데이터 티칭 시 RAM/ROM 의 값을 1 로 설정하거나 운전데이터 티칭 후 파라미터/운전 데이터 저장 명령(WRT)을 사용하여 수정한 파라미터 값을 ROM 에 저장하여야 합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 운전 데이터 티칭 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러 6" 이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)
   운전 데이터 항목번호에 설정할 값은 아래와 같습니다.

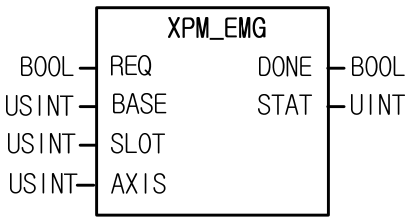
G1: 목표위치	10: 제어방식
2: 원호보간 보조위치	11: 운전방식
3: 운전속도	12: 운전패턴
4: 드웰시간	13: 원호크기
5: M 코드 번호	14: 가속번호
6: 종축설정	15: 감속번호
7: 헬리컬보간 축	16: 원호보간 방법
8: 원호보간 턴 수	17: 원호보간 방향
9: 좌표	

■ 프로그램 예

1. ST

```
INST_APM_SMD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, STEP:=STEP_UINT,
MD_VAL:=MD_DINT, MD_NO:=NO_USINT, RAM_ROM:=RAM_ROM_BOOL, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

XPM_EMG	적용 기종	발생플래그
비상 정지	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

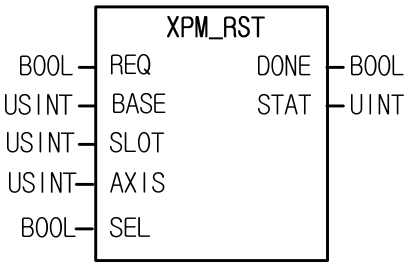
■ 기능

1. 이 명령은 위치결정 모듈에 비상 정지를 실행하는 명령입니다.
2. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 비상정지 지령을 내립니다.
3. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)
4. 긴급 상황으로 운전을 즉시 정지시키고자 할 때 사용되며, 이 지령이 실행되는 축은 비상정지상태가 됩니다.

■ 프로그램 예

1. ST

```
INST_XPM_EMG(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

XPM_RST	적용 기종	발생플래그
에러 리셋	XGI, XGR	-
펑션 블록	설 명	
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) SEL : 출력금지 해제 0: 축 에러 리셋, 1: 공통 에러 리셋 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력	

■ 기능

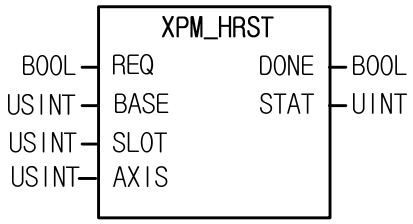
1. 이 위치결정 모듈에 에러 리셋을 실행하는 명령입니다.
2. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 에 지정된 축에 에러 리셋 지령을 내립니다.
3. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)
4. 외부 비상 정지, 상/하한 검출 등의 에러 상태를 해제하거나 또는 파라미터의 설정 범위 초과나 운전 중 에러가 발생하였을 때 발생한 에러를 리셋(Reset)하는데 사용합니다.

■ 프로그램 예

1. ST

```
INST_XPM_RST(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, SEL:=SEL_BOOL,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

XPM_HRST	적용 기종	발생플래그
에러 히스토리 리셋	XGI, XGR	-
펑션 블록	설 명	



입력 REQ : 펑션 블록 실행 요구
BASE : 모듈이 장착된 베이스의 번호를 설정
SLOT : 모듈이 장착된 슬롯의 번호를 설정
AXIS : 지령을 내릴 축 지정
XPM: 1 ~ 4 (1 축 ~ 4 축)
XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축)

출력 DONE : 최초 동작 후 1 을 유지
STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

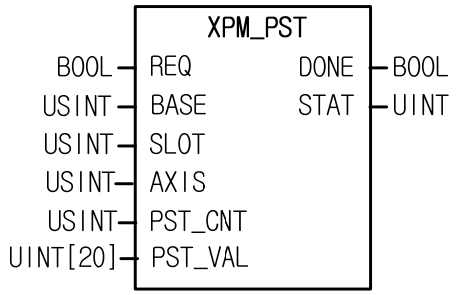
1. 이 위치결정 모듈에 에러 히스토리 리셋을 실행하는 명령입니다.
2. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 에 지정된 축에 에러 에러 히스토리 리셋 지령을 내립니다.
3. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1 축 ~ 4 축), XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축)
4. 파라미터의 설정 범위초과 등 이나 운전 중 에러가 발생하였을 때 발생한 에러 히스토리를 리셋(Reset)하는 데 사용합니다.

■ 프로그램 예

1. ST

```
INST_XPM_HRST(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, DONE=>DONE_BOOL,
STAT=>STAT_UINT);
```

XPM_PST	적용 기종	발생플래그
포인트 운전	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) PST_CMT : 포인트 운전 스텝 수 설정 0 ~ 19 PST_VAL : 포인트 운전 스텝 번호 설정 0 ~ 400 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

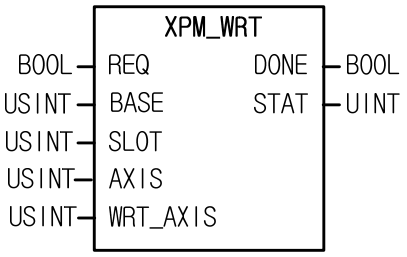
1. 이 명령은 위치결정 모듈에 포인트 운전을 실행하는 명령입니다.
2. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 에 지정된 축에 포인트 운전 기동 지령을 내립니다.
3. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1 축 ~ 4 축), XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축)
4. PTP(Point to Point) 운전시 최대 20 개의 운전 스텝을 설정하여 한번의 지령으로 정지 없이 연속으로 운전할 때 사용합니다.
5. PST_CNT 나 PST_VAL 에 설정값 이외의 값을 설정하면 “에러 6” 이 발생합니다.

■ 프로그램 예

1. ST

```
INST_XPM_PST(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, PST_CNT:=CNT_USINT,
PST_VAL:=ARY_PST, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

XPM_WRT	적용 기종	발생플래그
파라미터/운전 데이터 저장	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) WRT_AXIS : 저장축 설정(각 bit를 set 하여 설정) XPM: 0bit ~ 3bit (1축 ~ 4축) XGF-PN8A/B: 0bit ~ 7bit (1축 ~ 8축) **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

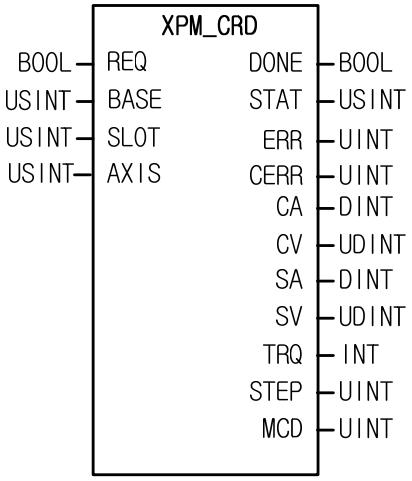
1. 이 명령은 위치결정 모듈에 파라미터/운전 데이터 저장을 실행하는 명령입니다.
2. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS에 지정된 축에 파라미터/운전데이터 저장 지령을 내립니다.
3. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)
4. 지령축에 WRT_AXIS 에 설정된 축의 현재 운전 중인 파라미터와 운전 데이터를 FRAM 에 저장하는 지령을 내립니다.

■ 프로그램 예

1. ST

```
INST_XPM_WRT(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, WRT_AXIS:=WRT_USINT,
DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

XPM_CRD	적용 기종	발생플래그
운전 정보 읽기	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력 ERR : 동작 중 에러 표시 CERR : 공통 에러 표시 CA : 지령 위치 어드레스 표시 CV : 지령 운전 속도 표시 SA : 사용하지 않음. SV : 사용하지 않음 TRQ : 사용하지 않음 STEP : 현재 운전데이터 스텝번호 표시 MCD : 현재 MCode 값 표시

■ 기능

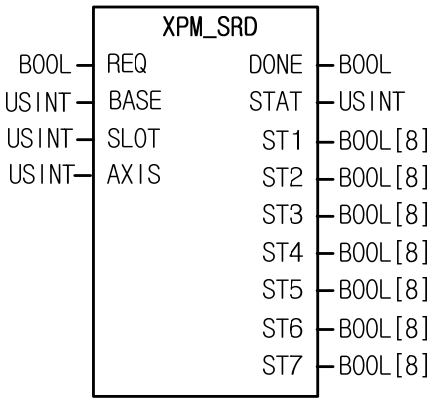
1. 이 명령은 위치결정 모듈의 현재 운전 정보 읽기를 실행하는 명령입니다.
2. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 에 지정된 축에 현재 운전 정보 읽기 지령을 내립니다.
3. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)
4. 설정된 축의 현재 위치 어드레스, 운전 속도, 운전 데이터 번호, MCode 값을 읽어 내어 모니터링 하거나 사용자 프로그램에서 조건으로 이용할 수 있습니다.

■ 프로그램 예

1. ST

```
INST_XPM_CRD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, DONE=>DONE_BOOL,
STAT=>STAT_UINT, ERR=>ERR_UINT, CERR=>CERR_UINT, CA=>CA_DINT, CV=>CV_UDINT, SA=>SA_DINT, SV=>SV_DINT,
TRQ=>TRQ_INT, STEP=>STEP_UINT, MCD=>MCD_UINT);
```

XPM_SRD	적용 기종	발생플래그
운전 상태 읽기	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력 ST1 : 상태 1 ST2 : 상태 2 ST3 : 상태 3 ST4 : 상태 4 ST5 : 상태 5 ST6 : 상태 6 ST7 : 상태 7

■ 기능

1. 이 명령은 위치결정 모듈의 현재 운전 상태 읽기를 실행하는 명령입니다.
2. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 에 지정된 축에 현재 운전 상태 읽기 지령을 내립니다.
3. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러 6" 이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)
4. 현재 운전 상태 비트 읽기 펑션 블록의 출력 변수 ST1 ~ ST7 의 내용은 프로그램에서 반드시 응용해야 하는 중요한 정보들입니다.
5. ST1 ~ ST4 의 각 Bit 가 의미하는 것은 아래와 같습니다.

	Bit	설명	Bit	설명
ST1	[0]	운전중(0: 정지, 1: BUSY)	[4]	원점 결정 상태 (0: 미결정, 1: 완료)
	[1]	Error 상태	[5]	-
	[2]	위치결정 완료	[6]	정지 상태
	[3]	MCode On 신호 (0: Off, 1: On)	[7]	-

	Bit	설명	Bit	설명
ST2	[0]	상한 검출	[4]	가속 중
	[1]	하한 검출	[5]	정속 중
	[2]	비상 정지 상태	[6]	감속 중
	[3]	방향 (0: 정방향, 1: 역방향)	[7]	드웰 중
ST3	[0]	1 축 위치 제어 중	[4]	원호 보간 중
	[1]	1 축 속도 제어 중	[5]	원점 복귀 운전 중
	[2]	직선 보간 중	[6]	위치 동기 운전 중
	[3]	토크제어 중	[7]	속도 동기 운전 중
ST4	[0]	조그 운전 중	[4]	수동 운전 이전 위치로 복귀 중
	[1]	-	[5]	CAM 제어
	[2]	인칭 운전 중	[6]	Feed 제어
	[3]	-	[7]	타원보간

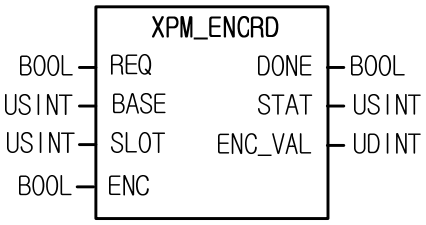
6. ST5 ~ ST7 의 각 Bit 가 의미하는 것은 아래와 같습니다.

	Bit	설명	Bit	설명
ST5	[0]	주축정보 1~4: 1 축~4 축 9: 엔코더	[4]	축 상태 (0: 종축, 1: 주축)
	[1]		[5]	-
	[2]		[6]	-
	[3]		[7]	-
ST6	[0]	비상 정지 신호	[4]	상한 신호
	[1]	-	[5]	하한 신호
	[2]	-	[6]	원점 신호
	[3]	-	[7]	근사 원점 신호
ST7	[0]	속도/위치 제어 전환 신호	[4]	-
	[1]	드라이버 레디 신호	[5]	-
	[2]	인포지션 신호	[6]	-
	[3]	-	[7]	-

■ 프로그램 예

1. ST

```
INST_XPM_SRD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, DONE=>DONE_BOOL,
STAT=>STAT_UINT,
ST1=>ARY_ST1, ST2=>ARY_ST2, ST3=> ARY_ST3, ST4=> ARY_ST4, ST5=> ARY_ST5, ST6=> ARY_ST6, ST7=> ARY_ST7);
```

XPM_ENCRD	적용 기종	발생플래그
엔코더값 읽기	XGI, XGR	-
펑션 블록	설 명	
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 ENC : 엔코더 번호 0: 엔코더 0 1: 사용하지 않음 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력 ENC_VAL : 엔코더의 현재값	

■ 기능

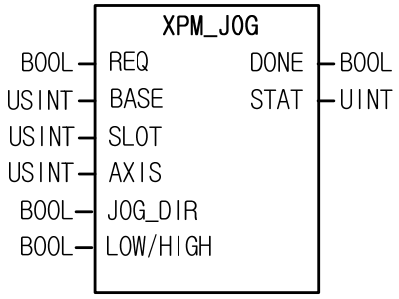
1. 이 명령은 위치결정 모듈에 엔코더값 읽기를 실행하는 명령입니다.
2. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈에 엔코더 값 읽기 지령을 내립니다.

■ 프로그램 예

1. ST

```
INST_XPM_ENCRD(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, ENC:=ENC_BOOL, DONE=>DONE_BOOL,
STAT=>STAT_UINT, ENC_VAL=>ENC_UDINT);
```

XPM_JOG	적용 기종	발생플래그
조그 운전	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) JOG_DIR : 조그 운전시 회전 방향을 설정 0:정방향, 1:역방향 LOW/HIGH : 조그 운전시 조그 속도를 설정 0:조그 저속 운전, 1:조그 고속 운전 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

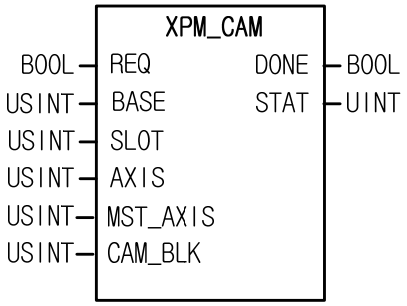
1. 이 명령은 위치결정 모듈에 조그 운전을 실행하는 명령입니다.
2. 테스트를 위한 수동 운전 기능으로 시스템의 동작, 배선상태 검사 및 티칭을 위한 위치 어드레스 확인용으로 사용되며 속도를 고속과 저속으로 구분하여 사용할 수 있습니다.
3. 입력 변수 REQ의 접속 조건이 On일 때 설정된 값에 의해 펄스가 출력되고 Off일 때 정지 됩니다.
4. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS로 지정된 축에 조그 운전 지령을 내립니다.
5. AXIS에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러6"이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)

■ 프로그램 예

1. ST

```
INST_XPM_JOG(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, JOG_DIR:=JOG_BOOL,
LOW_HIGH:=LOW_HIGH_BOOL, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

XPM_CAM	적용 기종	발생플래그
캠 운전	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) MST_AXIS : 주축 설정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) CAM_BLK : 캠 블록 설정 1 ~ 8: 1블록 ~ 8블록 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

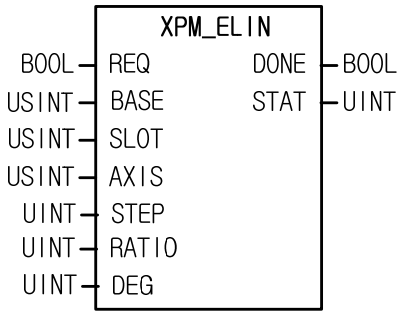
1. 이 명령은 위치결정 모듈에 캠 운전을 실행하는 명령입니다.
2. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS로 지정된 축에 캠 운전 지령을 내립니다.
3. AXIS에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러6"이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)

■ 프로그램 예

1. ST

```
INST_XPM_CAM(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT,
MST_AXIS:=MST_AXIS_USINT, CAM_BLK:=CAM_BLK_USINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

XPM_ELIN	적용 기종	발생플래그
타원 보간 운전	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1 축 ~ 4 축) XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축) STEP : 운전할 스텝번호 0~400 RATIO : 타원 비율 DEG : 운전 각도 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

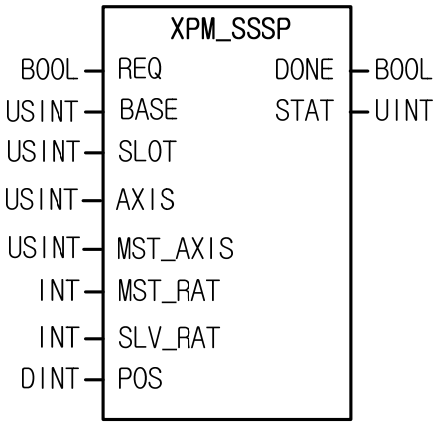
1. 이 명령은 위치결정 모듈에 타원 보간 운전을 실행하는 명령입니다.
2. 축의 STEP에 지정된 스텝을 RATIO에 설정된 비율로 DEG에 설정된 각도만큼 타원보간을 하는 것입니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS로 지정된 축에 타원 보간 운전 지령을 내립니다.
4. AXIS에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러6"이 발생합니다.
   XPM: 1 ~ 4 (1 축 ~ 4 축), XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축)

■ 프로그램 예

1. ST

```
INST_XPM_ELIN(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, STEP:=STEP_UINT,
RATIO:=RATIO_UINT, DEG:=DEG_UINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

XPM_SSSP	적용 기종	발생플래그
위치지정 속도 동기	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 지령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) MST_AXIS : 속도동기 주축 설정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) 9: 엔코더 MST_RAT : 주축의 속도비 설정 1 ~ 32767, -32768 ~ -1 SLV_RAT : 종축의 속도비 설정 1 ~ 32767, -32768 ~ -1 POS : 목표 위치 -2,147,483,648 ~ 2,147,483,647 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

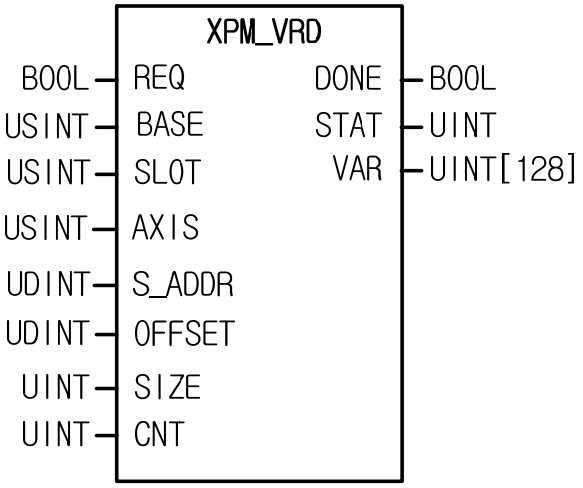
1. 이 명령은 위치결정 모듈에 위치지정 속도 동기를 실행하는 명령입니다.
2. 두 축간에 운전 속도를 설정한 비율로 제어 하고자 할 때 사용합니다. 종축의 위치가 POS 에 지정한 위치가 되면 속도 동기를 끝내고 정지합니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 위치지정 속도 동기 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1 축 ~ 4 축), XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축)
5. MST_AXIS 에는 속도동기의 주축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1 축 ~ 4 축), XGF-PN8A/B: 1 ~ 8 (1 축 ~ 8 축), 9: 엔코더

■ 프로그램 예

1. ST

```
INST_XPM_SSSP(REQ:=REQ_BOOL, BASE:=BASE_USINT, SLOT:=SLOT_USINT, AXIS:=AXIS_USINT, MST_AXIS:=AXIS_USINT,
MST_RAT:=MST_INT, SLV_RAT:=SLV_INT, POS:=POS_DINT, DONE=>DONE_BOOL, STAT=>STAT_UINT);
```

XPM_VRD	적용 기종	발생플래그
가변 데이터 읽기	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 명령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) S_ADDR : 읽을 데이터의 모듈 내부 메모리 선두 번지 0 ~ 53329 OFFSET : 읽을 데이터 블록 간 옵셋:0 ~ 53329 SIZE : 읽을 데이터 블록의 크기:1 ~ 128 CNT : 읽을 데이터 블록의 개수:1 ~ 128 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력 VAR : 읽은 데이터가 저장되는 PLC 디바이스

■ 기능

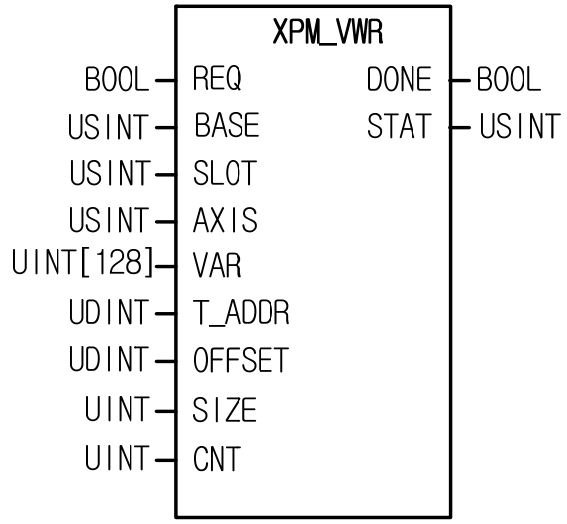
1. 이 명령은 위치결정 모듈에 파라미터, 운전데이터, 캠데이터 직접 읽기 지령을 내리는 명령입니다.
2. 파라미터와 운전데이터, 캠데이터의 모듈 내부 메모리 번지를 직접 지정하여 원하는 데이터를 읽어오는 데 사용할 수 있습니다.
3. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 파라미터, 운전데이터, 캠데이터 중 위치결정 모듈 내부 메모리의 "S_ADDR" 설정된 위치에서부터 "SIZE"만큼의 데이터를 WORD 단위로 "VAR"에 지정된 디바이스로 읽어오는 명령입니다. "CNT"가 2 이상인 경우 "S_ADDR"위치에서부터 "OFFSET"만큼 떨어져 있는 블록들을 "CNT"-1 횟수만큼 차례로 읽어와서 "VAR"에 지정된 디바이스에 저장합니다.
4. 한 명령으로 읽을 수 있는 최대 데이터 크기 (SIZE x CNT)는 128 WORD 입니다.
5. "가변 데이터 읽기" 명령은 운전 중에도 실행할 수 있습니다.
6. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정 값 이외의 값을 설정했을 경우 에러 "6"이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)
7. 읽을 데이터 크기(SIZE x CNT)가 0이거나 128 WORD 를 초과한 경우 STAT에 에러 "11"이 발생합니다.

■ 프로그램 예

1. ST

```
INST_XPM_VRD(REQ:=(*BOOL*),    BASE:=(*USINT*),    SLOT:=(*USINT*),    AXIS:=(*USINT*),    S_ADDR:=(*UDINT*),
OFFSET:=(*UINT*), SIZE:=(*UINT*), CNT:=(*UINT*), DONE=>(*BOOL*), STAT=>(*UINT*), VAR=>(*ARRAY[0..127]_OF_UINT*))
```

XPM_VWR	적용 기종	발생플래그
가변 데이터 쓰기	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 명령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) VAR : 쓸 데이터가 저장되어있는 PLC 디바이스 T_ADDR : 데이터가 쓰여지는 모듈 내부 메모리 선두 번지 0 ~ 53329 OFFSET : 쓸 데이터 블록 간 옵셋 0 ~ 53329 SIZE : 쓸 데이터 블록의 크기 1 ~ 128 CNT : 쓸 데이터 블록의 개수 1 ~ 128 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

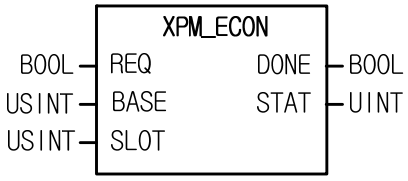
■ 기능

1. 이 명령은 위치결정 모듈에 파라미터, 운전데이터, 캠데이터 직접 쓰기 지령을 내리는 명령입니다.
2. 파라미터와 운전데이터, 캠데이터의 모듈 내부 메모리 번지를 직접 지정하여 원하는 데이터를 쓰는데 사용할 수 있습니다.
3. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈 내부 메모리의 파라미터, 운전데이터, 캠데이터 중 "T_ADDR" 위치에서부터 "SIZE" 만큼의 데이터를 PLC 프로그램에서 "VAR" 에 지정한 데이터들로 WORD 단위 쓰기를 실행하는 명령입니다. 블록 개수 "CNT" 가 2 이상인 경우 "T_ADDR" 에 위치한 블록부터 "OFFSET" 만큼 떨어져 있는 블록들에 나머지 데이터들을 "CNT" -1 횟수만큼 차례로 데이터 쓰기를 실행합니다.
4. 한 명령으로 쓸 수 있는 최대 데이터 크기 (SIZE x CNT)는 128 WORD 입니다.
5. "가변 데이터 쓰기" 명령은 운전 중에는 실행할 수 없습니다.
6. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정 값 이외의 값을 설정했을 경우 에러 "6" 이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)
7. 읽을 데이터 크기(SIZE x CNT)가 0 이거나 128 WORD 를 초과한 경우 STAT 에 에러 "11" 이 발생합니다.
8. 블록 개수 (CNT)가 2 이상이고, 블록 옵셋(OFFSET)이 블록 크기(CNT) 보다 작은 경우 데이터를 쓸 모듈 내부 메모리 블록이 서로 중복되므로 STAT 에 에러 "11" 이 발생합니다.

■ 프로그램 예

1. ST

```
INST_XPM_VWR(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), VAR:=(*ARRAY[0..127]_OF_UINT*),
T_ADDR:=(*UDINT*), OFFSET:=(*UINT*), SIZE:=(*UINT*), CNT:=(*UINT*), DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_ECON	적용 기종	발생플래그
서보 통신 연결	XGI, XGR	-
펑션 블록	설 명	
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력	

■ 기능

1. 이 명령은 위치결정 모듈에 EtherCAT 통신 연결 지령을 내리는 명령입니다.
2. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈에 서보와의 통신을 연결하라는 명령을 내립니다.
3. 서보 드라이브가 정상적으로 연결되면 글로벌 변수의 연결된 축에 해당하는 Bit 가 Set 됩니다.

	글로벌 변수	내 용
1 축	_xxyy_A1_RDY	1 축 운전 준비
2 축	_xxyy_A2_RDY	2 축 운전 준비
3 축	_xxyy_A3_RDY	3 축 운전 준비
4 축	_xxyy_A4_RDY	4 축 운전 준비
5 축	_xxyy_A5_RDY	5 축 운전 준비
6 축	_xxyy_A6_RDY	6 축 운전 준비
7 축	_xxyy_A7_RDY	7 축 운전 준비
8 축	_xxyy_A8_RDY	8 축 운전 준비

(* _xxyy 의 “xx” 는 모듈이 장착되어 있는 베이스번호, “yy” 는 슬롯번호입니다.)

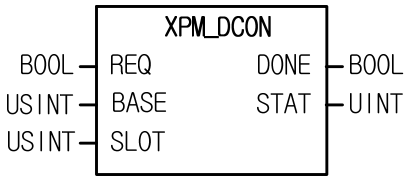
4. 이 명령어는 XGF-PN8A/B 모듈 전용입니다.

■ 프로그램 예

1. ST

```
INST_XPM_ECON(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_DCON	적용 기종	발생플래그
서보 통신 끊기	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 이 명령은 위치결정 모듈에 EtherCAT 통신 끊기 지령을 내리는 명령입니다.
2. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈에 서보와의 통신을 끊으라는 명령을 내립니다.
3. 서보 드라이브와의 연결이 끊기면 글로벌 변수의 연결된 축에 해당하는 Bit 가 Clear 됩니다.

	글로벌 변수	내 용
1 축	_xxyy_A1_RDY	1 축 운전 준비
2 축	_xxyy_A2_RDY	2 축 운전 준비
3 축	_xxyy_A3_RDY	3 축 운전 준비
4 축	_xxyy_A4_RDY	4 축 운전 준비
5 축	_xxyy_A5_RDY	5 축 운전 준비
6 축	_xxyy_A6_RDY	6 축 운전 준비
7 축	_xxyy_A7_RDY	7 축 운전 준비
8 축	_xxyy_A8_RDY	8 축 운전 준비

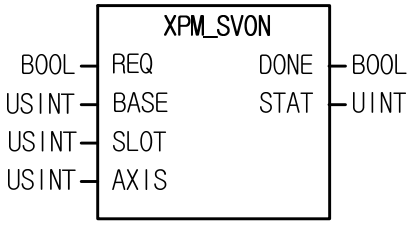
(*_xxyy 의 “xx” 는 모듈이 장착되어 있는 베이스번호, “yy” 는 슬롯번호입니다.)

4. 이 명령어는 XGF-PN8A/B 모듈 전용입니다.

■ 프로그램 예

1. ST

```
INST_XPM_DCON(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_SVON	적용 기종	발생플래그
서보 온	XGI, XGR	-
펑션 블록	설 명	
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 명령을 내릴 축 지정 1 ~ 8: 1축 ~ 8축 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력	

■ 기능

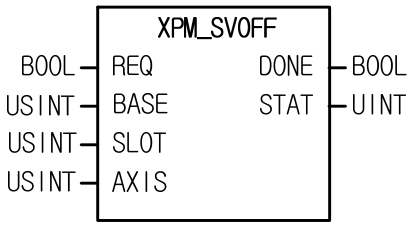
1. 이 명령은 위치결정 모듈에 서보 온 지령을 내리는 명령입니다.
2. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 서보 온 명령을 내립니다.
3. 모터를 기동하기 위해서는 서보 온 신호가 “ON”이 되어야 합니다.
4. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6”이 발생합니다.
   1 ~ 8: 1축 ~ 8축
5. 이 명령어는 XGF-PN8A/B 모듈 전용입니다.

■ 프로그램 예

1. ST

```
INST_XPM_SVON(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_SVOFF	적용 기종	발생플래그
서보 오프	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 명령을 내릴 축 지정 1 ~ 8: 1축 ~ 8축 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

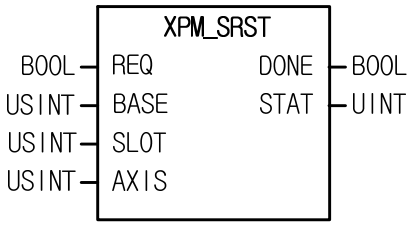
■ 기능

1. 이 명령은 위치결정 모듈에 서보 오프 지령을 내리는 명령입니다.
2. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 서보 오프 명령을 내립니다.
3. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6”이 발생합니다.
   1 ~ 8: 1축 ~ 8축
4. 이 명령어는 XGF-PN8A/B 모듈 전용입니다.

■ 프로그램 예

1. ST

```
INST_XPM_SVOFF(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_SRST	적용 기종	발생플래그
서보 에러 리셋	XGI, XGR	-
펑션 블록	설 명	
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 명령을 내릴 축 지정 1 ~ 8: 1축 ~ 8축 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력	

■ 기능

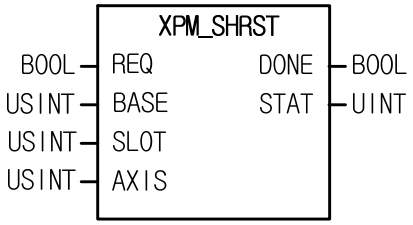
1. 이 명령은 위치결정 모듈에 서보 에러 리셋 지령을 내리는 명령입니다.
2. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 서보 에러 리셋 명령을 실행합니다.
3. 서보 드라이브 알람이 발생한 원인을 제거하지 않고 서보 에러 리셋 명령을 내리면 서보 드라이브의 알람이 Clear 되지 않을 수 있습니다. 따라서, 서보 드라이브 알람이 발생한 원인을 제거하고 서보 에러 리셋 명령을 수행하여야 합니다.
4. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   1 ~ 8: 1축 ~ 8축
5. 이 명령어는 XGF-PN8A/B 모듈 전용입니다.

■ 프로그램 예

1. ST

```
INST_XPM_SRST(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_SHRST	적용 기종	발생플래그
서보 에러 히스토리 리셋	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 명령을 내릴 축 지정 1 ~ 8: 1축 ~ 8축 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

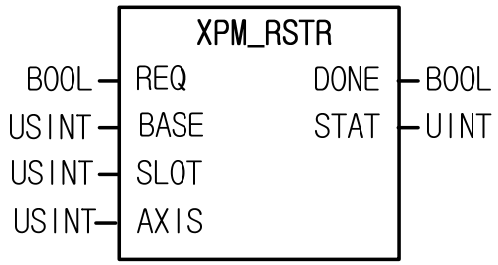
1. 이 명령은 위치결정 모듈에 서보 에러 히스토리 리셋 지령을 내리는 명령입니다.
2. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 서보 에러 히스토리 리셋 명령을 내립니다.
3. 모듈에 연결된 서보 중 선택된 축에 해당하는 서보에 발생한 알람 이력을 리셋하는 명령을 내립니다.
4. 서보 드라이브에서는 최대 10 개까지 서보 알람 이력을 저장하고 있습니다.
5. AXIS 에는 명령을 내릴 축을 설정하며, 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   1 ~ 8: 1 축 ~ 8 축
6. 이 명령어는 XGF-PN8A/B 모듈 전용입니다.

■ 프로그램 예

1. ST

```
INST_XPM_SHRST(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*))
```

XPM_RSTR	적용 기종	발생플래그
재기동	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 명령을 내릴 축 지정 XPM: 1 ~ 4 (1축 ~ 4축) XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축) **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

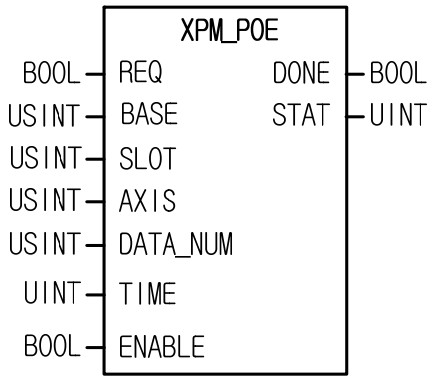
1. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS(명령축)에
   재기동 명령을 내립니다.
2. 운전 중 감속정지한 축을 재기동 할 때 사용되며, 이 명령이 실행되는 축은 이전 운전 정보를 가지고 다시 운전하게
   됩니다.
3. 축이 감속정지 후 재기동을 하기 전에 다른 운전을 수행하였다면 재기동 명령은 수행되지 않습니다.
4. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   XPM: 1 ~ 4 (1축 ~ 4축), XGF-PN8A/B: 1 ~ 8 (1축 ~ 8축)

■ 프로그램 예

1. ST

```
INST_XPM_RSTR(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*));
```

XPM_POE	적용 기종	발생플래그
설정 위치 출력 허용/금지	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 명령을 내릴 축 지정, 1 ~ 4: 1 축 ~4 축 DATA_NUM : 설정 위치 출력 데이터 개수(0~50) TIME : 설정 위치 출력 유지 시간 (단위: ms) 0~65,535ms ENABLE : 설정 위치 출력 허용/금지 0: 금지, 1: 허용 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

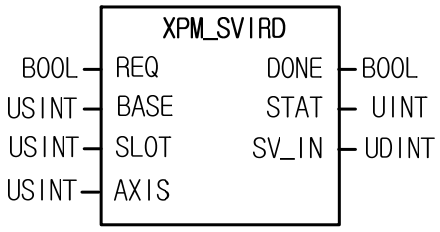
1. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 설정 위치 출력 기능을 허용/금지하는 명령입니다.
2. 설정 위치 출력 기능을 허용하고 현재위치가 위치 출력 설정 위치 영역에 지정된 위치가 되면 위치결정 모듈의 편차 카운트 클리어/설정위치출력 핀으로 신호가 출력됩니다.
3. DATA_NUM 에 설정 위치 출력 신호의 위치 출력 설정 위치 데이터 개수를 설정합니다. 데이터 개수는 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 11 이 발생합니다.
   0 ~ 50 (데이터 개수가 0인 경우는 ENABLE 를 0: 금지로 설정한 것과 같습니다.)
4. 설정 위치 출력 신호는 TIME 에 설정된 시간 동안 유지된 후 OFF 됩니다.
5. 설정 위치 출력 기능을 금지로 하고 명령을 내리면 현재 출력되던 신호가 바로 OFF 가 됩니다.
6. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   1 ~ 4: 1 축 ~ 4 축
7. 이 명령어는 XPM 모듈 전용입니다.

■ 프로그램 예

1. ST

```
INST_XPM_POE(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DATA_NUM:=(*USINT*), TIME:=(*UINT*),
ENABLE:=(*BOOL*), DONE=>(*BOOL*), STAT=>(*UINT*));
```

XPM_SVIRD	적용 기종	발생플래그
서보 외부 입력 정보 읽기	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 명령을 내릴 축 지정 1 ~ 8: 1축 ~ 8축 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력 SV_IN: 서보 입력 신호 정보

■ 기능

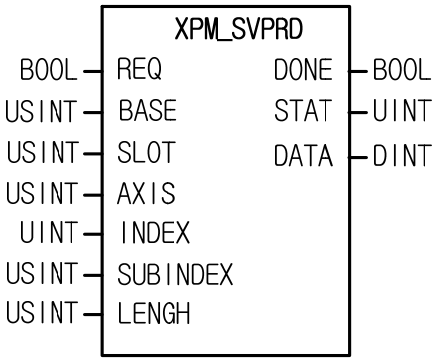
1. 이 명령은 BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS로 지정된 축에 서보 외보 입력 정보 읽기 명령을 내립니다.
2. 모듈에 연결된 서보 중 선택된 축에 해당하는 서보 드라이브의 입력 신호 상태를 읽어오는 명령입니다.
3. SV_IN에 읽어온 입력 신호 상태 정보가 출력됩니다.
4. AXIS 에는 명령을 내릴 축을 설정하며, 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6”이 발생합니다.
   1 ~ 8: 1축 ~ 8축
5. 이 명령어는 XGF-PN8B 모듈 전용입니다.

■ 프로그램 예

1. ST

```
INST_XPM_SVIRD(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*),
SV_IN=>(*UDINT*));
```

XPM_SVPRD	적용 기종	발생플래그
서보 드라이브 파라미터 읽기	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 명령을 내릴 축 지정 1 ~ 8: 1축 ~ 8축 INDEX: 변경할 서보 파라미터 Object Index SUBINDEX: 변경할 서보 파라미터 Object sub-index LENGTH: 변경할 서보 파라미터 Object 크기 1 ~ 4 Byte **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력 DATA : 읽은 서보파라미터 데이터 값

■ 기능

1. 이 명령은 위치결정 모듈에 연결된 서보드라이브의 파라미터 (CoE Object) 값을 읽는 펑션 블록입니다.
2. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 서보 파라미터 읽기 명령을 내립니다.
3. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 INDEX, SUBINDEX 로 지정된 서보파라미터 Object 에 LENGTH 크기의 값을 읽어서 DATA 에 저장합니다.
4. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 "에러 6" 이 발생합니다.
   1 ~ 8: 1 축 ~ 8 축
5. INDEX 값은 아래와 같이 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 STAT 에 에러 "11" 이 발생합니다.

설정값	내용
0x1000 ~ 0x1FFF	Communication Profile Area
0x2000 ~ 0x5FFF	Manufacturer Specific Profile Area
0x6000 ~ 0x9FFF	Standardized Device Profile Area

6. SUBINDEX 에 설정할 수 있는 값은 아래와 같습니다. 설정값 이외의 값을 설정했을 경우 STAT 에 에러 "11" 이 발생합니다.

설정값	내용
0x0~0xFF	서보파라미미터 Object Subindex

7. LENGTH 에 설정할 수 있는 값은 아래와 같습니다. 설정값 이외의 값을 설정했을 경우 STAT 에 에러 "11" 이 발생합니다.

설정값	내용
1~4	서보파라미터 Object Byte Length

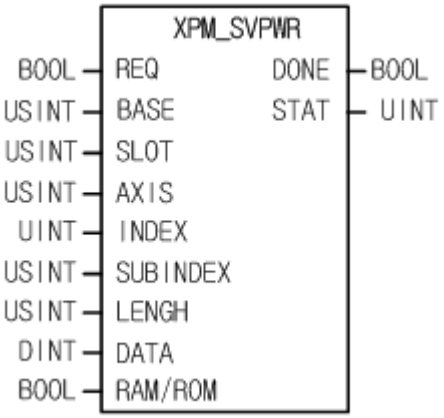
8. 이 명령어는 XGF-PN8B 모듈 전용입니다.

■ 프로그램 예

1. ST

```
INST_XPM_SVPRD(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), INDEX:=(*UINT*),
SUBINDEX:=(*USINT*), LENGH:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*), DATA=>(*DINT*));
```

XPM_SVPWR	적용 기종	발생플래그
서보 드라이브 파라미터 쓰기	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 명령을 내릴 축 지정 1 ~ 8: 1 축 ~ 8 축 INDEX: 변경할 서보 파라미터 Object Index SUBINDEX: 변경할 서보 파라미터 Object sub-index LENGTH: 변경할 서보 파라미터 Object 크기 1 ~ 4 Byte DATA : 변경할 서보 파라미터 값 RAM/ROM : 파라미터 저장 방법 0: RAM 에 저장, 1: ROM 에 저장 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 이 명령은 위치결정 모듈에 연결된 서보 드라이브의 파라미터 (CoE Object)를 변경하는 펑션 블록입니다.
2. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 서보 파라미터 쓰기 명령을 내립니다.
3. 서보 파라미터 쓰기 명령으로 서보드라이브 내부의 ROM 에 저장하려면 RAM/ROM 에 1 을 설정하여 명령을 실행하거나, RAM/ROM 에 0 을 설정하여 쓰기를 실행한 후 추후에 XPM_SVSAVE 명령으로 서보드라이브 EEPROM 저장을 실행 하여야 합니다.
4. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 INDEX, SUBINDEX 로 지정된 서보파라미터 Object 에 LENGTH 크기의 DATA 를 저장합니다.
5. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.

   1 ~ 8: 1 축 ~ 8 축

6. INDEX 값은 아래와 같이 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 STAT 에 에러 “11” 이 발생합니다.

설정값	내용
0x2000 ~ 0x5FFF	Manufacturer Specific Profile Area
0x6000 ~ 0x9FFF	Standardized Device Profile Area

7. SUBINDEX 에 설정할 수 있는 값은 아래와 같습니다. 설정값 이외의 값을 설정했을 경우 STAT 에 에러 "11" 이 발생합니다.

설정값	내용
0x0~0xFF	서보파라미미터 Object Subindex

8. LENGTH 에 설정할 수 있는 값은 아래와 같습니다. 설정값 이외의 값을 설정했을 경우 STAT 에 에러 "11" 이 발생합니다.

설정값	내용
1~4	서보파라미터 Object Byte Length

9. RAM/ROM 에 설정할 수 있는 값은 아래와 같습니다.

설정값	티칭 방법
0	RAM 티칭
1	ROM 티칭

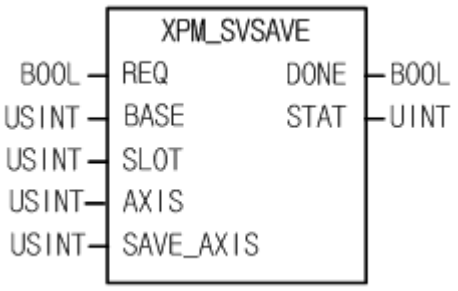
10. 이 명령어는 XGF-PN8B 모듈 전용입니다.

■ 프로그램 예

1. ST

```
INST_XPM_SVPWR(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), INDEX:=(*UINT*),
SUBINDEX:=(*USINT*), LENGTH:=(*USINT*), DATA:=(*DINT*), RAM_ROM:=(*BOOL*), DONE=>(*BOOL*), STAT=>(*UINT*));
```

XPM_SVSAVE	적용 기종	발생플래그
서보 파라미터 저장	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 명령을 내릴 축 지정 1 ~ 8: 1 축 ~ 8 축 SAVE_AXIS: 저장 축 지정 각 bit 를 set 하여 지정(bit0~7: 1 축~8 축) **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

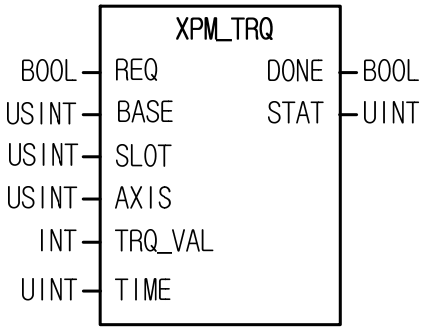
1. 위치결정 모듈에 연결된 서보 드라이브의 파라미터를 서보드라이브 내부의 EEPROM 에 저장하는 펑션 블록입니다.
2. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 서보 드라이브 파라미터 저장 명령을 내립니다.
3. 모듈에 연결된 서보 중 선택된 서보 드라이브의 파라미터를 서보드라이브 내부의 EEPROM 에 저장하도록 하는 명령입니다.
4. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6”이 발생합니다. 명령축은 실제 서보 파라미터가 저장되는 축과는 별개이며 명령축의 서보 파라미터를 저장하려면 SAVE_AXIS 에 해당 축의 BIT set 하여 설정하여야 합니다.
   1 ~ 8: 1 축 ~ 8 축
5. SAVE_AXIS 에 저장하고자 하는 서보 드라이브의 축을 비트로 설정합니다. 설정 값이 0 인 경우에는 “에러 11”이 발생합니다.
   Bit 0 ~ 7 : 1 축 ~ 8 축
6. 이 명령어는 XGF-PN8B 모듈 전용입니다.

■ 프로그램 예

1. ST

```
INST_XPM_SVSAVE(REQ:=(*BOOL*),  BASE:=(*USINT*),  SLOT:=(*USINT*),  AXIS:=(*USINT*),  SAVE_AXIS:=(*USINT*),
DONE=>(*BOOL*), STAT=>(*UINT*));
```

XPM_TRQ	적용 기종	발생플래그
토크 제어	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 명령을 내릴 축 지정 1 ~ 8: 1 축 ~ 8 축 TRQ_VAL : 토크값 (단위: %, -32768 ~ 32767) TIME : 토크 기울기 (단위: ms, 0~65535ms) **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

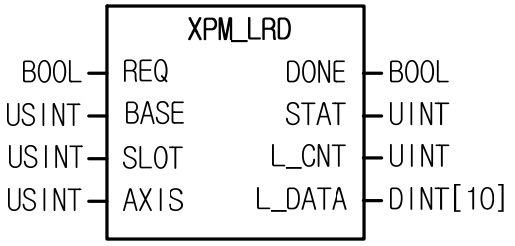
1. 이 명령은 BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS(명령축)에 토크제어를 수행하게 하는 명령입니다.
2. 토크값 및 토크 기울기를 설정하고 명령을 내리면 토크제어가 수행됩니다.
3. TRQ_VAL 에 운전하려는 토크값(%)을 설정합니다. 토크 값은 정격 토크에 대한 %로 동작합니다. (1 = 정격토크의 1%)
   예를 들어 200%의 토크로 토크제어를 수행하려면 TRQ_VAL 에 200 을 설정합니다.
   ※ 토크 값의 허용 범위는 연결된 서보드라이브 종류에 따라 다릅니다. 일반적으로, 목표 토크 값은 최대 토크 설정으로 제한됩니다.
4. TIME 은 목표 토크까지 도달하는 시간을 설정합니다. 명령이 수행되면 이 기울기로 TRQ_VAL 에 설정된 토크값까지 토크가 증가하게 됩니다.
5. 해당 축이 토크 제어 이외의 운전 중인 경우에는 수행할 수 없습니다.
6. AXIS 에는 명령을 내릴 축을 설정하며, 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   1 ~ 8: 1 축 ~ 8 축
7. 이 명령어는 XGF-PN8A/B 모듈 전용입니다.

■ 프로그램 예

1. ST

```
INST_XPM_TRQ(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), TRQ_VAL:=(*INT*), TIME:=(*UINT*), DONE=>(*BOOL*), STAT=>(*UINT*));
```

XPM_LRD	적용 기종	발생플래그
래치 위치 데이터 읽기	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 명령을 내릴 축 지정 1 ~ 8: 1 축 ~ 8 축 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력 L_CNT : 래치 위치 데이터 개수 L_DATA : 래치 위치 데이터 1 ~ 10

■ 기능

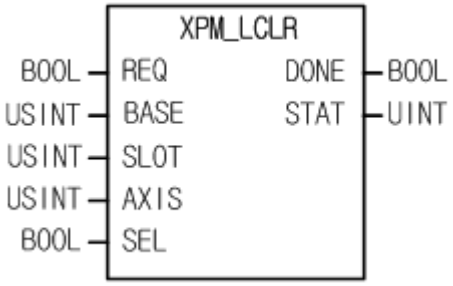
1. 이 명령은 위치결정 모듈의 외부 래치 명령 신호에 의해 저장된 래치된 데이터 개수 및 래치 위치 데이터를 읽고자 하는 경우에 사용되는 명령입니다.
2. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축의 래치 데이터를 읽어서 래치된 위치 데이터 개수를 L_CNT 에 래치 위치 데이터들은 L_DATA 에 저장합니다.
3. AXIS 에는 명령을 내릴 축을 설정하며 1 ~ 8 중 하나를 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
4. 이 명령어는 XGF-PN8A/B 모듈 전용입니다.

■ 프로그램 예

1. ST

```
INST_XPM_LRD(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), DONE=>(*BOOL*), STAT=>(*UINT*),
L_CNT=>(*UINT*), L_DATA=>(*ARRAY[0..9]_OF_UDINT*));
```

XPM_LCLR	적용 기종	발생플래그
래치 리셋	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 명령을 내릴 축 지정 1 ~ 8: 1 축 ~ 8 축 SEL : 래치 리셋 항목 선택 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

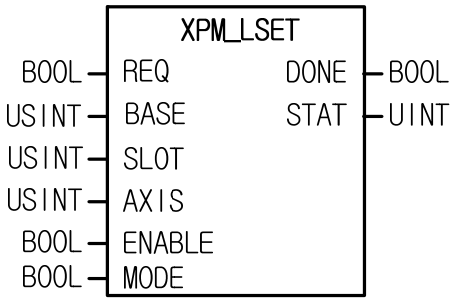
1. 이 명령은 위치결정 모듈에 저장된 래치된 데이터 개수 및 래치 위치 데이터 또는 래치 완료 상태를 초기화 하고자 하는 경우에 사용되는 명령입니다.
2. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 SEL 에 지정된 항목에 따라 래치 리셋 명령을 내립니다.
3. SEL 에 지정된 래치 리셋 항목에 따라 리셋되는 항목은 다음과 같습니다.
   0: 래치 완료 상태 리셋
   1: 래치 데이터 및 래치 완료 상태 리셋
4. SEL 에 1 을 설정하여 래치 리셋 명령을 실행 한 후 래치 위치 데이터 읽기 명령(XPM_LRD)으로 래치 위치 데이터를 읽으면 데이터는 모두 0 이 됩니다.
5. AXIS 에는 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정 값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   1 ~ 8: 1 축 ~ 8 축
6. 이 명령어는 XGF-PN8A/B 모듈 전용입니다.

■ 프로그램 예

1. ST

```
INST_XPM_LCLR(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), SEL:=(*BOOL*), DONE=>(*BOOL*), STAT=>(*UINT*));
```

XPM_LSET	적용 기종	발생플래그
래치 설정	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 명령을 내릴 축 지정 1 ~ 8: 1 축 ~ 8 축 ENABLE : 래치 허용/금지 MODE : 래치 모드 **출력** DONE : 최초 동작 후 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

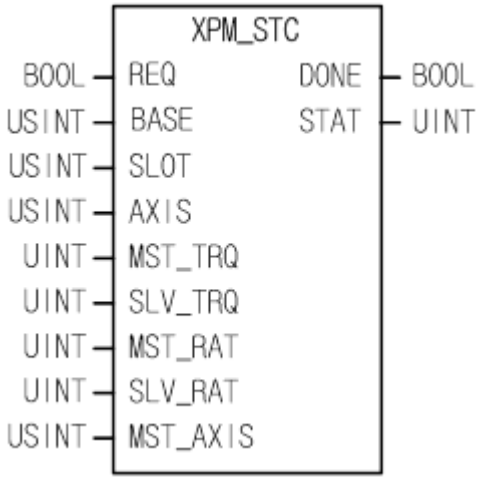
1. 이 명령은 위치결정 모듈의 외부 래치 기능을 허용/금지 하거나 래치 모드를 설정하기 위해 사용되는 명령입니다.
2. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 ENABLE 에 지정된 항목에 따라 래치 기능을 허용/금지하고, MODE 에 지정된 항목에 따라 래치 모드를 설정하는 명령을 내립니다.
3. ENABLE 에 지정된 래치 허용/금지 항목에 따른 동작은 다음과 같습니다.
   0: 래치 금지
   1: 래치 허용
4. MODE 에 지정된 래치 모드 항목에 따른 동작은 다음과 같습니다.
   0: 단일 트리거
   1: 연속 트리거
5. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정 값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.
   1 ~ 8 : 1 축 ~ 8 축
6. 이 명령어는 XGF-PN8B 모듈 전용입니다.

■ 프로그램 예

1. ST

```
INST_XPM_LSET(REQ:=(*BOOL*), BASE:=(*USINT*), SLOT:=(*USINT*), AXIS:=(*USINT*), ENABLE:=(*BOOL*), MODE:=(*BOOL*),
DONE=>(*BOOL*), STAT=>(*UINT*));
```

XPM_STC	적용 기종	발생플래그
토크 동기	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 BASE : 모듈이 장착된 베이스의 번호를 설정 SLOT : 모듈이 장착된 슬롯의 번호를 설정 AXIS : 명령을 내릴 축 지정 1 ~ 8: 1축 ~ 8축 MST_TRQ : 주축의 토크비 설정 0 ~ 65535 SLV_TRQ : 종축의 토크비 설정 0 ~ 65535 MST_RAT : 주축의 속도비 설정 0 ~ 65535 SLV_RAT : 종축의 속도비 설정 0 ~ 65535 MST_AXIS : 토크 동기 주축 설정 1 ~ 8 : 1축 ~ 8축 **출력** DONE : 최초 동작 후 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

■ 기능

1. 위치결정 모듈에 연결된 서보드라이브의 해당 축에 토크동기 명령을 내리는 펑션 블록입니다.
2. BASE(위치결정 모듈의 베이스번호)와 SLOT(위치결정 모듈의 슬롯번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 토크 동기 명령을 내립니다.
3. 명령을 수행하는 축은 MST_AXIS 에 설정된 축을 주축으로 토크동기 운전을 합니다.
4. 명령을 수행하는 축은 MST_TRQ, SLV_TRQ 에 설정된 토크비 및 MST_RAT, SLV_RAT 에 설정된 속도비로 토크동기 운전을 합니다.

   종축 토크 = (SLV_TRQ/MST_TRQ) * 주축 토크

   종축 토크동기 속도 = (SLV_RAT/MST_RAT) * 주축 속도
5. AXIS 에 명령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6”이 발생합니다.

   1 ~ 8: 1축 ~ 8축
6. MST_AXIS 에는 토크 동기의 주축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 11”이 발생합니다.

   1 ~ 8: 1축 ~ 8축

제12장 확장 펑션

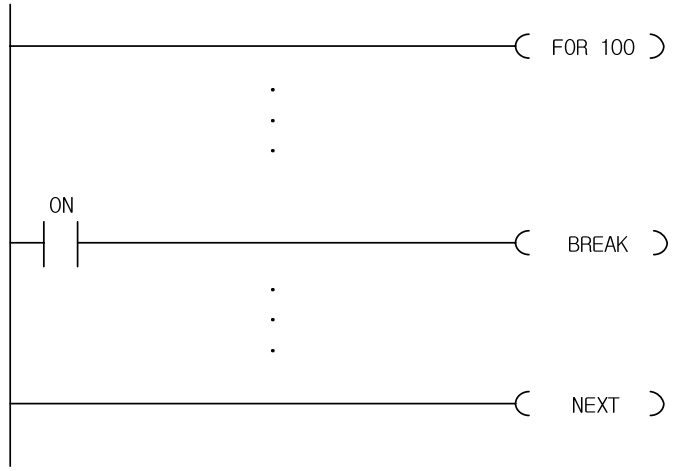
1. 각각의 확장 펑션에 대한 설명입니다.
2. 사용자 프로그램 수행 도중 특정한 부분의 프로그램 처리(FOR ~ NEXT, CALL 명령 등을 사용)를 할 때 사용합니다.

FOR/NEXT/BREAK	적용 기종	발생플래그
루프 명령	XGI, XGR, XEC	-
펑 션	설 명	
FOR | N	FOR ~ NEXT 구간을 n 번 실행	
NEXT		
BREAK	FOR ~ NEXT 구간을 빠져 나옴	

■ 기능

1. PLC 가 RUN 모드에서 FOR 을 만나면 FOR ~ NEXT 명령간의 처리를 n 회 실행한 후 NEXT 명령의 다음 스텝을 실행합니다.
2. n 은 1 ~ 65,535 까지 지정 가능합니다.
3. FOR ~ NEXT 의 가능한 NESTING 개수는 16 개 입니다.
4. FOR ~ NEXT 루프를 빠져 나오는 방법은 BREAK 명령을 사용합니다.
5. 스캔 시간이 길어 질 수 있으므로 WDT 설정치를 넘지 않도록 주의하시기 바랍니다.

■ 프로그램 예



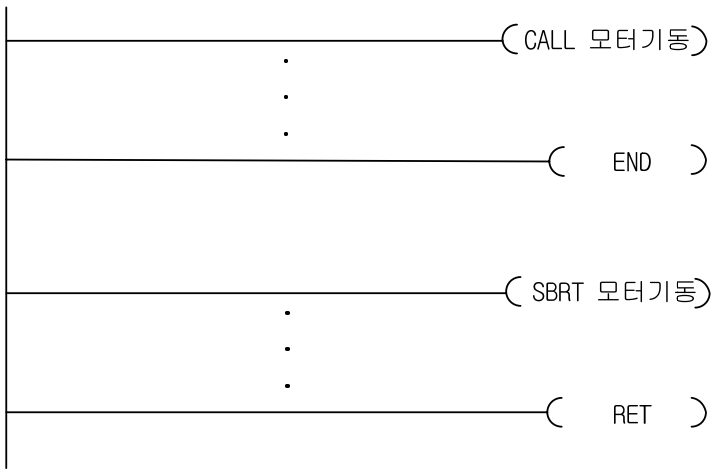
(1) 프로그램이 실행되면 FOR ~ NEXT 에 의해서 루프가 100 번 실행됩니다.
(2) 100 번의 루프를 실행하지 않고 중간에 루프를 빠져 나오려면 접점을 On 시켜서 BREAK 명령을 실행하면 루프를 중간에 빠져나올 수 있습니다.

CALL/SBRT/RET	적용 기종	발생플래그
호출 명령	XGI, XGR, XEC	-
펑 션	설 명	
CALL NAME	SBRT 루틴 호출	
SBRT NAME	CALL 에 의해 호출될 루틴 지정	
RET	RETURN	

■ 기능

1. 프로그램 수행 중 입력 조건이 성립하면 CALL n 명령에 따라 해당 SBRT n ~ RET 명령 사이의 프로그램을 수행합니다.
2. CALL n 은 중첩되어 사용 가능하며 반드시 SBRT n ~ RET 명령 사이에 프로그램은 END 명령 뒤에 있어야 합니다.
3. SBRT 내에서 다른 SBRT 를 CALL 하는 것이 가능합니다. 대신 SBRT 내에서는 END 명령을 사용하지 않습니다.
4. FOR ~ NEXT 루프를 빠져 나오는 방법은 BREAK 명령을 사용합니다.

■ 프로그램 예



(1) 프로그램이 실행 도중 CALL 명령어를 만나면 모터기동이라는 이름의 SBRT 를 호출하게 됩니다.
(2) SBRT 명령어는 반드시 END 뒤에 위치하여야 합니다.
(3) 모터기동이라는 SBRT 가 호출되면 해당 SBRT 내의 프로그램이 실행되며 RET 를 만나면 다시 모터기동이라는 CALL 을 실행한 자리로 되돌아 갑니다.

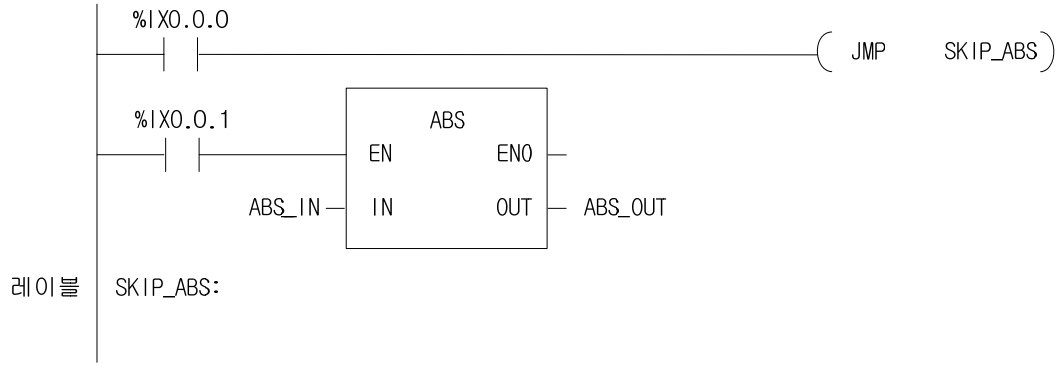
JMP	적용 기종	발생플래그
분기 명령	XGI, XGR, XEC	-
펑 션	설 명	
JMP LABLE	LABLE 위치로 점프	

■ 기능

1. JMP (레이블) 명령의 입력 접점이 On 되면 지정 레이블(LABLE) 이후로 Jump 하며 JMP 와 레이블 사이의 모든 명령은 처리되지 않습니다.
2. 레이블은 중복되게 사용할 수 없습니다. JMP 는 중복사용 가능합니다.
3. 비상사태 발생시 처리해서는 안 되는 프로그램을 JMP 와 레이블 사이에 넣으면 좋습니다.

■ 프로그램 예

(1) 입력 신호 %IX0.0.0 을 On 하였을 시 ABS 펑션을 실행하지 않는 프로그램



INIT_DONE	적용 기종	발생플래그
초기화 태스크 종료 명령	XGI, XGR, XEC	-
펑 션	설 명	
INIT_DONE	초기화 태스크 종료	

■ 기능

1. 초기화 태스크를 종료시키는 명령어입니다.
2. 초기화 태스크 프로그램 작성시에는 반드시 이 명령어를 사용해서 초기화 태스크 프로그램을 종료시켜야 합니다. 그렇지 않을 경우, 초기화 태스크 프로그램을 종료할 수 없게 되고 스캔 프로그램으로 진입할 수 없습니다.

■ 프로그램 예

(1) %IX0.0.0 접점이 On 되면 초기화 태스크가 종료되는 프로그램.

END	적용 기종	발생플래그
종료 명령	XGI, XGR, XEC	-
펑 션	설 명	
	프로그램의 종료	

■ 기능

1. 프로그램 종료를 표시합니다.
2. END 명령 처리 후 프로그램의 처음으로 돌아가 처리합니다.

제13장 프로세스 제어 라이브러리

프로세스 제어, 데이터 처리, 산술 명령어, 데이터 측정, 데이터 생성에 관련된 프로세스 제어 라이브러리를 설명합니다.

13.1. 프로세스 제어 라이브러리

1) STAT
프로세스 제어 라이브러리의 일부 펑션 및 펑션 블록에는 에는 STAT이 존재하는데 이것은 해당 명령어의 각종 이상 상태를 알리기 위해서 사용합니다. STAT이 0 이외의 값으로 출력될 경우 명령어 설정상에 이상 상태가 감지된 경우이며 STAT코드의 내용은 다음과 같습니다.

STAT	명칭	발생시 동작	내용
1	T_s 에러	스캔주기 동작	T_s 설정이 현재 스캔 시간보다 빨라서 설정한 대로 동작할 수 없을 때 우선 가능한 가장 빠른 시간으로 연산하고 이를 표시합니다.
2	X_min, X_max 역전	동작 정지 출력 리셋	입력값은 보통 X로 설계되었으며 이를 최대/최소로 제한하는 과정에서 X_min 값이 X_max값보다 큰 경우를 표시합니다.
4	Y_min, Y_max 역전	동작 정지 출력 리셋	출력값은 보통 Y로 설계되었으며 이를 최대/최소로 제한하는 과정에서 Y_min 값이 Y_max값보다 큰 경우를 표시합니다.
8	기타 설정 에러	동작 정지 출력 리셋	위의 내용에 해당하지 않는 각종 설정이 잘못 되었을 경우를 나타냅니다.

위의 STAT중 둘 이상의 경우에 해당이 되면 두 STAT의 합이 출력됩니다. 즉, X_min, X_max가 역전되어 STAT에 2를 출력해야 하고, 동시에 Y_min, Y_max 또한 역전되어 STAT에 4를 출력해야 하는 상황이 된다면 STAT에는 둘의 합인 6이 출력됩니다.
STAT이 1인 T_s에러를 제외한 에러들은 펑션 또는 펑션블록의 동작을 중지 시키고 0을 출력하며 DONE 과 ENO가 있다면 Off 시킵니다.

2) T_s
일부 명령어에 있는 T_s는 명령어의 연산 주기를 나타내며 T_s를 설정하면 명령어는 T_s 시간마다 동작을 하게 됩니다. 명령어에 접근했을 때 이전에 연산한 시각과 현재의 시각을 비교해서 T_s 이상의 시간이 지났을 때 연산을 하는 구조로서 시간상 오차E(T_s)를 가지며 연산 시간 오차가 발생하면 그 다음 연산 주기에 반영하므로 보통의 경우 시간 오차가 누적되지는 않습니다.

$$0 \leq E(T\_s) < T_{scan}$$

단, $T\_s < T_{scan}$ 인 경우, CPU의 스캔 시간보다 T_s가 더 빠르므로 CPU는 T_s마다 해당 명령어를 수행할 수 없습니다. 이러한 경우 T_s오차가 누적되며 명령어는 누적된 오차를 해결하기 위해서 매 스캔마다 연산하고 STAT값에 1을 출력합니다. 따라서 T_s를 0으로 설정하면 매 스캔마다 해당 명령어를 처리하게 됩니다.

3) 동일한 최대 제한값과 최소 제한값의 설정
프로세스 라이브러리에는 X 혹은 Y의 최대/최소 제한값을 설정하는 부분이 여러 곳에 존재합니다. 보통의 경우 최대값 및 최소값이 제한이 되면 출력 하단에 해당 제한이 이뤄지고 있음을 알리는 비트가 존재하는데 (예:X_max_AL) 특별히 최대 제한값과 최소 제한값이 같은 값으로 설정될 경우 두 가지 알람을 모두 켜고 있습니다. 이는 최대 및 최소 제한에 모두 제한됨을 나타내는 방법입니다.

4) 비정상 입력
실수를 입력으로 받는 명령어에 1.#inf00000 E+000, -1.#inf00000 E+000 또는 1.#QNAN0000 E+000 과 같이 비정상적인 입력이 설정되면 올바른 동작을 수행할 수 없습니다.

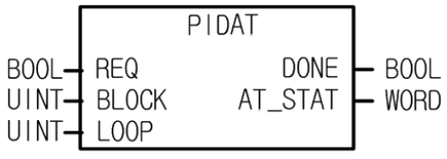
알아두기

STAT 1 (T_S 에러)의 깜빡임

PLC의 매 스캔에는 처리해야 할 데이터의 양이 다를 수 있으므로 그 수행 속도가 매 스캔 일정치 않을 수 있습니다. 이러한 경우 T_s 설정에 여유를 두지 않으면 항상 STAT 1이 표시된 채로 동작하거나 STAT 1이 깜빡일 수 있습니다. 예를 들어 사용자가 T_s를 3ms로 설정했는데 스캔 주기가 2~4ms 범위에서 오르락 내리락 한다면 스캔 주기가 2ms나 3ms일 때에는 정상 수행이 되고 4ms일 때에는 해당 명령어가 정상 수행 될 수 없으므로 STAT에 1을 표시하고 그 스캔에서는 4ms로 연산하게 됩니다. 다시 스캔이 2ms나 3ms로 짧아지면 STAT 1이 꺼져서 깜빡이게 됩니다.

13.2. 프로세스 제어 펑션, 펑션블록

PIDAT	적용 기종	발생플래그
PID 자동동조	XEC	-
펑션 블록	설 명	



입력 REQ : 펑션 블록 실행 요구
BLOCK : 블록 번호 (0)
LOOP : 루프 번호 (0~15)

출력 DONE : 에러 없이 수행 시 On
PID_STAT : PID 상태알람

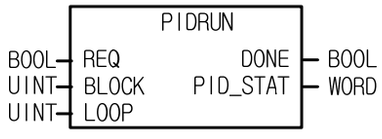
■ 기능

1. 해당 블록, 해당 루프의 PID 연산을 합니다.
2. 블록은 0 으로 고정되고, 각 블록에서 루프는 0~15 입력이 가능하므로 전부 16 개의 독립된 루프를 사용할 수 있습니다.
3. 출력 AT_STAT 은 16 진수로 <표 13.1> 과 같이 해당 루프의 자동동조 상태를 나타냅니다.

<표 13.1>

분류	표시	명칭	내용
STATE	16#0001	PID_STAT	해당 루프가 연산되고 있습니다.
	16#0080	AT_DONE	자동동조가 정상적으로 종료 되었습니다.
	16#0100	MV_MIN_MAX_ERR	최대 조작값이 최소 조작값 보다 작게 설정되었습니다.
	16#0300	PWM_PERIOD_ERR	PWM 출력 주기가 100(10ms)보다 작게 설정되었습니다.
	16#0400	SV_RANGE_ERR	정동작인 경우 오토튜닝 시작 시점의 목표값이 현재값 보다 작은 상태이고, 역동작인 경우 오도튜닝 시작 시점의 목표값이 현재값 보다 큰 상태 입니다.
	16#0500	PWM_ADDRESS_ERR	PWM 출력접점으로 지정된 접점이 %QX0.0.0~0.0.31 이외인 경우에 발생합니다.
	16#0A00	TUNE_DIR_CHG	오토튜닝 도중에 동작방향을 변경한 경우에 발생합니다
	16#0B00	AT_PERIOD_ERR;	자동동조 연산 주기가 100(10ms)보다 작게 설정되었습니다
	16#0E00	LOOP_EXCEED	자동동조 LOOP 번호가 15보다 큰 경우에 발생합니다.

4. 각각의 상태는 서로 동시에 나타날 수 있습니다.

PIDRUN	적용 기종	발생플래그
PID 연산기	XGI, XGR, XEC	-
펑션 블록	설 명	
	**입력** REQ : 펑션 블록 실행 요구 BLOCK : 블록 번호 (0~7) LOOP : 루프 번호 (0~31) **출력** DONE : 에러 없이 수행 시 On PID_STAT : PID 상태알람	

■ 기능

1. 해당 블록, 해당 루프의 PID 연산을 합니다.
2. 블록은 0~7 (XEC 는 0)입력이 가능하고, 각 블록에서 루프는 0~31(XEC 는 0~15) 입력이 가능하므로 전부 256(XEC 는 16)개의 독립된 PID 루프를 사용할 수 있습니다.
3. 출력 PID_STAT 은 16 진수로 해당 PID 루프의 다음의 <표 13.2> 와 같은 상태를 나타냅니다.

< 표 13.2 >

• XGI,XGR 의 경우

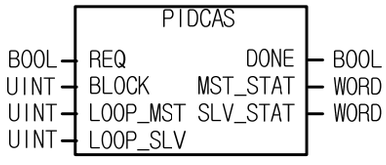
분류	표시	플래그 명	내용
ALARM	16#0001	T_s ERR	T_s 설정이 너무 작아서 T_s마다 연산할 수 없습니다.
	16#0002	K_p ERR	K_p 값이 0이므로 주의 하십시오.
	16#0004	dPV_AL	PV 가 dPV_max 설정에 의해 제한되고 있습니다.
	16#0008	dMV_AL	MV가 dMV_max 설정에 의해 제한되고 있습니다.
	16#0010	MVmax_AL	MV가 MV_max 설정에 의해 제한되고 있습니다.
	16#0020	MVmin_AL	MV가 MV_min 설정에 의해 제한되고 있습니다.
	16#0040	AT_fail	AT(Autotuning)가 비정상 적으로 종료되었습니다.
	16#0080	미사용	미사용
STATE	16#0100	PID_STAT	해당 루프가 연산되고 있습니다.
	16#0200	AT_STAT	AT(Autotuning) 중입니다.
	16#0400	AT_DONE	AT(Autotuning)가 종료 되었습니다.
	16#0800	EX_RUN	외부 런 신호에 의해서 기동되었습니다.
	16#1000	MAN_OUT	수동 출력 중입니다.
	16#2000	CAS_STAT	CAS(Cascade) 동작 중입니다.
	16#4000	CAS_MST	CAS(Cascade) 마스터로 동작하고 있습니다.
	16#8000	AW_STAT	AW1(Anti wind-up) 혹은 AW2가 동작되고 있습니다.

• XEC 의 경우

분류	표시	플래그 명	내용
ALARM	16#0001	PV_MIN_MAX_ALM	현재값이 설정된 최대, 최소 현재값의 범위를 벗어난 경우
	16#0002	PID_SCANTIME_ALM	연산주기 설정이 너무 작은 경우입니다.
	16#0003	PID_dPV_WARN	이번 PID주기의 현재값의 변화량이 현재값 변화량 제한 설정을 초과한 경우에 발생합니다
	16#0004	PID_dMV_WARN	이번 PID주기의 조작값의 변화량이 조작값 변화량 제한 설정을 초과한 경우에 발생합니다
	16#0005	PID_MV_MAX_WARN	이번 PID주기의 계산된 조작값이 최대 조작값 설정을 초과한 경우에 발생합니다.
	16#0006	PID_MV_MIN_WARN	이번 PID주기의 계산된 조작값이 최소 조작값 설정보다 작은 경우에 발생합니다.
ERROR	16#0100	MV_MIN_MAX_ERR	최대 조작값이 최소 조작값 보다 작게 설정되었습니다.
	16#0200	PV_MIN_MAX_ERR	최대 현재값이 최소 현재값 보다 작게 설정되었습니다.
	16#0300	PWM_PERIOD_ERR	PWM 출력 주기가 100(10ms)보다 작게 설정되었습니다.
	16#0400	SV_RANGE_ERR	정동작인 경우 오토튜닝 시작 시점의 목표값이 현재값 보다 작은 경우, 역동작인 경우 오토튜닝 시작 시점의 목표값이 현재값 보다 큰 경우입니다.
	16#0500	PWM_ADDRESS_ERR	PWM 출력접점으로 지정된 접점이 %QX0.0.0~0.0.31 이외인 경우에 발생합니다.
	16#0600	P_GAIN_SET_ERR	비례 상수가 0보다 작게 설정된 경우 발생합니다
	16#0700	I_TIME_SET_ERR	적분 시간이 0보다 작게 설정된 경우 발생합니다
	16#0800	D_TIME_SET_ERR	미분 시간이 0보다 작게 설정된 경우 발생합니다
	16#0900	CONTROL_MODE_ERR	제어 모드가 P, PI, PD, PID이외인 경우에 발생합니다.
	16#0B00	PID_PERIOD_ERR;	PID 연산 주기가 100(10ms)보다 작게 설정되었습니다
	16#0C00	HBD_WRONG_DIR	혼합운전시 정동작 루프의 방향 파라미터가 역동작으로 설정되거나 역동작 루프의 방향이 정동작으로 설정된 경우에 발생합니다
	16#0D00	HBD_SV_NOT_MATCH	혼합운전시 두 루프의 목표값이 서로 다른 경우에 발생합니다
	16#0E00	LOOP_EXCEED	PID LOOP 번호가 15보다 큰 경우에 발생합니다.

4. 각각의 상태는 서로 동시에 나타날 수 있습니다.

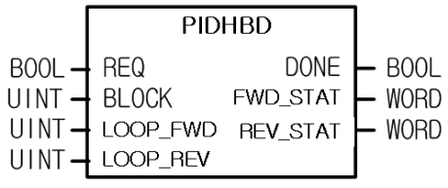
PIDCAS	적용 기종	발생플래그
Cascade PID 연산기	XGI, XGR, XEC	-

펑션 블록	설 명
	**입력** REQ :펑션 블록 실행 요구 BLOCK :블록 번호 LOOP_MST :마스터 루프 번호 LOOP_SLV :슬레이브 루프 번호 **출력** DONE : 에러 없이 수행 시 On MST_STAT : 마스터 루프 상태 알람 SLV_STAT : 슬레이브 루프 상태 알람

■ 기능

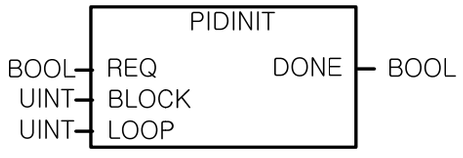
1. 해당 블록의 해당 루프 2 개를 조합하여 Cascade PID 연산을 합니다.
2. 블록은 0~7(XEC 는 0) 입력이 가능하고, 마스터 루프와 슬레이브 루프는 서로 같은 블록 안에서 0~31(XEC 는 0~15)의 서로 다른 숫자로 사용해야 합니다.
3. 출력 MST_STAT, SLV_STAT 은 16 진수로 각각 마스터와 슬레이브 루프의 <표 13.2> 와 같은 상태를 나타냅니다.
4. 각각의 상태는 서로 동시에 나타날 수 있습니다.

PIDHBD	적용 기종	발생플래그
정역 혼합 출력 PID 연산기	XEC	-

펑션 블록	설 명
	**입력** REQ :펑션 블록 실행 요구 BLOCK :블록 번호 LOOP_FWD :정방향 루프 번호 LOOP_REV :역방향 루프 번호 **출력** DONE : 에러 없이 수행 시 On FWD_STAT : 정방향 루프 상태 알람 REV_STAT : 역방향 루프 상태 알람

■ 기능

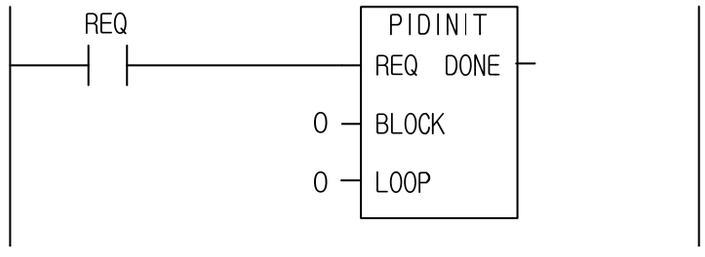
1. 해당 블록의 해당 루프 2 개를 조합하여 정/역 혼합출력 PID 연산을 합니다.
2. 블록은 0 이고, 마스터 루프와 슬레이브 루프는 서로 같은 블록 안에서 0~15 의 서로 다른 숫자로 사용해야 합니다.
3. 출력 FWD_STAT, REV_STAT 은 16 진수로 각각 정방향과 역방향 루프의 <표 13.2> 와 같은 상태를 나타냅니다.
4. 각각의 상태는 서로 동시에 나타날 수 있습니다.

PIDINIT	적용 기종	발생플래그
PID 초기화	XGI, XGR	-
펑션 블록	설 명	
	**입력** REQ : 펑션 블록 실행 요구 BLOCK : 블록 번호 LOOP : 루프 번호 **출력** DONE : 에러 없이 수행 시 On	

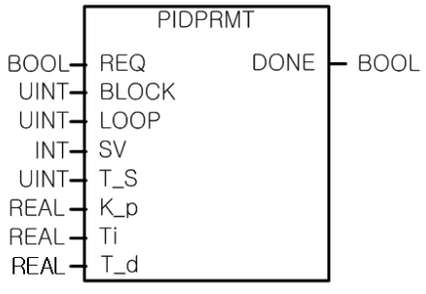
■ 기능

1. 해당 블록의 해당 루프 PID 설정을 모두 0 으로 초기화 합니다.

■ 프로그램 예



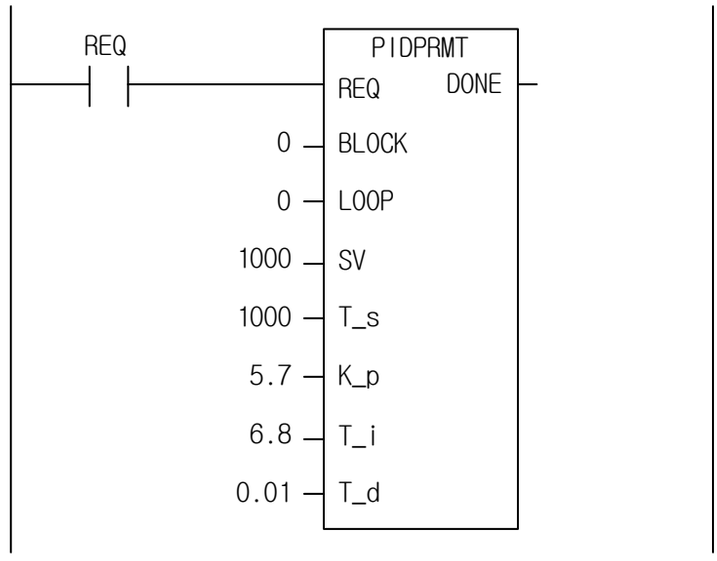
입력 접점 REQ 가 SET 되면 PID 블록0, 루프0 의 모든 설정을 0 으로 초기화 합니다.

PIDPRMT	적용 기종	발생플래그
PID 파라미터 변경	XGI, XGR	-
펑션 블록	설 명	
	**입력** REQ : 펑션 블록 실행 요구 BLOCK : 블록 번호 LOOP : 루프 번호 SV : 목표값 T_s : 연산 주기 K_p : 비례 상수 Ti : 적분 상수 T_d : 미분 상수 **출력** DONE : 에러 없이 수행시 On	

■ 기능

1. 해당 블록의 해당 루프 주요 PID 설정을 입력 값으로 변경합니다.
2. 설정 변경 항목은 입력에 표현된 바와 같이 SV, T_s, K_p, T_i, T_d 입니다.
3. PIDPRMT 명령어를 응용하여 한 PID 루프 내의 조건에 따른 계수를 변경할 수 있으므로 시스템의 반응에 따른 패턴 제어를 수행 할 수 있습니다.

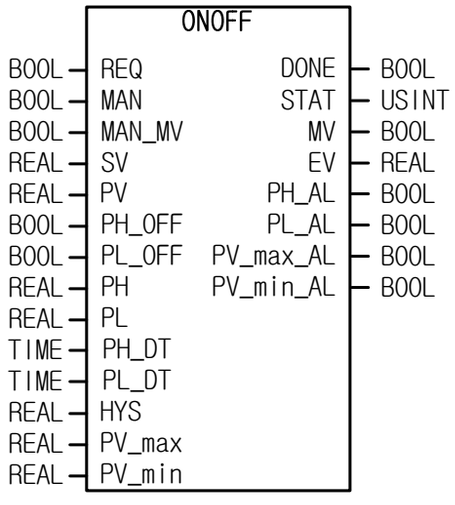
■ 프로그램 예



PID 블록0, 루프0의 주요 설정을 그림의 입력 변수들로 변경합니다.

ONOFF	적용 기종	발생플래그
ON / OFF 제어기	XGI, XGR	-

펑션 블록



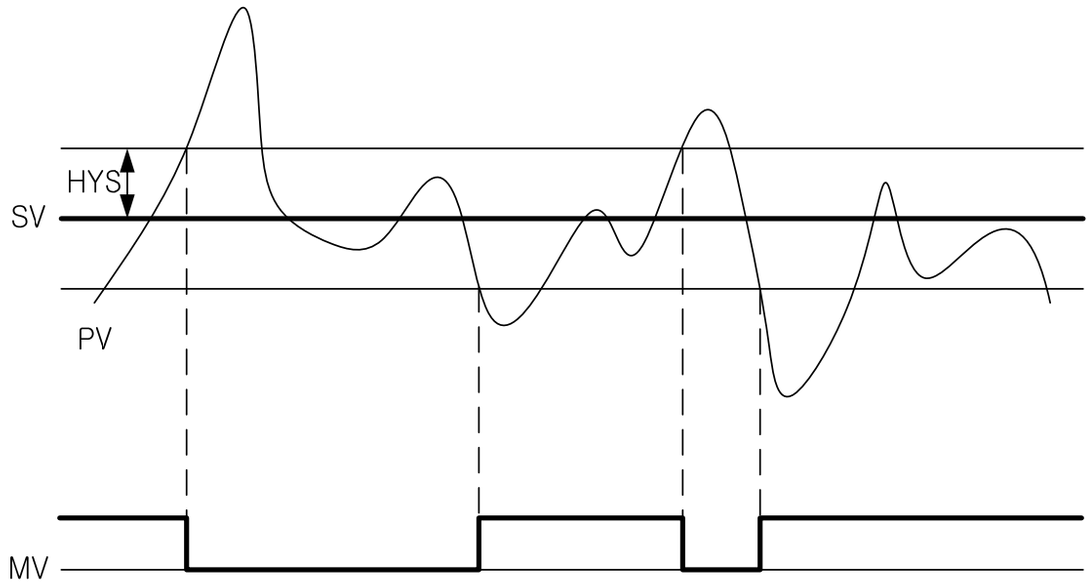
설 명

입력
- REQ : 펑션 블록 실행 요구
- MAN : 수동 모드 전환 비트
- MAN_MV : 수동 모드 전환 값
- SV : 목표값
- PV : 현재값
- PH_OFF : PV 이상 구간 해제 비트
- PL_OFF : PV 이하 구간 해제 비트
- PH : PV 이상 구간 설정값
- PL : PV 이하 구간 설정값
- PH_DT : PV 이상 구간 설정 대기 시간
- PL_DT : PV 이하 구간 설정 대기 시간
- HYS : 히스테리시스 반경 설정
- PV_max : PV 최대 제한값
- PV_min : PV 최소 제한값

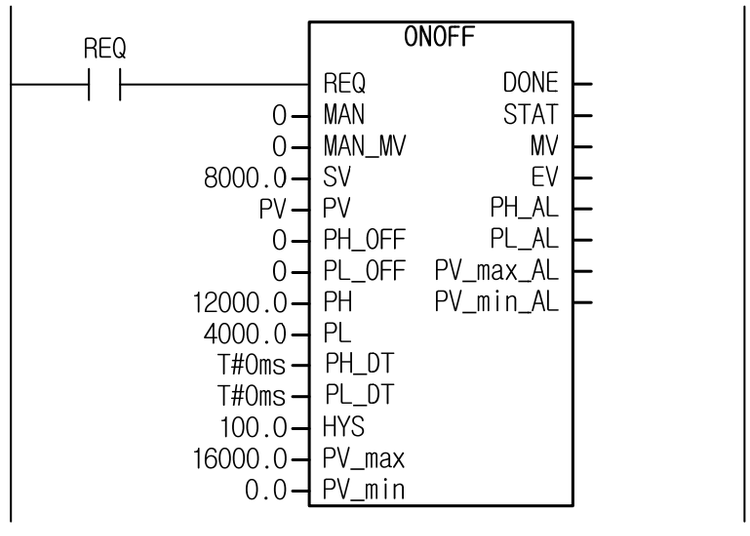
출력
- DONE : 에러 없이 수행 시 On
- STAT : 상태 알람
- MV : 출력값
- EV : 에러값
- PH_AL : PV 이상 알람
- PL_AL : PV 이하 알람
- PV_max_AL : PV 최대값 이상 알람
- PV_min_AL : PV 최소값 이하 알람

■ 기능

1. Bool 형식의 출력 MV 를 생성하는 ON / OFF 제어기 입니다.
2. 현재값 PV 를 입력보드(AD)에서 받을 경우 데이터형을 REAL 로 바꿔서 사용해야 합니다.
3. MAN 를 SET 시키게 되면 수동모드로 전환되어 MAN_MV 의 값이 연산 결과를 무시하고 MV 로 출력됩니다.
4. (SV - HYS) > PV 일 경우 MV = On
5. (SV + HYS) < PV 일 경우 MV = Off
6. (SV - HYS) ≤ PV ≤ (SV + HYS) 일 경우 MV = MV(이전값)
7. 에러값 EV = SV - PV 를 나타냅니다.
8. PH/PL 에 각각 PV 의 상/하구간을 설정하면 PV 의 값이 상/하구간을 벗어날 경우 그에 맞는 각각의 알람 PH_AL/PL_AL 을 표시합니다.
9. 단. PH_OFF/PL_OFF 비트가 On 인 경우에는 각각 PH_AL/PL_AL 동작을 수행하지 않습니다.
10. PH_DT/PL_DT 에는 PH_AL/PL_AL 의 출력 지연 시간을 설정 할 수 있습니다.
11. PV_max/PV_min 에는 각각 PV 의 최대값/최소값을 설정하여 PV 입력을 제한할 수 있습니다. 제한에 걸린 경우 PV_max_AL/PV_min_AL 알람을 On 합니다.
12. EV 가 실수(REAL) 데이터 표현범위를 벗어날 경우 출력이 '1.#inf00000 E+000' 혹은 '-1.#inf00000 E+000' 으로 표시될 수 있으나 EV 를 제외한 출력은 정상동작합니다.

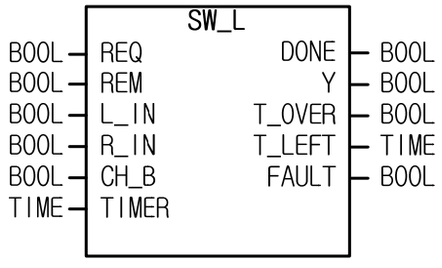


■ 프로그램 예



PV 가 8100(8000+100)을 초과하면 MV 가 Off 되고 PV 가 7900(8000-100) 미만이면 MV 가 On 됩니다.
PV 가 16000 이상이면 16000 으로 간주되며 PV_max_AL 이 SET 되고, 0 이하이면, 0 으로 간주되며 PV_min_AL 이 SET 됩니다.
PV 가 12000 이상이면 PH_AL 이 SET 되고 4000 이하이면 PL_AL 이 SET 됩니다.

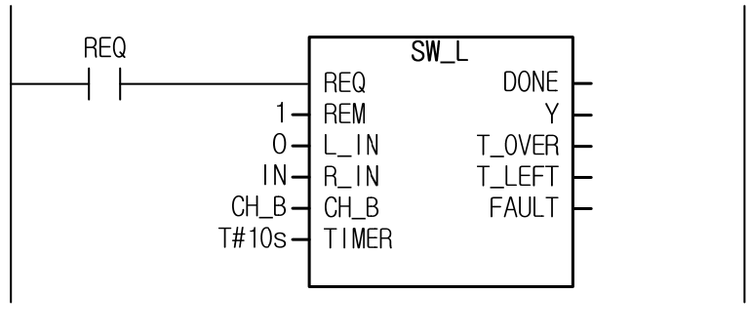
SW_L	적용 기종	발생플래그
1 입력 래치	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 REM : 리모트 입력 설정 L_IN : 로컬 입력 R_IN : 리모트 입력 CH_B : 체크백 입력 TIMER : 체크백 대기 시간 **출력** DONE : 에러 없이 수행 시 On Y : 출력값 T_OVER : 타임 오버 알람 T_LEFT : 남은 시간 표시 FAULT : 체크백 실패 알람

■ 기능

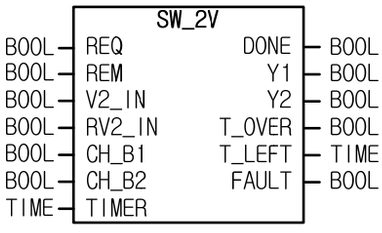
1. 펌프 제어를 할 경우 펌프 동작 명령을 주고 체크백 신호를 받아서 실제로 동작하는지 체크하지 않으면 펌프가 고장으로 작동이 안되거나 다른 이유로 실제로 펌프가 동작되지 않아도 계속 동작 명령을 출력해서 사고의 위험이 있습니다. 이런 경우를 대비해서 동작 명령 Y 를 출력하고 CHECK_BACK 신호(실제 PUMP 의 RUN 신호)가 입력되지 않으면 이를 이상 상태로 판정하여 동작 명령을 출력하지 않고 FAULT 를 출력하도록 되어 있습니다.
2. REM 이 OFF 이면 L_IN 을, REM 이 ON 이면 R_IN 을 입력으로 받습니다.
3. 처음 입력이 ON 되면 출력 Y 를 ON 하고 TIMER 에 설정한 시간 동안 CH_B(체크백)신호를 기다립니다.
4. 이 때 T_LEFT 에는 남은 시간이 표시되며 대기 상태가 끝나면 T_OVER 를 ON 합니다.
5. TIMER 시간만큼 지난 후 입력과 CH_B 가 ON 이면 계속해서 Y 를 ON 상태로 유지하고, 잠시라도 CH_B 가 OFF 된 경우에는 시스템 고장으로 판단하여 Y 에 OFF 를 출력하고 FAULT 를 ON 합니다. 이 때에는 다시 CH_B 가 ON 되도 Y 에 OFF 을 출력합니다.
6. 입력에 OFF 가 들어오면 처음부터 다시 동작합니다.

■ 프로그램 예



REQ 가 SET 인 상태에서 IN 이 ON 되면 Y 가 ON 되고 10 초간 타이머가 동작합니다. 이 동안 T_LEFT 에 남은 시간이 표시됩니다.
10 초가 모두 지나면 T_OVER 가 ON 되고 CH_B 가 ON 인 경우 Y 가 ON 으로 유지되고 CH_B 가 OFF 되면 Y 가 OFF 되면서 FAULT 가 ON 됩니다.

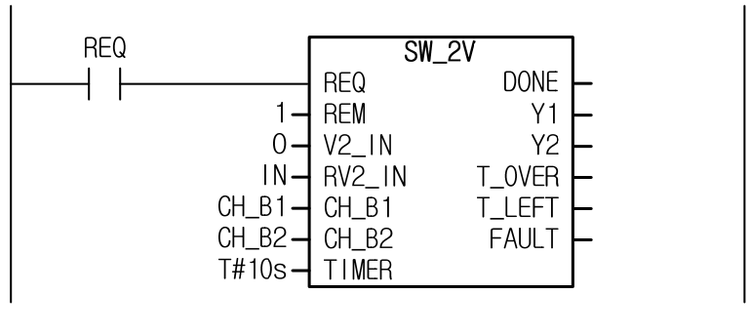
SW_2V	적용 기종	발생플래그
2-Way 밸브 제어	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 REM : 리모트 입력 설정 V2_IN : 로컬 밸브2/1 선택 RV2_IN : 리모트 밸브2/1 선택 CH_B1 : 밸브1 체크백 입력 CH_B2 : 밸브2 체크백 입력 TIMER : 체크백 대기 시간 **출력** DONE : 에러 없이 수행 시 On Y1 : 출력1 Y2 : 출력2 T_OVER : 타임 오버 T_LEFT : 남은 시간 표시 FAULT : 체크백 실패 알람

■ 기능

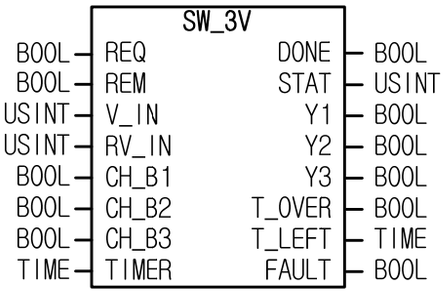
1. 2-Way 밸브의 경우 선택된 쪽만 OPEN 되고 선택이 안된 쪽은 CLOSE 되어야 합니다. 또한 체크백 신호가 입력되면 출력을 내보내지 않아야 밸브가 제대로 동작됩니다. OPEN 명령 후 체크백 입력 DELAY 설정시간 안에 체크백 신호가 입력되지 않을 경우 FAULT 가 출력됩니다.
2. REM 이 OFF 이면 V2_IN 을, REM 이 ON 이면 RV2_IN 을 입력으로 받습니다.
3. 입력이 OFF -> ON 로 변경되면 출력 Y2 를 ON 하고 TIMER 에 설정한 시간동안 CH_B2 신호를 기다립니다.
4. 입력이 ON -> OFF 로 변경되면 출력 Y1 를 ON 하고 TIMER 에 설정한 시간동안 CH_B1 신호를 기다립니다.
5. 이 때 T_LEFT 에는 남은 시간이 표시되며 대기 상태가 끝나면 T_OVER 를 ON 합니다.
6. TIMER 시간만큼 지난 후 해당 출력을 OFF 하고 해당 CH_B 가 ON 이면 FAULT 를 OFF 하고 해당 CH_B 가 OFF 이면 FAULT 를 ON 합니다.
7. 입력이 바뀌면 처음부터 다시 동작하고, TIMER 설정이 스캔주기의 2 배 이상으로 설정되어야 출력을 보장할 수 있습니다.

■ 프로그램 예



REQ가 SET인 상태에서 IN이 ON 되면 Y2가 ON되고 10초간 타이머가 동작합니다. 이 동안 T_LEFT에 남은 시간이 표시됩니다.
10초가 모두 지나면 T_OVER가 ON되고 출력 Y1, Y2가 OFF 됩니다.
이 때 CH_B2가 ON인 경우 FAULT가 OFF 되고 CH_B2가 OFF인 경우 FAULT가 ON 됩니다.

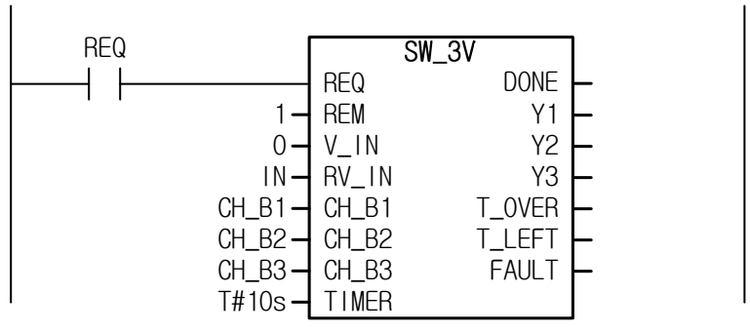
SW_3V	적용 기종	발생플래그
3-Way 밸브 제어	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 REM : 리모트 입력 설정 V_IN : 로컬 입력 선택(1~3) RV_IN : 리모트 입력 선택(1~3) CH_B1 : 밸브 1 체크백 입력 CH_B2 : 밸브 2 체크백 입력 CH_B2 : 밸브 3 체크백 입력 TIMER : 체크백 대기 시간 **출력** DONE : 에러 없이 수행 시 On Y1 : 출력 1 Y2 : 출력 2 Y3 : 출력 3 T_OVER : 타임 오버 T_LEFT : 남은 시간 표시 FAULT : 체크백 실패 알람

■ 기능

1. 3-Way 밸브의 경우 선택된 쪽만 OPEN 되고 선택이 안된 쪽은 CLOSE 되어야 합니다.또한 체크백 신호가 입력되면 출력을 내보내지 않아야 밸브가 제대로 동작됩니다. OPEN 명령 후 체크백 입력 DELAY 설정시간 안에 체크백 신호가 입력되지 않을 경우 FAULT 가 출력됩니다.
2. REM 이 OFF 이면 V_IN 을, REM 이 ON 이면 RV_IN 을 입력으로 받습니다.
3. 입력이 Vm -> Vn 으로 변경되면 출력 Yn 를 ON 하고 TIMER 에 설정한 시간동안 CH_Bn 신호를 기다립니다.
4. 이 때 T_LEFT 에는 남은 시간이 표시되며 대기 상태가 끝나면 T_OVER 를 ON 합니다.
5. TIMER 시간만큼 지난 후 해당 출력을 OFF 하고 CH_Bn 이 ON 이면 FAULT 를 OFF 하고 해당 CH_Bn 이 OFF 이면 FAULT 를 ON 합니다.
6. 입력이 바뀌면 처음부터 다시 동작하고, TIMER 설정이 스캔주기의 2 배 이상으로 설정되어야 출력을 보장할 수 있습니다.
7. 입력은 1~3 의 값 이어야 하며 이를 벗어날 경우 STAT 에 8 을 출력합니다.

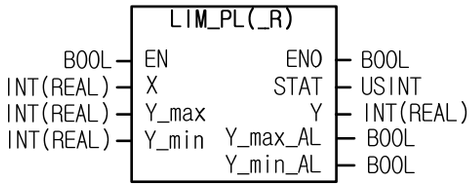
■ 프로그램 예



REQ가 SET인 상태에서 IN을 3으로 바꾸면 되면 Y3이 ON되고 10초간 타이머가 동작합니다.
이 동안 T_LEFT에 남은 시간이 표시됩니다.
10초가 모두 지나면 T_OVER가 ON되고 출력 Y1, Y2, Y3이 OFF 됩니다.
이 때 CH_B3가 ON인 경우 FAULT가 OFF 되고 CH_B3가 OFF인 경우 FAULT가 ON 됩니다.

13.3. 데이터 처리 펑션, 펑션블록

LIM_PL(_R)	적용 기종	발생플래그
최대/최소값 제한	XGI, XGR	-
펑션	설 명	



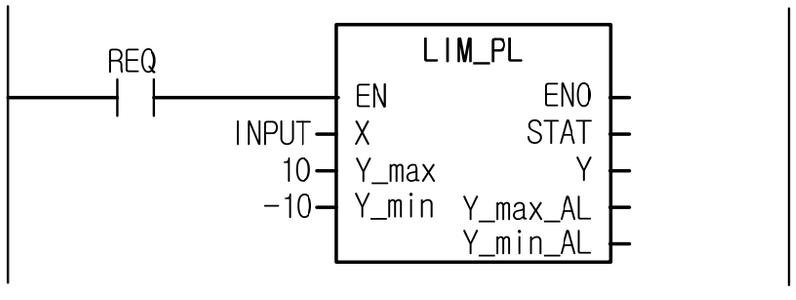
입력
- EN : 펑션 실행 요구
- X : 입력값
- Y_max : 출력 최대 제한값
- Y_min : 출력 최소 제한값

출력
- ENO : 에러 없이 수행 시 On
- STAT : 상태 알람
- Y : 출력
- Y_max_AL : 출력 최대값 이상 알람
- Y_min_AL : 출력 최소값 이하 알람

■ 기능

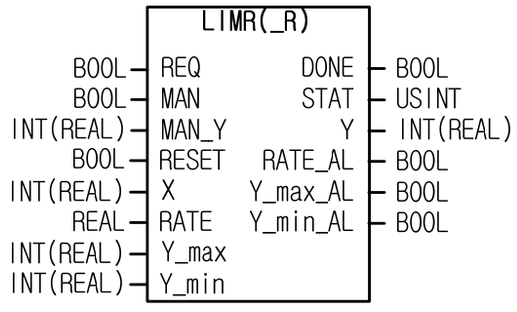
1. 입력 X 값을 최대값 및 최소값으로 제한하여 출력 Y를 생성합니다.
2. Y_max와 Y_min의 사이의 값은 그대로 통과합니다.
3. 최대 제한값 Y_max 이상이면 Y_max_AL이 ON 되고 Y_max값이 Y로 출력됩니다.
4. 최소 제한값 Y_min 이하이면 Y_min_AL이 ON 되고 Y_min값이 Y로 출력됩니다.
5. Y_max 설정이 Y_min설정보다 작은 경우 STAT은 4를 가리키고 0을 출력합니다.

■ 프로그램 예



1. INPUT이 20 인 경우 : Y에 10 (Y_max) 이 출력되고 Y_max_AL이 On됩니다.
2. INPUT이 3 인 경우 : Y에 그대로 3이 출력됩니다.
3. INPUT이 -12 인 경우 : Y에 -10 (Y_min)이 출력되고 Y_min_AL이 On됩니다.

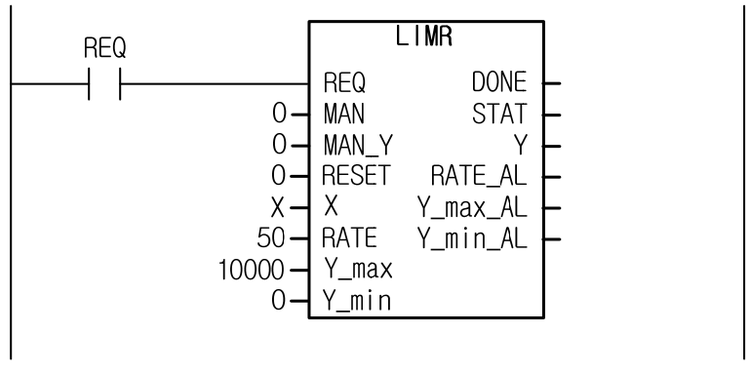
LIMR(_R)	적용 기종	발생플래그
최대/최소값, 최대 변화율 제한	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 MAN : 수동모드 설정 MAN_Y : 수동 출력 RESET : 블록 동작 리셋 X : 입력값 RATE : 최대 변화율 제한값 Y_max : 출력 최대 제한값 Y_min : 출력 최소 제한값 **출력** DONE : 에러 없이 수행 시 On STAT : 상태 알람 Y : 출력값 RATE_AL : 최대 변화율제한 상태 알람 Y_max_AL : 출력 최대값 이상 알람 Y_min_AL : 출력 최소값 이하 알람

■ 기능

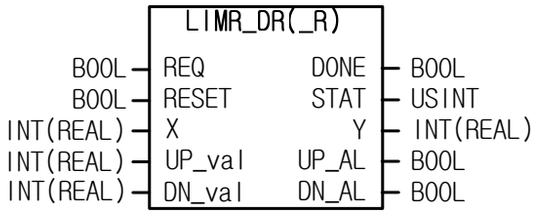
1. 입력 X 의 최대 변화율을 제한하고, 최대 / 최소값을 제한하여 출력합니다.
2. 본 펑션 블록은 REQ 가 Off 되더라도 내부 상태를 저장하며 다시 REQ 를 On 했을 때 이전에 수행했던 동작이 수행 중이라면 이어서 수행합니다.
3. 변화율 제한식 : $Y_{old} - \frac{RATE(Y_{max} - Y_{min})}{100} \leq Y \leq Y_{old} + \frac{RATE(Y_{max} - Y_{min})}{100}$
4. 변화율 제한된 경우 RATE_AL, 최대 / 최소값 제한된 경우 Y_max_AL 또는 Y_min_AL 표시합니다.
5. MAN 이 ON 인 경우 MAN_Y 의 값이 Y 로 출력되며 MAN 이 다시 OFF 된 경우 그 상태부터 다시 변화율이 제한됩니다.
6. RESET 이 ON 인 경우 출력 Y 를 0 으로 초기화합니다.
7. REQ에 양 변환 검출접점(P 접점)이나 클럭(_T1s 등)의 양 변환 검출접점을 사용하면 원하는 주기로 동작 시킬 수 있습니다.

■ 프로그램 예



1. X가 0 → 3000 으로 변경 : 최대 변화량은 $\frac{RATE(Y_{max}-Y_{min})}{100}$ = 5000 까지 허용되므로 변화율 제한 통과, 최대 / 최소값 제한 통과, Y = 3000 출력
2. X가 0 → 10000 으로 변경 : 최대 변화량은 5000 까지 허용되므로 변화율 제한에 2 스캔 동안 걸려서 5000 씩 증가 후 Y = 10000 출력 후 Y_max_AL은 ON
3. X가 0 → 30000 으로 변경 : 최대 변화량은 5000 까지 허용되므로 변화율 제한에 6 스캔 동안 걸려서 5000 씩 증가 후, 최대값 제한에 걸려서 Y = 10000 출력 후 Y_max_AL은 ON

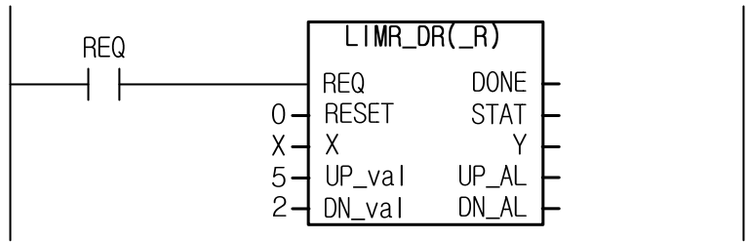
LIMR_DR(_R)	적용 기종	발생플래그
방향성 최대 변화량 제한	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 RESET : 블록 동작 리셋 X : 입력값 UP_val : 증가 제한값 DN_val : 감소 제한값 **출력** DONE : 에러 없이 수행 시 On STAT : 상태 알람 Y : 출력값 UP_AL : 증가량 제한 알람 DN_AL : 감소량 제한 알람

■ 기능

1. 입력 X 의 증가 / 감소 각각의 최대 변화량을 제한하여 출력합니다.
2. 본 펑션 블록은 REQ 가 Off 되더라도 내부 상태를 저장하며 다시 REQ 를 On 했을 때 이전에 수행했던 동작이 수행 중이라면 이어서 수행합니다.
3. X 의 변화에 대해서 Y 는 최대 UP_val 증가, 또는 DN_val 만큼 감소 할 수 있습니다.
4. 증가량 / 감소량 제한된 경우 각각 UP_AL 또는 DN_AL 비트를 통해 표시합니다.
5. RESET 비트가 On 되면 펑션 블록이 RESET 상태가 되고 0 이 Y 로 출력됩니다.
6. UP_val 혹은 DN_val 의 값이 음수 일 경우 STAT 에 8 을 출력합니다.
7. REQ에 양 변환 검출접점(P 접점)이나 클럭(_T1s 등)의 양 변환 검출접점을 사용하면 원하는 주기로 동작 시킬 수 있습니다.

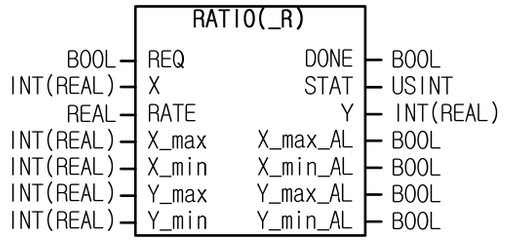
■ 프로그램 예



1. X가 0 → 3000 으로 변경 : 최대 증가량이 5 이므로 600 스캔 동안 Y가 5씩 증가하고 그 동안 UP_AL은 On을 유지하다가 Y = 3000을 출력하면 UP_AL은 Off
2. X가 1000 → 0 으로 변경 : 최대 감소량이 2 이므로 500 스캔 동안 Y가 2씩 감소하고 그 동안 DN_AL은 On을 유지하다가 Y = 0 출력을 출력하면 DN_AL은 Off

RATIO(_R)	적용 기종	발생플래그
비율 변환기	XGI, XGR	-

펑션 블록



설 명

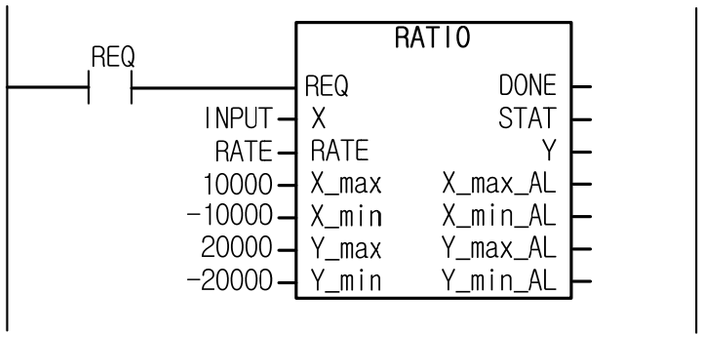
입력
- REQ : 펑션 블록 실행 요구
- X : 입력값
- RATE : 비율
- X_max : 입력 최대 제한값
- X_min : 입력 최소 제한값
- Y_max : 출력 최대 제한값
- Y_min : 출력 최소 제한값

출력
- DONE : 에러 없이 수행 시 On
- STAT : 상태알람
- Y : 출력값
- X_max_AL : 입력 이상 알람
- X_min_AL : 입력 이하 알람
- Y_max_AL : 출력 이상 알람
- Y_min_AL : 출력 이하 알람

■ 기능

1. 입력 X 의 일정 비율을 Y 에 출력합니다.
2. 이 때 기준점은 0 이 아니고 X_min 이므로 주의하십시오.
3. 출력 Y 는 $Y = (X - X\_min) \times \frac{RATE}{100} + X\_min$ 의 식을 거쳐서 나옵니다.
4. X_max, X_min 은 입력 X 의 최대 / 최소값을 제한하여 X 가 X_max 이상일 때 X 대신 X_max 로 연산하고 X 가 X_min 이하일 때 X 대신 X_min 으로 연산합니다.
5. Y_max, Y_min 은 출력 Y 의 최대 / 최소값을 제한하여 Y 가 Y_max 이상일 때 Y_max 를 출력하고, Y 가 Y_min 이하일 때 Y_min 을 출력합니다.
6. 입출력에 설정된 최대, 최소 값 이상이거나 이하일 때 해당 알람 X_max_AL, X_min_AL, Y_max_AL, Y_min_AL 이 표시됩니다.

■ 프로그램 예



1. **X = 20000, RATE = 50 인 경우 :** X 가 X_max 이상일 때 X_max 의 값인 10000 이 입력,

$$Y = (10000 - (-10000)) \times \frac{50}{100} + (-10000)$$, X_max_AL = On

Y = 0

2. **X=1000, RATE=20 인 경우 :** X 가 1000 으로 입력,

$$Y = (1000 - (-10000)) \times \frac{20}{100} + (-10000)$$

Y = -7800

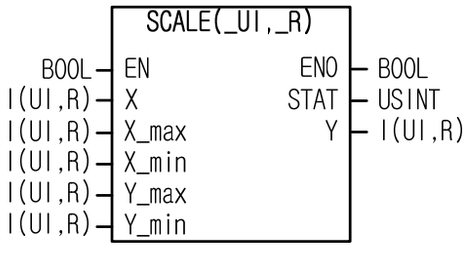
3. **X = 20000, RATE = -250인 경우** : X가 X_max 이상이므로 X_max 값인 10000으로 연산,

$$-60000 = (10000 - (-10000)) \times \frac{-250}{100} + (-10000)$$,X_max_AL = ON, Y_min_AL = ON

Y가 Y_min이하 이므로 Y_min으로 출력되어,

Y = -20000

SCALE(_UI, _R)	적용 기종	발생플래그
스케일 변환기	XGI, XGR	-

펑션	설 명
	**입력** EN : 펑션 실행 요구 X : 입력값 X_max : 입력 최대 제한값 X_min : 입력 최소 제한값 Y_max : 출력 스케일 최대값 Y_min : 출력 스케일 최소값 **출력** ENO : 에러 없이 수행 시 On STAT : 상태 알람 Y : 출력값

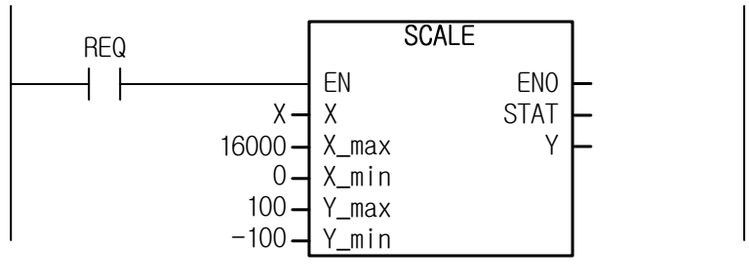
■ 기능

1. 입력 X 를 최대 / 최소 제한 후 설정된 스케일로 변경합니다.
2. 입력 X 의 범위를 X_max, X_min 에, 출력 Y 의 범위를 Y_max, Y_min 에 설정하고 사용합니다.
3. 출력 방정식은 다음과 같습니다.

$$Y = (X - X_{min}) \frac{Y_{max} - Y_{min}}{X_{max} - X_{min}} + Y_{min}$$

4. X_max, X_min 의 값이 같을 경우 출력방정식의 분모가 0 이기 때문에 STAT 에 8 을 출력합니다.
5. X 입력값이 X_MAX ~ X_MIN 값을 벗어날 경우 STAT 에 2 를 출력합니다.

■ 프로그램 예

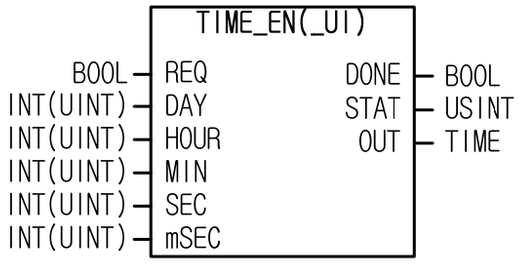


0 ~ 16000 의 값을 -100 ~ 100 값으로 스케일합니다.

1. X가 4000 일 경우: $Y = (4000 - 0)\ \frac{100 + 100}{16000 - 0} - 100 = -50$
2. X가 20000 일 경우: X를 16000으로 제한 후, $Y = (20000 - 0)\ \frac{100 + 100}{16000 - 0} - 100 = 150$

   Y = 150 이지만 Y_max = 100이므로 Y = 100을 출력합니다.

TIME_EN(_UI)	적용 기종	발생플래그
일,시,분,초,1/1000 초 를 TIME 형 데이터로 만든다.	XGI, XGR	-
펑션 블록	설 명	



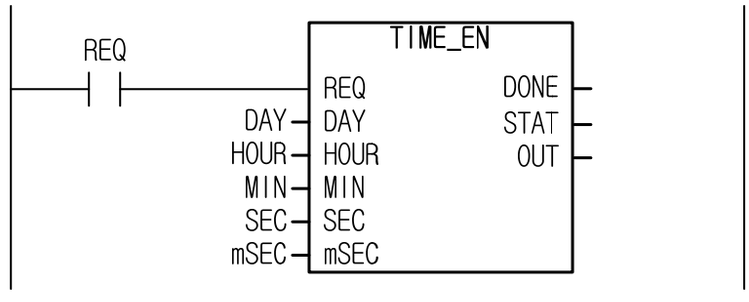
입력 REQ : 펑션 블록 실행 요구
DAY : 날짜
HOUR : 시간
MIN : 분
SEC : 초
mSEC : 1/1000 초

출력 DONE : 에러 없이 수행 시 On
STAT : 상태 알람
OUT : 시간 출력값

■ 기능

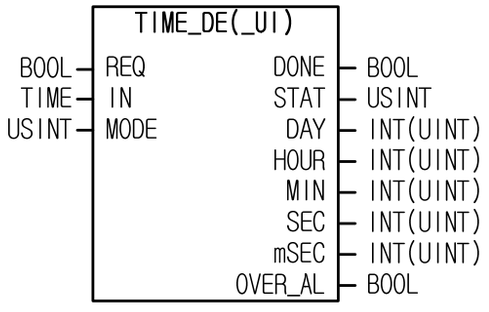
1. 날짜, 시, 분, 초, 1/1000 초 데이터를 시간형 변수로 변환합니다.
2. 입력값이 음수 이거나 출력결과가 TIME 형 데이터 표현 범위(0~49d17h2m47s295ms)를 초과할 경우 STAT 8 이 발생하고 연산이 수행되지 않습니다.

■ 프로그램 예



1. DAY=1, HOUR=1, MIN= 1, SEC=1, mSEC=1 일 경우 OUT = T#1d1h1m1s1ms
2. DAY=0, HOUR=0, MIN=30000, SEC=0, mSEC=0 일 경우 OUT = T#0d20h20m0s0ms

TIME_DE(_UI)	적용 기종	발생플래그
TIME 형 데이터를 일, 시, 분, 초, 1/1000 초로 분리한다.	XGI, XGR	-
펑션 블록	설 명	



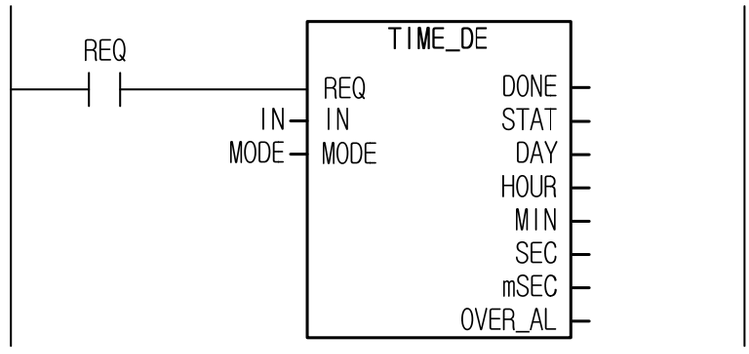
입력 REQ : 펑션 블록 실행 요구
IN : 시간 입력값
MODE : 출력 모드 선택(0~4)

출력 DONE : 에러 없이 수행 시 On
STAT : 상태 알람
DAY : 날짜
HOUR : 시간
MIN : 분
SEC : 초
mSEC : 1/1000 초
OVER_AL : 오버 플로우 알람

■ 기능

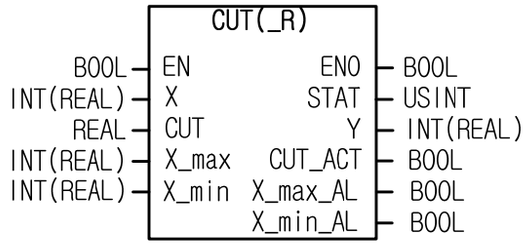
1. 시간형 입력을 날짜, 시, 분, 초, 1/1000 초 로 나누어 출력합니다.
2. MODE 에 따라서 다음과 같이 출력합니다.
   A. MODE 0 : 날/시/분/초/ms 모두 표시
   B. MODE 1 : 시/분/초/ms 로만 표시
   C. MODE 2 : 분/초/ms 로만 표시
   D. MODE 3 : 초/ms 로만 표시
   E. MODE 4 : ms 로만 표시
3. 출력 데이터의 범위를 넘으면 최대값 32767 (TIME_DE_UI 의 경우 65535)을 출력하고 OVER_AL 을 Set 한다.
4. MODE 가 5 이상 입력되면 STAT 8 을 표시하고 동작하지 않습니다.

■ 프로그램 예



1. IN =T#1d1h1m1s1ms, MODE = 0 일 경우 DAY =1, HOUR= 1, MIN= 1, SEC= 1, mSEC= 1, OVER_AL=Off
2. IN =T#1d1h1m1s1ms, MODE = 1 일 경우 DAY =0, HOUR=25, MIN= 1, SEC= 1, mSEC= 1, OVER_AL=Off
3. IN =T#1d1h1m1s1ms, MODE = 2 일 경우 DAY =0, HOUR= 0, MIN=1501, SEC= 1, mSEC= 1, OVER_AL=Off
4. IN =T#1d1h1m1s1ms, MODE = 3 일 경우 DAY =0, HOUR= 0, MIN= 0, SEC=32767, mSEC= 1, OVER_AL=On
5. IN =T#1d1h1m1s1ms, MODE = 4 일 경우 DAY =0, HOUR= 0, MIN= 0, SEC= 0, mSEC=32767, OVER_AL=On
6. IN =T#90061001ms, MODE = 0 일 경우 입력은 T#1d1h1m1s1ms로 수정 표시되며
   결과는 DAY=1, HOUR=1, MIN=1, SEC=1, mSEC=1, OVER_AL=Off

CUT(_R)	적용 기종	발생플래그
소신호 제거 필터	XGI, XGR	-

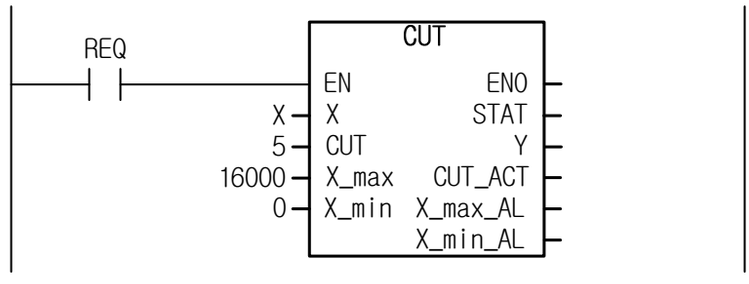
펑션	설 명
	**입력** EN : 펑션 실행 요구 X : 입력값 CUT : 소신호 제거 범위 (%) X_max : 입력 최대 제한값 X_min : 입력 최소 제한값 **출력** ENO : 에러 없이 수행 시 On STAT : 상태 알람 Y : 출력값 CUT_ACT : CUT 연산이 이뤄지고 있음. X_max_AL : 입력 최대 제한 알람 X_min_AL : 입력 최소 제한 알람

■ 기능

1. 입력이 [X_min]부터 [X_min ~ X_max 구간의 CUT% 만큼]사이의 값일 경우 입력을 무시하고 X_min 을 출력합니다.
2. 이 때 기준점은 0 이 아닌 [X_min]입니다.
3. 입력은 X_max, X_min 값에 의해 최대 최소값이 제한이 되고 이를 X_max_AL, X_min_AL 을 통해 알립니다.
4. 위에서 최대 최소값 제한된 후의 입력이 $X \leq X\_min + CUT\frac{X\_max - X\_min}{100}$ 라면

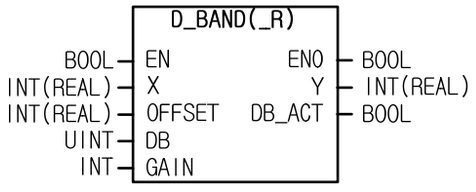
   Y = X_min 을 출력하고 CUT_ACT 를 ON 시킵니다.
5. X_min 의 값이 X_max 의 값보다 클 경우 STAT 은 2 를 가리키고 0 을 출력합니다.

■ 프로그램 예



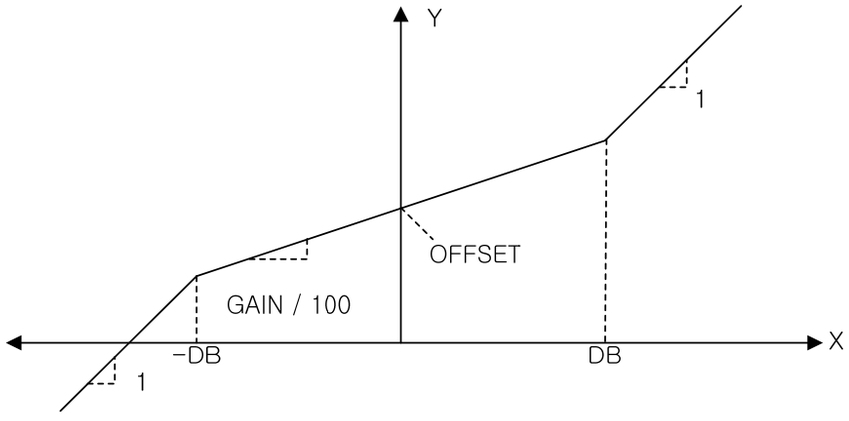
1. X가 4000인 경우 : 16000(Xmax-Xmin)의 5%(CUT) 내에 들지 않았으므로 변화 없이 4000이 출력됩니다.
2. X가 18000인 경우 : 16000으로 제한되어 16000이 출력되며 X_max_AL이 ON 됩니다.
3. X가 100 인 경우 : 16000의 5%인 800 보다 작으므로 0(X_min)이 출력되며 CUT_ACT가 ON 됩니다.

D_BAND(_R)	적용 기종	발생플래그
불감대 적용 출력	XGI, XGR	-

펑션	설 명
	**입력** EN : 펑션 실행 요구 X : 입력값 OFFSET : 출력 옵셋 DB : 데드밴드 반 폭 GAIN : 불감대 구간의 이득율(%) **출력** ENO : 에러 없이 수행 시 On Y : 출력값 DB_ACT : 입력값이 DB 안에 들어온 경우 알람

■ 기능

1. 입력 X 에 불감대를 적용시켜 출력 Y 를 계산합니다.
2. DB 값은 크기를 나타내므로 절대값 연산을 통해 |DB|로 사용합니다.
3. -|DB| ~ |DB| 만큼 데드밴드를 설정합니다.
4. 입력 X 가 데드밴드 안에 들어오면 DB_ACT 비트가 ON 됩니다.
5. 데드밴드 양 끝점은 데드밴드 바깥의 출력에도 영향을 줍니다.
6. 연산 결과가 정수(INT) 데이터 표현범위를 벗어날 경우 출력이 INT (-32768 ~ 32767)로 제한됩니다.
7. 연산 결과가 실수(REAL) 데이터 표현범위를 벗어날 경우 출력이 '1.#inf00000 E+000' 혹은 '-1.#inf00000 E+000'으로 표시되며 이 경우 ENO 비트는 Off 됩니다.
8. 데드밴드의 입출력 방정식은 다음과 같습니다.



A. UNDER THE BAND (X 가 -|DB| 보다 작을 때) :

$$Y = X - (\frac{GAIN}{100} \times DB) + DB + OFFSET$$

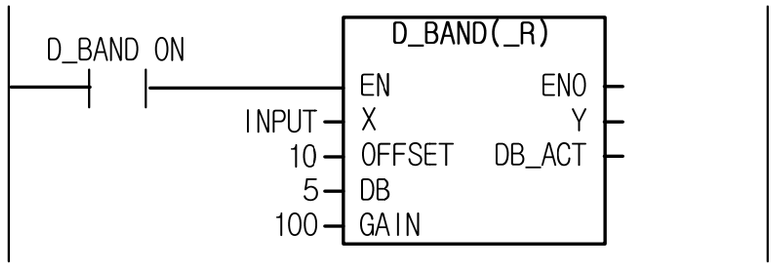
B. IN THE BAND (X 값이 -|DB| ~ |DB| 영역 안에 들어올 때) :

$$Y = (\frac{GAIN}{100} \times X) + OFFSET$$

C. OVER THE BAND (X 가 |DB| 보다 클 때) :

$$Y = X + (\frac{GAIN}{100} \times DB) - DB + OFFSET$$

■ 프로그램 예



1. INPUT이 -8이 입력된 경우 :

$$-8_{(X)} - (\frac{100_{(GAIN)}}{100} \times 5_{(DB)}) + 5_{(DB)} + 10_{(OFFSET)} = 2_{(Y)}$$

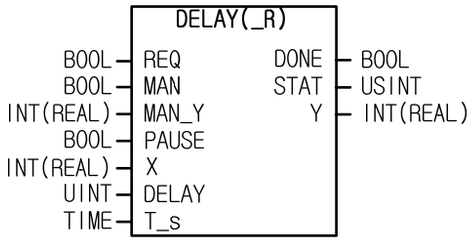
2. INPUT이 3이 입력된 경우 : X의 값이 DB = 5 안에 포함되므로 DB_ACT는 ON

$$(\frac{100_{(GAIN)}}{100} \times 3_{(X)}) + 10_{(OFFSET)} = 13_{(Y)}$$

3. INPUT이 16이 입력된 경우 :

$$16_{(X)} + (\frac{100_{(GAIN)}}{100} \times 5_{(DB)}) - 5_{(DB)} + 10_{(OFFSET)} = 26_{(Y)}$$

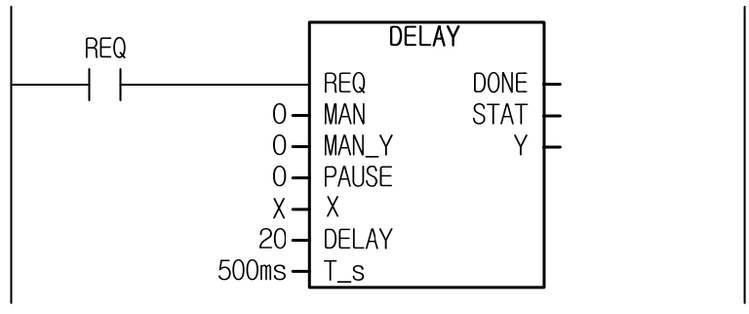
DELAY (_R)	적용 기종	발생플래그
시간 지연 출력	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 MAN : 수동 모드 MAN_Y : 수동 모드 출력 PAUSE : 일시정지 X : 입력값 DELAY : 딜레이 샘플 수 T_s : 연산 주기 **출력** DONE : 에러 없이 수행 시 On STAT : 상태 알람 Y : 출력값

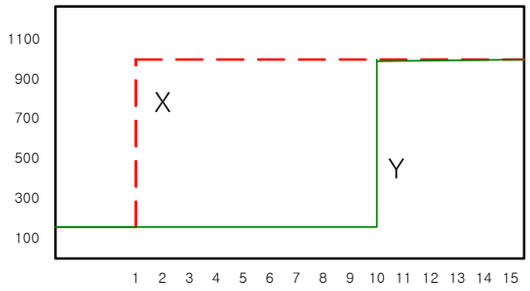
■ 기능

1. 입력 X 를 T_s * DELAY 만큼 지연시킨 출력 Y 를 생성합니다. (T_s 단위 : [sec])
2. 스캔주기마다 현재 입력을 저장하고 동시에 이전 입력을 출력합니다.
3. 처음 동작이 인가되면 저장된 이전값이 없기 때문에 T_s * DELAY 만큼 0 을 출력합니다.
4. DELAY 스캔은 최대 100 스캔까지 입력이 가능하며 그 이상의 값을 입력하면 해당 STAT 에 8 을 출력하고 동작하지 않습니다.
5. PAUSE 가 ON 이면 출력은 일시 정지하고 현재 데이터는 저장합니다.
6. MAN 이 ON 이면 수동 모드로 MAN_Y 를 출력하고 현재 데이터를 저장하지 않으므로 다시 자동 모드로 복귀시 T_s * DELAY 만큼 0 을 출력합니다.

■ 프로그램 예

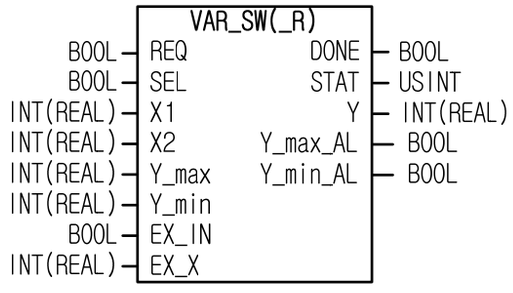


1. DELAY가 20이고 T_s가 500ms 이므로 Y는 10초 전의 X값을 출력한다.


1100
900
700
500
300
100
X
Y
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

VAR_SW(_R)	적용 기종	발생플래그
상수 선택 스위치	XGI, XGR	-

펑션 블록



설 명

입력

REQ	: 펑션 블록 실행 요구
SEL	: 입력 1/2 선택
X1	: 입력 1
X2	: 입력 2
Y_max	: 출력 최대 제한값
Y_min	: 출력 최소 제한값
EX_IN	: 외부 입력 선택
EX_X	: 외부 입력값

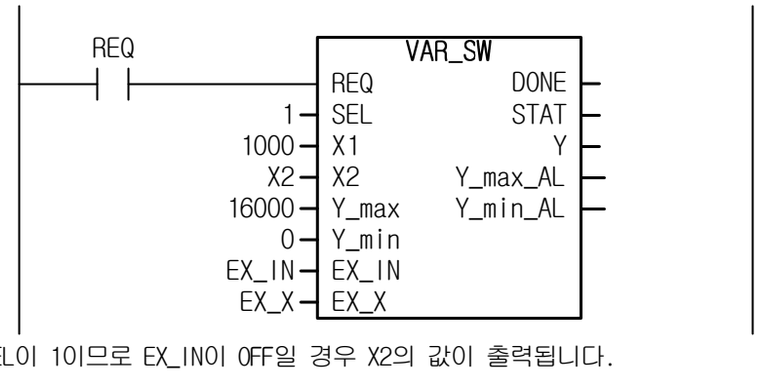
출력

DONE	: 에러 없이 수행 시 On
STAT	: 상태 알람
Y	: 출력값
Y_max_AL	: 출력 최대값 이상 알람
Y_min_AL	: 출력 최소값 이하 알람

■ 기능

1. SEL 비트의 설정에 따라 X1 또는 X2 를 출력합니다.
2. Y_max 및 Y_min 을 설정하여 출력의 최대 최소값을 제한할 수 있습니다.
3. EX_X 에 외부기기(MMI 등)를 연결하여 EX_IN 을 출력할 수 있습니다.
4. EX_X 역시 최대 최소값의 제한을 받습니다.
5. Y_min 의 값이 Y_max 의 값보다 높을 경우 STAT 는 4 를 출력합니다.

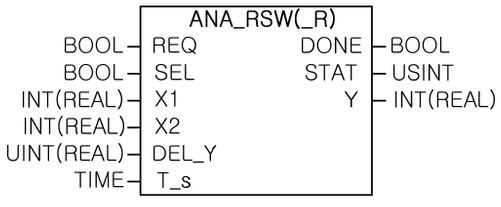
■ 프로그램 예



SEL이 1이므로 EX_IN이 OFF일 경우 X2의 값이 출력됩니다.

1. X2가 10000 이고 EX_IN이 OFF일 경우 : X2가 적용되어 10000이 출력됩니다.
2. X2가 20000 이고 EX_IN이 OFF일 경우 : X2가 적용되어 최대값에 제한된 후 16000이 출력되고 Y_max_AL은 ON이 됩니다.
3. X2가 1000 이고 EX_IN=ON, EX_X=-1000일 경우 : EX_IN이 적용되어 최소값에 제한된 후 0이 출력되고 Y_min_AL은 ON이 됩니다.

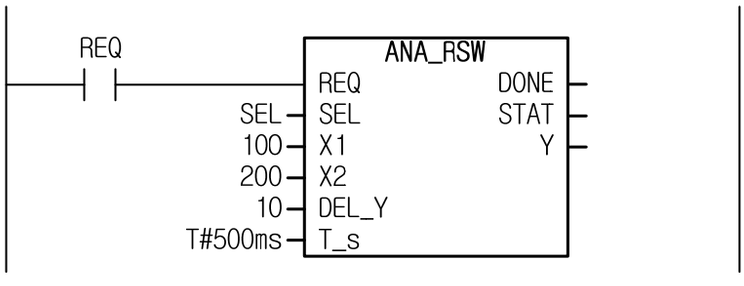
ANA_RSW(_R)	적용 기종	발생플래그
아날로그 증분 제한형 스위치	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 SEL : 입력 선택 X1 : 입력 1 X2 : 입력 2 DEL_Y : 출력 증분 허용치 T_s : 연산 주기 **출력** DONE : 에러 없이 수행 시 On STAT : 상태 알람 Y : 출력값

■ 기능

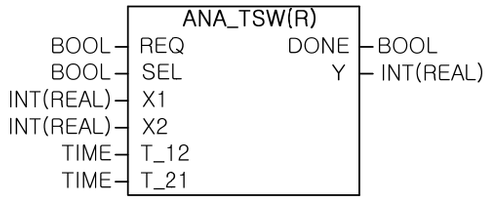
1. SEL 비트의 설정에 따라 X1, X2 입력을 선택 출력합니다.
2. REQ 가 ON 되는 순간, RESET 동작을 합니다. 따라서 SEL 에 의해 선택된 입력을 초기값으로 출력합니다.
3. SEL 비트가 바뀌면 연산주기 T_s 마다 DEL_Y 값 만큼 Y 가 증가/감소하여 선택된 값(X1 / X2) 에 도달합니다.
4. SEL 비트가 바뀌지 않아도 SEL 에 의해 선택된 값(X1 / X2)이 바뀌면 T_s 마다 DEL_Y 값 만큼 Y 가 증가/감소하여 선택된 값에 도달합니다.

■ 프로그램 예



1. SEL=OFF 인 상태에서 SEL=ON으로 바뀌면 500ms 마다 Y가 10씩 증가하여 Y=200 이 됩니다.
2. SEL=OFF 인 상태에서 X1을 300으로 변경하면 500ms 마다 Y가 10씩 증가하여 10초 후 Y=300 이 됩니다.

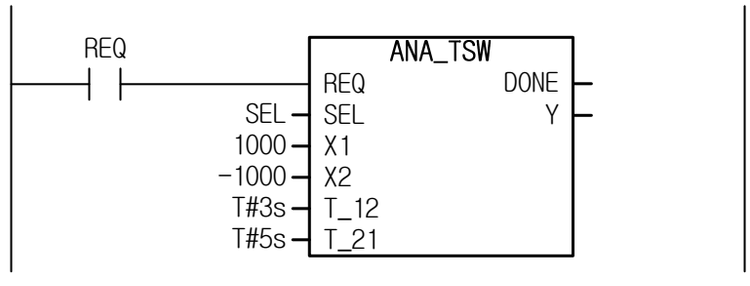
ANA_TSW(_R)	적용 기종	발생플래그
아날로그 시간 제한형 스위치	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 SEL : 입력 선택 X1 : 입력 1 X2 : 입력 2 T_12 : 입력 1->2 전환 시간 T_21 : 입력 2->1 전환 시간 **출력** DONE : 에러 없이 수행 시 On Y : 출력값

■ 기능

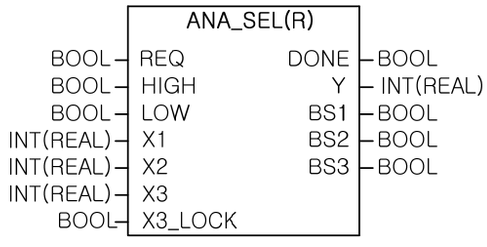
1. SEL 비트의 설정에 따라 X1, X2 입력을 선택 출력합니다.
2. REQ 가 ON 되는 순간, RESET 동작을 합니다. 따라서 SEL 에 의해 선택된 입력을 초기값으로 출력합니다.
3. SEL 변경 이전의 데이터로부터 이후의 데이터까지 정해진 시간을 기준으로 서서히(RAMP) 변화 시킵니다.
4. SEL 선택에 따라 X1 에서 X2 로 변경될 경우 T_12 시간을 따르며, X2 에서 X1 으로 변경될 경우 T_21 시간을 따릅니다.
5. 정수형 명령어 ANA_TSW 에서는 전환 중 반올림이 수행되기 때문에 최대 0.5 의 오차가 발생하므로 정해진 시간보다 일찍 목표입력에 도달할 수 있습니다.
6. 연산 결과가 정수(INT) 데이터 표현범위를 벗어날 경우 출력이 INT (-32768 ~ 32767)로 제한됩니다.
7. 연산 결과가 실수(REAL) 데이터 표현범위를 벗어날 경우 출력이 '1.#inf00000 E+000' 혹은 '-1.#inf00000 E+000'으로 표시되며 이 경우 DONE 비트가 Off 됩니다.

■ 프로그램 예



1. SEL=OFF → ON 인 경우 : 3초에 걸쳐서 Y=1000 → -1000 으로 서서히 감소합니다.
2. SEL=ON → OFF 인 경우 : 5초에 걸쳐서 Y=-1000 → 1000 으로 서서히 증가합니다.

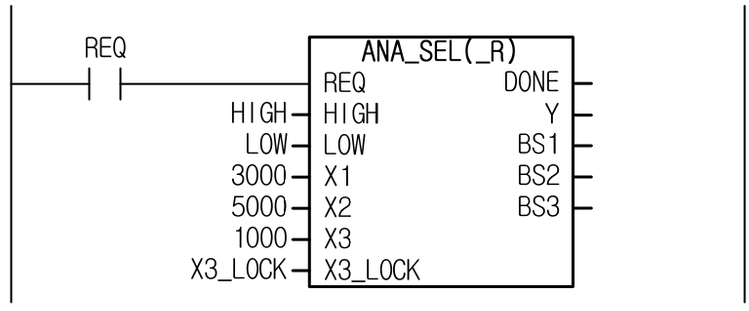
ANA_SEL(_R)	적용 기종	발생플래그
아날로그 크기 비교형 스위치	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 HIGH : 크기에 따른 입력 선택 LOW : 크기에 따른 입력 선택 X1 : 입력 1 X2 : 입력 2 X3 : 입력 3 X3_LOCK : 입력 3 유효 비트 **출력** DONE : 에러 없이 수행 시 On Y : 출력값 BS1 : 블록 셀렉트 1 BS2 : 블록 셀렉트 2 BS3 : 블록 셀렉트 3

■ 기능

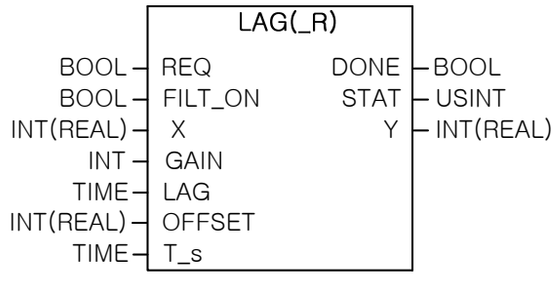
1. HIGH = ON, LOW = OFF 이면 X1 ~ X3 중 가장 큰 값을 출력하고 그에 해당하는 BS 를 ON 시킵니다.
2. HIGH = OFF, LOW = ON 이면 X1 ~ X3 중 가장 작은 값을 출력하고 그에 해당하는 BS 를 ON 시킵니다.
3. HIGH = LOW(모두 ON 혹은 모두 OFF) 으로 설정할 경우 중간값을 선택합니다. X1 ~ X3 중 중간값을 출력하고 그에 해당하는 BS 를 ON 시킵니다.
4. 위와 같이 중간값을 선택하고 2 개의 입력이 같을 경우, 해당 2 개의 값을 출력 Y 로 출력하고 그에 해당하는 2 개의 BS 는 ON 이 됩니다.
5. 중간값을 선택하고 3 개의 입력이 모두 같다면 같은 3 개의 값을 출력 Y 로 출력하고 모든 BS 를 ON 합니다.
6. X3_LOCK = ON 이면 입력 중 X3 은 무시되며 이 경우 2 입력이 되므로 중간값은 둘 중 큰 값이라고 정의합니다.

■ 프로그램 예



1. HIGH = ON, LOW = OFF, X3_LOCK = OFF 인 경우 : Y = 5000을 출력하고 BS2 = ON 시킵니다.
2. HIGH = ON, LOW = ON, X3_LOCK = OFF 인 경우 : Y = 3000을 출력하고 BS1 = ON 시킵니다.
3. HIGH = OFF, LOW = OFF, X3_LOCK = OFF 인 경우 : Y = 3000을 출력하고 BS1 = ON 시킵니다.
4. HIGH = OFF, LOW = ON, X3_LOCK = OFF 인 경우 : Y = 1000을 출력하고 BS3 = ON 시킵니다.
5. HIGH = OFF, LOW = ON, X3_LOCK = ON 인 경우 : Y = 3000을 출력하고 BS1 = ON 시킵니다.
6. HIGH = ON, LOW = ON, X3_LOCK = ON 인 경우 : Y = 5000을 출력하고 BS2 = ON 시킵니다.

LAG(_R)	적용 기종	발생플래그
고주파 제한 필터	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 FILT_ON : 필터 동작 허용 X : 입력값 GAIN : 필터 이득율(%) LAG : LAG 필터 계수 OFFSET : 출력 옵셋 T_s : 연산 주기 **출력** DONE : 에러 없이 수행 시 On STAT : 상태 알람 Y : 출력값

■ 기능

1. 고주파 성분을 제한하는 필터 처리를 합니다.
2. 입력 X 가 LAG 필터를 거쳐 출력 Y 로 출력됩니다.
3. 입력과 출력의 전달과정에서 0.001%미만의 오차가 발생할 수 있습니다.
4. FILT_ON 비트가 Off 일 경우 LAG 필터는 입력을 거르지 않으며 이 때의 출력 방정식은 다음과 같습니다.

$$Y' = \frac{GAIN}{100} \times X$$

5. FILT_ON 비트가 On 일 경우 LAG 필터가 동작하고 이 때의 내부 출력 방정식은 다음과 같습니다.

$$Y' = Y'_{old} + \frac{T\_s}{LAG + T\_s} \times \left(\frac{GAIN}{100} \times \frac{X + X_{old}}{2} - Y'_{old}\right)$$

T_s : [sec]

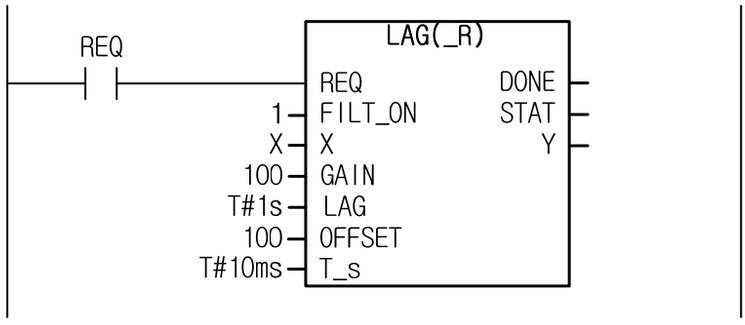
6. 필터 연산 후에는 내부 출력값에 OFFSET 이 더해지며, OFFSET 은 필터를 통과하지 않습니다.

$$Y = Y' + OFFSET$$

주) 위의 식에서 Y 는 실제 출력을 나타내며 Y' 는 내부 출력임.

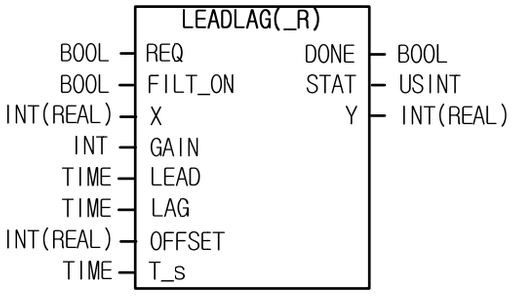
7. LAG_R 의 연산 과정에서 데이터가 실수형 변수(REAL)의 표현 범위를 벗어날 경우 STAT 8 을 가리키고 0 을 출력합니다.

■ 프로그램 예



REQ 와 FILT_ON 가 ON 된 상태에서 입력 X 의 값이 변하면 고주파 성분을 필터링하여 출력합니다.
10msec (T_s)마다 입출력 방정식에 의한 연산이 되어 출력을 생성합니다.

LEADLAG(_R)	적용 기종	발생플래그
고주파 / 저주파 제한 필터	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 FILT_ON : 필터 동작 허용 X : 입력값 GAIN : 필터 이득율(%) LEAD : LEAD 필터 계수 LAG : LAG 필터 계수 OFFSET : 출력 옵셋 T_s : 연산 주기 **출력** DONE : 에러 없이 수행 시 On STAT : 상태 알람 Y : 출력값

■ 기능

1. 고주파 및 저주파 제한 필터 처리를 합니다.
2. LEAD 필터와 LAG 필터를 통해 출력이 생성됩니다.
3. 입력과 출력의 전달과정에서 0.001%미만의 오차가 발생할 수 있습니다.
4. FILT_ON 비트가 Off 일 경우 LEADLAG 필터는 입력을 거르지 않으며 이 때의 출력 방정식은 다음과 같습니다.

$$Y' = \frac{GAIN}{100} \times X$$

5. FILT_ON 비트가 On 일 경우 LEADLAG 필터가 동작하고 이 때의 내부 출력 방정식은 다음과 같습니다.

$$Y' = \frac{LAG \times Y'_{old} + GAIN((LEAD + T\_s)X - LEAD \times X_{old})}{LAG + T\_s}$$

T_s : [sec]

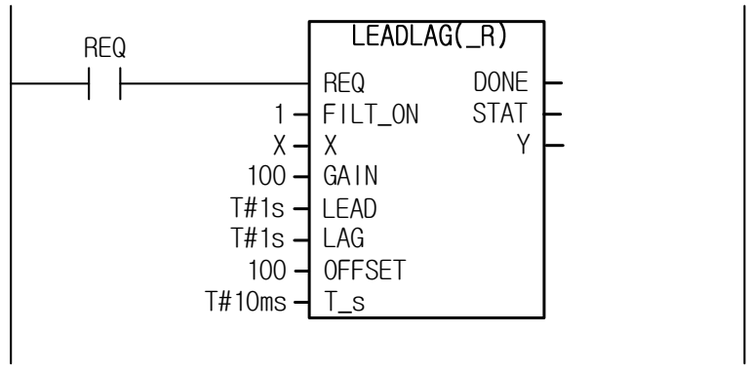
6. 필터 연산 후에는 내부 출력값에 OFFSET 이 더해지며, OFFSET 은 필터를 통과하지 않습니다.

$$Y = Y' + OFFSET$$

주) 위의 식에서 Y 는 실제 출력을 나타내며 Y' 는 내부 출력임.

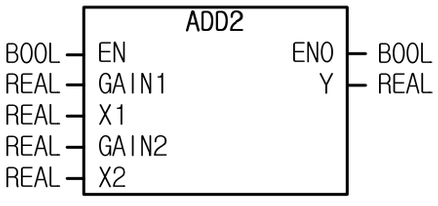
7. LEADLAG_R 의 연산 과정에서 데이터가 실수형 변수(REAL)의 표현 범위를 벗어날 경우 STAT 8 을 가리키고 0 을 출력합니다.

■ 프로그램 예



REQ 와 FILT_ON 가 ON 된 상태에서 입력 X 의 값이 변하면 저주파 및 고주파 성분을 필터링하여 출력합니다.
10msec (T_s)마다 입출력 방정식에 의한 연산이 되어 출력을 생성합니다.

13.4. 산술 연산 펑션, 펑션블록

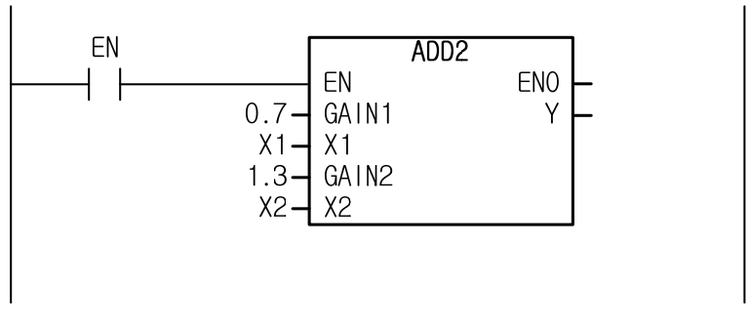
ADD2	적용 기종	발생플래그
Y = G1X1 + G2X2	XGI, XGR	-
펑션	설 명	
	**입력** EN : 펑션 실행 요구 GAIN1 : 연산 이득값 1 X1 : 입력 1 GAIN2 : 연산 이득값 2 X2 : 입력 2 **출력** ENO : 에러 없이 수행 시 On Y : 출력값	

■ 기능

1. 정해진 산술 연산을 수행합니다.
2. 연산 결과가 Y(REAL)의 데이터 표현 범위를 벗어날 경우 ENO 가 OFF 되며
'1.#inf00000 E+000', '-1.#inf00000 E+000', '1.#QNAN0000e+000' 으로 표시되며 이 경우 DONE 비트는 Off 됩니다.

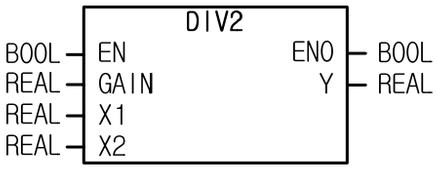
Y = GAIN1 * X1 + GAIN2 * X2

■ 프로그램 예



X1 = 10.0, X2 = 20.0 이면 Y = 0.7(10.0) + 1.3(20.0) = 7.0 + 26.0 = 33.0 입니다.

DIV2	적용 기종	발생플래그
Y = Gain (X1 / X2)	XGI, XGR	-

펑션	설 명
	**입력** EN : 펑션 실행 요구 GAIN : 연산 이득값 X1 : 입력 1 X2 : 입력 2 **출력** ENO : 에러 없이 수행 시 On Y : 출력값

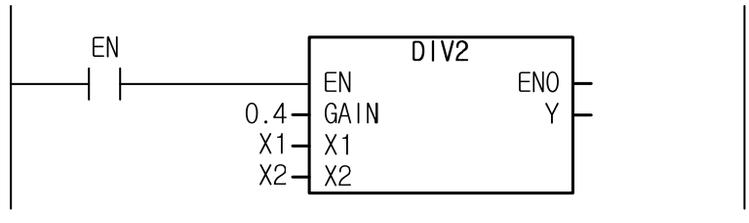
■ 기능

1. 정해진 산술 연산을 수행합니다.

   Y = GAIN (X1 / X2)

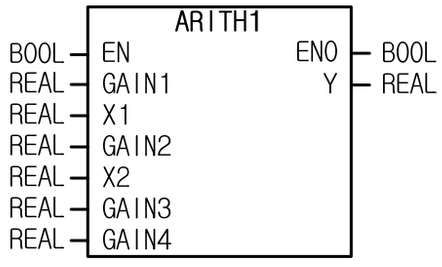
2. X2 의 값이 0 일 경우 분모가 0 이기 때문에 '1.#QNAN0000 E+000' 이 출력됩니다.
3. 연산 결과가 Y(REAL)의 데이터 표현범위를 벗어날 경우 ENO 가 OFF 되며 '1.#inf00000 E+000' 혹은 '-1.#inf00000 E+000' 으로 표시되며 이 경우 DONE 비트는 Off 됩니다.

■ 프로그램 예



X1 = 10.0, X2 = 20.0 이면 Y = 0.4(10.0 / 20.0) = 0.2 입니다.

ARITH1	적용 기종	발생플래그
Y = (G1X1+G2X2)G3 + G4	XGI, XGR	-

펑션	설 명
	**입력** EN : 펑션 실행 요구 GAIN1 : 연산 이득값 1 X1 : 입력 1 GAIN2 : 연산 이득값 2 X2 : 입력 2 GAIN3 : 연산 이득값 3 GAIN4 : 연산 이득값 4 **출력** ENO : 에러 없이 수행 시 On Y : 출력값

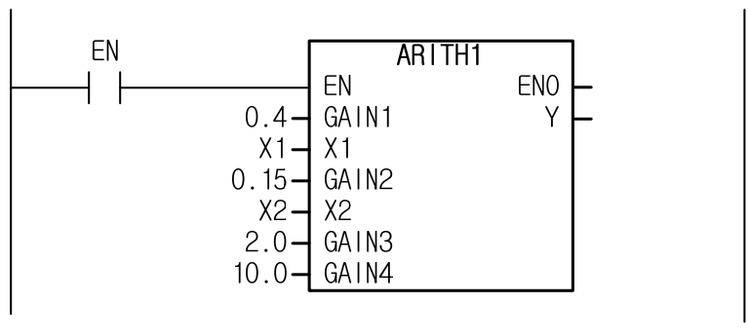
■ 기능

1. 정해진 산술 연산을 수행합니다.

$$Y = (GAIN1 \times X1 + GAIN2 \times X2)GAIN3 + GAIN4$$

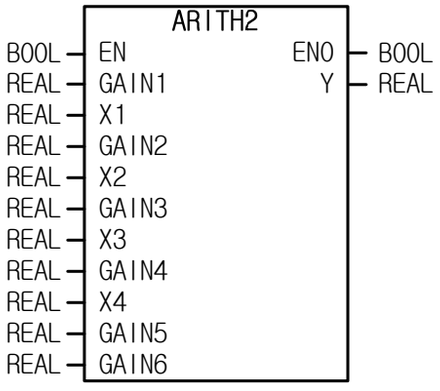
2. 연산 결과가 Y(REAL)의 데이터 표현 범위를 벗어날 경우 ENO 가 OFF 되며
'1.#inf00000 E+000', '-1.#inf00000 E+000', '1.#QNAN0000e+000' 으로 표시되며 이 경우 DONE 비트는 Off 됩니다.

■ 프로그램 예



X1 = 10.0, X2 = 20.0 이면 Y = (0.4(10.0)+0.15(20.0))2.0+10.0 = (4.0+3.0)2.0+10.0 = 24.0 입니다.

ARITH2	적용 기종	발생플래그
Y = (G1X1+G2X2+G3X3+G4X4)G5 + G6	XGI, XGR	-
펑션	설 명	



입력 EN : 펑션 실행 요구
GAIN1 : 연산 이득값 1
X1 : 입력 1
GAIN2 : 연산 이득값 2
X2 : 입력 2
GAIN3 : 연산 이득값 3
X3 : 입력 3
GAIN4 : 연산 이득값 4
X4 : 입력 4
GAIN5 : 연산 이득값 5
GAIN6 : 연산 이득값 6

출력 ENO : 에러 없이 수행 시 On
Y : 출력값

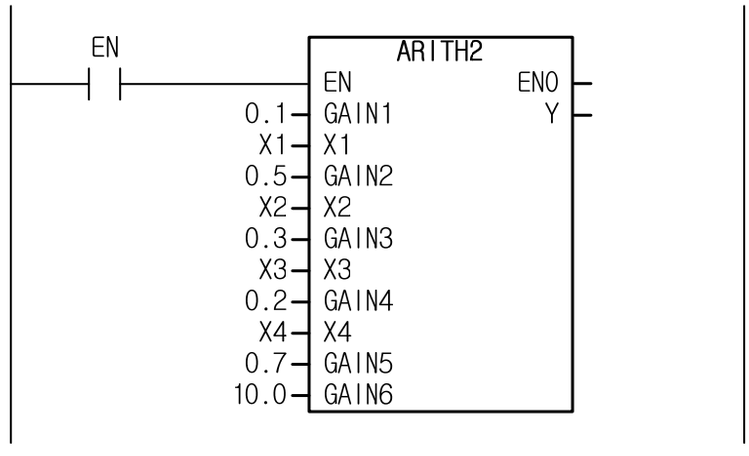
■ 기능

1. 정해진 산술 연산을 수행합니다.

$$Y = (GAIN1 \times X1 + GAIN2 \times X2 + GAIN3 \times X3 + GAIN4 \times X4)GAIN5 + GAIN6$$

2. 연산 결과가 Y(REAL)의 데이터 표현 범위를 벗어날 경우 ENO 가 OFF 되며
'1.#inf00000 E+000', '-1.#inf00000 E+000', '1.#QNAN0000e+000' 으로 표시되며 이 경우 DONE 비트는 Off 됩니다..

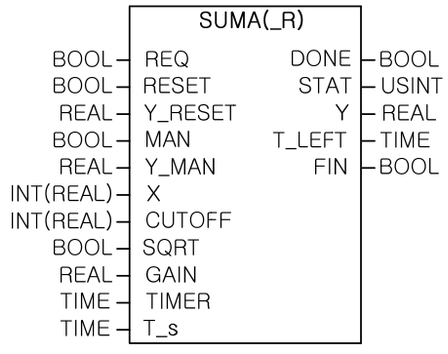
■ 프로그램 예



X1 = 10.0, X2 = 20.0, X3 = 10.0, x4 = 30.0 이면 Y = (0.1(10.0)+0.5(20.0)+0.3(10.0)+0.2(30.0))0.7+10.0
= (1+10+3+6)0.7+10.0 = 24.0 입니다.

SUMA(_R)	적용 기종	발생플래그
아날로그 합산기	XGI, XGR	-

펑션 블록



설 명

입력

REQ	: 펑션 블록 실행 요구
RESET	: 블록 동작 리셋
Y_RESET	: 리셋값
MAN	: 수동모드
Y_MAN	: 수동출력값
X	: 입력값
CUTOFF	: 소신호 무시 폭
SQRT	: 제곱근 설정
GAIN	: 입력 이득율(%)
TIMER	: 타이머 설정
T_s	: 연산 주기

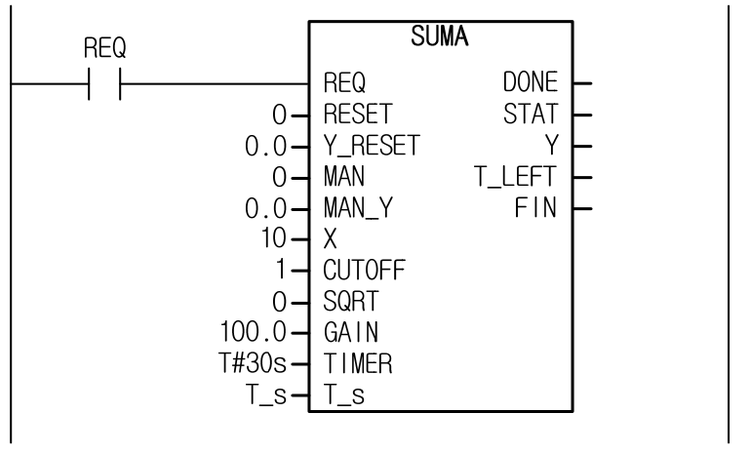
출력

DONE	: 에러 없이 수행 시 On
STAT	: 상태 알람
Y	: 출력값
T_LEFT	: 타이머 남은 시간
FIN	: 타이머 종료 표시

■ 기능

1. X 로 입력되는 아날로그 데이터를 정해진 시간마다 합산하여 Y 로 출력합니다.
2. SUMA (INT 형)명령어는 양이나 음, 한 방향으로 합산이 이뤄질 경우 출력이 크게 증가하므로 너무 빠른 출력 포화를 막기 위해 실수형 출력을 지원합니다.
3. RESET 비트가 On 되면 Y_RESET 값이 출력되고 RESET 비트가 Off 되면 Y_RESET 값부터 연산을 속행합니다.
4. MAN 비트가 On 되면 MAN_Y 값이 출력되다가 MAN 비트가 Off 가 되면 Y_RESET 부터 다시 TIMER 시간 만큼 처음부터 연산합니다.
5. |X|가 |CUTOFF|값 보다 작거나 같을 때 X = 0 으로 처리합니다.
6. SQRT 비트가 On 되면 제곱근 처리된 X 로 연산합니다.
7. 프로그램의 스캔 시간이 1msec 보다 큰 상황에서는 연산을 건너뛰는 구간이 발생하므로 TIMER 가 종료될 때 T_s 설정시간 이하의 오차가 발생할 수 있습니다.
8. 연산 결과가 Y(REAL)의 데이터 표현범위를 벗어날 경우 '1.#inf00000 E+000' 혹은 '-1.#inf00000 E+000'으로 표시되며 이 경우 DONE 비트는 Off 되지만 내부상태(T_LEFT, FIN 등)는 정상적으로 처리됩니다.

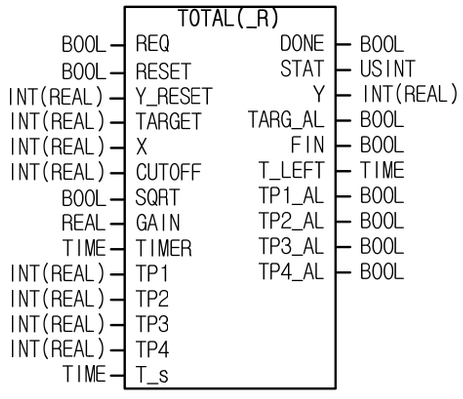
■ 프로그램 예



1. X=10, T_s= T#1s 인 경우 : REQ 를 On 하면 Y 는 매 초당 10 씩 증가하여 30 초 후 Y = 300 이 출력되고 FIN = On
2. X=10, T_s= T#2s 인 경우 : REQ 를 On 하면 Y 는 매 2 초당 10 씩 증가하여 30 초 후 Y = 150 이 출력되고 FIN = On

TOTAL(_R)	적용 기종	발생플래그
아날로그 적산기	XGI, XGR	-

펑션 블록



설 명

입력

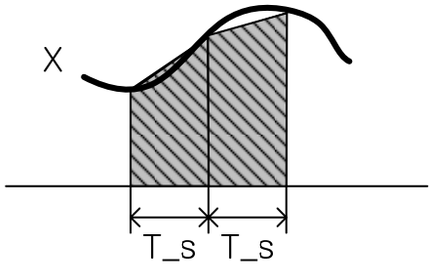
REQ	: 펑션 블록 실행 요구
RESET	: 블록 동작 리셋
Y_RESET	: 리셋값
TARGET	: 목표값
X	: 입력값
CUTOFF	: 소신호 무시 폭
SQRT	: 제곱근 설정
GAIN	: 입력 이득율(%)
TIMER	: 연산 시간
TP1	: 트립 포인트 1
TP2	: 트립 포인트 2
TP3	: 트립 포인트 3
TP4	: 트립 포인트 4
T_s	: 연산 주기

출력

DONE	: 에러 없이 수행 시 On
STAT	: 상태 알람
Y	: 출력값
TARG_AL	: 목표값 도달 알람
FIN	: 연산종료 알람
T_LEFT	: 연산시간 종료 알람
TP1_AL	: 트립 포인트 1 알람
TP2_AL	: 트립 포인트 2 알람
TP3_AL	: 트립 포인트 3 알람
TP4_AL	: 트립 포인트 4 알람

■ 기능

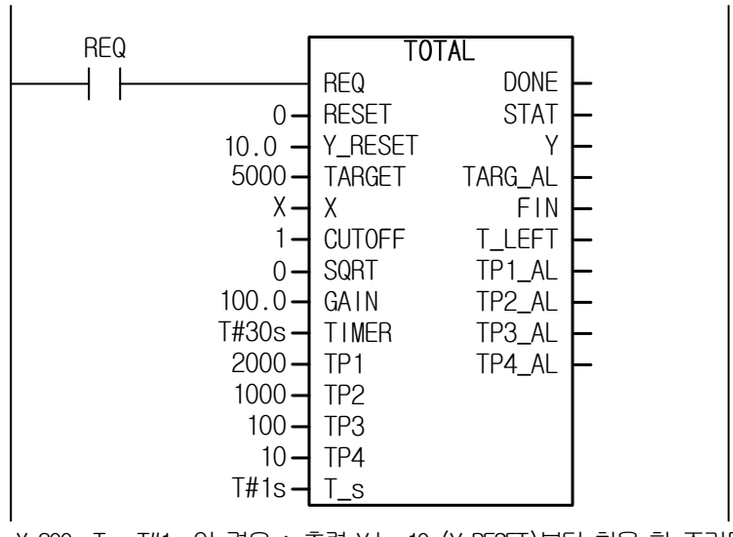
1. X 로 입력되는 아날로그 데이터를 적산합니다.
2. 적산은 초기에 Y_RESET 값부터 이뤄집니다.
3. 다음과 같이 연산 주기 T_s 마다 빗금친 부분의 넓이를 더해나가는 사다리꼴 적산법에 의한 연산을 통해 적산하며 GAIN 을 통해 전달율을 적용합니다.



$$Y = Y_{old} + GAIN/100\ (X_{old} + X)T\_s / 2 \qquad T\_s : [sec]$$

4. RESET 비트가 On 되면 RESET 상태가 되고 Y_RESET 값이 출력됩니다.
5. RESET 비트가 OFF 되어 RESET 상태가 해제되면 Y_RESET 값부터 연산을 다시 시작합니다.
6. 목표치 TARGET 값을 설정하여 출력값이 목표값 이상으로 증가하면 TARG_AL 을 통해 알려줍니다.
7. 출력값이 TARGET-TP[n] ≤ Y ≤ TARGET+TP[n] 범위 이내로 들어오면 해당 TP[n]_AL 을 On 하여 목표값으로 얼마나 접근했는지를 알립니다.
8. 출력 Y 는 TARGET 값에 제한을 받지 않고 증가 감소합니다.
9. |X|가 |CUTOFF|값 보다 작으면 X = 0 으로 처리합니다.
10. SQRT 비트가 On 되면 제곱근 처리된 X 로 연산합니다.
11. 프로그램의 스캔 시간이 1msec 보다 큰 상황에서는 연산을 건너뛰는 구간이 발생하므로 TIMER 가 종료될 때 T_s 설정시간 이하의 오차가 발생할 수 있습니다.
12. 입력과 출력 사이에서 0.001%미만의 오차가 발생할 수 있습니다.
13. |GAIN * X| 가 1.0e+38 이상의 매우 큰 범위일 경우 연산 과정이 부정확 할 수 있습니다.
14. 연산 결과가 정수(INT) 데이터 표현범위를 벗어날 경우 출력이 INT (-32768 ~ 32767)로 제한됩니다.
15. 연산 결과가 실수(REAL) 데이터 표현범위를 벗어날 경우 출력이 '1.#inf00000 E+000' 혹은 '-1.#inf00000 E+000'으로 표시되며 이 경우 DONE 비트는 Off 되지만 내부상태(T_LEFT, FIN 등)는 정상적으로 처리됩니다.

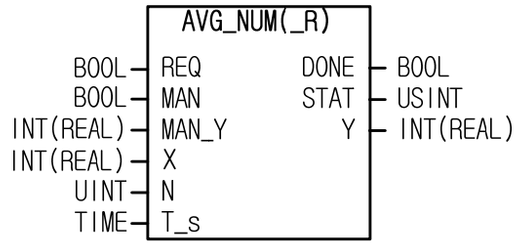
■ 프로그램 예



1. X=200, T_s=T#1s 인 경우 : 출력 Y 는 10 (Y_RESET)부터 처음 한 주기동안 100 증가하고(사다리꼴 적산법) 다음 주기부터 매 1 초당 200 씩 증가하여 30 초가 지나면 5910 출력
   TARG_AL 은 Y ≥ 5000 인 경우 On
   TP1_AL 은 5000 - TP1 ≤ Y ≤ 5000 + TP1 인 경우 On
   TP2_AL 은 5000 - TP2 ≤ Y ≤ 5000 + TP2 인 경우 On
   TP3_AL 은 5000 - TP3 ≤ Y ≤ 5000 + TP3 인 경우 On
   TP4_AL 은 5000 - TP4 ≤ Y ≤ 5000 + TP4 인 경우 On

2. X=200, T_s=T#5s 인 경우 : 출력 Y 는 10 (Y_RESET)부터 처음 한 주기동안 500 증가하고(사다리꼴 적산법) 다음 주기부터 매 5 초당 1000 씩 증가하여 30 초가 지나면 5510 출력
   TARG_AL 은 Y ≥ 5000 인 경우 On
   TP1_AL 은 5000 - TP1 ≤ Y ≤ 5000 + TP1 인 경우 On
   TP2_AL 은 5000 - TP2 ≤ Y ≤ 5000 + TP2 인 경우 On
   TP3_AL 은 5000 - TP3 ≤ Y ≤ 5000 + TP3 인 경우 On
   TP4_AL 은 5000 - TP4 ≤ Y ≤ 5000 + TP4 인 경우 On

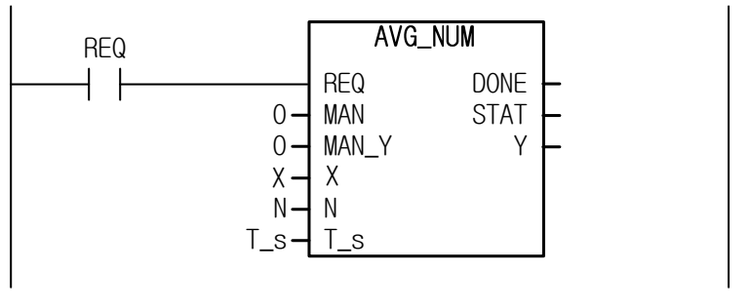
AVG_NUM(_R)	적용 기종	발생플래그
횟수 평균 출력	XGI, XGR	_LER

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 MAN : 수동모드 설정 MAN_Y : 수동 출력 X : 입력값 N : 평균 횟수 T_s : 연산 주기 **출력** DONE : 에러 없이 수행 시 On STAT : 상태 알람 Y : 출력값

■ 기능

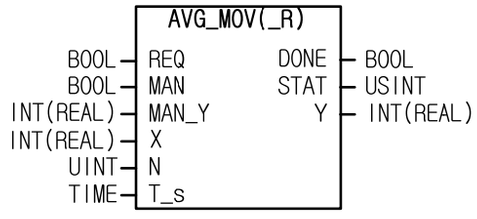
1. T_s 마다 입력 X 를 받아 N 회 평균값을 출력합니다.
2. 출력 Y 는 N * T_s 시간마다 새로운 평균값으로 업데이트 됩니다.
3. MAN 비트 On 시 T_s 는 무시하며 출력 Y 에는 MAN_Y 가 출력됩니다.
4. N 의 값이 0 이거나 30001 이상일 경우 STAT 에 8 을 출력합니다.
5. 연산 결과가 정수(INT) 데이터 표현범위를 벗어날 경우 출력이 INT (-32768 ~ 32767)로 제한됩니다.
6. 연산 과정에서 X * N 이 실수(REAL) 데이터 표현범위를 벗어날 경우 출력이 '1.#inf00000 E+000' 혹은 '-1.#inf00000 E+000' 으로 표시되며 연산에러 플래그(_LER)가 발생합니다. 이 경우 DONE 비트는 Off 됩니다.

■ 프로그램 예



1. X 가 0 부터 초당 1 증가, T_s= T#1s, N=3 : Y 는 3 초마다 3 씩 증가
2. X 가 0 부터 초당 1 증가, T_s= T#2s, N=3 : Y 는 6 초마다 6 씩 증가
3. X 가 0 부터 초당 1 증가, T_s= T#1s, N=6 : Y 는 6 초마다 6 씩 증가

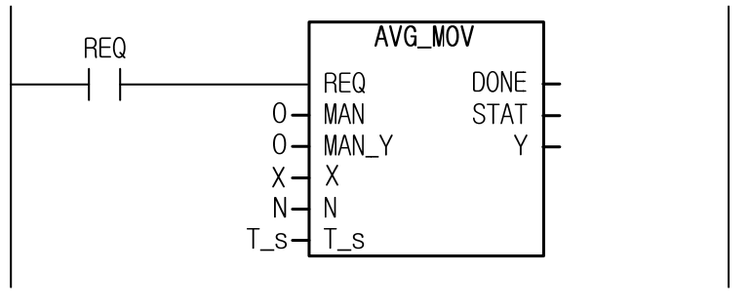
AVG_MOV(_R)	적용 기종	발생플래그
이동 평균 출력	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 MAN : 수동모드 설정 MAN_Y : 수동 출력 X : 입력값 N : 평균 횟수 T_s : 연산 주기 **출력** DONE : 에러 없이 수행 시 On STAT : 상태 알람 Y : 출력값

■ 기능

1. T_s 마다 입력 X 를 받아 현재 시각 이전의 값들과 N 회 이동 평균값을 출력합니다.
2. 출력 Y 는 T_s 시간마다 새로운 평균값으로 업데이트 됩니다.
3. MAN 비트 On 시 T_s 는 무시하며 출력 Y 에는 MAN_Y 가 출력됩니다.
4. N 의 값이 0 이거나 101 이상일 경우 STAT 에 8 을 출력합니다.
5. 연산 결과가 정수(INT) 데이터 표현범위를 벗어날 경우 출력이 INT (-32768 ~ 32767)로 제한됩니다.
6. 연산 과정에서 X * N 이 실수(REAL) 데이터 표현범위를 벗어날 경우 출력이 '1.#inf00000 E+000' 혹은 '-1.#inf00000 E+000' 으로 표시되며 이 경우 DONE 비트는 Off 됩니다.

■ 프로그램 예

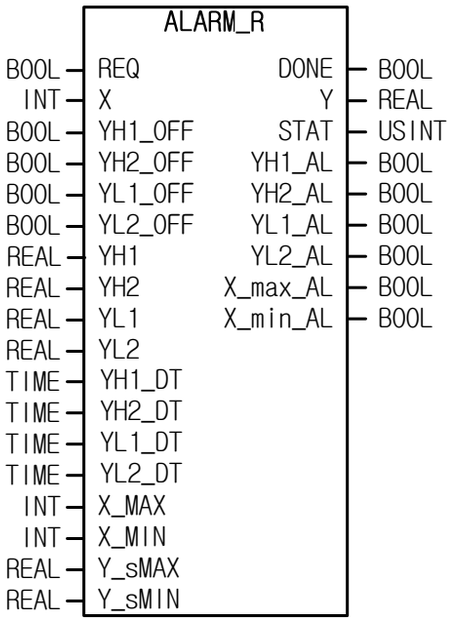


1. X 가 0 부터 초당 1 증가, T_s= T#1s, N=3 : Y 는 1 초마다 1 씩 증가
2. X 가 0 부터 초당 1 증가, T_s= T#2s, N=3 : Y 는 2 초마다 2 씩 증가
3. X 가 0 부터 초당 1 증가, T_s= T#1s, N=6 : Y 는 1 초마다 1 씩 증가

13.5. 데이터 측정 펑션, 펑션블록

ALARM_R	적용 기종	발생플래그
알람 지시기	XGI, XGR	-

펑션 블록



설 명

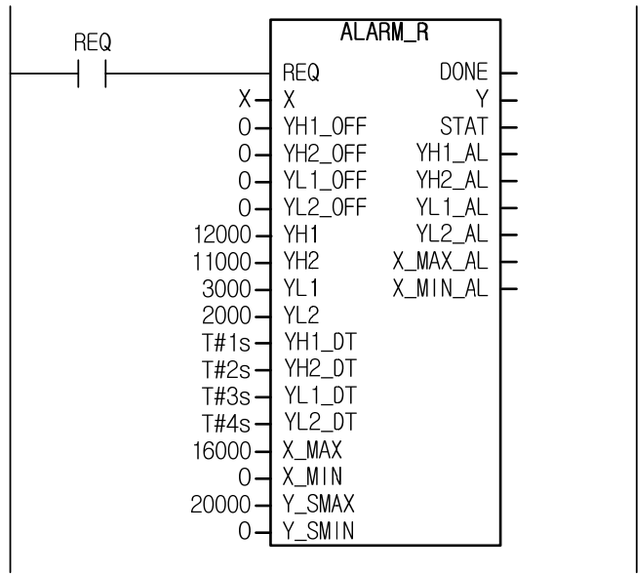
입력
- REQ : 펑션 블록 실행 요구
- X : 입력값
- YH1_OFF : 출력값 상한 1 구간 해제 비트
- YH2_OFF : 출력값 상한 2 구간 해제 비트
- YL1_OFF : 출력값 하한 1 구간 해제 비트
- YL2_OFF : 출력값 하한 2 구간 해제 비트
- YH1 : 출력 상한 1 구간 설정값
- YH2 : 출력 상한 2 구간 설정값
- YL1 : 출력 하한 1 구간 설정값
- YL2 : 출력 하한 2 구간 설정값
- YH1_DT : 출력 상한 1 구간 설정 대기시간(초)
- YH2_DT : 출력 상한 2 구간 설정 대기시간(초)
- YL1_DT : 출력 하한 1 구간 설정 대기시간(초)
- YL2_DT : 출력 하한 2 구간 설정 대기시간(초)
- X_MAX : 입력 최대 제한값
- X_MIN : 입력 최소 제한값
- Y_sMAX : 출력 스케일 최대값
- Y_sMIN : 출력 스케일 최소값

출력
- DONE : 에러 없이 수행 시 On
- Y : 출력값
- STAT : 상태 알람
- YH1_AL : 출력 이상 1 구간 알람
- YH2_AL : 출력 이상 2 구간 알람
- YL1_AL : 출력 이하 1 구간 알람
- YL2_AL : 출력 이하 2 구간 알람
- X_max_AL: 입력 이상 알람
- X_max_AL: 입력 이상 알람

■ 기능

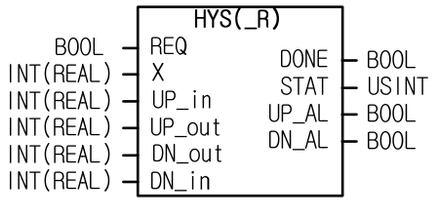
1. 정수 입력 X 를 실수로 바꾸어 출력하며, 2 개의 상한과 2 개의 하한 및 스케일 연산을 할 수 있습니다.
2. 입력이 정수형이기 때문에 특수모듈 및 외부 입력값을 받아서 변환없이 입력으로 사용할 수 있습니다.
3. X_MIN ~ X_MAX 범위의 값을 Y_sMIN ~ Y_sMAX 범위의 값으로 스케일 연산합니다.
4. YH1, YH2 는 상한값을 설정하여 이상일 때 알려주며 기능 사용 여부(YH_OFF)와 알리는 지연시간(YH_DT)을 설정할 수 있습니다.
5. YL1, YL2 는 하한값을 설정하여 이하일 때 알려주며 기능 사용 여부(YL_OFF)와 알리는 지연시간(YL_DT)을 설정할 수 있습니다.
6. X_max = X_min 일 경우 연산과정 중 분모가 0 이 되기 때문에 동작하지 않으며 STAT 는 8 을 출력합니다.

■ 프로그램 예



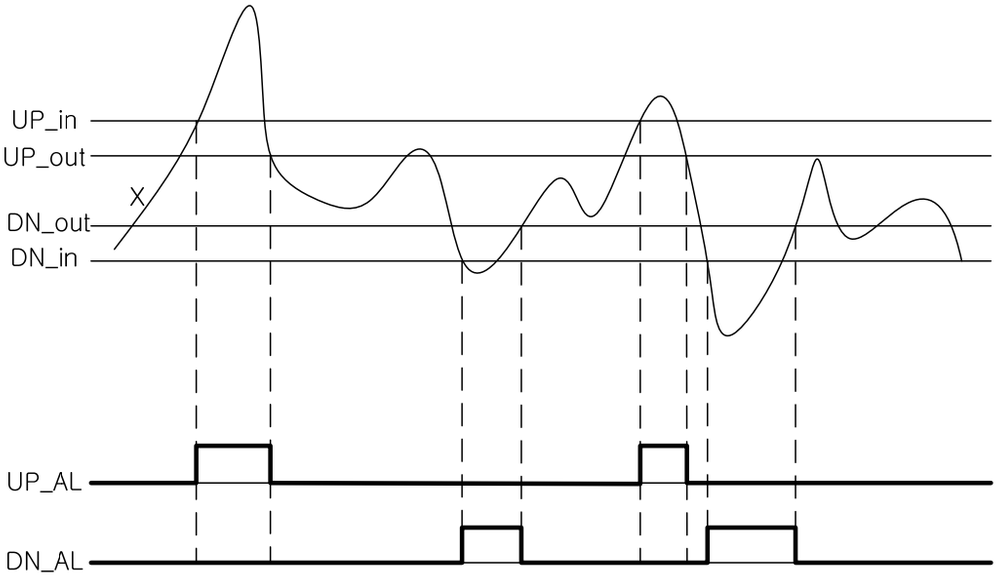
1. X = 8900 인 경우 : Y = 11125, 2 초 후 YH2_AL On
2. X = 11000 인 경우 : Y = 13750, 1 초 후 YH1_AL On, 2 초 후 YH2_AL On
3. X = 2100 인 경우 : Y = 2625, 3 초 후 YL1_AL On
4. X = 1200 인 경우 : Y = 1500, 3 초 후 YL1_AL On, 4 초 후 YL2_AL On

HYS(_R)	적용 기종	발생플래그
방향성 데드밴드	XGI, XGR	-

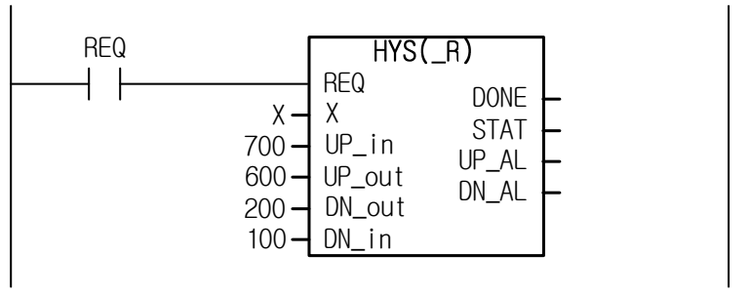
펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 X : 입력값 UP_in : 상승 셋 트리거 UP_out : 상승 리셋 트리거 DN_out : 하강 리셋 트리거 DN_in : 하강 셋 트리거 **출력** DONE : 에러 없이 수행 시 On STAT : 상태 알람 UP_AL : 최대값 이상 알람 DN_AL : 최소값 이상 알람

■ 기능

1. 입력 X 를 받아 방향성 불감대(Hysteresis)를 적용하여 UP/DOWN 상태를 알립니다.
2. UP_in < X 이면 UP_AL 을 On 합니다.
3. UP_out ≤ X ≤ UP_in 이면 이전 UP_AL 상태를 유지합니다.
4. X < UP_out 이면 UP_AL 을 Off 합니다.
5. X < DN_in 이면 DN_AL 을 ON 합니다.
6. DN_in ≤ X ≤ DN_out 이면 이전 DN_AL 상태를 유지합니다.
7. DN_out < X 이면 DN_AL 을 Off 합니다.
8. UP_in 의 값이 UP_out 의 값보다 작을 경우 STAT 에 8 을 출력합니다.
9. DN_out 의 값이 DN_in 의 값보다 작을 경우 STAT 에 8 을 출력합니다.

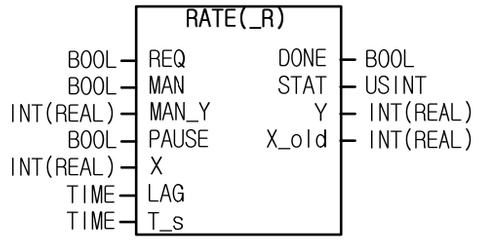


■ 프로그램 예



1. X 가 0 에서 800 으로 변경된 경우 : UP_AL On, DN_AL Off
2. X 가 800 에서 650 으로 변경된 경우 : UP_AL On, DN_AL Off
3. X 가 650 에서 300 으로 변경된 경우 : UP_AL Off, DN_AL Off
4. X 가 300 에서 50 으로 변경된 경우 : UP_AL Off, DN_AL On
5. X 가 50 에서 150 으로 변경된 경우 : UP_AL Off, DN_AL On

RATE(_R)	적용 기종	발생플래그
초당 변화량 측정	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 MAN : 수동 모드 전환 MAN_Y : 수동 출력 값 PAUSE : 일시 정지 X : 입력값 LAG : LAG 필터 계수 T_s : 연산 주기 **출력** DONE : 에러 없이 수행 시 On STAT : 상태 알람 Y : 출력값 X_old : 이전 X 값

■ 기능

1. RATE 함수는 입력 X 의 초당 변화량을 표시해 주는 명령어 입니다.
2. MAN 비트가 On 이면 MAN_Y 를 출력합니다.
3. PAUSE 비트가 ON 이면 블록을 일시정지합니다.
4. LAG 에 시상수를 설정하면 입력에 저역 통과 필터 처리를 합니다.
5. LAG 를 포함한 RATE 명령어의 입출력 방정식은 다음과 같습니다.

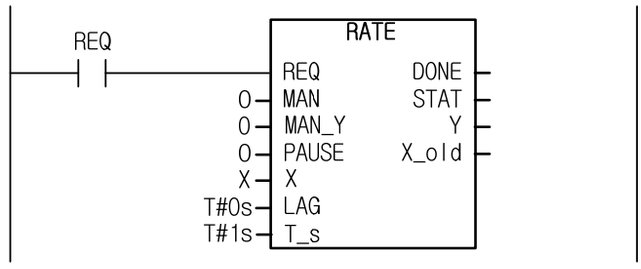
$$Y = Y_{old} + \frac{T\_s}{LAG + T\_s} \times \left(\frac{X + X_{old}}{Ts} - Y_{old}\right) \qquad [T\_s : sec]$$

6. 위의 식에서 LAG 가 0 일 경우 다음과 같이 정리됩니다.

$$Y = \frac{X - X_{old}}{Ts} \qquad [T\_s : sec]$$

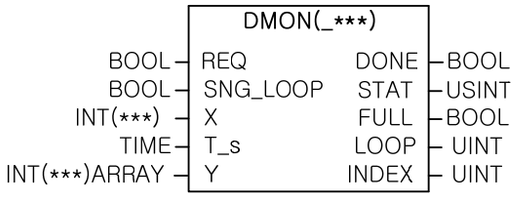
7. 연산 결과가 정수(INT) 데이터 표현범위를 벗어날 경우 출력이 INT (-32768 ~ 32767)로 제한됩니다.
8. 연산 결과가 실수(REAL) 데이터 표현범위를 벗어날 경우 출력이 '1.#inf00000 E+000' 혹은 '-1.#inf00000 E+000'으로 표시되며 이 경우 DONE 비트는 Off 되지만 내부상태(X_old 등)는 정상적으로 처리됩니다.

■ 프로그램 예



1. X가 0부터 초당 1증가하면 Y는 1 출력
2. X가 10부터 초당1증가 하면 Y는 1 출력
3. X가 10부터 초당30감소 하면 Y는 -30 출력

DMON(_***)	적용 기종	발생플래그
입력어레이를 출력어레이 만큼 저장	XGI, XGR	-
펑션 블록	설 명	



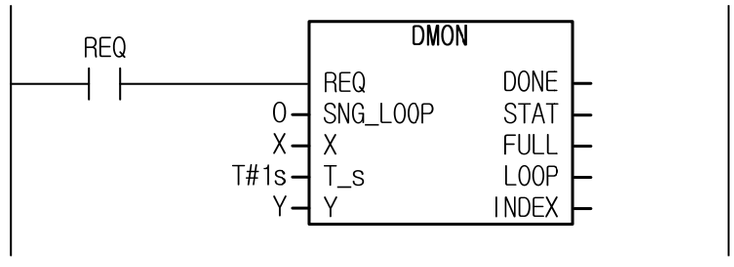
입력
- REQ : 펑션 블록 실행 요구
- SNG/LOOP : 싱글동작 / 루프동작 여부
- X : 입력값
- T_s : 연산 주기
- Y : 출력값

출력
- DONE : 에러 없이 수행 시 On
- STAT : 상태 알람
- FULL : 출력 어레이 꽉참
- LOOP : 출력 어레이가 꽉 찬 횟수
- INDEX : 현재 저장될 위치의 어레이 번호

■ 기능

1. 시간에 따라 변화하는 데이터를 저장하고 싶을 경우에 사용합니다.
2. 연산 주기(T_s)마다 입력 X 를 Y(ARRAY)에 순서대로 저장합니다.
3. DMON 펑션블록은 INT 형 명령어이며 DMON 뒤에 붙는 데이터형은 _DI(DINT), _R(REAL), _UI(UINT), _UDI(UDINT), _W(WORD), _DW(DWORD) 와 같이 입출력 데이터별로 선택하여 사용할 수 있습니다.
4. SNG_LOOP 가 Off 이면 싱글 동작으로, 어레이 수 만큼 입력을 저장하고 FULL 을 On 시키며 정지합니다.
5. SNG_LOOP 가 On 이면 루프 동작으로, 어레이 수 만큼 입력을 저장한 후에도 계속해서 어레이의 처음부터 새로운 값을 덮어쓰며 멈추지 않고 동작합니다.
6. SNG_LOOP 를 바꾸어 싱글/루프 동작으로 전환 할 경우 REQ 를 다시 인가하여 초기화 후 사용 바랍니다.
7. 루프 동작 시 어레이가 꽉 찰 때마다 LOOP 는 증가합니다. LOOP 값이 65535 를 넘으면 0 으로 재설정 됩니다.

■ 프로그램 예



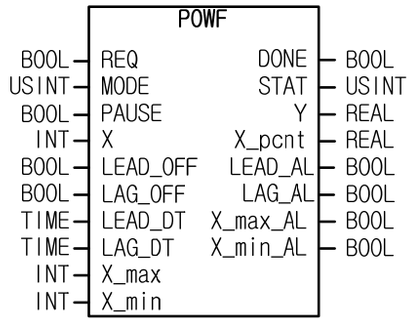
Y 는 ARRAY [0..10] of INT 형으로 설정합니다.

1. X 가 0 부터 초당 1 증가 : 매 초당 Y[0]=0 … Y[10]=10 의 순서대로 값이 저장되며 12 초부터 FULL=On
2. X 가 10 부터 초당 1 증가 : 매 초당 Y[0]=10 … Y[10]=20 의 순서대로 값이 저장되며 12 초부터 FULL=On
3. X 가 10 부터 초당 3 감소 : 매 초당 Y[0]=10 … Y[10]=-20 의 순서대로 값이 저장되며 12 초부터 FULL=On

13.6. 데이터 생성 펑션, 펑션블록

POWF	적용 기종	발생플래그
역율 계기	XGI, XGR	-

펑션 블록



설 명

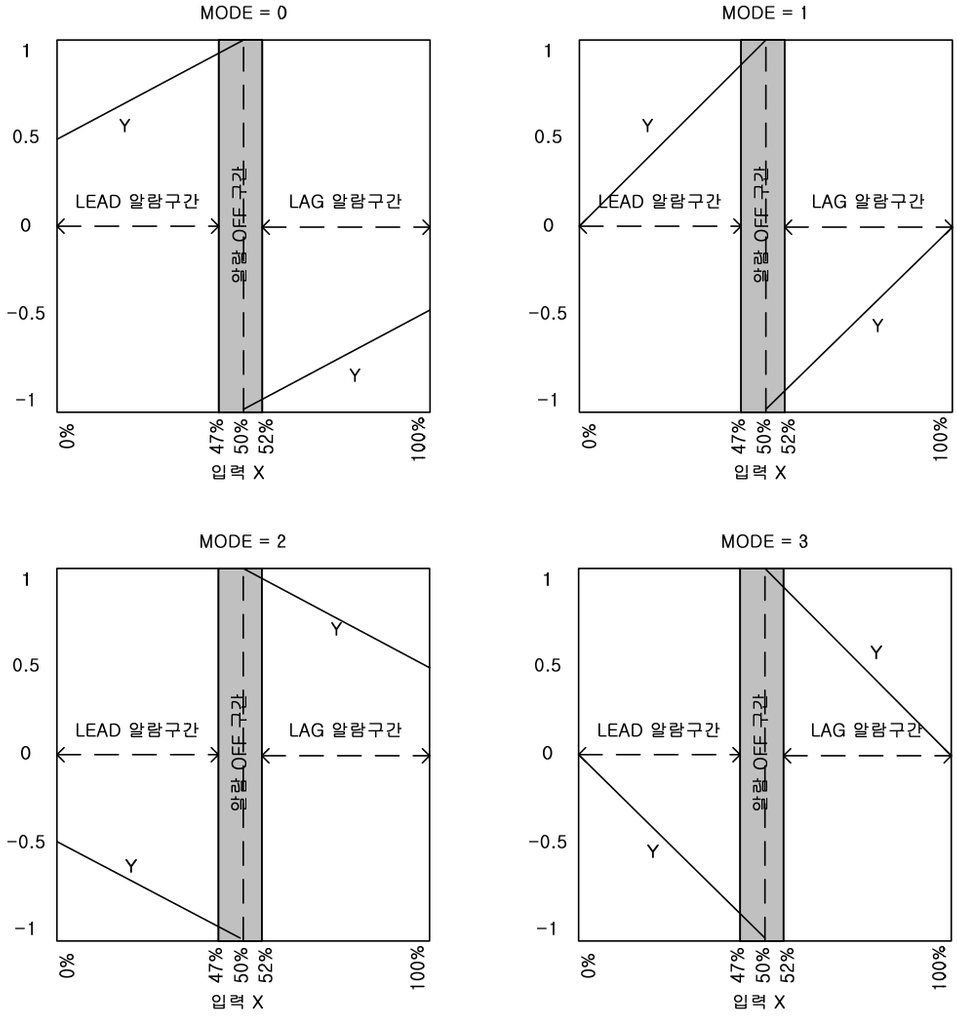
입력
- REQ : 펑션 블록 실행 요구
- MODE : 모드 전환
- PAUSE : 일시 정지
- X : 입력값
- LEAD_OFF : 진상파 알람 잠금
- LAG_OFF : 지상파 알람 잠금
- LEAD_DT : 진상파 알람 ON 지연 시간
- LAG_DT : 지상파 알람 ON 지연 시간
- X_max : 입력 최대 제한값
- X_min : 입력 최소 제한값

출력
- DONE : 에러 없이 수행 시 On
- STAT : 상태 알람
- Y : 출력값
- X_pcnt : 퍼센트 출력
- LEAD_AL : 진상파 알람
- LAG_AL : 지상파 알람
- X_max_AL : 최대값 이상 알람
- X_min_AL : 최소값 이하 알람

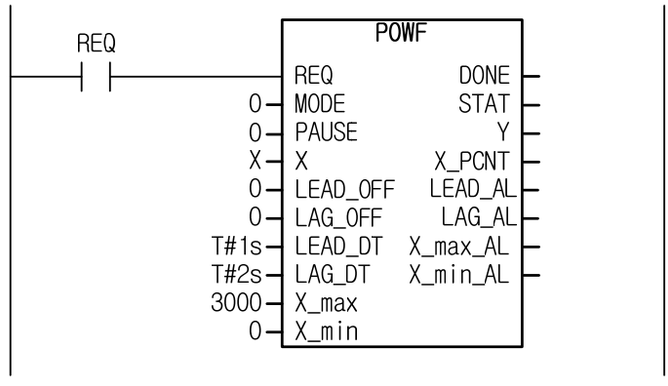
■ 기능

1. 역율 센서로부터 받은 입력 X 를 참조하여 정해진 역율 프로파일에 따라 출력 Y 를 생성합니다.
2. 입력 X 는 X_max, X_min 설정에 의해서 최대, 최소값이 제한됩니다.
3. 입력 X 는 X_max, X_min 설정에 의해 % 단위로 변환되어 X_PCNT 에 표시되며 % 값으로 연산을 수행합니다.
4. 모드(0 ~ 3 선택 가능)에 따라 프로파일의 종류가 선택됩니다. 각 모드에 대한 출력은 아래의 그림과 같습니다.
   A. MODE 0 : 기울기가 0.5, 진상 옵셋 1, 지상 옵셋 -1.
   B. MODE 1 : 기울기가 1, 진상 옵셋 1, 지상 옵셋 -1.
   C. MODE 2 : 기울기가 -0.5, 진상 옵셋 -1, 지상 옵셋 1.
   D. MODE 3 : 기울기가 -1, 진상 옵셋 -1, 지상 옵셋 1.

5. X 가 50%인(그래프의 중심) 특이점에서는 출력 Y 를 0 으로 정의합니다.
6. PAUSE 가 On 이면 연산을 정지하고 연산이 속계될 때까지 알람 비트 역시 표시하지 않습니다.
7. LEAD_AL, LAG_AL 에 진상, 지상을 표시하며 표시여부(_OFF)와 표시 지연시간(_DT)을 설정할 수 있습니다.
8. X_max, X_min 에 입력 X 의 최대, 최소값을 설정할 수 있습니다.
9. MODE 가 3 초과 할 시에 STAT 에 8 을 출력.
10. X_max = X_min 일 경우 연산과정 중 분모가 0 이 되기 때문에 동작하지 않으며 STAT 는 8 을 표시 합니다.
11. 입력과 출력사이에 0.001%미만의 오차가 발생할 수 있습니다.

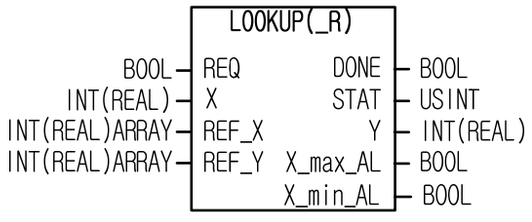


■ 프로그램 예



1. X가 0인 경우 : X_PCNT = 0 이며, Y = 0.5, 1초 후 LEAD_AL = On, LAG_AL = Off
2. X가 1500인 경우 : X_PCNT = 50 이며, Y = 0, LEAD_AL = Off, LAG_AL = Off
3. X가 2000인 경우 : X_PCNT = 66 이며, Y = -0.84, LEAD_AL = Off, 2초 후 LAG_AL = On

LOOKUP(_R)	적용 기종	발생플래그
LOOK-UP 테이블 출력	XGI, XGR	-
펑션 블록	설 명	

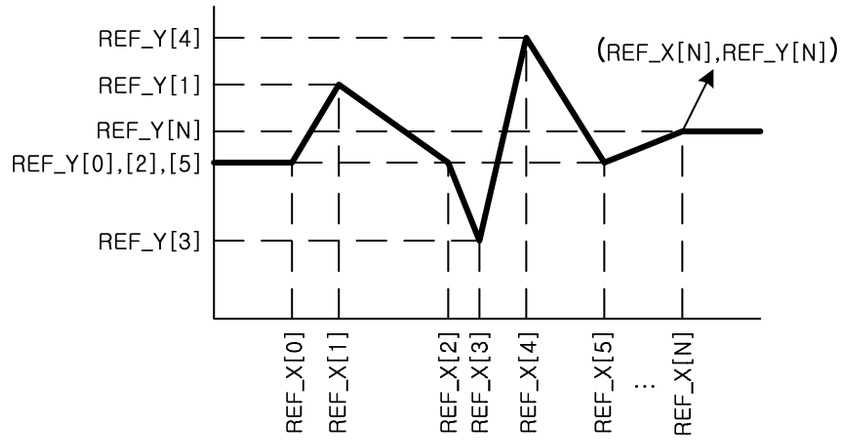


입력 REQ : 펑션 블록 실행 요구
X : 입력값
REF_X : LOOK-UP 테이블의 X 좌표 배열
REF_Y : LOOK-UP 테이블의 Y 좌표 배열

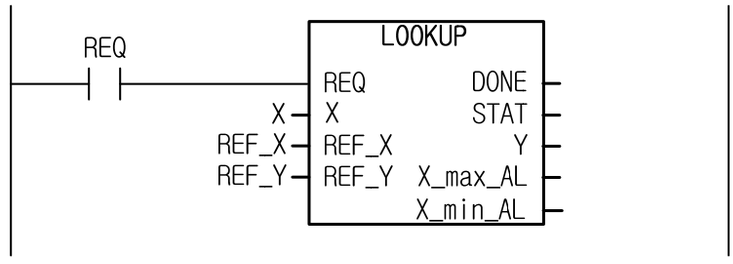
출력 DONE : 에러 없이 수행 시 On
STAT : 상태 알람
Y : 출력값
X_max_AL : REF_X 이상 알람
X_min_AL : REF_X 이하 알람

■ 기능

1. 입력 배열(REF_X)및 출력 배열(REF_Y)을 이용해 구간 별로 선형 LOOK-UP 테이블을 생성하고 입력 X 를 적용하여 출력을 얻어냅니다.
2. 입력 배열 REF_X 는 오름차순으로 정렬되어야 하며 배열의 원소가 같을 경우 에러 처리합니다.
3. 입력 X 를 통해 입력된 값이 입력 배열(REF_X)의 범위와 같거나 벗어날 경우 X_max_AL, X_min_AL 을 표시합니다.
4. REF_X 의 원소가 오름차순 정렬이 되지 않았을 경우 STAT 은 8 을 출력합니다.
5. REF_X 와 REF_Y 의 배열 원소 개수가 다를 경우 STAT 은 8 을 출력합니다.
6. 연산 결과가 정수(INT) 데이터 표현범위를 벗어날 경우 출력이 INT (-32768 ~ 32767)로 제한됩니다.
7. 연산 결과가 실수(REAL) 데이터 표현범위를 벗어날 경우 출력이 '1.#inf00000 E+000' 혹은 '-1.#inf00000 E+000'으로 표시되며 이 경우 DONE 비트는 Off 되지만 내부상태(X_max_AL, X_min_AL 등)는 정상적으로 처리됩니다.



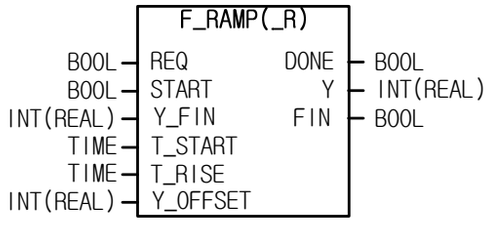
■ 프로그램 예



REF_X 는 ARRAY [0..4] of INT 로 설정하고 배열의 원소는 [10, 20, 30, 40, 50]으로 설정합니다.
REF_Y 는 ARRAY [0..4] of INT 로 설정하고 배열의 원소는 [10, 20, 10, 50, 20]으로 설정합니다.

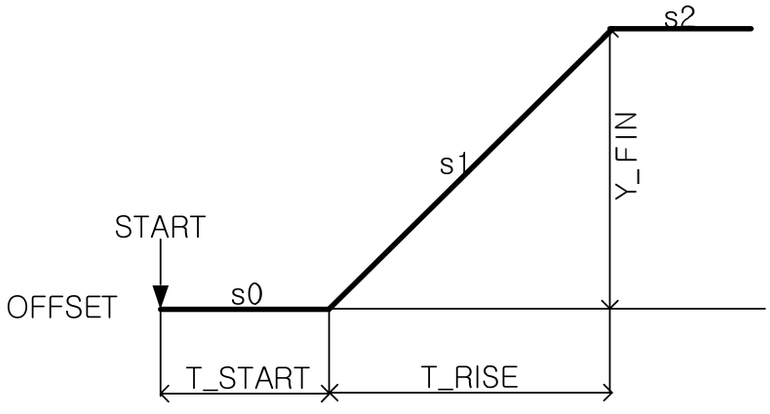
1. X 가 5 일 경우 : Y = 10, X_min_AL = On, X_max_AL = Off
2. X 가 15 일 경우 : Y = 15, X_min_AL = Off, X_max_AL = Off
3. X 가 45 일 경우 : Y = 35, X_min_AL = Off, X_max_AL = Off
4. X 가 100 일 경우 : Y = 20, X_min_AL = Off, X_max_AL = On

F_RAMP(_R)	적용 기종	발생플래그
단발성 RAMP 함수 출력	XGI, XGR	-

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 START : 연산 시작 Y_FIN : RAMP 함수 목표값 T_START : 연산 대기 시간 T_RISE : 전체 상승 구간 Y_OFFSET : 출력 옵셋 **출력** DONE : 에러 없이 수행 시 On Y : 출력 FIN : 정상 상태 알람

■ 기능

1. RAMP 함수를 출력합니다.
2. START On 시에 파형 출력을 시작합니다.
3. REQ 가 Off 이면 연산 마지막 상태의 값을 유지합니다.
4. REQ 가 On 상태에서 START 가 Off 이면 초기값으로 초기화 후 연산 시작(START On)을 대기합니다.
5. Y_FIN 은 RAMP 함수의 목표값, T_START 는 START 후 대기 시간, T_RISE 는 파형 상승 시간, Y_OFFSET 에는 옵셋을 설정합니다.
6. 파형 상승이 끝나면 FIN 이 On 됩니다.
7. F_RAMP : Y_FIN + Y_OFFSET 가 Y(INT)의 데이터 표현범위를 벗어날 경우 -32768 ≤ Y ≤ 32767 로 제한됩니다.
8. F_RAMP_R : Y_FIN + Y_OFFSET 가 Y(REAL)의 데이터 표현범위를 벗어날 경우 연산 도중에 결과가 '1.#inf00000 E+000' 혹은 '-1.#inf00000 E+000'으로 표시되며 이 경우 DONE 비트는 Off 되지만 내부상태(FIN 등)는 정상적으로 처리됩니다.



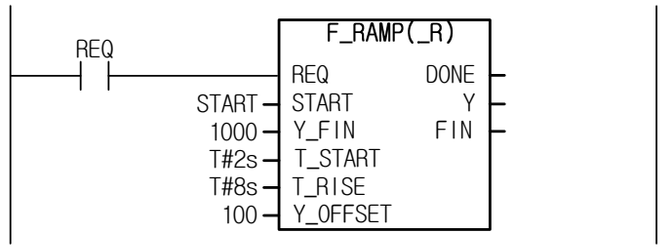
각 구간의 식은 다음과 같습니다.

s0 : Y = Y_OFFSET

s1 : Y = Y_FIN * (t - T_START) / T_RISE + Y_OFFSET

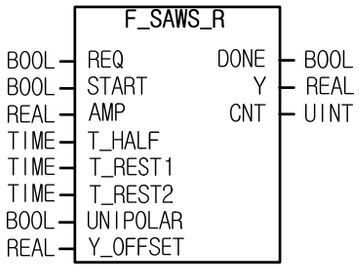
s2 : Y = Y_FIN + Y_OFFSET
(여기서 t 는 START 후의 경과 시간입니다.)

■ 프로그램 예



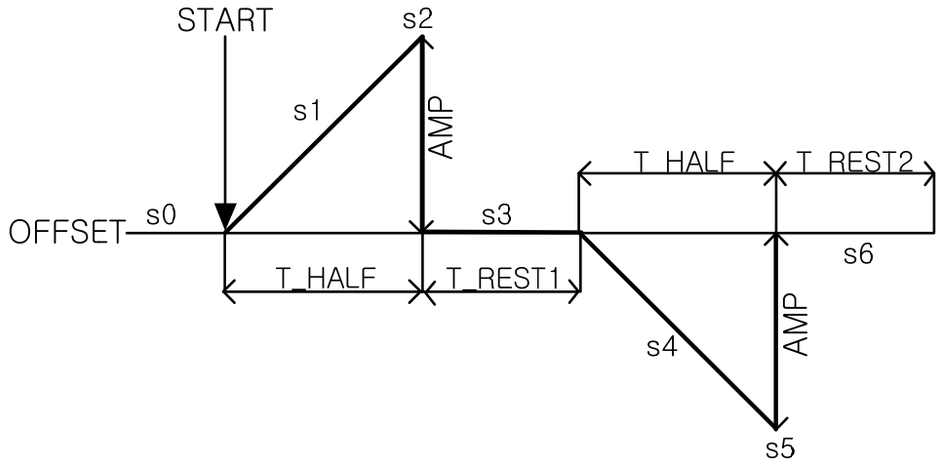
위의 그림과 같이 설정한 상태에서 START 를 On 하면 2 초 후 100 부터 1000 까지 증가하는 파형을 얻을 수 있습니다.

F_SAWS_R	적용 기종	발생플래그
톱니파 출력	XGI, XGR	-

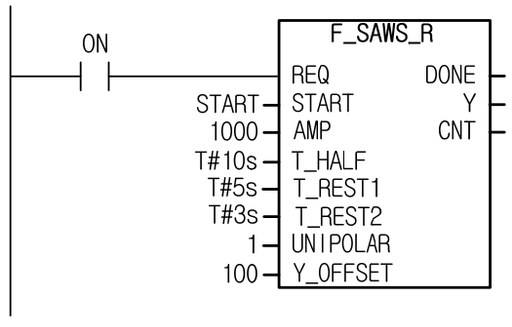
펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 START : 연산 시작 AMP : SAWS 함수 목표값 T_HALF : 함수 반주기 T_REST1 : 파형간 대기시간 1 T_REST2 : 파형간 대기시간 2 UNIPOLAR : 단극성 함수 출력 Y_OFFSET : 출력 옵셋 **출력** DONE : 에러 없이 수행 시 On Y : 출력값 CNT : 출력 반복 횟수

■ 기능

1. 톱니파를 출력합니다.
2. START On 시에 파형 출력을 시작합니다.
3. REQ 가 Off 이면 연산 마지막 상태의 값을 유지합니다.
4. REQ 가 On 상태에서 START 가 Off 이면 초기값으로 초기화 후 연산 시작(START On)을 대기합니다.
5. AMP 는 SAWS 함수의 진폭값, T_HALF 는 톱니파 상승 시간, Y_OFFSET 에는 옵셋을 설정합니다.
6. UNIPOLAR 가 On 인 경우 단극성 함수를, Off 인 경우 양극성 함수를 출력합니다.
7. 함수의 출력 카운트 CNT 값은 한 주기의 출력이 종료되고 나면 증가하고 UINT 의 범위인 65535 를 넘으면 0 부터 다시 증가합니다.
8. 스캔을 건너뛴 경우 (스캔이 1msec 이상인 경우) 극대/극소값인 S2, S5 점에서 스캔이 오차가 발생할 수 있으며 T_HALF 값이 작을수록 그래프의 기울기가 커지므로 오차가 커집니다.
9. F_SAWS : Y_FIN + Y_OFFSET 가 Y(INT)의 데이터 표현범위를 벗어날 경우 -32768 ≤ Y ≤ 32767 로 제한됩니다.
10. F_SAWS_R : Y_FIN + Y_OFFSET 가 Y(REAL)의 데이터 표현범위를 벗어날 경우 연산 도중에 결과가 '1.#inf00000 E+000' 혹은 '-1.#inf00000 E+000' 으로 표시되며 이 경우 DONE 비트는 Off 되지만 내부상태(CNT 등)는 정상적으로 처리됩니다.

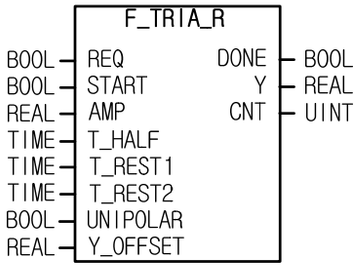


■ 프로그램 예



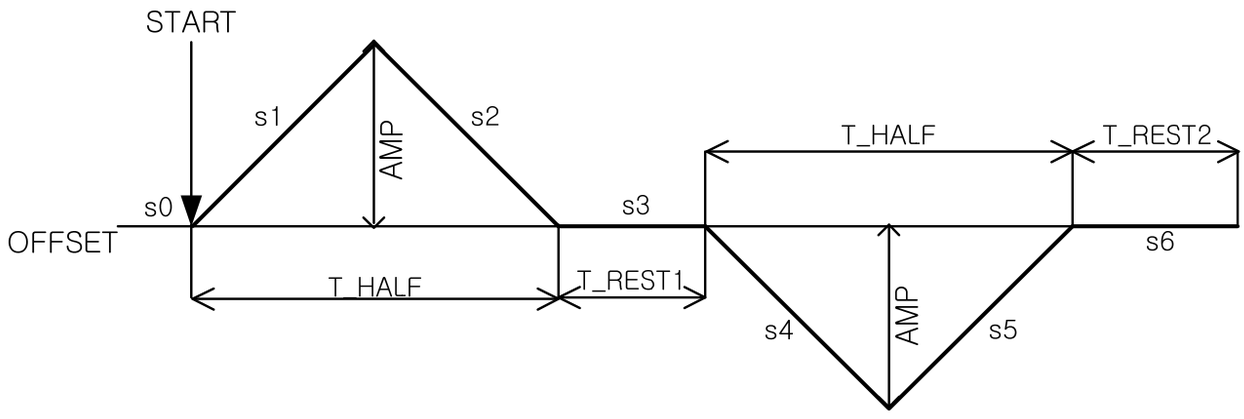
위의 그림과 같이 설정한 상태에서 START 를 On 하면 파형이 출력됩니다.

F_TRIA_R	적용 기종	발생플래그
삼각파 출력	XGI, XGR	-

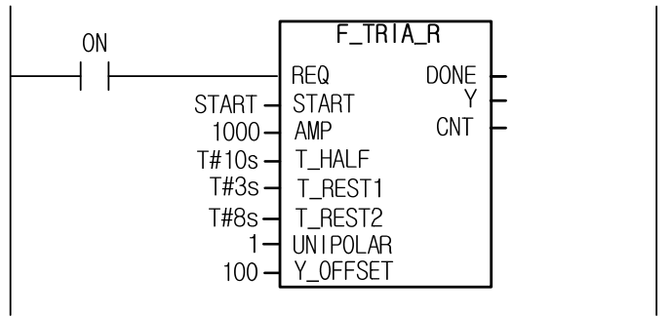
펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 START : 연산 시작 AMP : TRIA 함수 목표값 T_HALF : 함수 반주기 T_REST1 : 파형간 대기시간 1 T_REST2 : 파형간 대기시간 2 UNIPOLAR : 단극성 함수 출력 Y_OFFSET : 출력 옵셋 **출력** DONE : 에러 없이 수행 시 On Y : 출력값 CNT : 출력 반복 횟수

■ 기능

1. 삼각파를 출력합니다.
2. START On 시에 파형 출력을 시작합니다.
3. REQ 가 Off 이면 연산 마지막 상태의 값을 유지합니다.
4. REQ 가 On 상태에서 START 가 Off 이면 초기값으로 초기화 후 연산 시작(START On)을 대기합니다.
5. AMP 는 TRIA 함수의 진폭값, T_HALF 는 삼각파 상승 시간, Y_OFFSET 에는 옵셋을 설정합니다.
6. UNIPOLAR 가 On 인 경우 단극성 함수를, Off 인 경우 양극성 함수를 출력합니다.
7. 함수의 출력 카운트 CNT 값은 한 주기의 출력이 종료되고 나면 증가하고 UINT 의 범위인 65535 를 넘으면 0 부터 다시 증가합니다.
8. 스캔을 건너뛴 경우 (스캔이 1msec 이상인 경우) 극대/극소값인 S2, S5 점에서 스캔이 오차가 발생할 수 있으며 T_HALF 값이 작을수록 그래프의 기울기가 커지므로 오차가 커집니다.
9. F_TRIA : Y_FIN + Y_OFFSET 가 Y(INT)의 데이터 표현범위를 벗어날 경우 -32768 ≤ Y ≤ 32767 로 제한됩니다.
10. F_TRIA_R : Y_FIN + Y_OFFSET 가 Y(REAL)의 데이터 표현범위를 벗어날 경우 연산 도중에 결과가 '1.#inf00000 E+000' 혹은 '-1.#inf00000 E+000' 으로 표시되며 이 경우 DONE 비트는 Off 되지만 내부상태(CNT 등)는 정상적으로 처리됩니다.

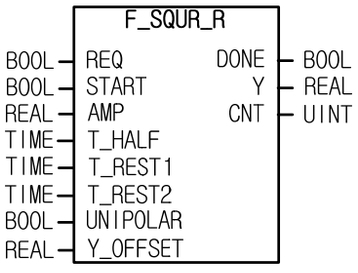


■ 프로그램 예



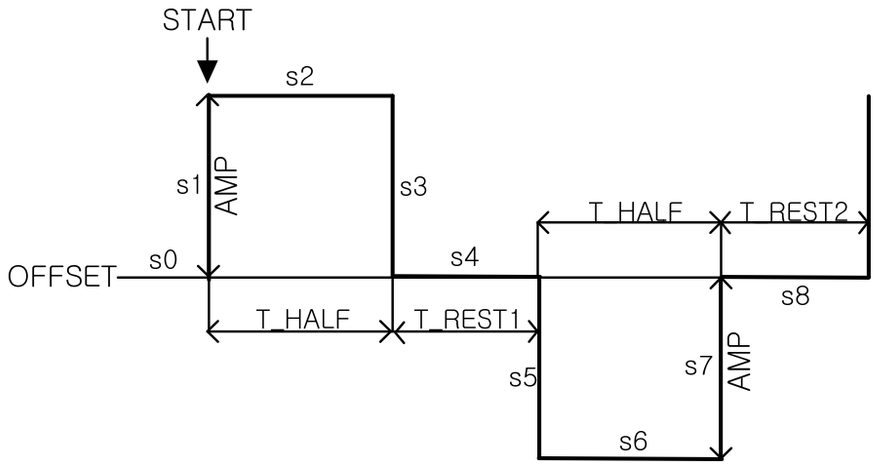
아래의 그림과 같이 설정한 상태에서 START 를 On 하면 파형이 출력됩니다.

F_SQUR_R	적용 기종	발생플래그
사각파 출력	XGI, XGR	-

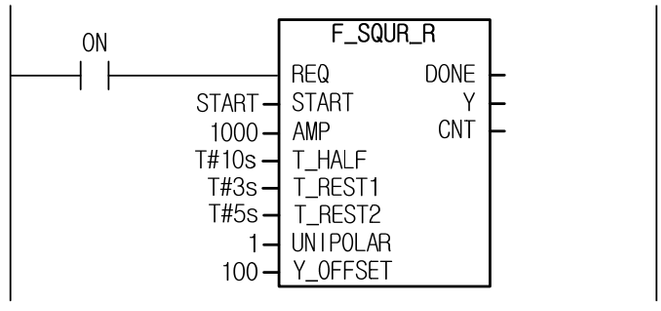
펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 START : 연산 시작 AMP : SQUR 함수 목표값 T_HALF : 함수 반주기 T_REST1 : 파형 대기시간 1 T_REST1 : 파형 대기시간 2 UNIPOLAR : 단극성 함수 출력 Y_OFFSET : 출력 옵셋 **출력** DONE : 에러 없이 수행 시 On Y : 출력값 CNT : 출력 반복 횟수

■ 기능

1. 사각파를 출력합니다.
2. START On 시에 파형 출력을 시작 합니다.
3. REQ 가 Off 이면 연산 마지막 상태의 값을 유지합니다.
4. REQ 가 On 상태에서 START 가 Off 이면 초기값으로 초기화 후 연산 시작(START On)을 대기합니다.
5. AMP 는 SQUR 함수의 진폭값, T_HALF 는 사각파 반주기, Y_OFFSET 에는 옵셋을 설정합니다.
6. UNIPOLAR 가 On 인 경우 단극성 함수를, Off 인 경우 양극성 함수를 출력합니다.
7. 함수의 출력 카운트 CNT 값은 한 주기의 출력이 종료되고나면 증가하고 UINT 의 범위인 65535 를 넘으면 0 부터 다시 증가합니다.
8. F_SQUR : Y_FIN + Y_OFFSET 가 Y(INT)의 데이터 표현범위를 벗어날 경우 -32768 ≤ Y ≤ 32767 로 제한됩니다.
9. F_SQUR_R : Y_FIN + Y_OFFSET 가 Y(REAL)의 데이터 표현범위를 벗어날 경우 연산 도중에 결과가 '1.#inf00000 E+000' 혹은 '-1.#inf00000 E+000' 으로 표시되며 이 경우 DONE 비트는 Off 되지만 내부상태(CNT 등)는 정상적으로 처리됩니다.

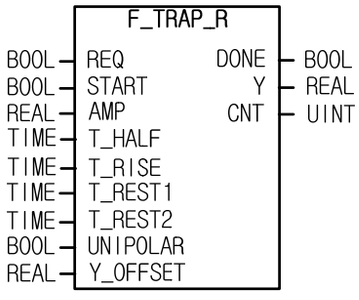


■ 프로그램 예



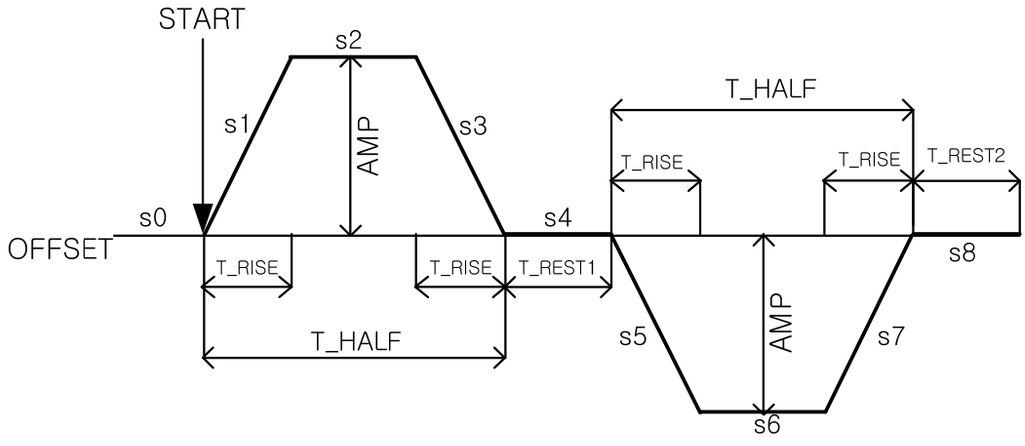
아래의 그림과 같이 설정한 상태에서 START 를 On 하면 파형이 출력됩니다.

F_TRAP_R	적용 기종	발생플래그
사다리꼴파 출력	XGI, XGR	-

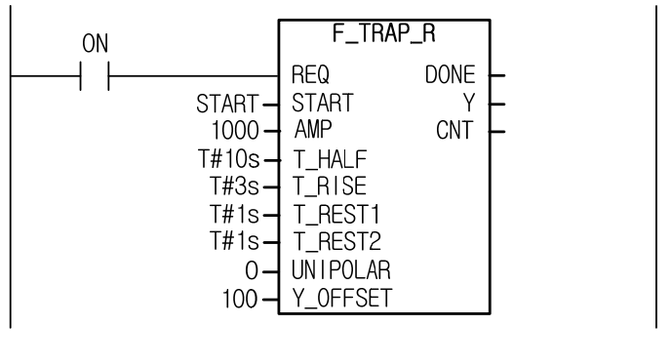
펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 START : 연산 시작 AMP : TRAP 함수 목표값 T_HALF : 함수 반주기 T_RISE : 사다리꼴 출력 시간 T_REST1 : 파형 대기시간 1 T_REST1 : 파형 대기시간 2 UNIPOLAR : 단극성 함수 출력 Y_OFFSET : 출력 옵셋 **출력** DONE : 에러 없이 수행 시 On Y : 출력값 CNT : 출력 반복 횟수

■ 기능

1. 사다리꼴파를 출력합니다.
2. START On 시에 파형을 출력합니다.
3. REQ 가 Off 이면 연산 마지막 상태의 값을 유지합니다.
4. REQ 가 On 상태에서 START 가 Off 이면 초기값으로 초기화 후 연산 시작(START On)을 대기합니다.
5. AMP 는 TRAP 함수 진폭값, T_RISE 는 사다리꼴 출력 시간, T_HALF 는 파형의 반주기, Y_OFFSET 에는 옵셋을 설정합니다.
6. UNIPOLAR 가 On 인 경우 단극성 함수를, Off 인 경우 양극성 함수를 출력합니다.
7. 함수의 출력 카운트 CNT 값은 한 주기의 출력이 종료되고나면 증가하고 UINT 의 범위인 65535 를 넘으면 0 부터 다시 증가합니다.
8. 만일 T_RISE 가 T_HALF 의 절반 이상이 되면 삼각파가 출력되며 AMP 크기의 출력이 보장되지 않습니다.
9. F_TRAP : Y_FIN + Y_OFFSET 가 Y(INT)의 데이터 표현범위를 벗어날 경우 -32768 ≤ Y ≤ 32767 로 제한됩니다.
10. F_TRAP_R : Y_FIN + Y_OFFSET 가 Y(REAL)의 데이터 표현범위를 벗어날 경우 연산 도중에 결과가 '1.#inf00000 E+000' 혹은 '-1.#inf00000 E+000'으로 표시되며 이 경우 DONE 비트는 Off 되지만 내부상태(CNT 등)는 정상적으로 처리됩니다.

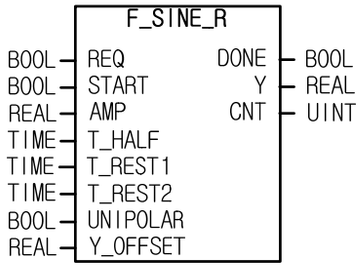


■ 프로그램 예



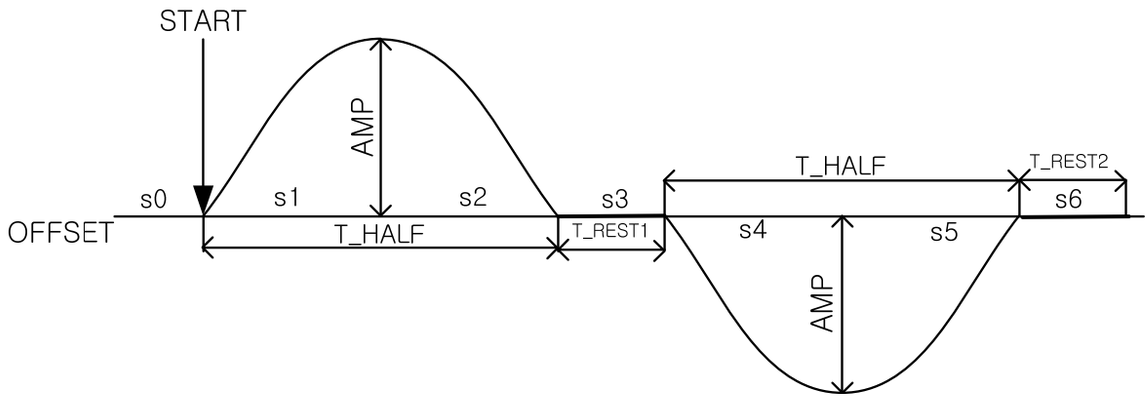
아래의 그림과 같이 설정한 상태에서 START 를 On 하면 파형이 출력됩니다.

F_SINE_R	적용 기종	발생플래그
사인파 출력	XGI, XGR	_LER

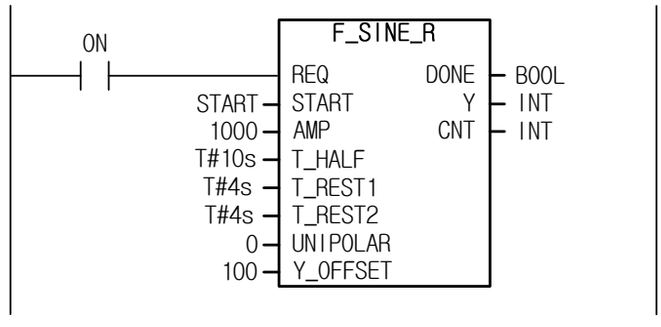
펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 START : 연산 시작 AMP : SINE 함수 목표값 T_HALF : 함수 반주기 T_REST1 : 파형대기 시간 1 T_REST2 : 파형대기 시간 2 UNIPOLAR : 단극성 함수 출력 Y_OFFSET : 출력 옵셋 **출력** DONE : 에러 없이 수행 시 On Y : 출력값 CNT : 출력 반복 횟수

■ 기능

1. 사인파를 출력합니다.
2. START On 시에 파형 출력을 시작합니다.
3. REQ 가 Off 이면 연산 마지막 상태의 값을 유지합니다.
4. REQ 가 On 상태에서 START 가 Off 이면 초기값으로 초기화 후 연산 시작(START On)을 대기합니다.
5. AMP 는 SINE 함수의 진폭값, T_HALF 는 사인파의 반주기, Y_OFFSET 에는 옵셋을 설정합니다.
6. UNIPOLAR 가 On 인 경우 단극성 함수를, Off 인 경우 양극성 함수를 출력합니다.
7. 함수의 출력 카운트 CNT 값은 한 주기의 출력이 종료되고나면 증가하고 UINT 의 범위인 65535 를 넘으면 0 부터 다시 증가합니다.
8. 스캔을 건너뛴 경우 (스캔이 1msec 이상인 경우) 극대/극소값인 S2, S5 점에서 스캔이 오차가 발생할 수 있으며 T_HALF 값이 작을수록 그래프의 기울기가 커지므로 오차가 커집니다.
9. F_SINE : Y_FIN + Y_OFFSET 가 Y(INT)의 데이터 표현범위를 벗어날 경우 -32768 ≤ Y ≤ 32767 로 제한됩니다.
10. F_SINE_R : Y_FIN + Y_OFFSET 가 Y(REAL)의 데이터 표현범위를 벗어날 경우 연산 도중에 결과가 '1.#inf00000 E+000' 혹은 '-1.#inf00000 E+000' 으로 표시되며 연산에러 플래그(_LER)가 발생합니다. 이 경우 DONE 비트는 Off 되지만 내부상태(CNT 등)는 정상적으로 처리됩니다.

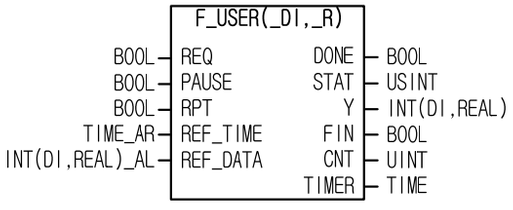


■ 프로그램 예



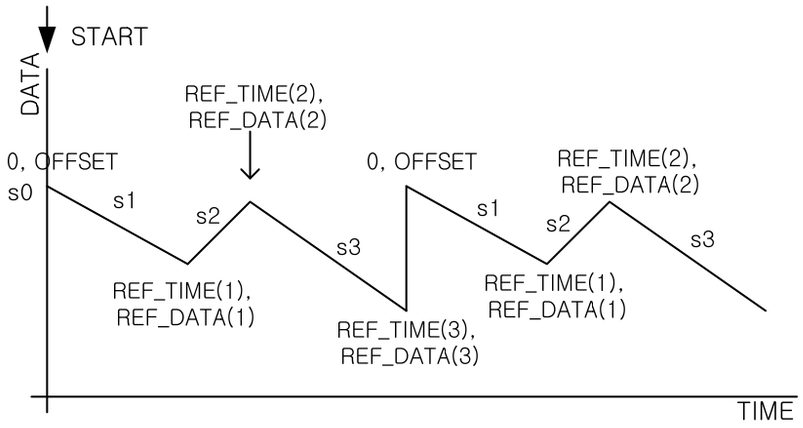
아래의 그림과 같이 설정한 상태에서 START 를 On 하면 파형이 출력됩니다.

F_USER(_DI, _R)	적용 기종	발생플래그
사용자 정의 파형 출력	XGI, XGR	-

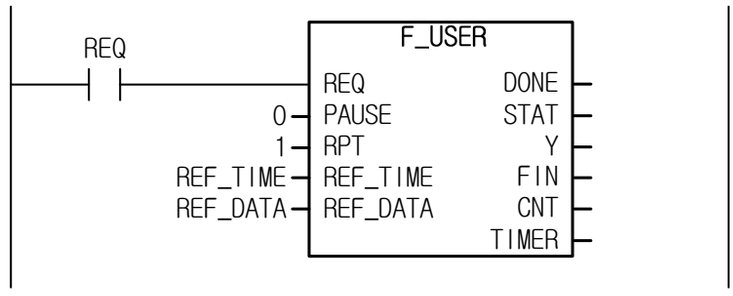
펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 PAUSE : 일시 정지 RPT : 반복 설정 REF_TIME : 시간 배열 REF_DATA : 데이터 배열 **출력** DONE : 에러 없이 수행 시 On STAT : 상태 알람 Y : 출력값 FIN : 출력 완료(반복하지 않을 경우) CNT : 반복 횟수 TIMER : FB 내의 타이머 값

■ 기능

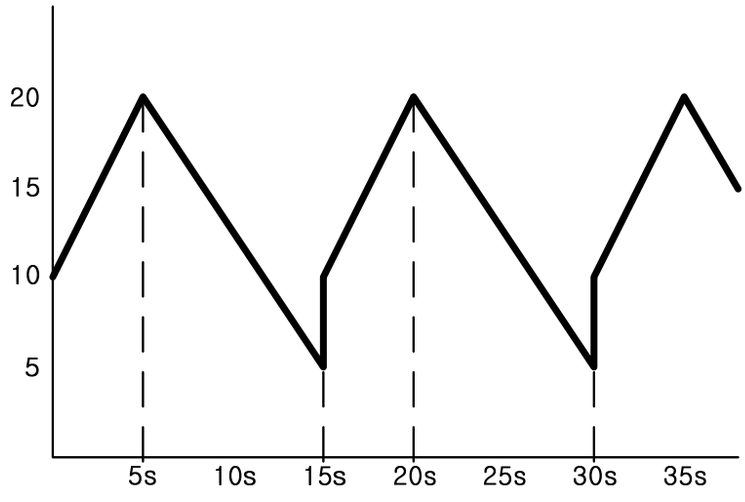
1. 사용자가 정의한 파형을 출력합니다.
2. REQ 가 Off 이면 연산 마지막 상태의 값을 유지합니다.
3. 시작상태(0 초)의 데이터가 정의 되지 않은 경우 REF_DATA 의 첫 값으로 간주합니다. 즉 첫 데이터를 (2 초,3000)으로 정의했다면 파형 시작 후 2 초 동안 3000 으로 출력합니다.
4. PAUSE 비트 On 시 출력이 일시 정지됩니다. 단, REQ On 시 첫 초기화 출력은 PAUSE 에 의해 제한되지 않습니다.
5. RPT 비트가 On 이면 파형이 반복 출력됩니다.
6. REF_TIME 과 REF_DATA 를 이용하여 사용자가 파형을 정의합니다.
7. 단발성의 경우(RPT = Off)출력이 완료되면 FIN 이 On 되고 TIMER 에 진행 시간이 표시됩니다.
8. 반복성의 경우(RPT = On)출력이 완료되면 처음부터 반복해서 출력되고 CNT 에는 함수 출력 카운트 값이, TIMER 에는 이번 주기의 진행 시간이 표시됩니다.
9. 반복성 함수의 출력 카운트 CNT 값은 한 주기의 출력이 종료되고나면 증가하고 UINT 의 범위인 65535 를 넘으면 0 부터 다시 증가합니다.
10. 한 파형이 끝나는 순간 RPT 가 체크되며 RPT 가 On 이라면 반복성 함수로, Off 라면 단발성 함수로 인식됩니다. 반복성으로 파형을 반복하던 경우에도 파형이 끝나는 순간 RPT 가 Off 이면 단발성 함수로 인식합니다.
11. 단발성 함수에서 파형의 출력이 종료되면 FIN 이 On 되고 이후로 RPT 가 변해도 파형 출력이 재계되지는 않으며 REQ 를 Off 해야 상태가 초기화 됩니다.
12. REF_TIME 의 원소가 오름차순 정렬이 되지 않았을 경우 STAT 은 8 을 출력합니다.
13. REF_TIME 와 REF_DATA 의 개수가 다를 경우 STAT 는 8 을 출력합니다.



■ 프로그램 예



REF_TIME 는 ARRAY [0..2] of INT 로 설정하고 배열의 원소는 [T#0s, T#5s, T#15s]로 설정합니다.
REF_DATA 는 ARRAY [0..2] of INT 로 설정하고 배열의 원소는 [10, 20, 5]으로 설정합니다.
위와 같이 실행하면 아래의 블록에서 REQ 를 인가 했을 때 다음과 같은 파형이 출력됩니다.



제14장 ST(Structured Text)

14.1. 개요

14.1.1. ST 언어

오픈 컨트롤러에서 로직(논리)의 기술 방식에 대해서 규정한 국제 규격 IEC61131-3 에서 정의된 언어입니다.
ST 언어에서는 연산자, 제어구문, 함수를 지원하며, 아래와 같은 기술이 가능합니다.

- 조건문에 의한 선택분기, 반복문에 의한 반복 등의 제어구문
- 연산자(*, /, +, -, <, >, = 등)를 사용한 식
- 사용자가 정의한 펑션 블록(FB)의 호출
- 함수의 호출(IEC 함수)
- 한자/한글 등의 전각 문자를 포함한 코멘트 기술

14.1.2. 특징

1) 텍스트 형식의 자유로운 기술

ST 언어는 영/숫자의 텍스트 형식으로 기술됩니다. 코멘트, 문자열 내에서는 한자/한글 등의 전각 문자도 사용 할 수 있습니다.

```
//A valve is closed when the LIMIT switch OR a tank turns on.
//A valve is opend when turned off.
IF Limit_switch = TRUE THEN
    Valve := FALSE;
ELSE
    Valve := TRUE;
END_IF;
```

2) C 언어 등의 고급 언어와 동등한 프로그래밍 가능

ST 언어는 C 언어 등의 고급 언어와 같이 조건문에 의한 선택 분기나, 반복문에 의한 반복 등, 구문에 의한 제어를 기술할 수 있습니다. 따라서 프로그램을 간결하게 나타낼 수 있습니다.

```
(* Lines A, B, AND C are controlled. *)
CASE Line OF
    1: Start_switch :=TRUE; (* Conveyer operation start *)
    2: Start_switch := FALSE; (* conveyer STOP *)
    3: Start_switch := TRUE; (* warning OR a conveyer STOP *)
    ELSE Warnig_lamp := TRUE;
  END_CASE;

  IF Start_switch = TRUE THEN (* It processes 100 times *)
  FOR Num_of_process := 0 TO 100 BY 1 DO
   Parts_A := Parts_A +1 ;
   END_FOR;
   END_IF;
```

3) 연산 처리를 용이하게 기술 가능

ST 언어는 IL 이나 LD 에서는 기술하기 어려운 연산 처리를 간결하고 보기 쉽게 기술할 수 있기 때문에, 프로그램의 가시성이 양호하며, 복잡한 산술 연산 및 비교 연산 등을 실행하는 분야에 적용되고 있습니다.

```
1
2  //FUNCTION 예제
3   CMD_TMR(IN:=%IX5.0.0, PT:=T#300ms) ;
4   bb := CMD_TMR.Q ;
5
6  // IF문 예제
7  A := 1.0;
8  B := 1.000e+3;
9  C := 2.0;
10 D := B*B - 4*A*C ;
11 IF D < 0.0 THEN NROOTS := 0 ;
12 ELSIF D = 0.0 THEN
13     NROOTS := 1 ;
14     X1 := - B/(2.0*A) ;
15 ELSE
16     NROOTS := 2 ;
17     X1 := (- B + SQRT(D))/(2.0*A) ;
18     X2 := (- B - SQRT(D))/(2.0*A) ;
19 END_IF ;
20
```

14.2. ST 언어의 구성

14.2.1. 표현식

1) 표현식은 연산자들과 피연산자들로 구성되어 있습니다. 피연산자는 정의된 문자(숫자 문자, 문자열, 시간문자), 정의된 변수(일반변수, 직접변수), 정의된 함수(펑션, 펑션 블록) 또는 다른 표현식일 수 있습니다. ST 언어의 연산자들은 <표 1>에 요약합니다.

2) 표현식의 계산은 <표 1>에 있는 연산자 우선 순위로 정의된 순서대로 연산자들을 피연산자에 적용함으로써 이루어집니다. 표현식에서 가장 높은 우선 순위를 가진 연산자가 첫 번째로 수행하고, 다음 우선순위를 가진 연산자가 차례로 수행합니다. 이러한 순서가 계산이 끝날 때까지 계속됩니다.

예) A+B*C: 먼저 B 와 C 를 곱하고, 그 연산 결과를 A 에 더합니다.

번호	연산	기호	우선순위
1	괄호로 묶음	(표현식)	가장 높음
2	함수 계산	함수 식별자 (파라미터 리스트) 예) ADD(X, Y)	
3	부정 보수	- NOT	
4	지수	**	
5	곱셈 나눗셈 나머지	* / MOD	
6	더하기 빼기	+ -	
7	비교	<, >, <=, >=	
8	동등 (일치) 부등 (불일치)	= <>	
9	부울 논리곱 부울 논리곱	& AND	
10	부울 배타적 논리합	XOR	가장 낮음
11	부울 논리합	OR	

<표 1> ST 언어의 연산자들

3) 동일한 우선순위를 가진 연산자들은 표현식의 왼쪽에서 오른쪽 순으로 수행합니다.
예) A+B-C: 먼저 A 와 B 를 더하고, 그 연산 결과에 C 를 뺍니다.
4) 연산자가 2 개의 피연산자를 가질 때 왼쪽 끝의 피연산자가 먼저 수행합니다.
예) SIN(A)*COS(B): SIN(A)를 먼저 실행 후 COS(B)를 연산합니다.
5) 연산자를 수행 할 경우, 다음 조건은 에러로 처리합니다.
(1) 0 의 값으로 나눌 경우
예) A/(B*C): B*C 연산 결과가 0 일 경우 CPU 에서 연산 에러 발생
(2) 피연산자가 연산을 위한 정확한 데이터 타입이 아닌 경우
예) ADD(1,2,3): 숫자의 데이터 타입을 결정할 수 없어서 컴파일할 때 에러 발생
(3) 산술 연산 결과가 데이터 타입의 값의 범위를 벗어난 경우.
예) B*C: B, C 가 UINT 타입일 경우, 연산 결과가 65,535 이상이면 CPU 에서 연산 에러 발생

14.2.1.1. + 연산자

1) 덧셈 연산자는 연산자 양쪽의 두 값을 더하는데 사용합니다.
2) 문법

result := expression1 + expression2

항목	설명
Result	변수 또는 직접변수
expression1	ANY_NUM 타입의 표현식
expression2	ANY_NUM 타입의 표현식

사용 예	설명
Val1 := 20; Val2 := 4; Result := Val1 + Val2;	오른쪽의 두 개의 변수 값을 찾아 더하고 그 결과를 변수 Result에 대입하라고 지시합니다. 결과 값은 24가 됩니다. 피연산자(Val1, Val2)는 상수와 변수가 모두 사용 가능합니다.

알아두기

ANY_NUM 타입은 ANY_REAL 타입과 ANY_INT 타입을 포함합니다. 자세한 사항은 3.2.2 의 데이터 타입 계층도를 참고하시기 바랍니다.

14.2.1.2. - 연산자

1) 뺄셈 연산자는 - 연산자 앞에 있는 값에서 뒤에 있는 값을 빼는데 사용합니다.

2) 문법

result := *expression1* - *expression2*

항목	설명
result	변수 또는 직접변수
expression1	ANY_NUM 타입의 표현식
expression2	ANY_NUM 타입의 표현식

사용 예	설명
Val1 := 20; Val2 := 4; Result := Val1 - Val2;	- 기호 앞의 변수 값에서 뒤에 있는 변수 값을 빼서 그 결과를 변수 Result에 대입하라고 지시합니다. 결과 값은 16이 됩니다. 피연산자(Val1, Val2)는 상수와 변수가 모두 사용 가능합니다.

14.2.1.3. * 연산자

1) 곱셈 연산자는 연산자 양쪽의 두 값을 곱하는데 사용합니다.

2) 문법

result := *expression1* * *expression2*

항목	설명
result	변수 또는 직접변수
expression1	ANY_NUM 타입의 표현식
expression2	ANY_NUM 타입의 표현식

사용 예	설명
In1 := 2 ; Result := 20 * In1 ;	20과 변수 In1의 값을 곱하여 Result 값에 대입합니다. 결과 값은 40이 됩니다. 피연산자는 상수와 변수가 모두 사용 가능합니다.

14.2.1.4. / 연산자

1) 나눗셈 연산자는 / 연산자의 왼쪽에 있는 값을 연산자 오른쪽에 있는 값으로 나눕니다.

2) 나눗셈은 연산자와 함께 사용된 변수의 타입이 정수형인지 실수형인지에 따라서 다르게 계산됩니다. 실수형 나눗셈은 실수형 값을 출력하고, 정수형 나눗셈은 정수형 값을 출력합니다. 정수는 소수점이 없으므로 5 를 3 으로 나누면, 일반적인 수학적 개념으로는 소수점이 있는 실수가 되므로 정수형이 아니라서 소수점 이하 부분을 버린 결과가 출력됩니다.

```
7 Result := 20 / INT_TYPE ;
8
9 Result1 := 20 / REAL_TYPE ;
```

```
7 Result = 6, INT_TYPE = 3
8
9 Result1 = 6.666666508e+000, REAL_TYPE = 3.000000000e+000
```

3) 문법

result := expression1 / expression2

항목	설명
result	변수 또는 직접변수
expression1	ANY_NUM 타입의 표현식
expression2	ANY_NUM 타입의 표현식

사용 예	설명
In1 := 2 ; Result := 20 / In1 ;	20을 변수 In1의 값으로 나눠서 Result 값에 대입합니다. 결과 값은 10이 됩니다. 피연산자는 상수와 변수가 모두 사용 가능합니다.

알아두기

0 값으로 나눈 경우에 연산 에러 플래그(_ERR)가 On 됩니다. 이 경우에도 CPU 는 런 모드를 유지합니다.

14.2.1.5. MOD 연산자

1) 나머지 연산자는 MOD 연산자의 왼쪽에 있는 값을 연산자 오른쪽에 있는 값으로 나누었을 때의 나머지를 결과로 출력합니다.

2) 문법

result := *expression1* MOD *expression2*

항목	설명
result	변수 또는 직접변수
expression1	ANY_NUM 타입의 표현식
expression2	ANY_NUM 타입의 표현식

사용 예	설명
In1 := 10 ; Result := 12 MOD In1 ;	12를 변수 In1의 값으로 나눈 나머지 2 값을 Result 값에 대입합니다. 피연산자는 상수와 변수가 모두 사용 가능합니다.

알아두기

0 값으로 나눈 경우에 연산 에러 플래그(_ERR)가 On 됩니다. 이 경우에도 CPU는 런 모드를 유지합니다.

14.2.1.6. ** 연산자

1) 지수 연산자는 **연산자 왼쪽의 값을 연산자 오른쪽의 승수만큼 곱하는 것입니다.

2) 문법

result := *expression1* ** *expression2*

항목	설명
result	변수 또는 직접변수
expression1	ANY_REAL 타입의 표현식
expression2	ANY_REAL 타입의 표현식

사용 예	설명
In1 := 3 ; Result := 10 ** In1 ;	10를 변수 In1의 값만큼 곱하여 Result 값에 대입합니다. 결과 값은 1000이 됩니다. 피연산자는 상수와 변수가 모두 사용 가능합니다.

14.2.1.7. AND 또는 & 연산자

1) 이항 연산자(AND 또는 &)는 두 피연산자 사이의 비트끼리 비교합니다. 두 피연산자에 대응하는 비트가 모두 1이면 결과 비트도 1이 됩니다.

2) 문법

result := *expression1 AND expression2* 또는 *result* := *expression1 & expression2*

항목	설명
result	변수 또는 직접변수
expression1	ANY_BIT 타입의 표현식
expression2	ANY_BIT 타입의 표현식

AND 비트 연산 수행은 아래와 같습니다.

expression1	*expression2*	*result*
0	0	0
0	1	0
1	0	0
1	1	1

사용 예	설명
Result := 2#10010011 AND 2#00111101 ;	두 피연산자에서 5번째 비트와 1번째 비트만 둘다 1이므로 Result 값은 2#00010001이 됩니다. 피연산자는 상수와 변수가 모두 사용 가능합니다.

14.2.1.8. OR 연산자

1) 이항 연산자(OR)는 두 피연산자 사이의 비트끼리 비교합니다. 두 피연산자에 대응하는 비트 중 어느 하나라도 1 이면 결과 비트도 1이 됩니다.

2) 문법

result := *expression1* OR *expression2*

항목	설명
result	변수 또는 직접변수
expression1	ANY_BIT 타입의 표현식
expression2	ANY_BIT 타입의 표현식

OR 비트 연산 수행은 아래와 같습니다.

expression1	*expression2*	*result*
0	0	0
0	1	1
1	0	1
1	1	1

사용 예	설명
Result := 2#10010011 OR 2#00111101 ;	7번째 비트를 제외한 모든 비트 위치에서 적어도 하나의 1이 존재하므로 Result 값은 2#10111111이 됩니다.

14.2.1.9. XOR 연산자

1) 이항 연산자(XOR)는 두 피연산자 사이의 비트끼리 비교합니다. 두 피연산자에 대응하는 비트 중 어느 하나만 1 이면 (둘 다 1이면 안됨) 결과 비트도 1이 됩니다.

2) 문법

result := expression1 XOR expression2

항목	설명
result	변수 또는 직접변수
expression1	ANY_BIT 타입의 표현식
expression2	ANY_BIT 타입의 표현식

XOR 비트 연산 수행은 아래와 같습니다.

expression1	*expression2*	*result*
0	0	0
0	1	1
1	0	1
1	1	0

사용 예	설명
Result := 2#10010011 XOR 2#00111101;	두 피연산자의 1번째 비트는 모두 1이므로 그 연산 결과의 1번째 비트는 0이 되는 것입니다 Result 값은 2#10101110 이 됩니다.

14.2.1.10. = 연산자

1) 비교 연산자(=)는 두 피연산자가 같은지 비교합니다.

2) 문법

result := expression1 = expression2

항목	설명
result	변수 또는 직접변수
expression1	ANY 타입의 표현식
expression2	ANY 타입의 표현식

= 비트 연산 수행은 아래와 같습니다.

expression1	*expression2*	*result*
0	0	1
0	1	0
1	0	0
1	1	1

사용 예	설명
Val1 := 20; Val2 := 20 ; Result := Val1 = Val2 ;	두 피연산자 Val1과 Val2가 같은지 비교한 결과를 변수 Result에 대입합니다. 결과 값은 1이 됩니다.

14.2.1.11. <> 연산자

1) 비교 연산자(<>)는 두 피연산자가 같지 않은지 비교합니다.

2) 문법

result := *expression1* <> *expression2*

항목	설명
result	변수 또는 직접변수
expression1	ANY 타입의 표현식
expression2	ANY 타입의 표현식

<> 비트 연산 수행은 아래와 같습니다.

expression1	*expression2*	*result*
0	0	0
0	1	1
1	0	1
1	1	0

사용 예	설명
Val1 := 20; Val2 := 20 ; Result := Val1 <> Val2 ;	두 피연산자 Val1과 Val2가 같지 않은지 비교한 결과를 변수 Result에 대입합니다. 결과 값은 0이 됩니다.

14.2.1.12. > 연산자

1) 관계 연산자(>)는 연산자 왼쪽의 피연산자가 오른쪽의 피연산자 보다 큰지 비교합니다.

2) 문법

result := expression1 > expression2

항목	설명
result	변수 또는 직접변수
expression1	ANY 타입의 표현식
expression2	ANY 타입의 표현식

> 비트 연산 수행은 아래와 같습니다.

expression1	*expression2*	*result*
0	0	0
0	1	0
1	0	1
1	1	0

사용 예	설명
Val1 := 20; Val2 := 10 ; Result := Val1 > Val2 ;	피연산자 Val10이 피연산자 Val2보다 큰지 비교한 결과를 변수 Result에 대입합니다. 결과 값은 10이 됩니다.

14.2.1.13. < 연산자

1) 관계 연산자(<)는 < 연산자 앞의 피연산자가 연산자 뒤의 피연산자 보다 작은지 비교합니다.

2) 문법

result := expression1 < expression2

항목	설명
result	변수 또는 직접변수
expression1	ANY 타입의 표현식
expression2	ANY 타입의 표현식

< 비트 연산 수행은 아래와 같습니다.

expression1	*expression2*	*result*
0	0	0
0	1	1
1	0	0
1	1	0

사용 예	설명
Val1 := 20; Val2 := 10 ; Result := Val1 < Val2 ;	피연산자 Val1이 피연산자 Val2보다 작은지 비교한 결과를 변수 Result에 대입하라고 지시합니다. 결과 값은 0이 됩니다.

14.2.1.14. >= 연산자

1) 관계 연산자(>=)는 앞의 피연산자가 뒤의 피연산자 보다 크거나 같은지 비교합니다.

2) 문법

result := expression1 >= expression2

항목	설명
result	변수 또는 직접변수
expression1	ANY 타입의 표현식
expression2	ANY 타입의 표현식

>= 비트 연산 수행은 아래와 같습니다.

expression1	*expression2*	*result*
0	0	1
0	1	0
1	0	1
1	1	1

사용 예	설명
Val1 := 20; Val2 := 20 ; Result := Val1 >= Val2 ;	피연산자 Val1이 피연산자 Val2보다 크거나 같은지 비교한 결과를 변수 Result에 대입합니다. 결과 값은1이 됩니다.

14.2.1.15. <= 연산자

1) 관계 연산자(<=)는 앞의 피연산자가 뒤의 피연산자 보다 작거나 같은지 비교합니다.

2) 문법

result := expression1 <= expression2

항목	설명
result	변수 또는 직접변수
expression1	ANY 타입의 표현식
expression2	ANY 타입의 표현식

<= 비트 연산 수행은 아래와 같습니다.

expression1	*expression2*	*result*
0	0	1
0	1	0
1	0	1
1	1	1

사용 예	설명
Val1 := 2; Val2 := 20 ; Result := Val1 <= Val2 ;	피연산자 Val10이 피연산자 Val2보다 작거나 같은지 비교한 결과를 변수 Result에 대입합니다. 결과 값은 10이 됩니다.

14.2.1.16. NOT 연산자

1) NOT 연산자는 비트 값을 반전합니다.

2) 문법

result := NOT expression

항목	설명
result	변수 또는 직접변수
expression	ANY_BIT 타입의 표현식

사용 예	설명
Val1 = 2#1100; Result:= NOT Val1 ;	Val1의 값을 반전하여 Result값에 넣습니다. 결과 값은2#0011이 됩니다.

14.2.1.17. - 연산자

1) 연산자는 부호를 변경 시킵니다.

2) 문법

result := - expression

항목	설명
result	변수 또는 직접변수
expression	ANY_NUM 타입의 표현식

사용 예	설명
Val1 = 10; Result:= - Val1 ;	Val1의 부호를 반전하여 Result값에 넣습니다. 결과 값은 -10이 됩니다.

14.2.2. 명령문

1) ST 언어의 명령문들이 <표 2>에 요약되어 있습니다.

2) 명령문들은 세미 콜론(;)에 의해 마칩니다.

번호	명령문 형태	예
1	할당문	A:=B; CV:= CV+1; C:=SIN(X);
2	평션 블록 호출 펑션 블록 출력 사용	CMD_TMR(IN:=%IX5, PT:= T#300ms); A:=CMD_TMR.Q;
3	리턴	**RETURN**;
4	IF 문	D:=B*B -4*A*C; **IF** D<1.0 **THEN** NROOTS :=0; **ELSIF** D= 0.0 **THEN** NROOTS := 1; X1:= -B/(2.0*A); **ELSE** X1:= (-B+SQRT(D))/(2.0*A); X2:= (-B-SQRT(D))/(2.0*A); **END_IF**;
5	CASE 문	TW := WORD_BCD_TO_INT(THUMBWHEEL); TW_ERROR := 0; **CASE** TW **OF** **1,5:** DISPLAY := OVEN_TEMP; **2:** DISPLAY := MOTOR_SPEED; **3:** DISPLAY := GROSS - TARE; **4, 6..10:** DISPLAY := ADD(TW , 4); **ELSE** DISPLAY := 0 ; TW_ERROR := 1; **END_CASE**; %MW100 := INT_TO_BCD_WORD(DISPLAY);

6	FOR 문	J := 101;
		FOR I := 1 **TO** 100 **BY** 2 **DO**
		IF WORDS[I] = 'KEY' THEN
		J := I;
		EXIT;
		END_IF;
		END_FOR ;

번호	명령문 형태	예
7	WHILE 문	J := 1; **WHILE** J <= 100 & WORDS[J] <> 'KEY' **DO** J := J+2; **END_WHILE**;
8	REPEAT 문	J := -1; **REPEAT** J := J+2; **UNTIL** J = 101 OR WORDS[J] = 'KEY' **END_REPEAT ;**
9	EXIT 문	**EXIT**;
10	널/공백 명령문	;
EXIT 문은 지원되는 모든 반복문(FOR, WHILE, REPEAT)에 사용됩니다.		

<표 2> ST 언어 명령문들

14.2.2.1. 할당문

1) 할당문은 왼쪽에 변수, 그 뒤를 따르는 할당문 연산자(:=), 마지막으로 연산할 표현식으로 구성됩니다.
   예) A := B + C ;
2) 할당문은 평션 이름을 할당문 연산자 왼쪽에 위치 시킴으로써 펑션의 리턴값을 대입하는데도 사용합니다.

14.2.2.2. 선택문

1) 선택문은 IF 문과 CASE 문 두 가지 종류가 있습니다.
2) 선택문은 특정 조건에 기초하여 수행하는 동안 선택문을 구성하는 명령문들 중 하나(혹은 그룹)를 선택합니다.
   (1) IF 문
      (a) 관련 부울 표현식이 1 의 값(참)으로 결과가 나오면 명령문 그룹을 실행합니다.
      (b) 조건이 거짓이면 어떠한 명령문도 실행되지 않습니다. 그러나 ELSE 가 있는 경우 ELSE 를 따르는 명령문 그룹이 실행됩니다. 만약 ELSIF 관련 조건이 참인 경우 ELSIF 를 따르는 명령문 그룹이 실행됩니다.
   (2) CASE 문
      (a) INT 형의 변수("선택자")를 계산하는 표현식과 명령문 그룹의 리스트로 구성됩니다.
      (b) 각 그룹의 라벨은 하나 이상의 정수와 정수 값의 범위로 설정할 수 있습니다.
      (c) 선택자의 계산 값을 포함하는 범위 내에 있는 명령문의 그룹이 수행되며, 선택자의 어떠한 값도 CASE 문의 각 경우에 해당되지 않으면 ELSE 를 뒤따르는 명령문 그룹이 수행됩니다. 만일 ELSE 가 없다면 어떠한 명령문도 수행되지 않습니다.

14.2.2.3. 반복문

1) 반복문은 FOR 문, WHILE 문 그리고 REPEAT 문 세가지 종류가 있습니다.
2) 반복문은 관련 명령문의 그룹이 반복적으로 수행되는 것을 말합니다.
   (1) FOR 문
      (a) 반복 횟수가 미리 결정되어 있는 경우에 사용됩니다.
      (b) FOR 문에서는 명령문 시퀀스가 END_FOR 까지 반복적으로 수행되며 값의 진행은 FOR 루프의 제어 변수에 지정됩니다.
      (c) 제어 변수, 초기값과 최종 값은 같은 정수형(SINT, INT, DINT)의 표현식으로 나타내며 반복되는 문장에 의해 바뀌지 않습니다. 종료 조건에 대한 검사는 각 반복의 시작 시 행해져 초기값이 최종값을 초과한다면 명령문 시퀀스가 더 이상 수행되지 않습니다.
   (2) WHILE 문과 REPEAT 문
      (a) WHILE 문은 관련 부울 표현식이 거짓일 때까지 END_WHILE 까지의 명령문 시퀀스를 반복적으로 수행됩니다.
      (b) REPEAT 문에서는 관련 부울 조건이 참일 때까지 UNTIL 까지의 명령문 시퀀스를 반복적으로 (최소한 한번은) 수행됩니다.
      (c) WHILE 문과 REPEAT 문은 외부적으로 결정되는 종료 조건을 가진 “대기 루프(wait loop)”와 같은 프로세스간 동기화 하는데 사용하지 않습니다.
      (d) EXIT 문은 종료 조건이 만족되기 전에 반복을 중단하는데 사용됩니다. EXIT 문이 중첩 반복 구조 내에 사용될 때 해당 EXIT 는 그 EXIT 가 위치한 가장 안쪽 루프에 적용됩니다. 따라서, 제어는 EXIT 문의 뒤에 위치하는 첫번째 루프 종료자(END_FOR, END_WHILE, END_REPEAT) 이후의 명령문으로 전달됩니다.
      (e) WHILE 문과 REPEAT 문은 루프 종료 조건의 만족하거나 EXIT 문의 수행이 보장될 수 없는 알고리즘에 사용되면 에러입니다.

14.2.2.4. IF 문

1) IF 문은 프로그램이 한 가지 이상 선택해야 하는 경우에 갈림길을 제공하기 때문에 분기문이라 불립니다.

2) 문법

IF *condition* THEN *statements* [ELSE *elsestatements*] END_IF

또는 아래와 같이 사용할 수 있습니다.

```
IF condition THEN
    statements
[ELSIF condition-n THEN
    elseifstatements] . . .
[ELSE
    elsestatements]
END_IF
```

항목	설명
condition	*condition* 이 TRUE 이면 THEN 이하의 *statements* 를 수행합니다. FLASE 인 경우는 ELSIF 또는 ELSE 로 분기됩니다.
statements	만약 *condition* 이 TRUE 이면 하나 이상의 상태문을 수행합니다..
condition-n	*condition* 을 N개 사용할 수 있습니다.
elseifstatements	만약 *condition-n* 이 TRUE이면 하나 이상의 상태문을 수행 합니다.
elsestatements	만약에 이전 *condition* 또는 *condition-n* 이 FALSE 인 경우 하나 이상의 상태문을 수행합니다.

사용 예	설명
IF Val1 <= 10 THEN Result := 10; END_IF;	조건 Val1 <= 10 이 참이면 Result에 10의 값을 할당합니다.

IF Val1 <= 10 THEN
  Result := 10;
ELSE
  Result := 20;
END_IF;
	조건 Val1 <= 10 이 참이면 Result에 10의 값을 할당합니다. 만약 FALSE 이면 Result에 20의 값을 할당합니다.

IF Val1 <= 10 THEN
  Result := 10;
ELSIF Val1 <= 20 THEN
  Result := 20;
ELSE
  Result := 30;
END_IF;
	조건 Val1 <= 10 이 참이면 Result에 10의 값을 할당합니다. 만약 FALSE 이면 ELSIF의 조건을 수행합니다. 두번째 조건 Val1 <= 20 이 참이면 Result에 20의 값을 할당합니다. 만약 FALSE 이면 ELSE 이하의 상태문을 수행합니다. 즉, Result에 30의 값을 할당합니다.

14.2.2.5. CASE 문

1) 프로그램의 제어가 CASE 다음에 오는 표현식의 값으로 분기합니다. 표현식은 모두 정수값(INT 형)이어야 합니다. 표현식의 값이 케이스 목록(case list) 범위에 포함되지 않는 경우 ELSE 이후에 있는 상태문이 실행됩니다. 만약 ELSE 가 없는 경우 케이스 목록 내의 어떠한 상태문 리스트(statement list)도 실행하지 않습니다. 다른 분기 명령이 없다면 나머지 모든 문을 실행하면서 프로그램의 흐름을 진행합니다.

2) 문법

```
CASE expression OF
case_list : statement_list
{ case_list : statement_list}
[ELSE
    statement_list]
END_CASE
```

항목	설명
expression	INT형 상태문만 가능합니다.
case_list	*case_list_element* {',' *case_list_element*} 위와 같이 조건이 여러 개 올 수 있습니다.
case_list_element	*subrange* 또는 *signed_integer* 만 가능합니다.
subrange	*signed_integer* .. *signed_integer* 형태입니다.
statement_list	하나 이상의 상태문을 수행합니다.

사용 예	설명
CASE Val1 OF 1 : Result := 10 ; 2..5 : Result := 20 ; 7, 10 : Result := 30 ; ELSE Result := 40 ; END_CASE ;	만약 Val1의 값이 정수값 10이면 Result 변수에 10의 값을 할당합니다. 만약 2와 5의 범위의 값이면 Result 변수에 20의 값을 할당합니다. 만약 Val1의 값이 정수값 7 또는 10 이면 Result 변수에 30의 값을 할당합니다. 이외의 다른 값일 경우에는 Result 변수에 40의 값을 할당합니다.

14.2.2.6. FOR 문

1) FOR 문은 순환 과정을 처리하기 위해서 구분자로 분리된 세 개의 제어문을 사용합니다. FOR 문의 구성 요소 중 먼저 초기화하는 수식이 실행됩니다. 만약 TO 수식이 참이면(현재 counter 의 값이 end 값보다 작은 경우) 루프는 한번 실행됩니다. 그 다음 BY 수식의 값만큼 counter 의 값이 갱신되고 조건문을 다시 검사합니다. FOR 문은 선 조건 검사 루프입니다. 즉, 루프를 통과하기 전에 순환 여부를 검사합니다. 따라서 루프를 전혀 수행하지 않을 수도 있습니다.

2) 문법

```
FOR counter := start TO end [BY step] DO
    statements
END_FOR
```

항목	설명
counter	정수형(SINT, INT, DINT) 변수입니다. start, end, step 은 모두 동일한 타입이어야 합니다.
start	*counter*의 초기값
end	*counter*의 마지막 값
step	*counter* 변수가 루프 수행할 때 마다 증가되는 수를 표시합니다. 만약 사용하지 않았다면 기본적으로 1이 증가됩니다.
statements	세 개의 제어문에 의하여 설정된 수만큼 수행되는 하나 이상의 상태문입니다.

사용 예	설명
SUM := 0; FOR counter := 0 TO 10 DO SUM := SUM + 1; END_FOR ;	counter 변수가 0에서 10까지 1씩 자동 증가 시킵니다. SUM 변수에 1의 값을 계속 더하는 프로그램입니다. 최종 SUM의 값은 11 입니다.
SUM := 0; FOR counter = 0 TO 10 BY 2 DO SUM := SUM + 1; END_FOR ;	counter 변수가 0에서 10까지 2씩 증가 시킵니다. SUM 변수에 1의 값을 계속 더하는 프로그램입니다. 최종 SUM의 값은 6 입니다.

알아두기

1. 스캔 타입이 오래 걸려 워치독이 걸릴 수 있습니다.
2. BY 부분은 생략 가능합니다. 생략된 경우 루프 카운터는 기본적으로 1씩 증가합니다.
3. 초기값이 마지막 값보다 큰 경우 FOR 문을 수행하지 않습니다.

14.2.2.7. WHILE 문

1) WHILE 문은 조건 수식이 거짓이나 0 이 될 때까지 반복하는 순환문을 만듭니다. WHILE 문은 선 조건 검사 루프입니다. 즉, 루프를 통과하기 전에 순환 여부를 검사합니다. 따라서 조건이 만족하지 않는 경우 WHILE 문 내의 상태문을 전혀 수행하지 않을 수 있습니다.

2) 문법

```
WHILE condition DO
    statements
END_WHILE
```

항목	설명
condition	*condition* 이 TRUE 이면 DO 이하의 *statements* 를 수행합니다. FLASE 인 경우는 순환문을 빠져 나갑니다.
statements	*condition* 이 TRUE 이면 하나 이상 상태문이 처리됩니다.

사용 예	설명
Counter := 0 **WHILE** Counter < 20 **DO** Counter := Counter + 1; **END_WHILE** ;	Counter 변수가 20보다 작다는 조건이 맞으면 상태문을 수행 합니다. Counter 변수가 20이 되면 조건이 FALSE 가 되어 순환문을 빠져 나갑니다.

알아두기

WHILE 문은 조건 수식이 거짓이나 0 이 되지 않는 경우 무한 루프에 빠질 수 있습니다. 이 경우 스캔 타임이 오래 걸려 스캔 워치독이 걸릴 수 있습니다. 따라서 조건 수식이 항상 TRUE 가 되지 않도록 주의하시기 바랍니다.

14.2.2.8. REPEAT 문

1) REPEAT 문은 조건식이 TRUE 가 될 때까지 반복하는 순환문을 만듭니다. REPEAT 문(후 조건 검사 루프)은 루프 통과 후에 루프 반복 여부를 검사해서 결정합니다. 따라서 루프는 적어도 한번은 실행합니다.

2) 문법

```
REPEAT
    statements
UNTIL condition
END_REPEAT
```

항목	설명
condition	*condition* 이 FALSE 면 반복 수행하고 TRUE 면 빠져 나갑니다.
statements	조건이 TRUE 일 때까지 반복 수행합니다.

사용 예	설명
Counter := 0; REPEAT Counter := Counter + 1; UNTIL Counter > 20 END_REPEAT ;	우선 Counter 변수가 1 증가됩니다. Counter 변수가 20 보다 크다는 조건을 만족하면 순환문을 빠져 나가고 아니면 상태문을 수행합니다. Counter 변수가 21이 되면 조건이 TRUE 가 되어 순환문을 빠져 나갑니다.

알아두기

REPEAT 문은 조건 수식이 거짓이나 0 이 되지 않는 경우 무한 루프에 빠질 수 있습니다. 이 경우 스캔 타임이 오래 걸려 스캔 워치독이 걸릴 수 있습니다. 따라서 조건 수식이 항상 TRUE 가 되지 않도록 주의하시기 바랍니다.

14.2.2.9. EXIT 문

1) EXIT 문은 반복문(WHILE, FOR, REPEAT)에서 빠져 나가는데 사용됩니다.

2) EXIT 문을 반복문 밖에서 사용하는 경우 에러가 발생합니다.

3) 문법

EXIT

사용 예	설명
SUM := 0; FOR Counter := 0 TO 10 DO SUM := SUM + 1; EXIT; END_FOR ;	Counter 변수가 0 에서 10 까지 1씩 자동 증가하는 FOR 문 입니다. 그러나 상태문에 EXIT를 사용하여 바로 빠져 나옵니다. Counter 변수는 0의 되고, SUM 변수는 1의 됩니다.

Counter := 0;
WHILE Counter < 20 DO
Counter := Counter + 1 ;
IF Counter = 10 THEN
EXIT;
END_IF;
END_WHILE ;
	Counter 변수가 20보다 작다는 조건이 맞으면 상태문을 수행 합니다. Counter 변수가 20이 되면 조건이 FALSE가 되어 순환문을 빠져 나갈 수 있습니다. 그러나 상태문에 IF문과 EXIT문을 사용하여 Counter 변수가 10일 경우 반복문을 빠져 나갑니다.

Counter := 0;
REPEAT
Counter := Counter + 1 ;
IF Counter = 10 THEN
EXIT;
END_IF;
UNTIL Counter > 20
END_REPEAT ;
	우선 Counter 변수가 1만큼 증가합니다. Counter 변수가 20보다 크다는 조건이 맞으면 순환문을 빠져나가고 아니면 상태문을 수행합니다. 그러나 상태문에 IF문과 EXIT문을 사용하여 Counter 변수가 10 일 경우 반복문을 빠져 나갑니다.

14.2.3. 비 실행문(설명문)

1) 비 실행문(설명문)은 2 가지 형태를 제공 합니다. 한라인 비 실행문과 블록 비 실행문 형태가 있습니다.
2) 한라인 비 실행문은 "//" 사용하며, 해당 라인의 끝까지 비 실행문 처리 됩니다.
3) 블록 비 실행문은 "(*" 과 "*)" 사이의 문자들을 비 실행문 처리 합니다.
4) 설명문도 동일한 형태로 작성 가능 합니다.

예 1) 블록 비 실행문 설정 시

```
(* CASE Line OF
    1: Start_switch :=TRUE;
    2: Start_switch := FALSE;
    3: Start_switch := TRUE;
    ELSE Warnig_lamp := TRUE;
   END_CASE;

   IF Start_switch = TRUE THEN
   FOR Num_of_process := 0 TO 100 BY 1 DO
    Parts_A := Parts_A +1 ;
    END_FOR;
    END_IF;  *)
```

예 2) 한라인 비 실행문 설정 시

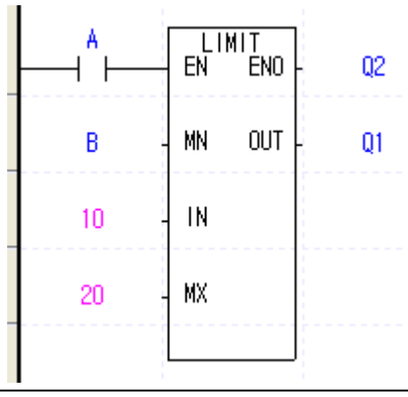
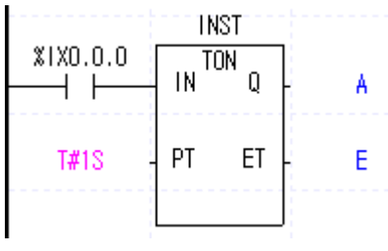
```
 CASE Line OF
//  1: Start_switch :=TRUE;
    2: Start_switch := FALSE;
//  3: Start_switch := TRUE;
    ELSE Warnig_lamp := TRUE;
   END_CASE;

   IF Start_switch = TRUE THEN
 // FOR Num_of_process := 0 TO 100 BY 1 DO
    Parts_A := Parts_A +1 ;
    END_FOR;
    END_IF;
```

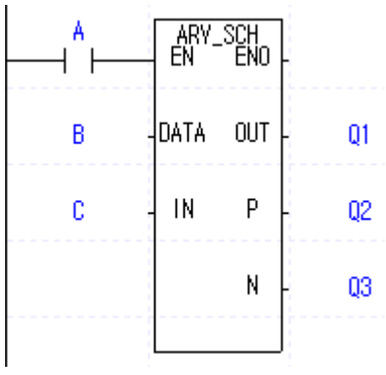
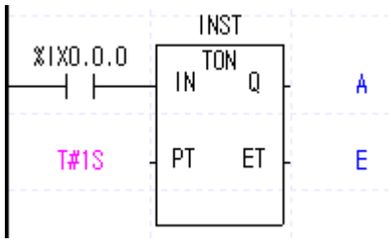
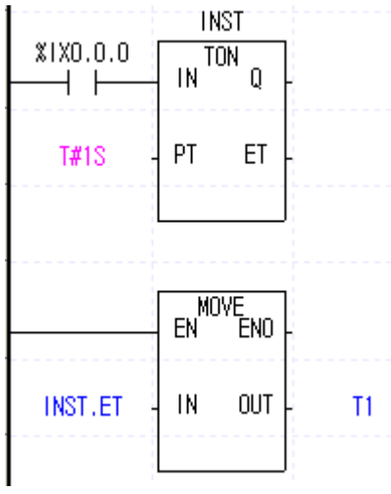
14.3. 펑션 및 펑션 블록

14.3.1. 사용 방법

1) 펑션 및 펑션 블록 입력 방법은 정형화 형태, 비정형화 형태 2 가지가 있습니다. 상황에 따라 어느 형태도 사용 가능 합니다.

(1) 정형화 형태:
펑션 및 펑션블록의 입력, 출력 파라미터 이름을 표기하는 형태입니다.

파라미터	펑션	펑션블록
공통	파라미터 순서는 임의의 순서로 사용 가능. Q1 := LIMIT(MN := B, MX := 20, IN := 10) ; Q1 := LIMIT(MX := 20, MN := B, IN := 10) ; EN, ENO은 사용 또는 생략 가능 Q1 := LIMIT(**EN := A,** MN := B, MX := 20, IN := 10, ENO => Q2) ; 	파라미터 순서는 임의의 순서로 사용 가능. INST(IN := %IX0.0.0, PT := T#1s, Q => A, ET => E) ; INST(PT := T#1s, IN := %IX0.0.0, Q => A, ET => E) ; 
입력	입력, 입출력 파라미터 할당에 := 기호 사용. C := LIMIT(**MN := B, MX := 20**, IN := 10) ;	입력, 입출력 파라미터 할당에 := 기호 사용. INST(**IN := %IX0.0.0, PT := T#1s**, Q => A, ET => B) ;
출력	출력 파라티미터 이름이 OUT 또는 Y(사용자 정의 펑션은 펑션 이름)일 경우 리턴값 할당. 나머지 출력 파라미터 할당은 => 기호 사용. **Q1 :=** ARY_SCH(DATA := B, IN := C, **P => Q2, N => Q3**) ;	모든 출력 파라미터 할당에 => 기호 사용 출력 파라미터 할당은 생략 가능. INST(IN := %IX0.0.0, PT := T#1s, **Q => A, ET => E**) ;

파라미터	펑션	펑션블록
	 단, 아래와 같이 사용하지 않는 출력 파라미터는 생략 가능합니다. (Q2, Q3 생략) Q1 := ARY_SCH(DATA := B, IN := C) ;	 INST(IN := %IX0.0.0, PT := T#1s) ; T1 := INST.ET; 

알아두기

펑션 블록은 인스턴스 이름으로 사용합니다. 즉, 펑션블록을 변수처럼 선언하고 이 변수 명(인스턴스 명)을 사용하여야 합니다.

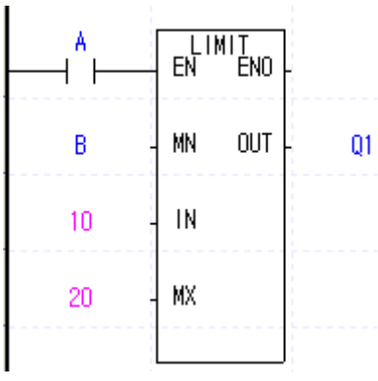
예) 타이머 사용

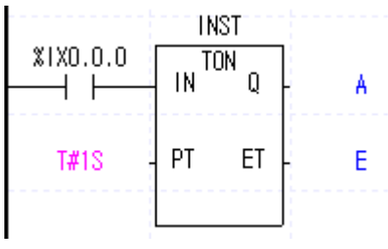
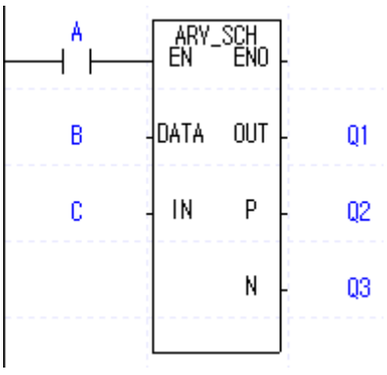
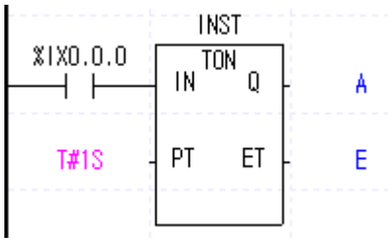
	변수 종류	변수	타입
1	VAR	INST_TON1	TON

INST_TON1(IN := TRUE, PT := T#100MS, Q => Q_OUT, ET => ET_OUT);

(2) 비 정형화된 형태

펑션 및 펑션블록의 입력, 출력 파라미터 이름을 생략하는 형태입니다.

<table>
<tr><th>파라미터</th><th>펑션</th><th>펑션블록</th></tr>
<tr><td>공통</td><td>모든 파라미터 순서는 변경 불가.
모든 파라미터는 생략 불가
Q1 := LIMIT(B, 20, 10) ;
 

EN, ENO는 사용 불가.</td><td>모든 파라미터 순서는 변경 불가.
모든 파라미터는 생략 불가.
INST(%IX0.0.0, T#1s, A, E) ;
 
</td></tr>
<tr><td>입력</td><td>입력 파리미터 순서는 변경 불가.
C := LIMIT(B, 20, IN := 10) ;</td><td>입력 파리미터 순서는 변경 불가.
INST(%IX0.0.0, T#1s, A, E) ;</td></tr>
<tr><td>출력</td><td>출력 파라티미터 이름이 OUT 또는 Y(사용자 정의 펑션은 펑션 이름)일 경우 리턴값 할당.
나머지 출력 파라미터 할당은 순서대로 입력.
Q1 := ARY_SCH(B, C, Q2, Q3) ;
 
</td><td>모든 출력 파라미터 할당은 순서대로 입력.
INST(%IX0.0.0, T#1s, A, E) ;
 
</td></tr>
</table>

알아두기

파라미터 타입이 가변인 펑션은 입력 파라미터 타입이 결정 되어야 합니다.

사용 예	설명
INT1 := ADD(1, 2, 3);	펑션 타입 결정 중에 에러가 발생합니다.

정상 동작 하기 위해서는 아래 중에 한가지 방법으로 입력하여야 합니다.

사용 예	설명
INT1 := ADD(**INT#1**, 2, 3);	상수에 타입을 설정할 수 있습니다.
INT1 := ADD(**B**, 2, 3);	변수(B)를 사용할 수 있습니다.
INT1 := **ADD_INT**(1, 2, 3);	타입이 설정된 펑션을 사용할 수 있습니다.

알아두기

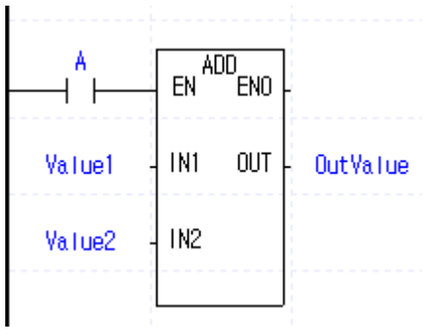
1. 입력 파라미터 EN 은 펑션을 실행하기 위한 조건입니다. 다음과 같이 EN 을 사용할 경우 A 값이 1 일 경우에만 LIMTIT 펑션이 실행됩니다.
   OUT := LIMIT(EN := A, MX := 20, MN := B, IN := 10) ;
2. ENO 파라미터는 펑션이 에러 없이 실행될 경우에 1 이 됩니다. ST 에서는 사용할 수 없고 LD 에서만 사용 가능합니다.

알아두기

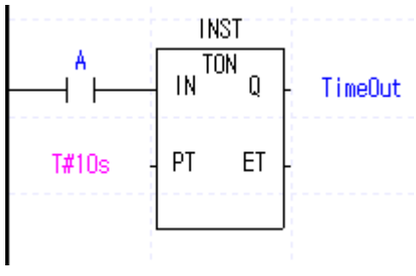
1. ST 는 확장 명령어(BREAK, CALL, END, FOR, INIT_DONE, JMP, NEXT, RET, SBRT)를 지원하지 않습니다.
2. 연산자 명과 동일한 이름의 펑션은 사용할 수 없습니다.(OR, XOR, AND, MOD, NOT)

14.3.2. 사용 예

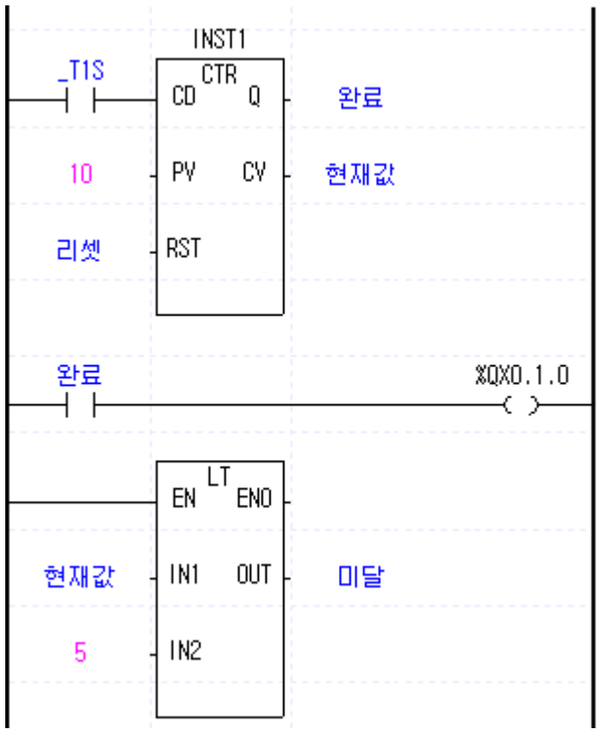
1) 펑션

LD 사용 예	ST 사용 예
	1) 정형화된 형태 EN 사용 OutValue := ADD(EN := A, IN1 := Value1, IN2 := Value2); EN 사용 안 함 OutValue := ADD(IN1 := Value1, IN2 := Value2); 2) 비정형화된 형태 OutValue := ADD(Value1, Value2); EN, ENO을 사용할 수 없습니다.

2) 펑션 블록

LD 사용 예	ST 사용 예
	1) 정형화된 형태 INST(IN := A, PT := T#10S, Q => TimeOut); 2) 비정형화된 형태 INST(A, T#10S, TimeOut, TimeValue); 출력 변수를 생략할 수 없습니다. 따라서 출력 파라미터 ET에 해당하는 변수를 연결해야 합니다. (TimeValue)

3) 응용

LD 사용 예	ST 사용 예
	INST1(CD := _T1S, PV := 10, RST := 리셋, Q => 완료, CV => 현재값); %QX0.1.0 := 완료; 미달 := LT(IN1 := 현재값, IN2 := 5);

14.4. ST 편집

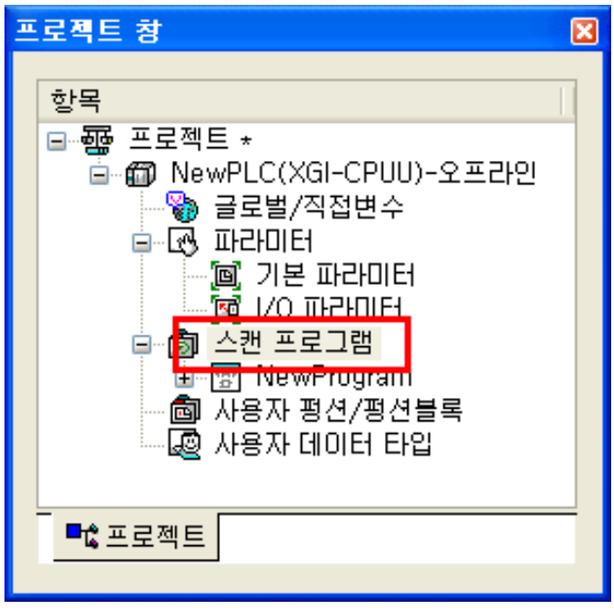
14.4.1. ST 프로그램 작성

PLC 의 스캔 프로그램, 사용자 펑션/펑션 블록, SFC 등을 ST 프로그램으로 작성할 수 있습니다.

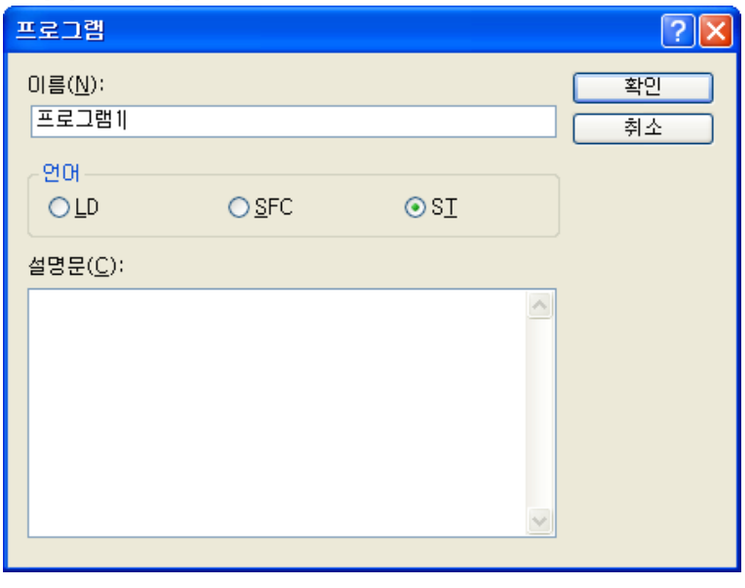
14.4.1.1. 스캔 프로그램 추가

[순서]

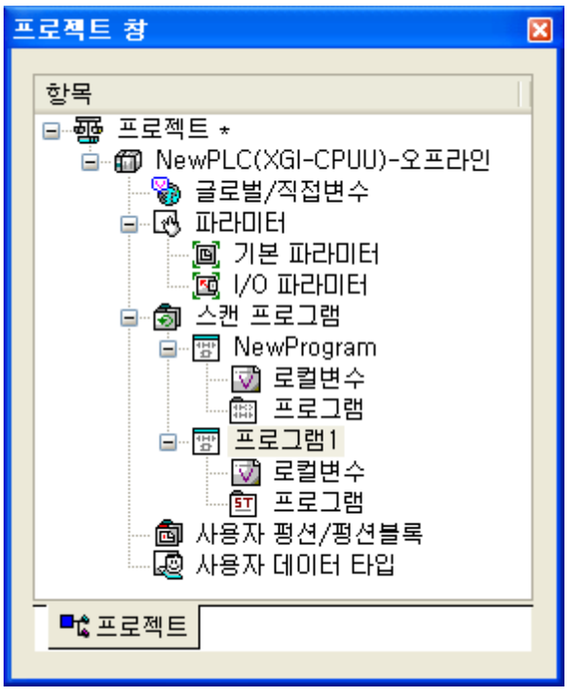
1. 프로젝트 창에서 스캔 프로그램을 선택합니다.



2. 메뉴 [프로젝트]-[항목 추가]-[프로그램]을 선택합니다.



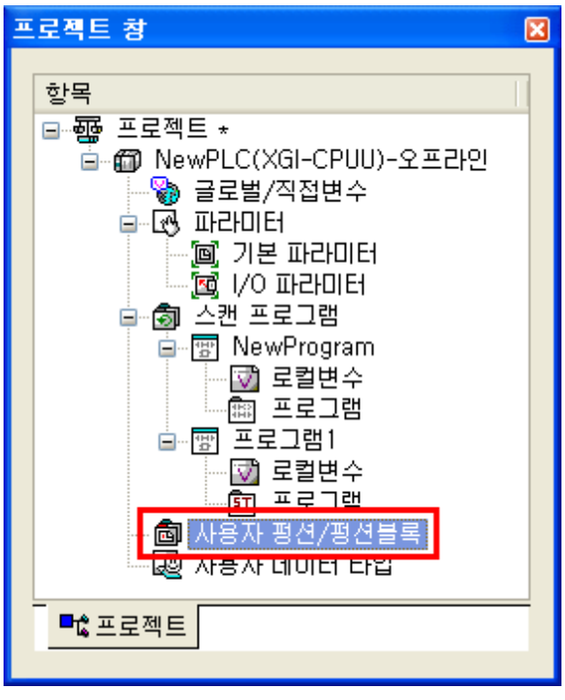
3. 프로그램 이름과 설명문을 입력하고 언어를 ST로 선택한 후 확인을 누릅니다.



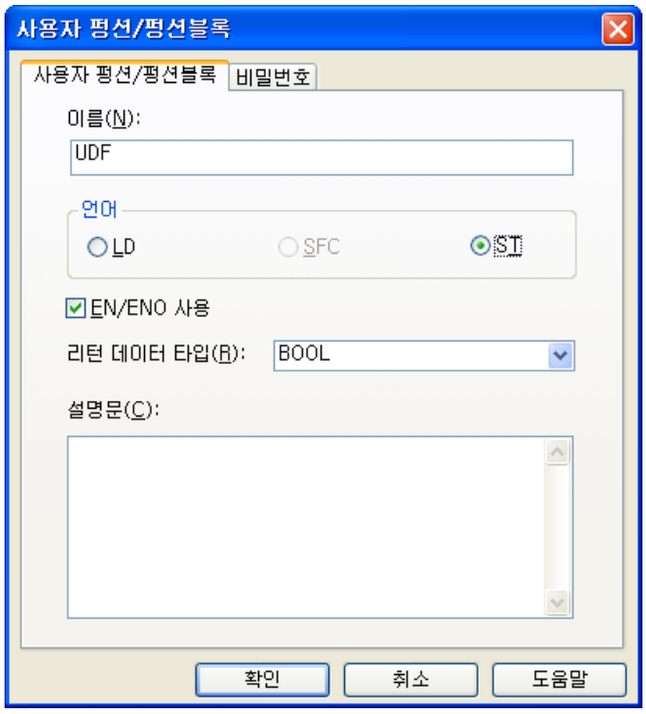
14.4.1.2. 사용자 펑션/펑션 블록 추가

[순서]

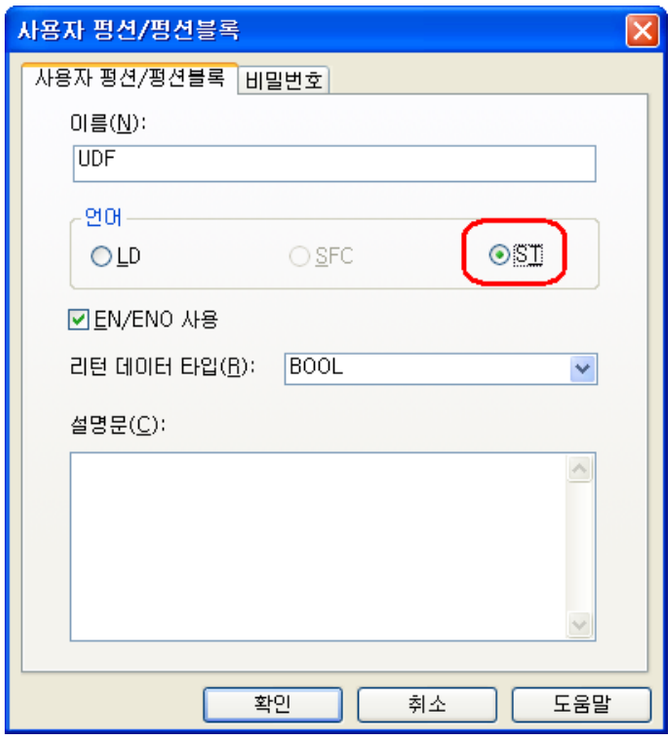
1. 프로젝트 창에서 사용자 펑션/펑션 블록을 선택합니다.



2. 메뉴 [프로젝트]-[항목 추가]-[펑션] 또는 [펑션 블록]을 선택합니다.



3. 사용자 펑션/펑션 블록 이름과 설명문을 입력하고 언어를 ST로 선택한 후 확인을 누릅니다. 사용자 펑션의 경우 리턴 데이터 타입을 설정해야 합니다.



14.4.2. 프로그램 편집

14.4.2.1. 복사/붙여넣기

선택된 문자열을 클립보드로 복사한 후 붙여넣기 시 복사된 문자열을 붙여 넣습니다.

[순서]

1. 복사하고자 하는 문자열을 선택합니다.

```
22    // SETTING ERROR FLAG!
23    IF LWORD_TMP <> LINT_VAL THEN ERROR := TRUE;
24    ELSE ERROR := FALSE;
25    END_IF;
26
27    DWORD_LOWER := LWORD_TO_DWORD(LWORD_TMP AND 16#00000000FFFFFFFF);
28    DWORD_HIGHER := LWORD_TO_DWORD(SHR(LWORD_TMP AND 16#FFFFFFFF00000000, 32));
```

2. 메뉴 [편집]-[복사]를 선택합니다.
3. 붙여넣기 할 위치로 이동합니다.

```
22    // SETTING ERROR FLAG!
23    IF LWORD_TMP <> LINT_VAL THEN ERROR := TRUE;
24    ELSE ERROR := FALSE;
25    END_IF;
26
27    DWORD_LOWER := LWORD_TO_DWORD(LWORD_TMP AND 16#00000000FFFFFFFF);
28    DWORD_HIGHER := LWORD_TO_DWORD(SHR(LWORD_TMP AND 16#FFFFFFFF00000000, 32));
29
30END_IF;
```

4. 메뉴 [편집]-[붙여넣기]를 선택합니다.

```
22    // SETTING ERROR FLAG!
23    IF LWORD_TMP <> LINT_VAL THEN ERROR := TRUE;
24    ELSE ERROR := FALSE;
25    END_IF;
26
27    DWORD_LOWER := LWORD_TO_DWORD(LWORD_TMP AND 16#00000000FFFFFFFF);
28    DWORD_HIGHER := LWORD_TO_DWORD(SHR(LWORD_TMP AND 16#FFFFFFFF00000000, 32));
29    LWORD_TO_DWORD
30END_IF;
```

알아두기

1. 클립보드: 컴퓨터에서 임시 저장 공간으로 사용하기 위해 확보된 메모리 영역
2. 붙여넣기 동작 시 선택된 영역이 있을 시는 선택된 영역에 덮어쓰기 되고, 없을 시는 삽입됩니다.
3. 텍스트만 붙여넣기 됩니다
4. 펑션블록을 복사/붙여넣기 시 인스턴트명이 동일하게 설정되므로 펑션블록 삽입을 통해 새 인스턴트 명으로 등록해야 합니다.

14.4.2.2. 편집취소/재실행

편집취소는 편집 되기 전 이전 상태로 되돌립니다. 재실행은 편집 취소된 상태에서 편집 취소 하기 이전 상태로 되돌립니다.

[순서]

1. 붙여넣기를 실행한 후 메뉴 [편집]-[편집취소]를 선택합니다.
   => 붙여넣기를 한 내용이 삭제됩니다.

```
22      // SETTING ERROR FLAG!
23      IF LWORD_TMP <> LINT_VAL THEN ERROR := TRUE;
24      ELSE ERROR := FALSE;
25      END_IF;
26
27      DWORD_LOWER := LWORD_TO_DWORD(LWORD_TMP AND 16#00000000FFFFFFFF);
28      DWORD_HIGHER := LWORD_TO_DWORD(SHR(LWORD_TMP AND 16#FFFFFFFF00000000, 32));
29
30  END_IF;
```

2. 메뉴 [편집]-[재실행]을 선택합니다.

   => 붙여넣기 동작이 다시 실행됩니다.

```
22      // SETTING ERROR FLAG!
23      IF LWORD_TMP <> LINT_VAL THEN ERROR := TRUE;
24      ELSE ERROR := FALSE;
25      END_IF;
26
27      DWORD_LOWER := LWORD_TO_DWORD(LWORD_TMP AND 16#00000000FFFFFFFF);
28      DWORD_HIGHER := LWORD_TO_DWORD(SHR(LWORD_TMP AND 16#FFFFFFFF00000000, 32));
29      LWORD_TO_DWORD
30  END_IF;
```

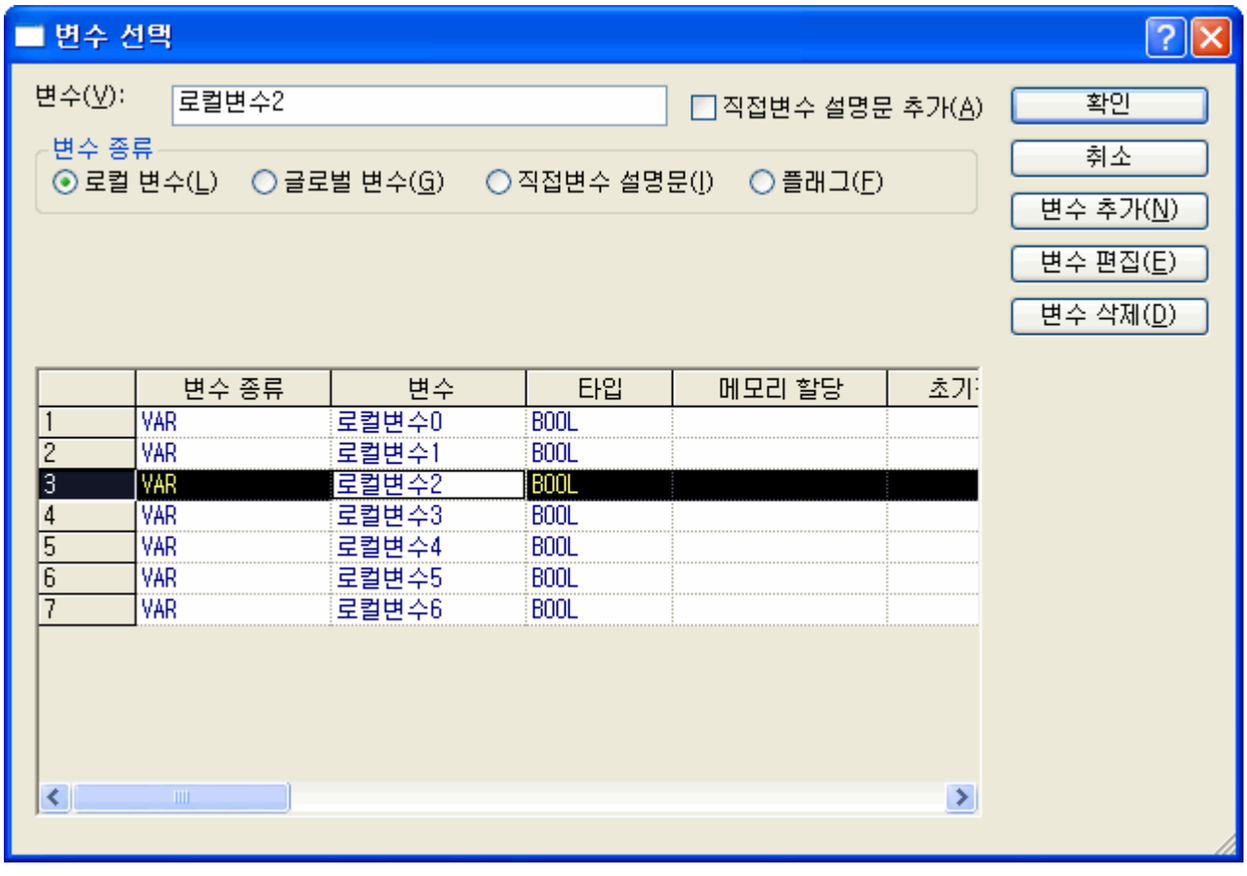
14.4.2.3. 변수 선택/추가

선택된 영역 또는 커서 위치에 변수를 입력합니다.

[순서]

1. 입력하고자 하는 위치로 커서를 이동 시킨 후 메뉴 [편집]-[변수선택/추가]를 선택합니다.
2. 변수를 커서 위치에 삽입합니다.

[대화 상자]

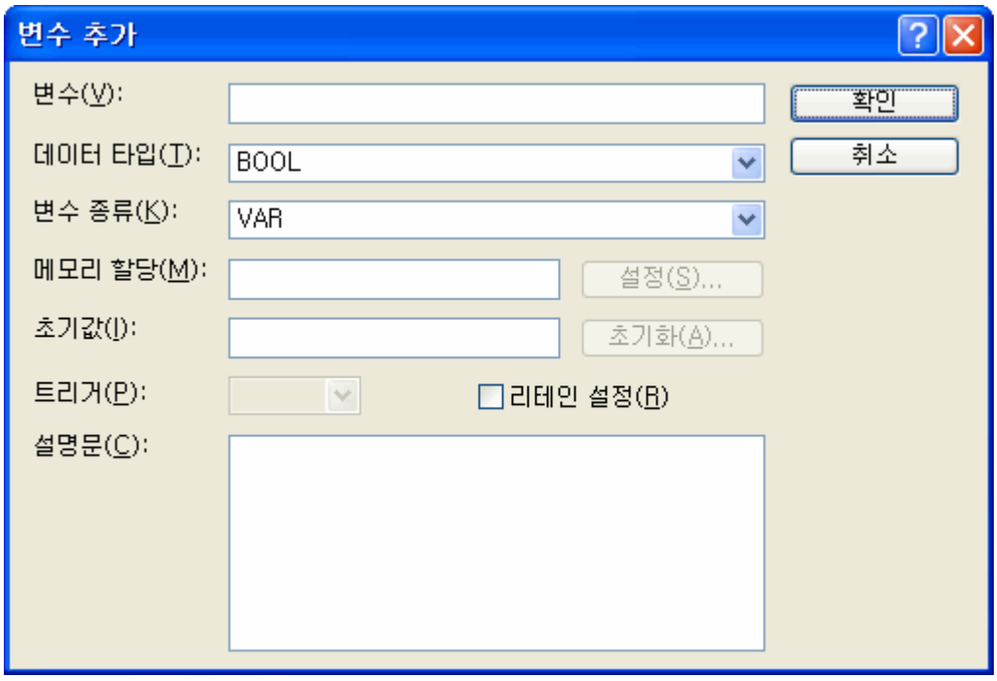


[대화 상자 설명]

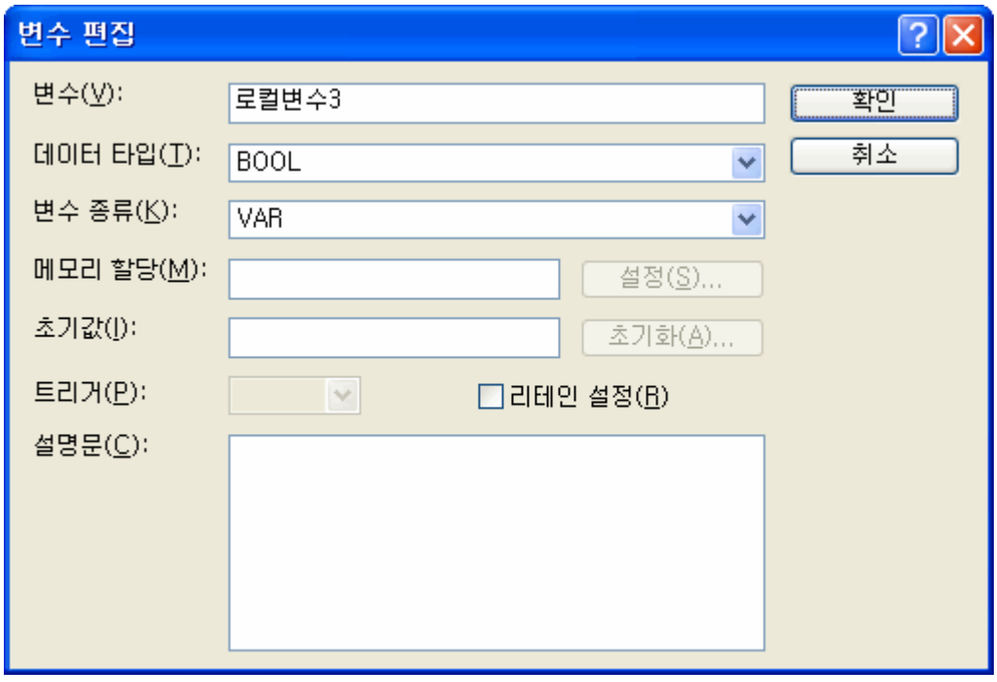
a. 변수: 상수, 직접 변수 또는 선언된 변수 명을 입력 할 수 있습니다. 입력한 문자열이 변수 형태이며 해당 문자열 이 로컬 변수 목록에 변수로 등록되어 있지 않은 경우, 변수 추가 대화 상자가 표시됩니다.

b. 로컬 변수: 선언된 로컬 변수 목록을 표시합니다.

c. 변수 추가: 로컬 변수 목록에 변수를 추가 할 수 있는 대화 상자를 호출 합니다.



d. 변수 편집: 선택된 변수를 편집 할 수 있는 대화 상자를 호출 합니다.

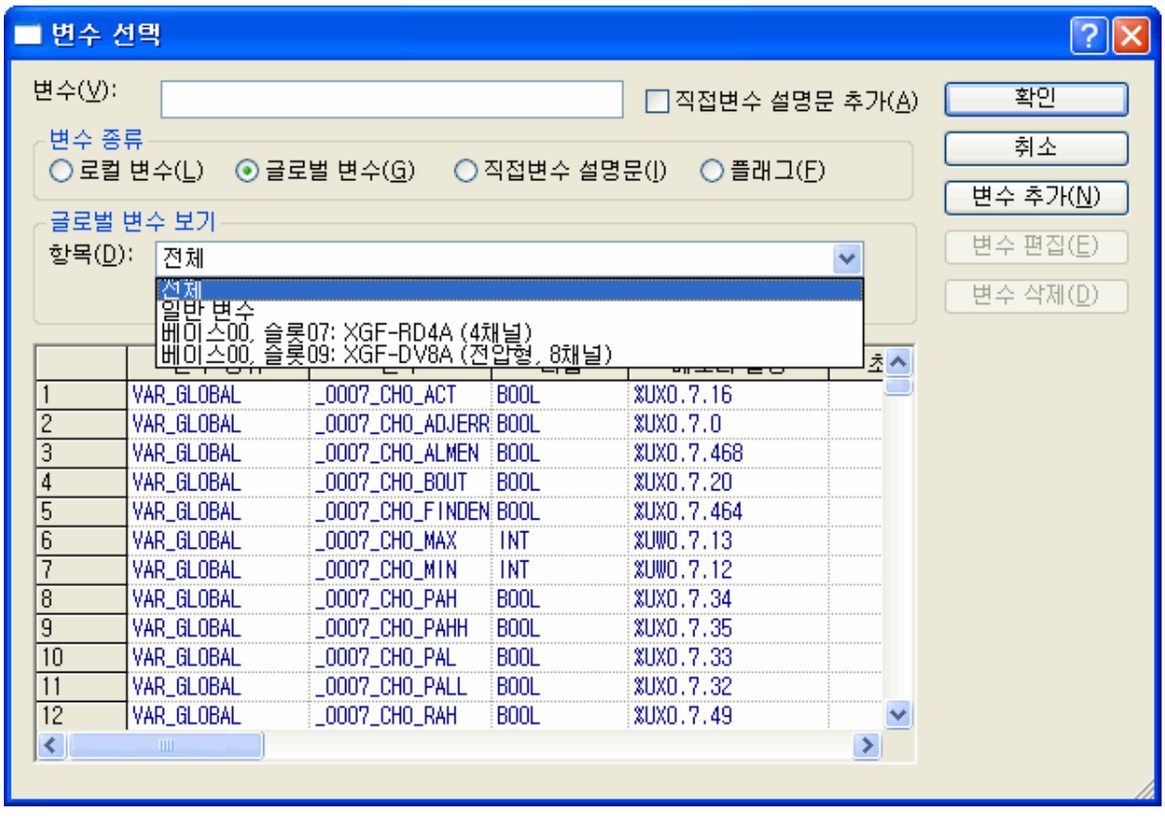


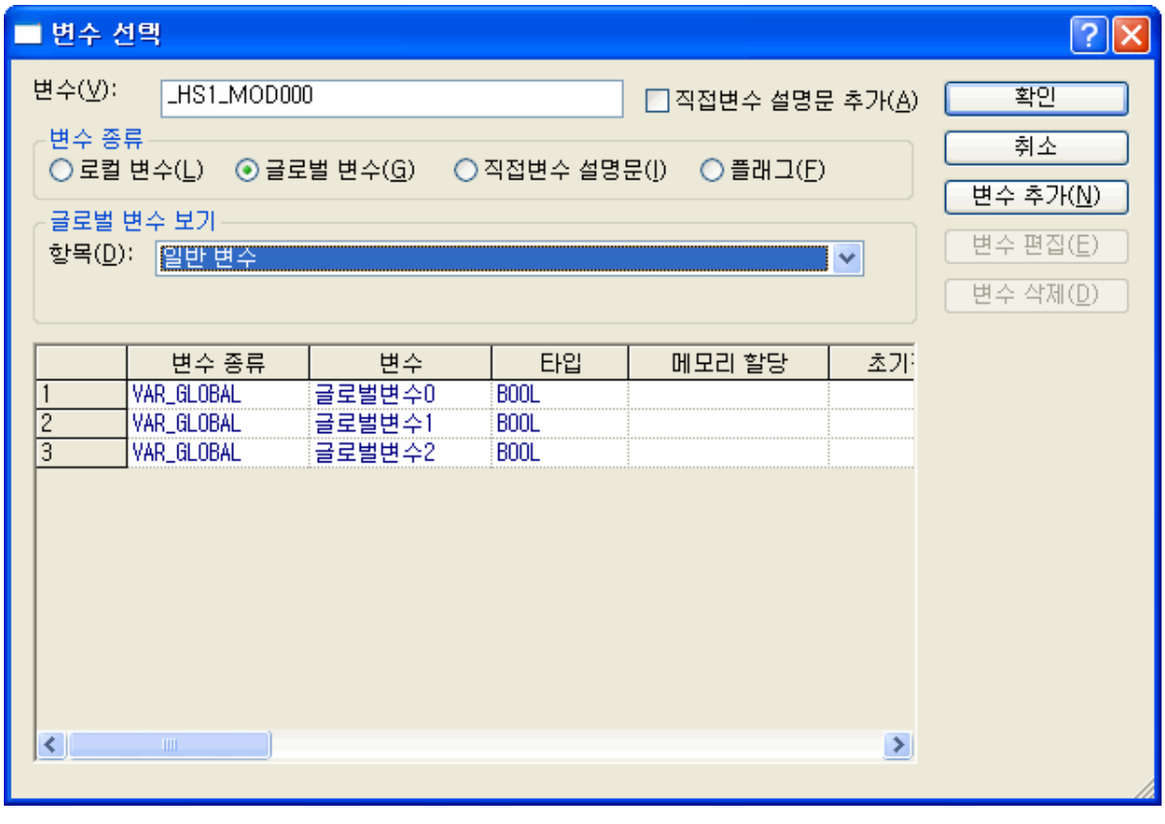
e. 변수 삭제: 선택된 변수를 로컬 변수 목록에서 삭제 합니다.

f. 확인: 입력 또는 선택한 사항을 적용하고 대화 상자를 닫습니다.

g. 취소: 대화 상자를 닫습니다.

[대화 상자]


변수 선택
변수(V):
직접변수 설명문 추가(A)
확인
취소
변수 추가(N)
변수 편집(E)
변수 삭제(D)
변수 종류
로컬 변수(L)
글로벌 변수(G)
직접변수 설명문(I)
플래그(F)
글로벌 변수 보기
항목(D):
전체
전체
일반 변수
베이스00, 슬롯07: XGF-RD4A (4채널)
베이스00, 슬롯09: XGF-DV8A (전압형, 8채널)
1 VAR_GLOBAL _0007_CHO_ACT BOOL %UX0.7.16
2 VAR_GLOBAL _0007_CHO_ADJERR BOOL %UX0.7.0
3 VAR_GLOBAL _0007_CHO_ALMEN BOOL %UX0.7.468
4 VAR_GLOBAL _0007_CHO_BOUT BOOL %UX0.7.20
5 VAR_GLOBAL _0007_CHO_FINDEN BOOL %UX0.7.464
6 VAR_GLOBAL _0007_CHO_MAX INT %UW0.7.13
7 VAR_GLOBAL _0007_CHO_MIN INT %UW0.7.12
8 VAR_GLOBAL _0007_CHO_PAH BOOL %UX0.7.34
9 VAR_GLOBAL _0007_CHO_PAHH BOOL %UX0.7.35
10 VAR_GLOBAL _0007_CHO_PAL BOOL %UX0.7.33
11 VAR_GLOBAL _0007_CHO_PALL BOOL %UX0.7.32
12 VAR_GLOBAL _0007_CHO_RAH BOOL %UX0.7.49

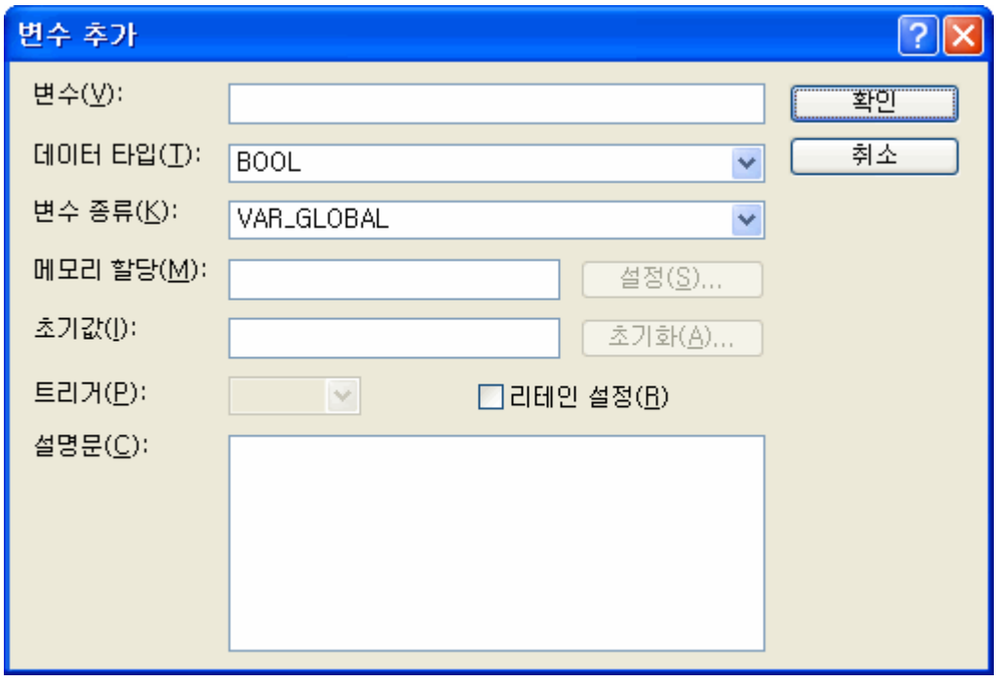

변수 선택
변수(V):
_HS1_MOD000
직접변수 설명문 추가(A)
확인
취소
변수 추가(N)
변수 편집(E)
변수 삭제(D)
변수 종류
로컬 변수(L)
글로벌 변수(G)
직접변수 설명문(I)
플래그(F)
글로벌 변수 보기
항목(D):
일반 변수
변수 종류 변수 타입 메모리 할당 초기
1 VAR_GLOBAL 글로벌변수0 BOOL
2 VAR_GLOBAL 글로벌변수1 BOOL
3 VAR_GLOBAL 글로벌변수2 BOOL

[대화 상자 설명]

a. 글로벌 변수: 선언된 글로벌 변수 목록을 표시합니다. EXTERNAL 변수로 등록 할 수 있습니다.

b. 글로벌 변수 보기: 글로벌 변수 목록을 전체, 일반 변수, 특수 모듈 관련 변수로 구분하여 표시 합니다.

c. 변수 추가: 글로벌 변수 목록에 변수를 추가 할 수 있는 대화 상자를 호출 합니다.



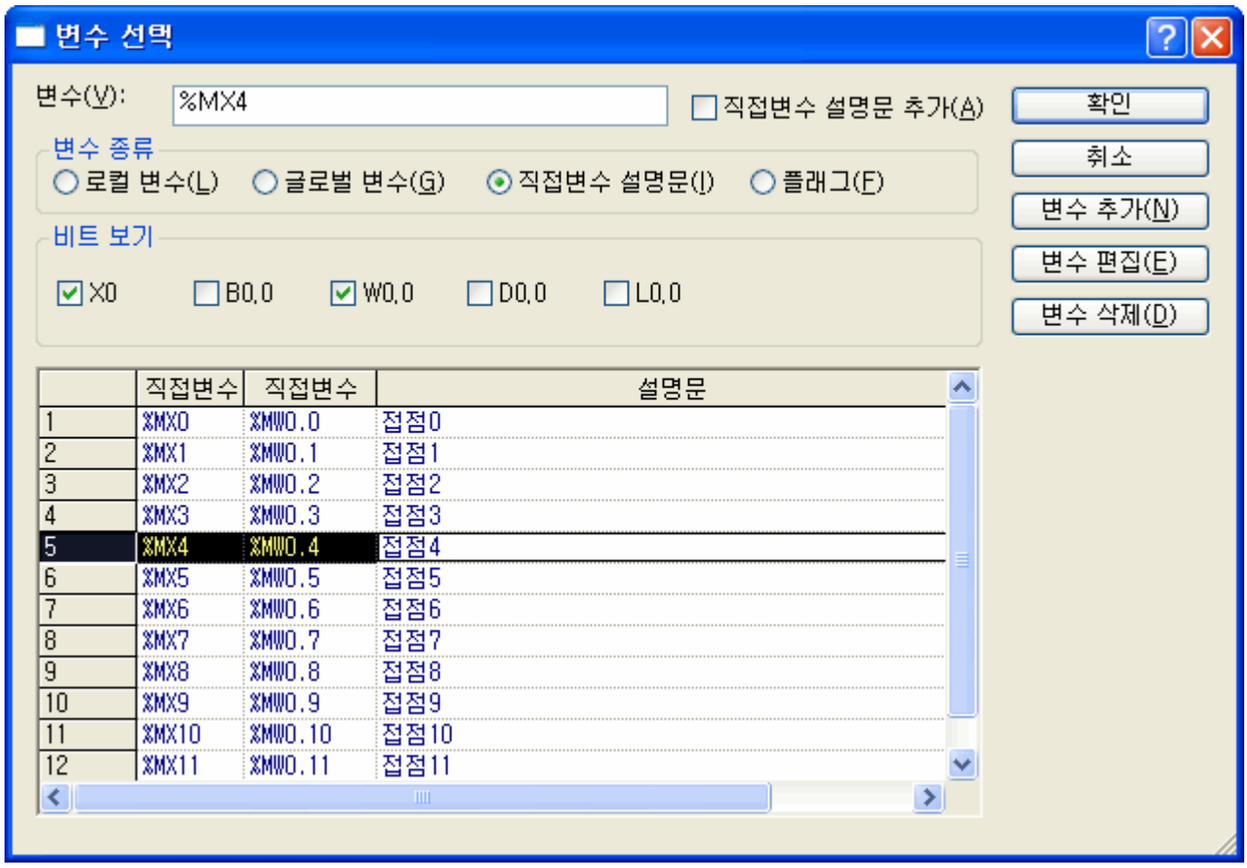
d. 변수 편집: 글로벌 변수의 편집은 제공하지 않습니다.

e. 변수 삭제: 글로벌 변수의 삭제는 제공하지 않습니다.

f. 확인: 입력 또는 선택된 변수를 로컬 변수 목록에 External 변수로 등록되고, 선택한 사항을 적용하고, 대화 상자를 닫습니다.

g. 취소: 대화 상자를 닫습니다.

[대화 상자]

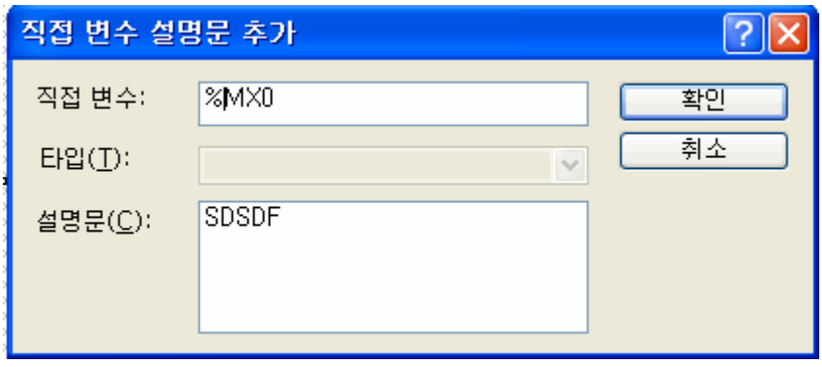


[대화 상자 설명]

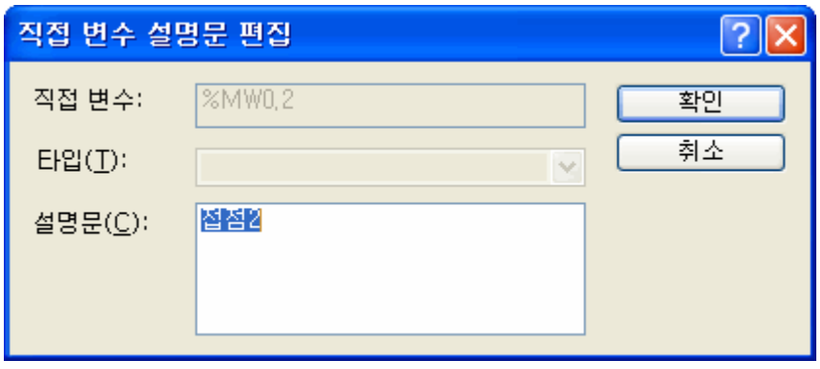
a. 직접 변수 설명문: 직접 변수 설명문 목록을 표시합니다.

b. 비트 보기: 비트형 직접 변수에 대해서 여러 가지 타입으로 직접변수를 표시 합니다. 비트(X0), 바이트(B0.0), 워드(W0.0), 더블 워드(D0.0), 롱 워드(L0.0) 형으로 표시 합니다.

c. 변수 추가: 직접 변수 설명문 목록에 직접 변수의 설명문을 추가 할 수 있는 대화 상자를 호출 합니다.



d. 변수 편집: 선택된 직접 변수 설명문을 편집 할 수 있는 대화 상자를 호출 합니다.

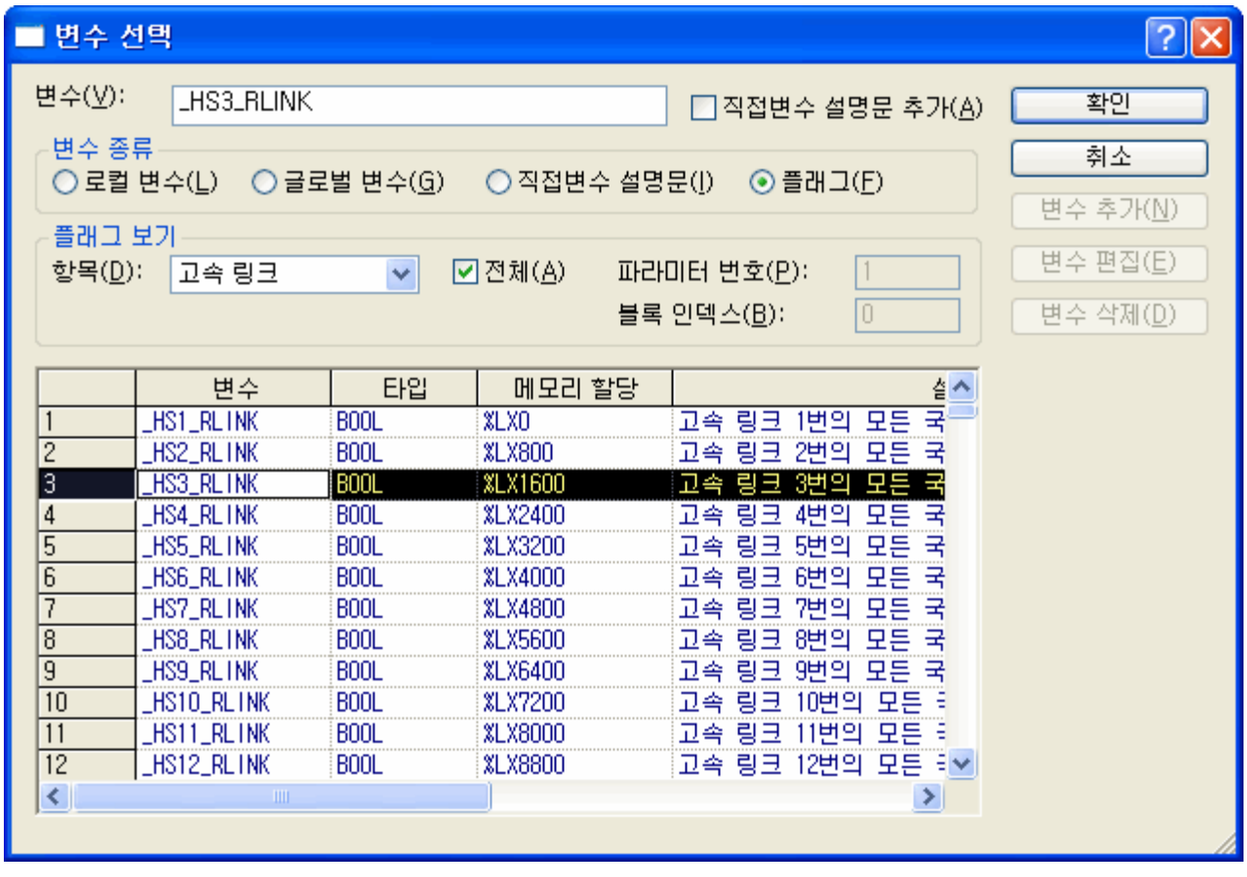


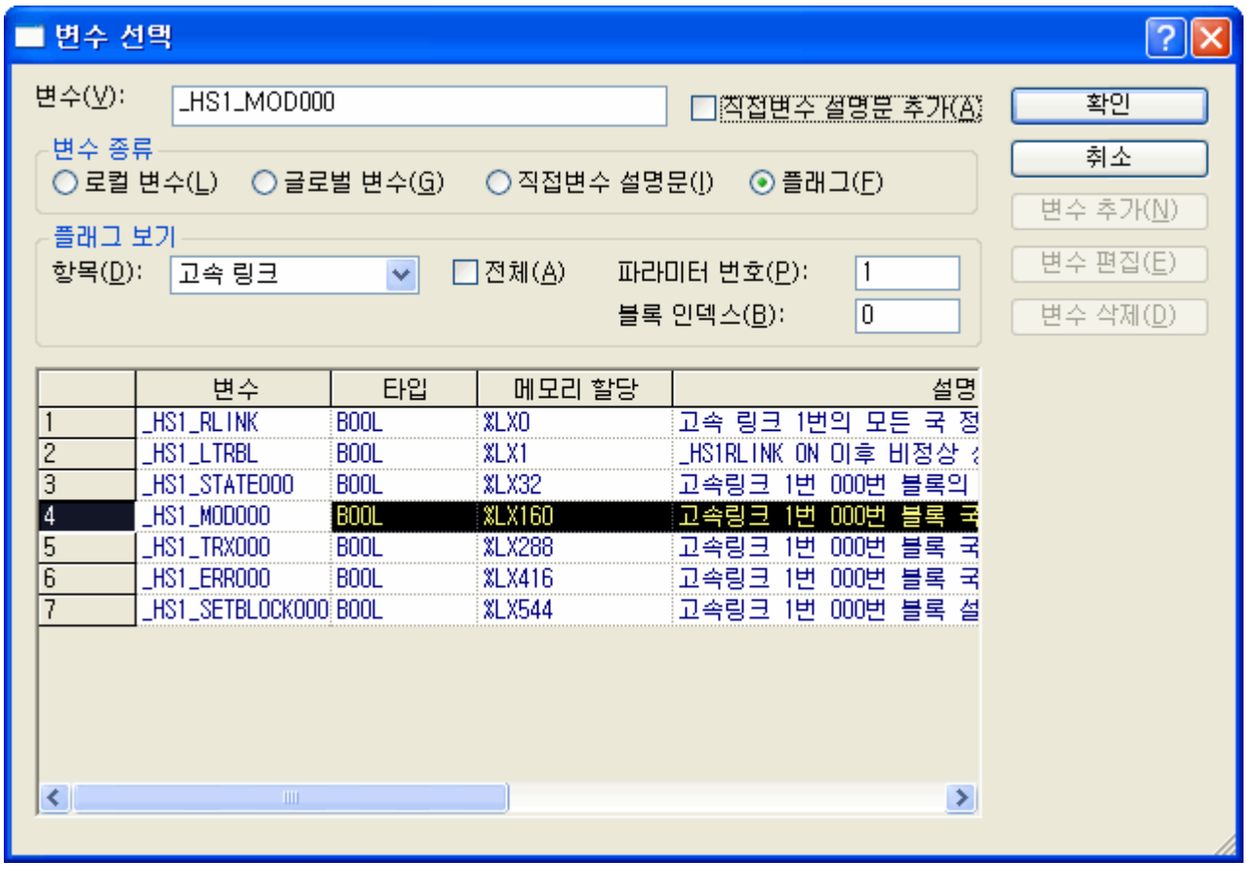
e. 변수 삭제: 선택된 직접 변수를 직접 변수 설명문 목록에서 삭제 합니다.

f. 확인: 입력 또는 선택한 사항을 적용하고 대화 상자를 닫습니다.

g. 취소: 대화 상자를 닫습니다.

[대화 상자]





[대화 상자 설명]

a. 플래그: 목록에 플래그를 표시합니다. 플래그의 상세 종류는 플래그 항목에서 선택할 수 있습니다.

b. 항목: 플래그의 종류를 표시하는 선택 상자로, 시스템/고속링크/P2P/PID 플래그를 선택할 수 있습니다.

c. 전체: 항목에서 선택한 플래그 전체를 표시할 지, 입력한 파라미터 번호/블록 인덱스에 해당하는 플래그만 표시할 지 여부를 선택합니다.

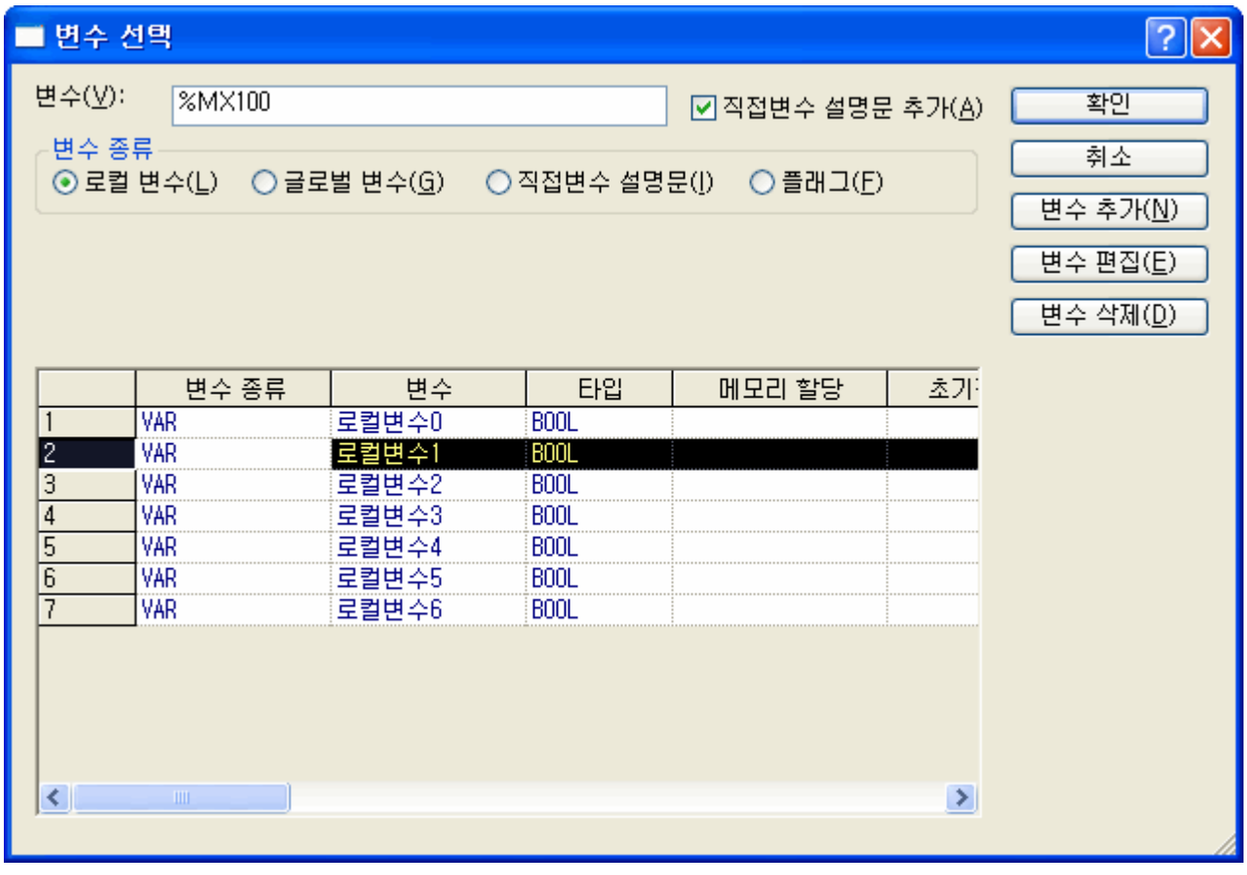
d. 파라미터 번호: 선택한 플래그 항목별 설정 번호를 입력합니다. 고속링크는 0~12, P2P 는 0~8, PID 는 0~63 입니다. PLC 종류에 따라서 달라 질 수 있습니다.

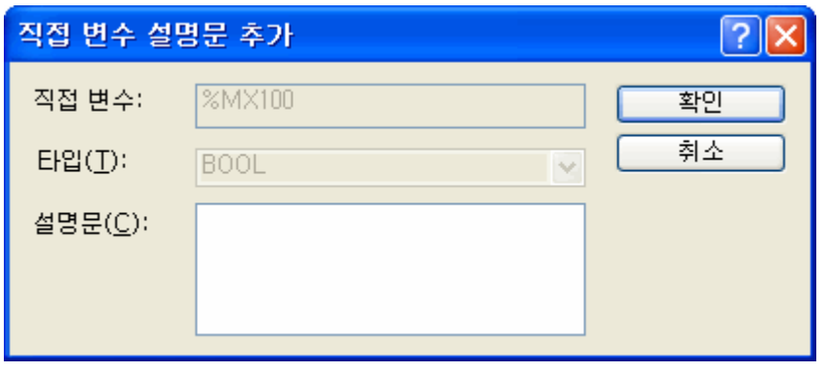
e. 블록 인덱스: 선택한 플래그의 항목별 블록 번호를 입력합니다. 고속링크는 0~127, P2P 는 0~63 입니다. PLC 종류에 따라서 달라 질 수 있습니다.

f. 확인: 입력 또는 선택한 사항을 적용하고 대화 상자를 닫습니다.

g. 취소: 대화 상자를 닫습니다.

[대화 상자]





[대화 상자 설명]

a. 직접 변수 설명문 추가: 입력된 변수가 직접 변수이고 설명문 없는 경우, 직접 변수 설명문 추가 대화 상자를 호출 합니다.

b. 직접 변수: 입력된 직접 변수를 표시 합니다.

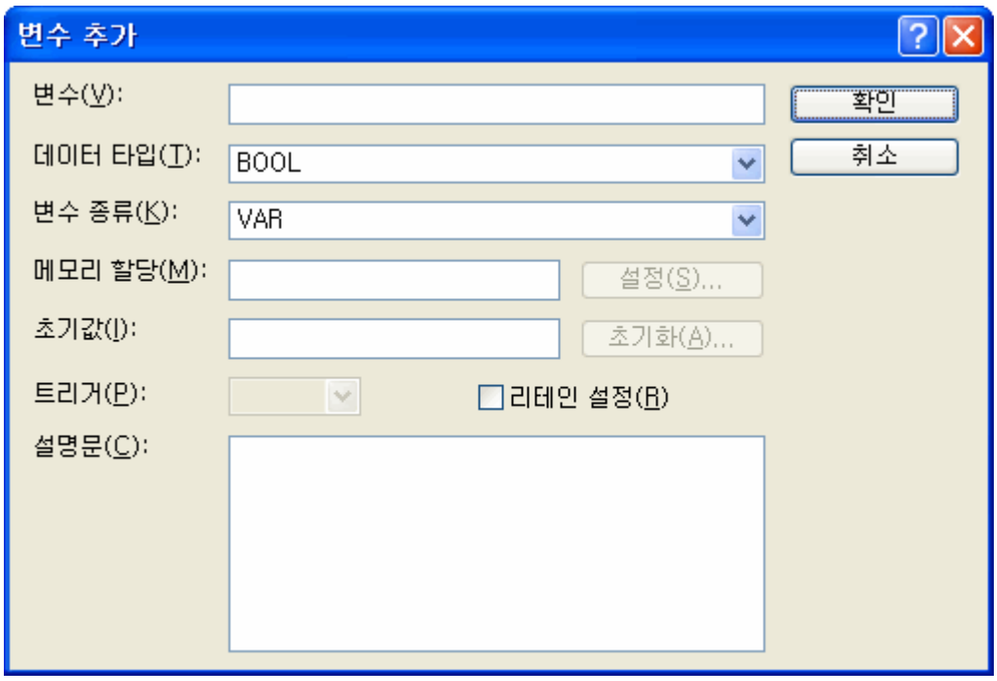
c. 타입: 직접 변수의 타입을 표시 합니다.

d. 설명문: 직접 변수에 대한 설명문을 입력 합니다.

e. 확인: 직접 변수 설명문 목록에 추가 하고, 대화 상자를 닫습니다.

f. 취소: 대화 상자를 닫습니다.

[대화 상자]



[대화 상자 설명]

a. 변수: 추가할 변수 명을 입력합니다.

b. 변수 종류: 추가할 변수에 대한 종류를 선택 합니다.

c. 변수 타입: 추가할 변수에 대한 종류를 선택 합니다.

d. 메모리 할당: 추가할 변수에 대한 직접 주소를 할당 합니다. 만일, 타입이 구조체인 경우는 설정 버튼이 활성화 됩니다. 구조체의 멤버에 대한 메모리 할당을 할 수 있습니다.

e. 초기값: 추가할 변수의 초기값을 입력합니다. 만일, 타입이 배열이거나 구조체인 경우는 초기화 버튼이 활성화 됩니다. 배열 및 구조체의 멤버에 대한 초기값을 입력 할 수 있습니다.

f. 트리거: 사용자 펑션 블록의 변수 종류가 VAR_INPUT 인 경우, 추가할 변수에 대한 트리거 상태를 설정 할 수 있습니다.

g. 리테인 설정: 추가할 변수에 대한 리테인 상태를 입력합니다.

h. 설명문: 추가할 변수에 대한 설명문을 입력합니다.

알아두기

1. 변수 목록에 없는 문자열에 커서가 위치한 후 [변수 추가/선택] 명령을 수행할 시에는 새로운 변수를 추가할 수 있습니다.
2. 문자열을 그룹으로 선택한 후 [변수 추가/선택] 명령을 수행할 시에는 선택된 문자열 그룹이 변수로 대치됩니다.

14.4.2.4. 펑션/펑션블록 삽입

커서 위치에 펑션/펑션블록 문자열을 삽입합니다.

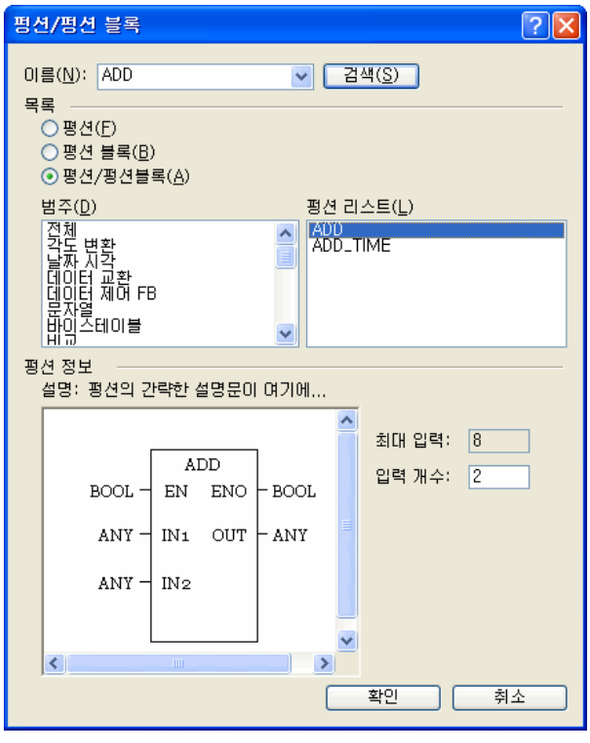
[순서]

1. 입력하고자 하는 위치로 커서를 이동시킵니다.

```
22      // SETTING ERROR FLAG!
23      IF LWORD_TMP <> LINT_VAL THEN ERROR := TRUE;
24      ELSE ERROR := FALSE;
25      END_IF;
26
27      DWORD_LOWER := LWORD_TO_DWORD(LWORD_TMP AND 16#00000000FFFFFFFF);
28      DWORD_HIGHER := LWORD_TO_DWORD(SHR(LWORD_TMP AND 16#FFFFFFFF00000000, 32));
29
30  END_IF;
```

2. 메뉴 [편집]-[펑션/펑션블록]을 선택합니다.

[대화 상자]



[대화 상자 설명]

a. 이름: 사용할 펑션(블록)의 이름을 입력합니다.

b. 검색: 입력한 이름의 펑션(펑션)블록을 검색합니다.

c. 목록: 대화 상자에 펑션만, 펑션블록만, 혹은 모두 표시할지 여부를 선택합니다.

d. 범주: 펑션(블록)의 범주를 표시합니다.

e. 펑션 리스트: 선택된 범주에 속한 펑션(블록)의 목록을 표시합니다.

f. 펑션 정보: 펑션의 정보 및 속성을 표시합니다. 펑션의 경우 입력 파라미터에 관한 사항을 설정할 수 있으며, 펑션 블록의 경우 인스턴스 명 및 인스턴스의 클래스를 선택할 수 있습니다.

g. 확인: 입력한 내용을 적용하고 대화 상자를 닫습니다.

h. 취소: 대화 상자를 닫습니다.

3. 펑션/펑션블록이 삽입됩니다.

```
22      // SETTING ERROR FLAG!
23      IF LWORD_TMP <> LINT_VAL THEN ERROR := TRUE;
24      ELSE ERROR := FALSE;
25      END_IF;
26
27      DWORD_LOWER := LWORD_TO_DWORD(LWORD_TMP AND 16#00000000FFFFFFFF);
28      DWORD_HIGHER := LWORD_TO_DWORD(SHR(LWORD_TMP AND 16#FFFFFFFF00000000, 32));
29      ADD( ANY_IN1, ANY_IN2, ANY_IN3, ANY_IN4, ANY_IN5, ANY_IN6, ANY_IN7, ANY_IN8 )
30  END_IF;
```

알아두기

1. 펑션/펑션 블록의 각 입/출력 파라미터는 자동으로 입력되는 것이 아닙니다. 사용자 의도에 따라 추가적이 수정이 필요합니다.
2. 펑션블록을 복사/붙여넣기 시 인스턴트명이 동일하게 설정되므로 펑션블록 삽입을 통해 새 인스턴트 명으로 등록해야 합니다.

14.4.3. 프로그램 보기

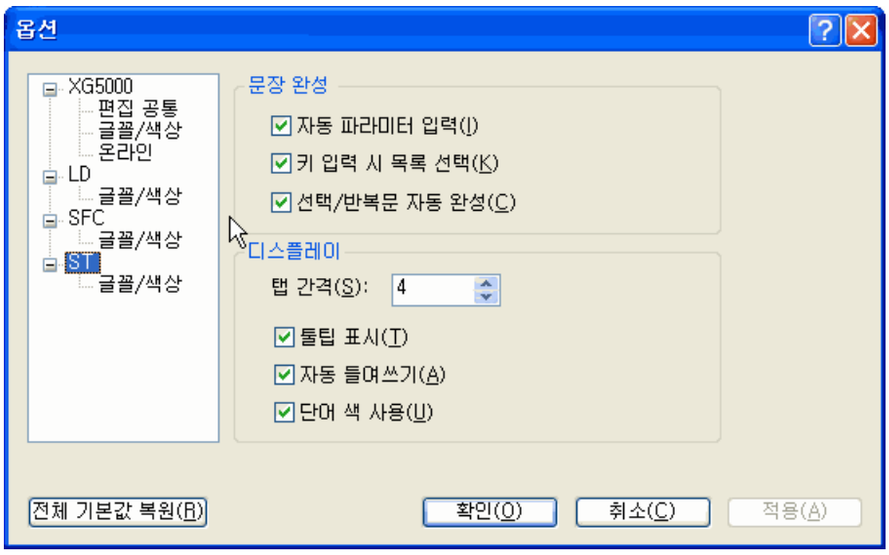
14.4.3.1. ST 옵션

ST 프로그램에 관한 옵션 대화상자 설명입니다.

[순서]

1. 메뉴 [도구]-[옵션]을 선택합니다.
2. ST 를 선택합니다.

[대화 상자]



[대화 상자 설명]

a. 자동 파라미터 입력: 펑션/펑션블록에서 펑션/펑션블록 선택 후 프로그램에 입력시 입출력 파라미터의 예를 같이 입력합니다.

b. 키 입력 시 목록 선택: 키보드로 문자 입력 시 입력된 문자열로 시작하는 펑션/펑션 블록 및 변수 이름을 나열합니다.

c. 선택/반복문 자동 완성: ST 프로그램의 제어문인 IF, WHILE, SWITCH 등의 제어문을 입력 후 엔터키 입력 시 ST 문법에 맞게 자동완성합니다.

d. 탭 간격: 탭 키 입력 시 띄어쓰기 할 개수를 입력합니다.

e. 툴팁 표시: ST 프로그램 내 문자열 위로 마우스 이동 시 문자열의 설명하는 내용이 표시됩니다.

f. 자동 들여쓰기: 엔터키 입력으로 줄 바꾸기시 이전 열의 탭 수만큼 들여 씁니다.

g. 단어 색 사용: ST 프로그램에 문자열을 변수, 예약어, 설명문, 펑션/펑션 블록 등에 따라 다양한 색깔을 표시합니다.

14.4.3.2. 글꼴/색상

ST 프로그램에서 글꼴이나 단어 단위로 색상을 지정할 수 있습니다.

1) 글꼴

[순서]

1. 메뉴 [도구]-[옵션]을 선택합니다.
6. ST 글꼴/색상을 선택합니다.
7. 글꼴을 변경합니다.

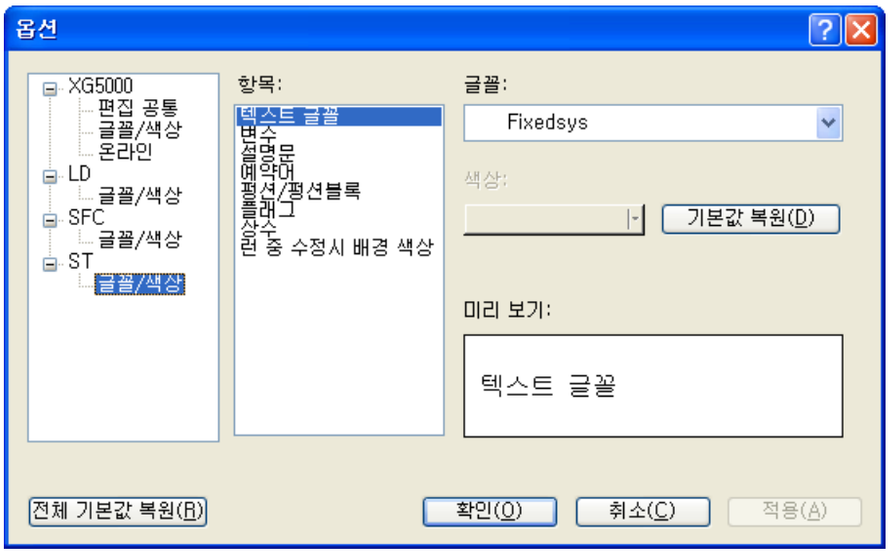
알아두기

1. 글자 크기는 변경할 수 없습니다.
2. 기본 글꼴은 “Fixedsys” 입니다.
3. 자세한 옵션 설정사항은 2.6 장 [옵션]을 참고하십시오.

2) 색상

[순서]

1. 메뉴 [도구]-[옵션]을 선택합니다.
2. ST 글꼴/색상을 선택합니다.
3. 색상을 변경합니다.



알아두기

1. 색상 변경 가능한 항목은 “변수”, “설명문”, “예약어”, “펑션/펑션블록”, “플래그”, “상수”, “런 중 수정시 배경 색상” 등이 있습니다.
2. 자세한 옵션 설정사항은 2.6 장 [옵션]을 참고하십시오.

14.4.3.3. 확대/축소

ST 프로그램은 확대/축소 기능을 지원하지 않습니다.

14.4.3.4. 탭 간격

들여쓰기 시 탭 문자의 간격을 지정합니다.

[순서]

1. 메뉴 [도구]-[옵션]을 선택합니다.
2. ST 편집 설정을 선택합니다.
3. 탭 간격을 변경합니다.

```
1
2 CLOCK_SOURCE := _T1S;
3
4 // LEFT rotate, FIND transition
5 IF (* _T1S XOR *) 0 = PREV_STATUS THEN
6 //IF (* _T1S XOR *) 0 = PREV_STATUS AND (PREV_STATUS XOR CLOCK_SOURCE) THEN
7     LINT_VAL := LINT_VAL + 1;
8     %ML0 := %ML0 + 1;
9
10    FOR IDX := 0 TO MAX_VALUE - 1 DO
11        mask := SHL(LWORD#1, IDX);
12        LONG_ARRAY[IDX, IDX] := mask = (LINT_VAL AND mask);
13    END_FOR;
14
```

[탭 크기 4 인 화면]

```
1
2 CLOCK_SOURCE := _T1S;
3
4 // LEFT rotate, FIND transition
5 IF (* _T1S XOR *) 0 = PREV_STATUS THEN
6 //IF (* _T1S XOR *) 0 = PREV_STATUS AND (PREV_STATUS XOR CLOCK_SOURCE) THEN
7         LINT_VAL := LINT_VAL + 1;
8         %ML0 := %ML0 + 1;
9
10        FOR IDX := 0 TO MAX_VALUE - 1 DO
11                mask := SHL(LWORD#1, IDX);
12                LONG_ARRAY[IDX, IDX] := mask = (LINT_VAL AND mask);
13        END_FOR;
```

[탭 크기 8 인 화면]

알아두기

1. 기본 탭 크기는 4 입니다.
2. 탭 크기의 범위는 1~100 사이입니다

14.4.3.5. 라인 번호 보이기

ST 프로그램에서 라인 번호를 보이거나 숨기거나 합니다.

[순서]

1. 메뉴 [도구]-[옵션]을 선택합니다.
2. XG5000 편집 공통을 선택합니다.
3. 라인 번호 표시를 선택합니다.

14.4.4. 편집 부가 기능

14.4.4.1. 북마크

북 마크를 설정하여, 관심 있는 부분으로 쉽게 이동할 수 있습니다.

1) 북 마크 설정

[순서]

1. 북 마크를 설정하고자 하는 위치로 커서를 이동시킵니다.

```
1
2 CLOCK_SOURCE := _T1S;
3
4 // LEFT rotate, FIND transition
5 IF (* _T1S XOR *) 0 = PREV_STATUS THEN
6 //IF (* _T1S XOR *) 0 = PREV_STATUS AND (PREV_STATUS XOR CLOCK_SOURCE) THEN
7     LINT_VAL := LINT_VAL + 1;
8     %ML0 := %ML0 + 1;
9
10    FOR IDX := 0 TO MAX_VALUE - 1 DO
11        mask := SHL(1, IDX);
12        LONG_ARRAY[IDX] := mask = (LINT_VAL AND mask);
13    END_FOR;
14
```

2. 메뉴 [편집]-[북 마크]-[설정/해제]를 선택합니다.

```
1
2 CLOCK_SOURCE := _T1S;
3
4 // LEFT rotate, FIND transition
5 IF (* _T1S XOR *) 0 = PREV_STATUS THEN
6 //IF (* _T1S XOR *) 0 = PREV_STATUS AND (PREV_STATUS XOR CLOCK_SOURCE) THEN
7     LINT_VAL := LINT_VAL + 1;
8     %ML0 := %ML0 + 1;
9
10    FOR IDX := 0 TO MAX_VALUE - 1 DO
11        mask := SHL(1, IDX);
12        LONG_ARRAY[IDX] := mask = (LINT_VAL AND mask);
13    END_FOR;
14
```

2) 북 마크 해제

[순서]

1. 북 마크를 해제하고자 하는 위치로 커서를 이동시킵니다.

```
1
2 CLOCK_SOURCE := _T1S;
3
4 // LEFT rotate, FIND transition
5 IF (* _T1S XOR *) 0 = PREV_STATUS THEN
6 //IF (* _T1S XOR *) 0 = PREV_STATUS AND (PREV_STATUS XOR CLOCK_SOURCE) THEN
7     LINT_VAL := LINT_VAL + 1;
8     %ML0 := %ML0 + 1;
9
10    FOR IDX := 0 TO MAX_VALUE - 1 DO
11        mask := SHL(1, IDX);
12        LONG_ARRAY[IDX] := mask = (LINT_VAL AND mask);
13    END_FOR;
14
```

2. 메뉴 [편집]-[북 마크]-[설정/해제]를 선택합니다.

```
1
2 CLOCK_SOURCE := _T1S;
3
4 // LEFT rotate, FIND transition
5 IF (* _T1S XOR *) 0 = PREV_STATUS THEN
6 //IF (* _T1S XOR *) 0 = PREV_STATUS AND (PREV_STATUS XOR CLOCK_SOURCE) THEN
7     LINT_VAL := LINT_VAL + 1;
8     %ML0 := %ML0 + 1;
9
10    FOR IDX := 0 TO MAX_VALUE - 1 DO
11        mask := SHL(1, IDX);
12        LONG_ARRAY[IDX] := mask = (LINT_VAL AND mask);
13    END_FOR;
14
```

3) 모든 북 마크 해제

[순서]

1. 메뉴 [편집]-[북 마크]-[모두 해제]를 선택합니다.

```
1
2 CLOCK_SOURCE := _T1S;
3
4 // LEFT rotate, FIND transition
5 IF (* _T1S XOR *) 0 = PREV_STATUS THEN
6 //IF (* _T1S XOR *) 0 = PREV_STATUS AND (PREV_STATUS XOR CLOCK_SOURCE) THEN
7     LINT_VAL := LINT_VAL + 1;
8     %ML0 := %ML0 + 1;
9
10    FOR IDX := 0 TO MAX_VALUE - 1 DO
11        mask := SHL(1, IDX);
12        LONG_ARRAY[IDX] := mask = (LINT_VAL AND mask);
13    END_FOR;
14
15    LWORD_TMP := 0;
16    FOR IDX := 0 TO MAX_VALUE - 1 DO
17        //MASK := MASK OR SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
18        mask := SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
19        LWORD_TMP := mask OR LWORD_TMP;
20    END_FOR;
```

4) 이전 북마크 이동

[순서]

1. 메뉴 [편집]-[북 마크]-[이전 북마크]를 선택합니다.

5) 다음 북마크 이동

[순서]

1. 메뉴 [편집]-[북 마크]-[다음 북마크]를 선택합니다.

알아두기

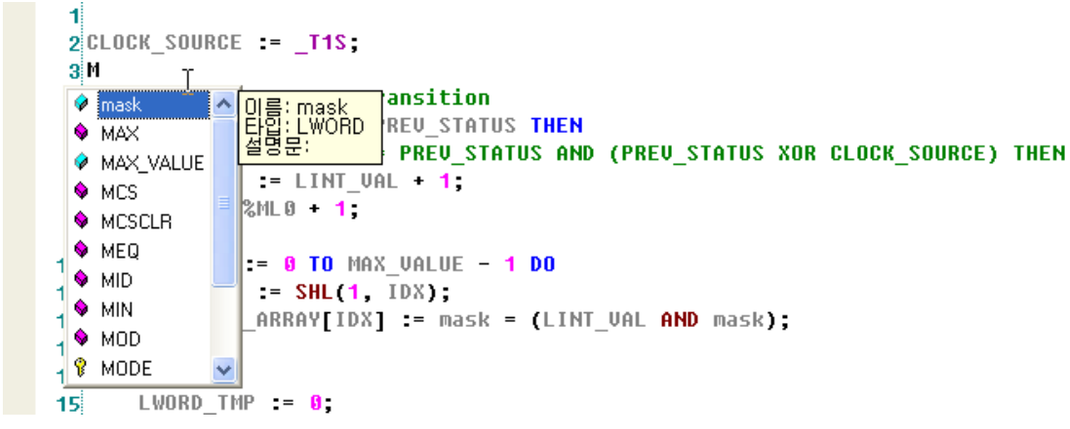
1. 북 마크는 라인 단위로 설정됩니다.
2. 북 마크는 편집 사항이 아니므로, 설정/해제에 관한 사항은 편집 취소 및 재 실행에 포함되지 않습니다.

14.4.4.2. 문자열 목록에서 선택

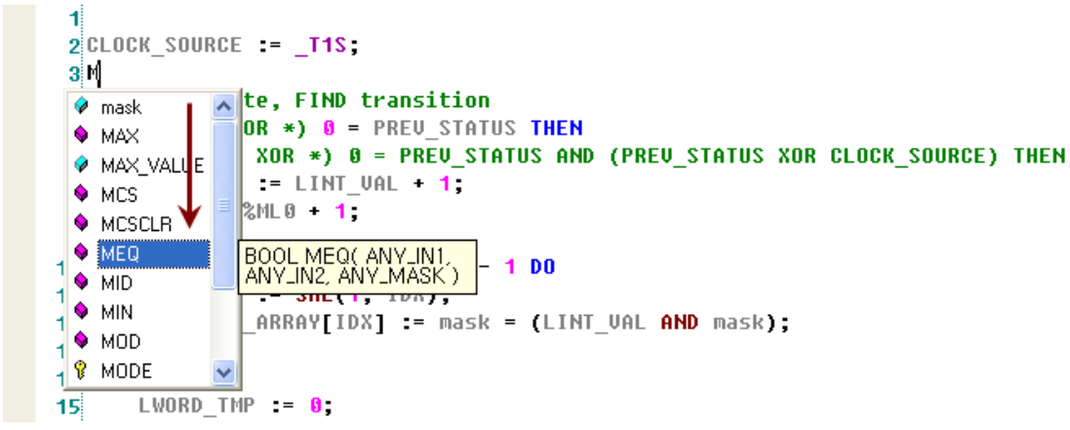
키보드로 문자 입력 시 같은 문자열로 시작하는 문자들을 보여주어 사용자가 편리하게 선택할 수 있습니다.

[순서]

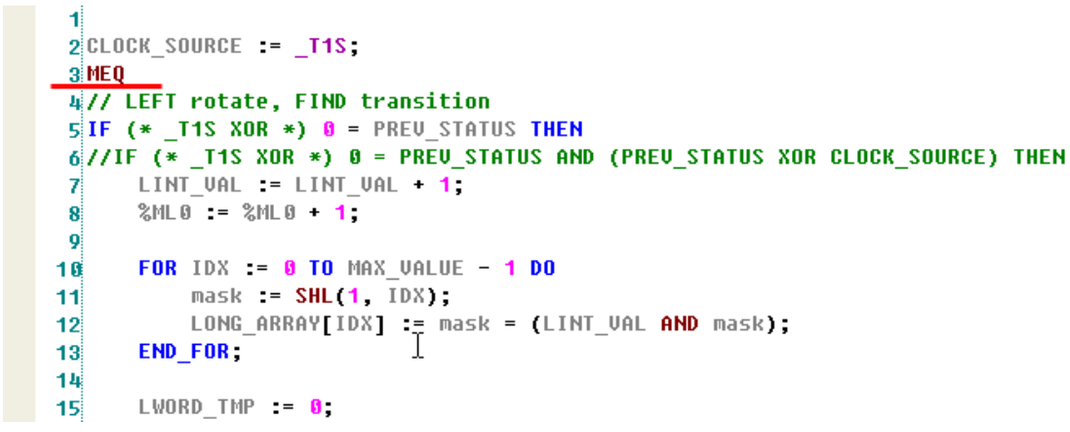
1. 키보드로 문자를 입력합니다.



2. 입력된 문자로 시작하는 목록에서 입력할 문자열을 선택합니다.



3. 엔터 키를 누르거나 마우스로 더블클릭 합니다.



알아두기

문자열 목록 비트맵 설명

1. : IF, CASE WHILE 등의 ST 언어 키워드
2. : 변수 이름
3. : 플래그 변수 이름
4. : 펑션 이름
5. : 펑션블록 인스턴스 이름
6. : 사용자 정의 펑션 이름
7. : 사용자 정의 펑션블록 인스턴스 이름

14.4.4.3. 문자열 목록에서 맴버 변수 선택

펑션블록 또는 사용자 데이터 타입 인스턴스 이름으로 문자열 목록에서 멤버 변수 선택하는 기능입니다.

[순서]

1. 키보드로 펑션블록 또는 사용자 데이터 타입 인스턴스 이름 다음에 '.' 을 입력합니다.

예) TON 펑션블록의 인스턴스 이름 TON_Inst 인 경우

```
1
2 CLOCK_SOURCE := _T1S;
3 TON_Inst.
4 // LEF
5 IF (*                                    S THEN
6 //IF (                                   TUS AND (PREV_STATUS XOR CLOCK_SOURCE) THEN
7     LI                                   NT_VAL + 1;
8     %ML                                  1;
9
10    FOR IDX := 0 TO MAX_VALUE - 1 DO
11        mask := SHL(1, IDX);
12        LONG_ARRAY[IDX] := mask = (LINT_VAL AND mask);
13    END_FOR;
```

TON_Inst.ET
TON_Inst.IN
TON_Inst.PT
TON_Inst.Q

이름: TON_Inst.ET
타입: TIME
설명문:

2. 입력할 멤버 변수를 선택합니다.

```
1
2 CLOCK_SOURCE := _T1S;
3 TON_Inst.
4 // LEF                                   ND transition
5 IF (*                                    0 = PREV_STATUS THEN
6 //IF (                                   TUS AND (PREV_STATUS XOR CLOCK_SOURCE) THEN
7     LI
8     %ML
9
10    FOR IDX := 0 TO MAX_VALUE - 1 DO
11        mask := SHL(1, IDX);
12        LONG_ARRAY[IDX] := mask = (LINT_VAL AND mask);
13    END_FOR;
```

TON_Inst.ET
TON_Inst.IN
TON_Inst.PT
TON_Inst.Q

이름: TON_Inst.PT
타입: TIME
설명문:

3. 엔터 키를 누르거나 마우스로 더블클릭 합니다.

```
1
2 CLOCK_SOURCE := _T1S;
3 TON_Inst.PT
4 // LEFT rotate, FIND transition
5 IF (* _T1S XOR *) 0 = PREV_STATUS THEN
6 //IF (* _T1S XOR *) 0 = PREV_STATUS AND (PREV_STATUS XOR CLOCK_SOURCE) THEN
7     LINT_VAL := LINT_VAL + 1;
8     %ML0 := %ML0 + 1;
9
10    FOR IDX := 0 TO MAX_VALUE - 1 DO
11        mask := SHL(1, IDX);
12        LONG_ARRAY[IDX] := mask = (LINT_VAL AND mask);
13    END_FOR;
```

14.4.4.4. 비 실행문 설정/해제

ST 프로그램 내용 중 PLC 에서 실행되지 않을 영역을 설정하거나 해제합니다.

기호 “(*” 와 “*)” 를 사용하여 비 실행문으로 설정합니다.

1) 비 실행문 설정

[순서]

1. 비 실행문을 설정할 영역을 선택합니다.

```
1
2  CLOCK_SOURCE := _T1S;
3
4  // LEFT rotate, FIND transition
5  IF (* _T1S XOR *) 0 = PREV_STATUS THEN
6  //IF (* _T1S XOR *) 0 = PREV_STATUS AND (PREV_STATUS XOR CLOCK_SOURCE) THEN
7      LINT_VAL := LINT_VAL + 1;
8      %ML0 := %ML0 + 1;
9
10     FOR IDX := 0 TO MAX_VALUE - 1 DO
11         mask := SHL(1, IDX);
12         LONG_ARRAY[IDX] := mask = (LINT_VAL AND mask);
13     END_FOR;
14
15     LWORD_TMP := 0;
16     FOR IDX := 0 TO MAX_VALUE - 1 DO
17         //MASK := MASK OR SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
18         mask := SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
19         LWORD_TMP := mask OR LWORD_TMP;
20     END_FOR;
```

2. 메뉴 [편집]-[비실행문 설정/해제]을 선택합니다.

```
1
2  CLOCK_SOURCE := _T1S;
3
4  // LEFT rotate, FIND transition
5  IF (* _T1S XOR *) 0 = PREV_STATUS THEN
6  //IF (* _T1S XOR *) 0 = PREV_STATUS AND (PREV_STATUS XOR CLOCK_SOURCE) THEN
7      LINT_VAL := LINT_VAL + 1;
8      %ML0 := %ML0 + 1;
9
10 (*  FOR IDX := 0 TO MAX_VALUE - 1 DO
11         mask := SHL(1, IDX);
12         LONG_ARRAY[IDX] := mask = (LINT_VAL AND mask);
13     END_FOR;
14 *)
15     LWORD_TMP := 0;
16     FOR IDX := 0 TO MAX_VALUE - 1 DO
17         //MASK := MASK OR SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
18         mask := SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
19         LWORD_TMP := mask OR LWORD_TMP;
20     END_FOR;
```

2) 비 실행문 해제

[순서]

1. 이미 설정되어 있는 비 실행문을 선택합니다.

```
1
2 CLOCK_SOURCE := _T1S;
3
4 // LEFT rotate, FIND transition
5 IF (* _T1S XOR *) 0 = PREV_STATUS THEN
6 //IF (* _T1S XOR *) 0 = PREV_STATUS AND (PREV_STATUS XOR CLOCK_SOURCE) THEN
7     LINT_VAL := LINT_VAL + 1;
8     %ML0 := %ML0 + 1;
9
10 (*  FOR IDX := 0 TO MAX_VALUE - 1 DO
11         mask := SHL(1, IDX);
12         LONG_ARRAY[IDX] := mask = (LINT_VAL AND mask);
13     END_FOR;
14 *)
15     LWORD_TMP := 0;
16     FOR IDX := 0 TO MAX_VALUE - 1 DO
17         //MASK := MASK OR SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
18         mask := SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
19         LWORD_TMP := mask OR LWORD_TMP;
20     END_FOR;
```

2. 메뉴 [편집]-[비실행문 설정/해제]를 선택합니다.

```
1
2 CLOCK_SOURCE := _T1S;
3
4 // LEFT rotate, FIND transition
5 IF (* _T1S XOR *) 0 = PREV_STATUS THEN
6 //IF (* _T1S XOR *) 0 = PREV_STATUS AND (PREV_STATUS XOR CLOCK_SOURCE) THEN
7     LINT_VAL := LINT_VAL + 1;
8     %ML0 := %ML0 + 1;
9
10 (*  FOR IDX := 0 TO MAX_VALUE - 1 DO
11         mask := SHL(1, IDX);
12         LONG_ARRAY[IDX] := mask = (LINT_VAL AND mask);
13     END_FOR;
14 *)
15     LWORD_TMP := 0;
16     FOR IDX := 0 TO MAX_VALUE - 1 DO
17         //MASK := MASK OR SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
18         mask := SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
19         LWORD_TMP := mask OR LWORD_TMP;
20     END_FOR;
```

14.4.4.5. 라인 비 실행문 설정/해제

비 실행문 설정/해제와 다르게 선택된 영역이 아니고, 선택 시작된 위치부터 라인 단위로 비 실행문을 설정합니다. 기호 "//" 를 사용하여 라인 비 실행문을 설정합니다.

1) 라인 비 실행문 설정

[순서]

1. 라인 비 실행문을 설정할 영역을 선택합니다.

```
1
2 CLOCK_SOURCE := _T1S;
3
4 // LEFT rotate, FIND transition
5 IF (* _T1S XOR *) 0 = PREV_STATUS THEN
6 //IF (* _T1S XOR *) 0 = PREV_STATUS AND (PREV_STATUS XOR CLOCK_SOURCE) THEN
7     LINT_VAL := LINT_VAL + 1;
8     %ML0 := %ML0 + 1;
9
10    FOR IDX := 0 TO MAX_VALUE - 1 DO
11        mask := SHL(1, IDX);
12        LONG_ARRAY[IDX] := mask = (LINT_VAL AND mask);
13    END_FOR;
14
15    LWORD_TMP := 0;
16    FOR IDX := 0 TO MAX_VALUE - 1 DO
17        //MASK := MASK OR SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
18        mask := SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
19        LWORD_TMP := mask OR LWORD_TMP;
20    END_FOR;
```

2. 메뉴 [편집]-[라인 비실행문 설정/해제]를 선택합니다.

```
1
2 CLOCK_SOURCE := _T1S;
3
4 // LEFT rotate, FIND transition
5 IF (* _T1S XOR *) 0 = PREV_STATUS THEN
6 //IF (* _T1S XOR *) 0 = PREV_STATUS AND (PREV_STATUS XOR CLOCK_SOURCE) THEN
7     LINT_VAL := LINT_VAL + 1;
8     %ML0 := %ML0 + 1;
9
10 //  FOR IDX := 0 TO MAX_VALUE - 1 DO
11 //      mask := SHL(1, IDX);
12 //      LONG_ARRAY[IDX] := mask = (LINT_VAL AND mask);
13 //  END_FOR;
14
15    LWORD_TMP := 0;
16    FOR IDX := 0 TO MAX_VALUE - 1 DO
17        //MASK := MASK OR SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
18        mask := SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
19        LWORD_TMP := mask OR LWORD_TMP;
20    END_FOR;
```

2) 라인 비 실행문 해제

[순서]

1. 라인 비 실행문 해제할 영역을 선택합니다.

```
1
2 CLOCK_SOURCE := _T1S;
3
4 // LEFT rotate, FIND transition
5 IF (* _T1S XOR *) 0 = PREV_STATUS THEN
6 //IF (* _T1S XOR *) 0 = PREV_STATUS AND (PREV_STATUS XOR CLOCK_SOURCE) THEN
7     LINT_VAL := LINT_VAL + 1;
8     %ML0 := %ML0 + 1;
9
10 //  FOR IDX := 0 TO MAX_VALUE - 1 DO
11 //      mask := SHL(1, IDX);
12 //      LONG_ARRAY[IDX] := mask = (LINT_VAL AND mask);
13 //  END_FOR;
14
15     LWORD_TMP := 0;
16     FOR IDX := 0 TO MAX_VALUE - 1 DO
17         //MASK := MASK OR SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
18         mask := SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
19         LWORD_TMP := mask OR LWORD_TMP;
20     END_FOR;
```

2. 메뉴 [편집]-[라인 비실행문 설정/해제]를 선택합니다.

```
1
2 CLOCK_SOURCE := _T1S;
3
4 // LEFT rotate, FIND transition
5 IF (* _T1S XOR *) 0 = PREV_STATUS THEN
6 //IF (* _T1S XOR *) 0 = PREV_STATUS AND (PREV_STATUS XOR CLOCK_SOURCE) THEN
7     LINT_VAL := LINT_VAL + 1;
8     %ML0 := %ML0 + 1;
9
10     FOR IDX := 0 TO MAX_VALUE - 1 DO
11         mask := SHL(1, IDX);
12 //      LONG_ARRAY[IDX] := mask = (LINT_VAL AND mask);
13 //  END_FOR;
14
15     LWORD_TMP := 0;
16     FOR IDX := 0 TO MAX_VALUE - 1 DO
17         //MASK := MASK OR SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
18         mask := SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
19         LWORD_TMP := mask OR LWORD_TMP;
20     END_FOR;
```

14.4.4.6. 들여쓰기/내어쓰기

선택된 영역을 들여쓰기 또는 내어쓰기 합니다.

1) 들여쓰기

[순서]

1. 들여쓰기 할 영역을 선택합니다.

```
1
2 CLOCK_SOURCE := _T1S;
3
4 // LEFT rotate, FIND transition
5 IF (* _T1S XOR *) 0 = PREV_STATUS THEN
6 //IF (* _T1S XOR *) 0 = PREV_STATUS AND (PREV_STATUS XOR CLOCK_SOURCE) THEN
7     LINT_VAL := LINT_VAL + 1;
8     %ML0 := %ML0 + 1;
9
10    FOR IDX := 0 TO MAX_VALUE - 1 DO
11        mask := SHL(1, IDX);
12        LONG_ARRAY[IDX] := mask = (LINT_VAL AND mask);
13    END_FOR;
14
15    LWORD_TMP := 0;
16    FOR IDX := 0 TO MAX_VALUE - 1 DO
17        //MASK := MASK OR SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
18        mask := SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
19        LWORD_TMP := mask OR LWORD_TMP;
20    END_FOR;
```

2. TAB 키를 누릅니다.

```
1
2 CLOCK_SOURCE := _T1S;
3
4 // LEFT rotate, FIND transition
5 IF (* _T1S XOR *) 0 = PREV_STATUS THEN
6 //IF (* _T1S XOR *) 0 = PREV_STATUS AND (PREV_STATUS XOR CLOCK_SOURCE) THEN
7     LINT_VAL := LINT_VAL + 1;
8     %ML0 := %ML0 + 1;
9
10        FOR IDX := 0 TO MAX_VALUE - 1 DO
11            mask := SHL(1, IDX);
12            LONG_ARRAY[IDX] := mask = (LINT_VAL AND mask);
13        END_FOR;
14
15    LWORD_TMP := 0;
16    FOR IDX := 0 TO MAX_VALUE - 1 DO
17        //MASK := MASK OR SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
18        mask := SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
19        LWORD_TMP := mask OR LWORD_TMP;
20    END_FOR;
```

2) 내어쓰기

[순서]

1. 내어쓰기 할 영역을 선택합니다.

```
1
2 CLOCK_SOURCE := _T1S;
3
4 // LEFT rotate, FIND transition
5 IF (* _T1S XOR *) 0 = PREV_STATUS THEN
6 //IF (* _T1S XOR *) 0 = PREV_STATUS AND (PREV_STATUS XOR CLOCK_SOURCE) THEN
7     LINT_VAL := LINT_VAL + 1;
8     %ML0 := %ML0 + 1;
9
10        FOR IDX := 0 TO MAX_VALUE - 1 DO
11            mask := SHL(1, IDX);
12            LONG_ARRAY[IDX] := mask = (LINT_VAL AND mask);
13        END_FOR;
14
15    LWORD_TMP := 0;
16    FOR IDX := 0 TO MAX_VALUE - 1 DO
17        //MASK := MASK OR SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
18        mask := SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
19        LWORD_TMP := mask OR LWORD_TMP;
20    END_FOR;
```

2. Shift + TAB 키를 누릅니다.

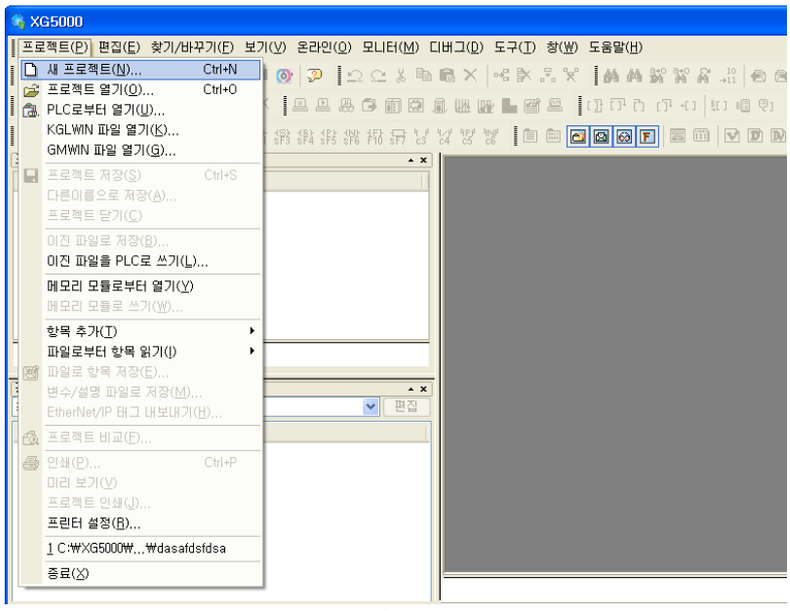
```
1
2 CLOCK_SOURCE := _T1S;
3
4 // LEFT rotate, FIND transition
5 IF (* _T1S XOR *) 0 = PREV_STATUS THEN
6 //IF (* _T1S XOR *) 0 = PREV_STATUS AND (PREV_STATUS XOR CLOCK_SOURCE) THEN
7     LINT_VAL := LINT_VAL + 1;
8     %ML0 := %ML0 + 1;
9
10    FOR IDX := 0 TO MAX_VALUE - 1 DO
11        mask := SHL(1, IDX);
12            LONG_ARRAY[IDX] := mask = (LINT_VAL AND mask);
13        END_FOR;
14
15    LWORD_TMP := 0;
16    FOR IDX := 0 TO MAX_VALUE - 1 DO
17        //MASK := MASK OR SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
18        mask := SHL(BOOL_TO_LWORD(LONG_ARRAY[IDX]), IDX);
19        LWORD_TMP := mask OR LWORD_TMP;
20    END_FOR;
```

14.5. 프로그래밍 가이드

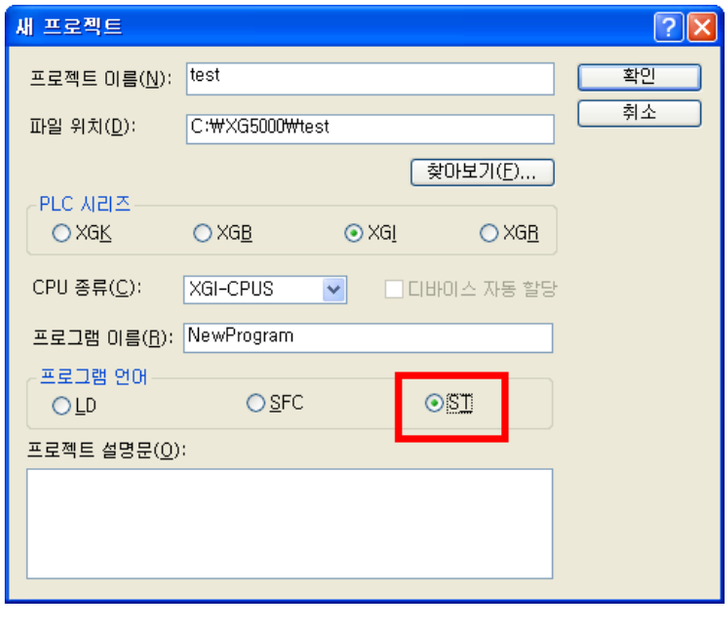
ST 프로그램의 입력에서 쓰기 및 모니터링까지의 일련의 기본 조작을 설명

1) ST 용 프로젝트 신규 작성
2) ST 프로그램에서 사용하는 변수 정의
3) ST 프로그램 작성
4) 작성한 ST 프로그램 PLC 에 쓰기 및 모니터링

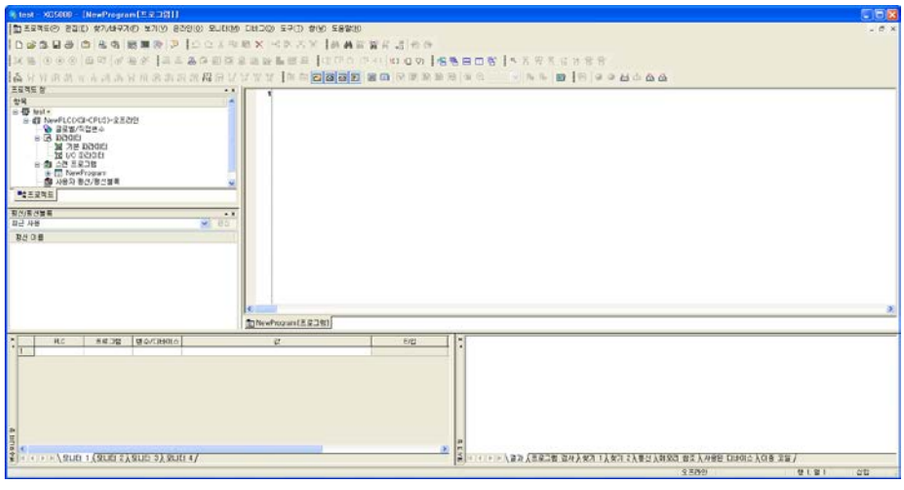
14.5.1. ST 용 프로그램 신규 생성



① 메뉴[프로젝트] -> [새 프로젝트]를 클릭



② 프로그램 언어 ST 설정 후 [확인]



③ ST 편집화면이 열립니다.

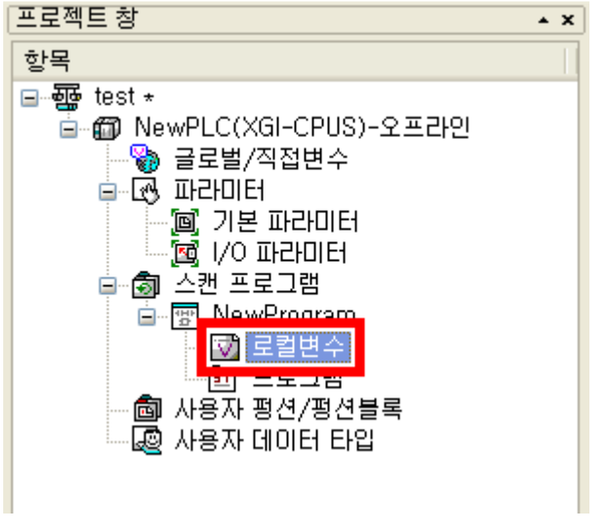
** ST 언어 사용 가능 CPU 기종

XGK (디바이스 자동 할당 체크시 사용가능)	XGB	XGI	XGR
XGK-CPUA	XGB-XECH	XGI-CPUE	XGR-CPUH
XGK-CPUE	XGB-XECS	XGI-CPUS	XGR-INC
XGK-CPUS	XGB-XECE	XGI-CPUH	
XGK-CPUH		XGI-CPUU	
XGK-CPUU		XGI-CPUU/D	

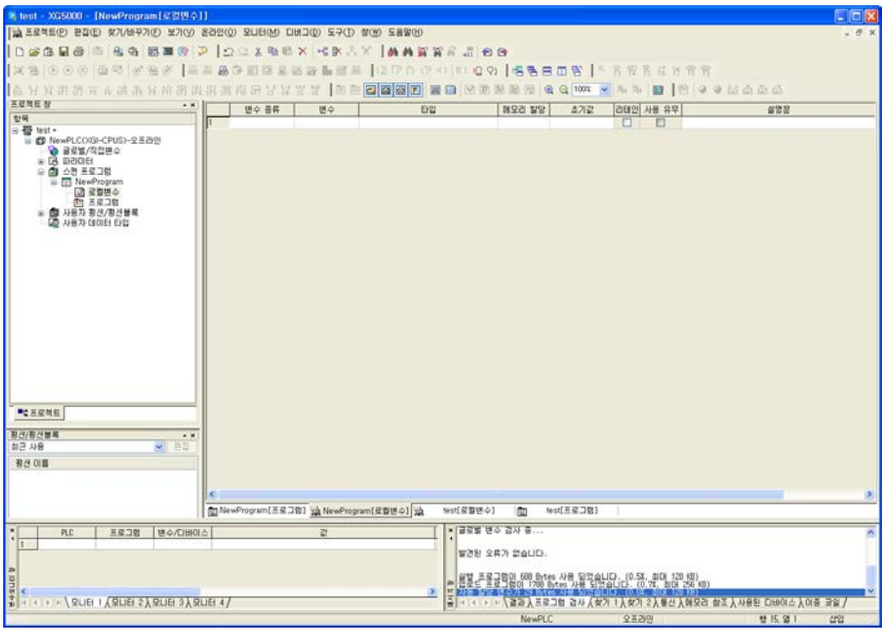
14.5.2. 변수 등록

ST 프로그램에서 변수를 사용하기 위해서는 사용할 변수를 명확하게 정의할 필요가 있으며, 정의되지 않은 변수를 사용한 프로그램을 변환하면 에러가 되어 프로그램을 작성할 수 없습니다.

변수에는 글로벌 변수와 로컬 변수의 2 종류가 있습니다. 글로벌 변수는 프로젝트 전체에서 사용할 수 있고, 로컬변수는 해당 프로그램에서만 사용할 수 있습니다. 여기에서는 이후에 입력할 프로그램에 사용할 로컬변수를 실제로 정의해 봅시다.



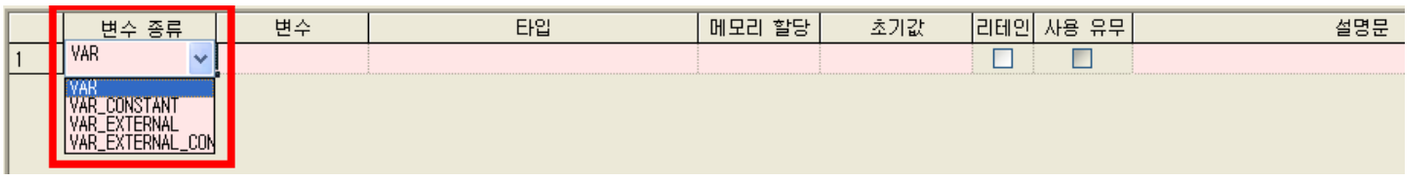
①[프로젝트 창] -> [스캔 프로그램] -> [NewProgram] -> [로컬변수 클릭]



②로컬변수 설정화면이 표시됩니다.

③ 변수 종류를 입력합니다.

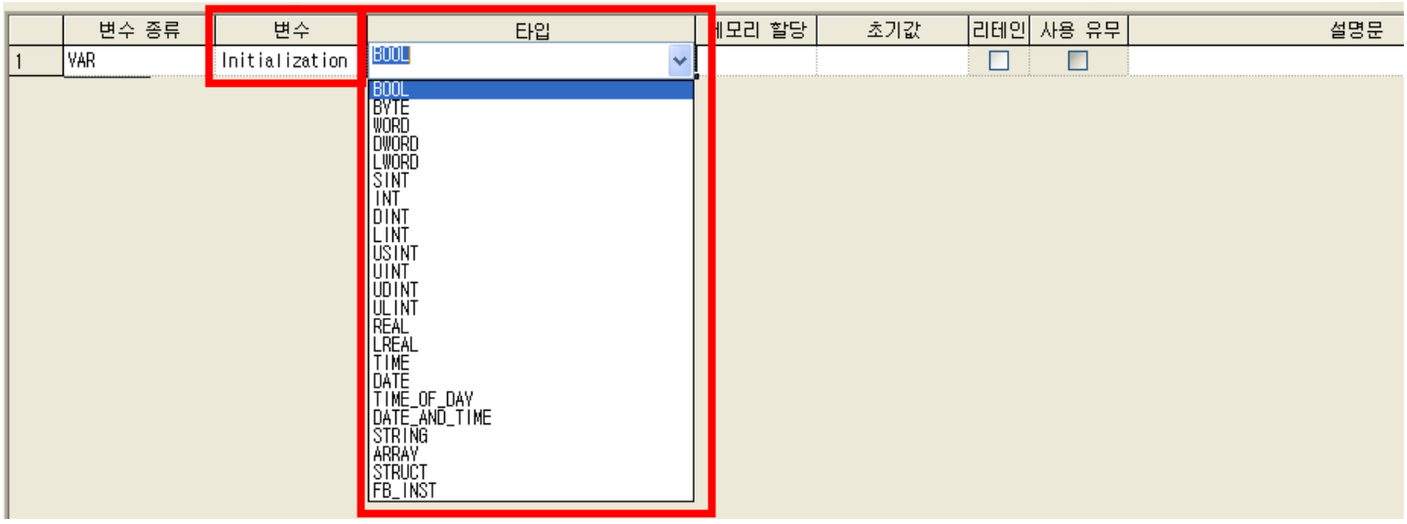
직접입력 또는 드롭다운 메뉴에서 선택합니다.



④ 변수의 이름과 타입을 입력합니다.

메모리할당은 입력하지 않을 경우 자동으로 할당되며, 특정 메모리를 사용하고자 할 경우 디바이스 주소를 입력합니다.

필요시 초기값과 설명문을 입력합니다.



⑤ 변수를 계속해서 입력할 때에는 [TAB] 키 또는 [마우스 우클릭 메뉴] -> [라인삽입]을 이용하여 변수를 추가 입력합니다.

	변수 종류	변수	타입	메모리 할당	초기값	리테인	사용 유무	설명문
1	VAR	Initialization	BOOL			☐	☑	the process is initialized
2						☐	☐	

⑥ 아래와 같이 변수를 입력하면 변수 등록이 완료됩니다.

	변수 종류	변수	타입	메모리 할당	초기값	리테인	사용 유무	설명문
1	VAR	Initialization	BOOL			☐	☑	the process is initialized
2	VAR	Defective	DINT			☐	☑	total number of defective goods
3	VAR	Good	DINT			☐	☑	total number of quality items
4	VAR	Yield	REAL			☐	☑	yield
5	VAR	Inspection	BOOL			☐	☑	inspection flag

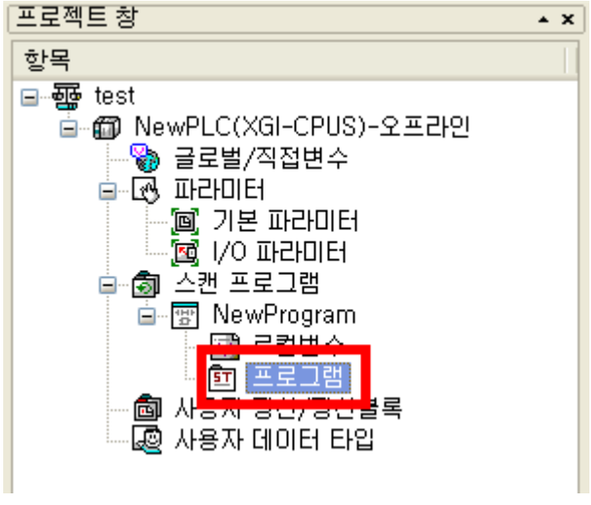
14.5.3. 프로그램 입력

1) ST 편집화면을 사용하여 텍스트 형식으로 자유롭게 입력할수 있습니다.
2) 정의된 변수, 제어구문, 코멘트는 입력하면 문자색이 바뀝니다.
3) 문자색이 바뀌지 않는 경우에는 오타 또는 변수 정의 여부를 확인하세요.

아래의 예제 프로그램을 입력해 봅시다.

```
IF Initialization THEN
    Good := 0; Defective := 0; Yield := 0.0;

ELSE
    IF Inspection THEN
        Good := Good +1;

    ELSE
        Defective := Defective + 1;
    END_IF;

    Yield:=DINT_TO_REAL(Good)/DINT_TO_REAL(Good+Defective);
END_IF;
```

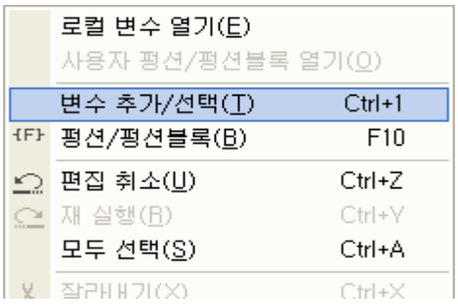
<예제 프로그램>



①[프로젝트 창] -> [스캔프로그램]
-> [프로그램]을 더블클릭 합니다.

② " IF" 를 입력합니다.

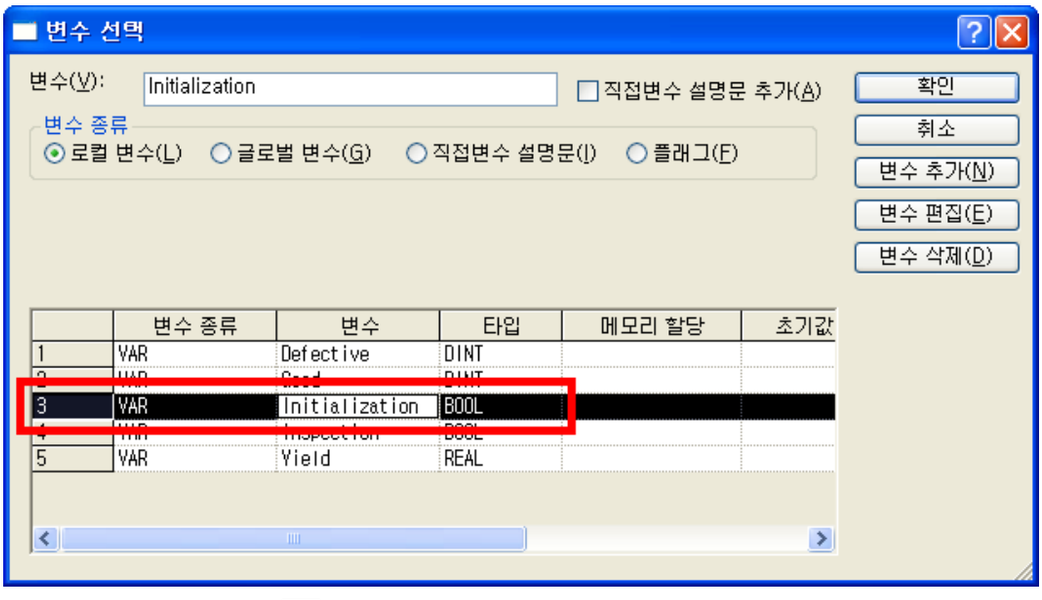
* 표현식은 소문자로 입력해도 대문자로 자동변환됩니다.



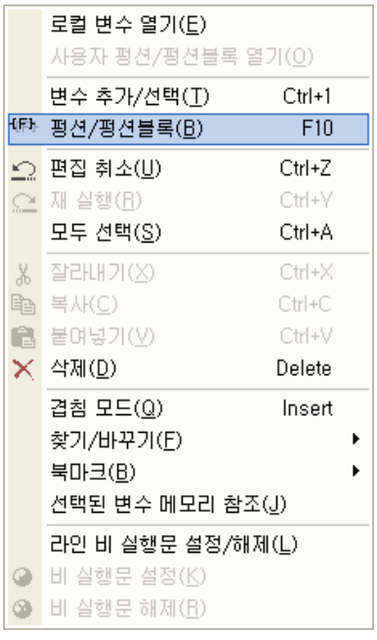
③[마우스 우클릭 메뉴] ->
[변수 추가/선택]을 클릭합니다.

④ 입력할 변수를 선택한 후 확인을 클릭합니다.

*ST 편집기에서 자동완성기능을 통한 변수 입력도 가능합니다. 단, 미리 등록된 변수에 한해서 자동완성기능이 적용됩니다.

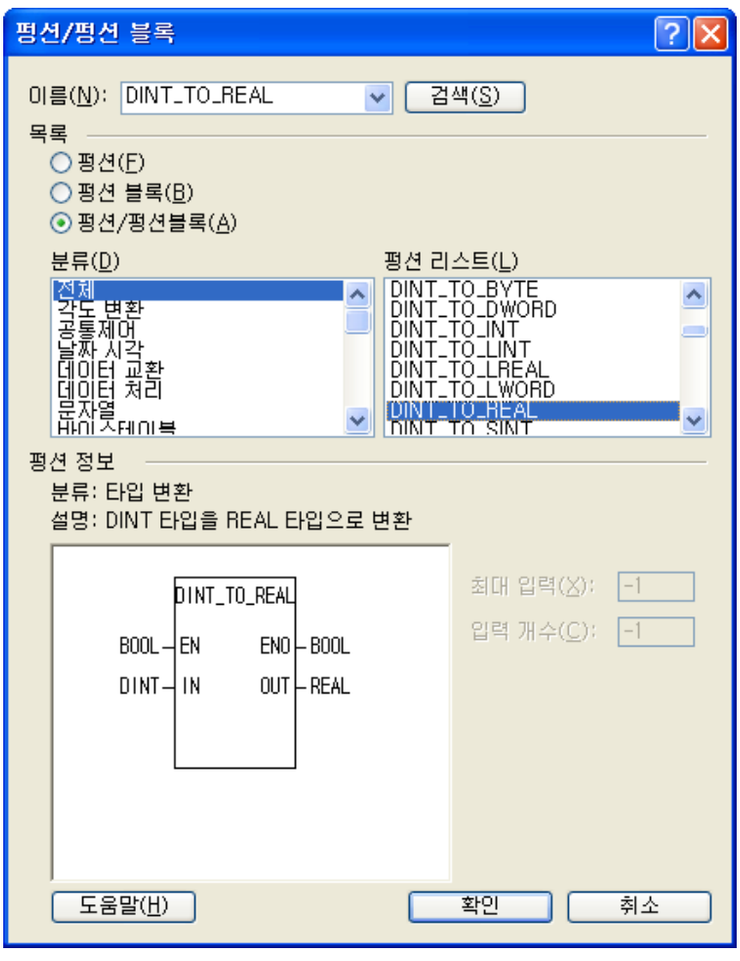


⑤변수가 입력되었습니다.



⑥이번엔 함수를 입력해봅시다.

[마우스 우클릭 메뉴] -> [펑션/펑션블록]을 클릭합니다.



⑦ 'DINT_TO_REAL' 을
선택후 확인을 누릅니다.

⑧ 아래와 같이 함수가 입력되었습니다.

NewProgram[프로그램]

```
1  IF Initialization THEN
2      Good := 0; Defective := 0; Yield := 0.0;
3
4  ELSE
5      IF Inspection THEN
6          Good := Good +1;
7
8      ELSE
9          Defective := Defective + 1;
10     END_IF;
11
12     (*REAL*):=DINT_TO_REAL(IN:=(*DINT*))
```

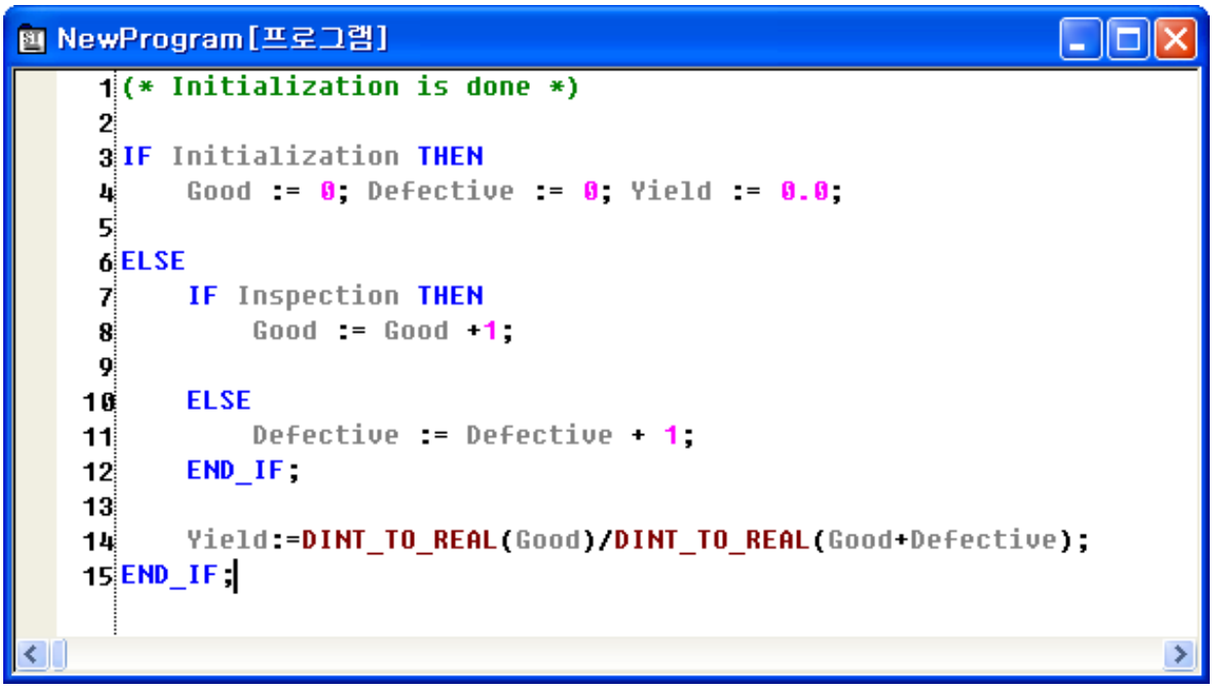
⑨ 표시된 함수인수의 타입을 참고하여 인수를 입력하고 완성시킵니다..

NewProgram[프로그램]

```
1  IF Initialization THEN
2      Good := 0; Defective := 0; Yield := 0.0;
3
4  ELSE
5      IF Inspection THEN
6          Good := Good +1;
7
8      ELSE
9          Defective := Defective + 1;
10     END_IF;
11
12     Yield:=DINT_TO_REAL(Good)/DINT_TO_REAL(Good+Defective);
13 END_IF;
```

⑩ 코멘트를 입력합니다. 코멘트는 프로그램의 동작에는 아무런 영향을 주지 않습니다.

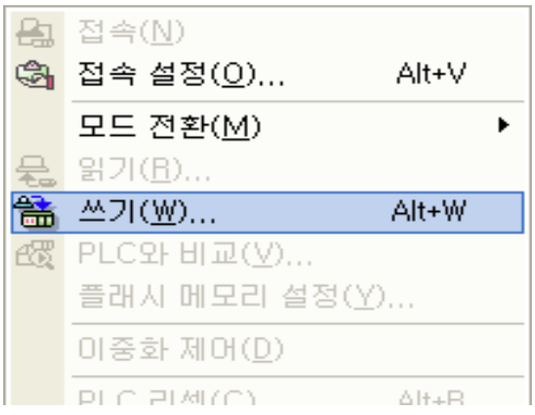
코멘트는 “(” 과 “*)” 사이에 입력합니다.



프로그램 입력이 완료되었습니다.

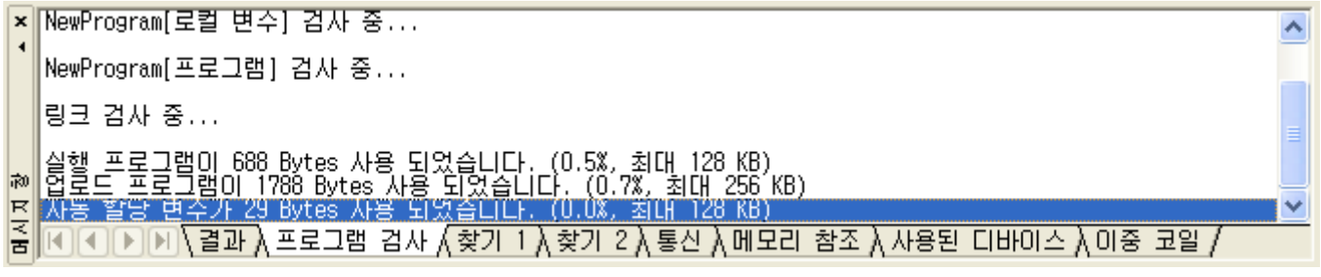
14.5.4. 프로그램 쓰기 및 모니터링

작성한 프로그램을 PLC 에 쓰는 과정과 모니터링 방법에 대해 설명합니다.

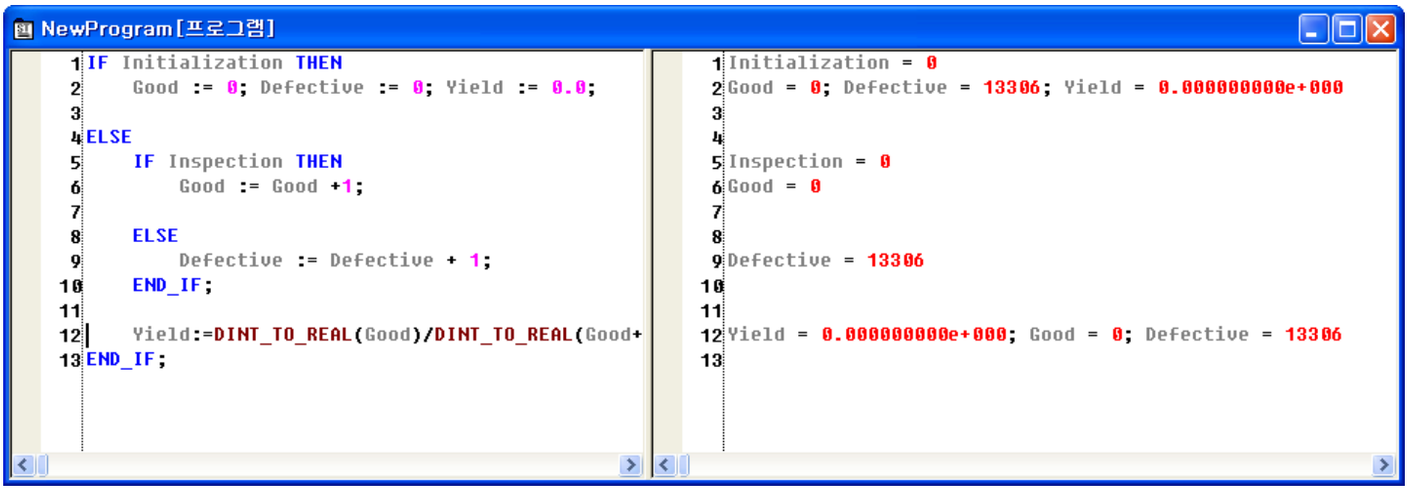


①[온라인] -> [쓰기] 를 클릭합니다.

② 프로그램 검사가 완료되었습니다.



③ 아래와 같이 우측에 변수값들이 모니터링 되는 것을 확인 할 수 있습니다.



14.6. 부록

14.6.1. 단축키

ST 편집은 키보드를 이용하여 입력, 복사, 붙여넣기, 잘라내기 등의 기능을 이용하여 프로그램을 작성할 수 있습니다.

다음의 단축키는 메뉴 [도구]-[단축키 설정]에서 변경할 수 있습니다.

동작	단축키	설명
변수 등록	Ctrl + 1	변수 등록창을 엽니다.
함수 입력	F10	함수 입력창을 엽니다.
복사	Ctrl + C	선택된 문자열을 복사합니다.
붙여넣기	Ctrl + V	복사된 문자열을 붙여 넣습니다.
삭제	Del	선택된 문자열을 삭제합니다.
잘라내기	Ctrl + X	선택된 문자열을 복사한 후 삭제합니다.
편집취소	Ctrl + Z	편집 전 상태로 되돌립니다.
재실행	Ctrl + Y	편집 전 상태에서 이전 편집된 상태로 되돌립니다.
모두선택	Ctrl + A	전체 문자열을 선택합니다.

다음의 단축키는 커서 이동에 관한 단축키입니다. 해당 단축키는 XG5000 에서 변경할 수 없습니다.

단축키	설명
Home	열의 시작으로 이동합니다.
Ctrl + Home	프로그램의 시작으로 이동합니다.
→	현재 커서를 오른쪽으로 한 칸 이동합니다.
←	현재 커서를 왼쪽으로 한 칸 이동합니다.
↑	현재 커서를 위쪽으로 한 칸 이동합니다.
↓	현재 커서를 아래쪽으로 한 칸 이동합니다.
End	열의 끝으로 이동합니다.
Page up	한 페이지 위로 이동합니다.
Page down	한 페이지 아래로 이동합니다
Ctrl + End	편집된 가장 마지막 줄로 이동합니다.
Ctrl + →	다음 단어의 시작 위치로 이동합니다.
Ctrl + ←	이전 단어의 시작 위치로 이동합니다.
Ctrl + Del	다음 단어의 시작 위치까지 삭제합니다.
Ctrl + BS	현재 단어의 시작 위치까지 삭제합니다.

단축키	설명
Shift + 이동	현재 커서 위치에서 이동할 위치까지 선택합니다.

알아두기

1. [도구]-[사용자 정의]-[도구모음]의 단축키 표현에서 s는 Shift 키를, c는 Ctrl 키를 a는 Alt 키를 표시합니다.
2. 편집 도구에서 설명한 단축키는 XG5000 에서 기본으로 제공하는 단축키를 기준으로 설명합니다.
3. 사용자 정의 단축키 설정은 제 2 장 기본사용법의 2.4 단축키 설정하기를 참고하시기 바랍니다.

14.6.2. 제한 사항

ST 프로그램 편집 시 다음과 같은 기능 제한이 있습니다.

항목	내용	제한사항
한 줄 최대 글자 수	한 줄에 입력될 수 있는 최대 문자 개수로 영문 2,048 개 한글 1,024 개 입니다.	2,048 개

알아두기

1. 하나의 스캔 프로그램 내에서는 하나의 언어 종류만 사용할 수 있습니다.
2. 스캔 프로그램 언어와 사용자 펑션/펑션 블록, SFC의 트랜지션, 액션 등은 각각 다른 언어를 사용할 수 있습니다.
3. 하나의 언어로 작성된 프로그램은 다른 종류의 언어로 변환할 수 없습니다.

부록 1 수치체계 및 데이터 구조

부1.1. 수치(데이터)의 표현

PLC CPU 에서는 모든 정보를 On 과 Off, 또는 '1' 과 '0' 의 상태로 기억하고 처리합니다. 따라서 수치 연산도 1 과 0 으로 처리된 수치, 즉 2 진수 (Binary number …. BIN)로 처리합니다. 한편, 일상 생활에서는 10 진수가 알기 쉽고 가장 널리 사용되고 있습니다. 그래서 PLC 에 수치를 Write 할 경우, 또는 PLC 의 수치정보를 Read 할 경우에는 10 진수에서 16 진수로, 16 진수에서 10 진수로 변환이 필요합니다. 여기에서는 10 진수와 2 진수, 16 진수, 2 진화 10 진수(BCD)의 표현과 그 상호관계에 대해 설명합니다.

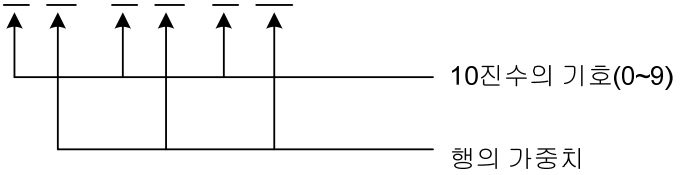
1) 10 진수(Decimal)

10 진수란 "0~9 의 종류의 기호를 사용하여 순서와 크기(량)를 표현하는 수" 를 말합니다.
그리고 0, 1, 2, 3, 4, …, 9 다음에 '10' 으로 자리올림하고 계속 진행됩니다.
예를 들면, 10 진수 153 을 행과 "행의 가중치" 란 측면에서 보면 아래와 같습니다.

153=100+50+3

=1*100+5*10+3*1

$=1*10^{2}+5*10^{1}+3*10^{0}$



2) 2 진수 (Binary …… Bin)

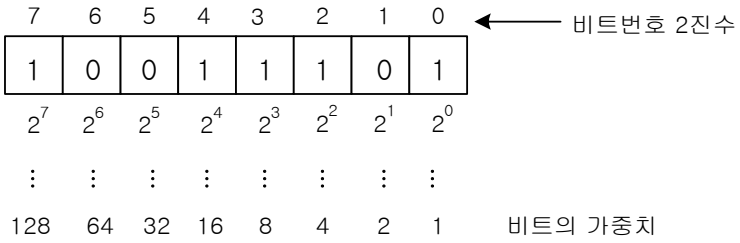
2 진수란 "0 과 1 의 두 종류 기호를 사용하여 순서와 크기를 나타내는 수" 를 말합니다. 그래서 0, 1 다음에 '10' 으로 자리올림을 하고, 계속 진행됩니다.
즉, 0,1 의 한 자리 수를 비트라고 합니다.

2 진수	10 진수
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
.....	.....

예를 들면 다음의 2 진수는 10 진수로 얼마나 되는지 생각해 봅시다.

“10011101”

10 진수에서 행번호와 행의 가중치를 고려하였듯이 우측부터 비트번호와 비트가중치를 붙여 봅시다.



10 진수와 같이 각 비트의 코드의 가중치의 곱의 합을 생각해 봅시다.

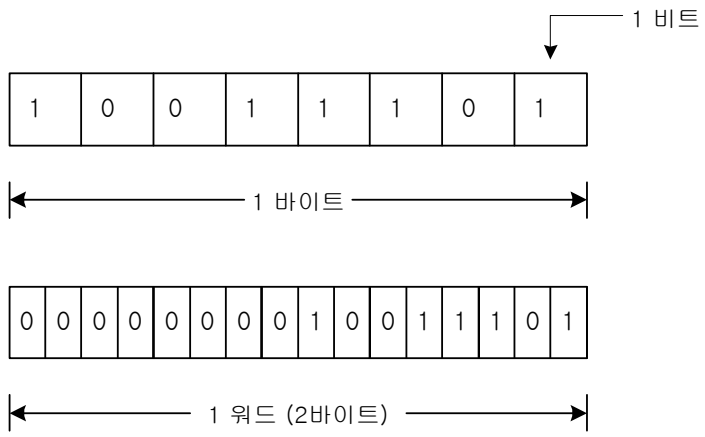
= 1 × 128 + 0 × 64 + 0 × 32 + 1 × 16 + 1 × 8 + 1 × 4 + 0 × 2 + 1 × 1

= 128 + 16 + 8 + 4 + 1

= 157

즉, 2 진수는 “코드가 1 인, 비트의 가중치를 가산한 것”이 10 진수로 되는 것입니다.

일반적으로 8 비트를 1 바이트, 16 비트(2 바이트)를 1 워드라 말합니다.

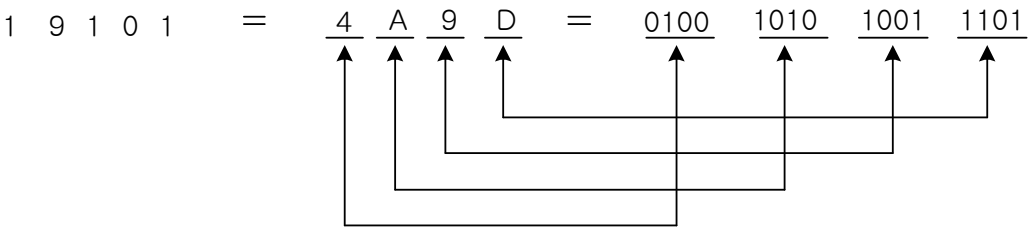


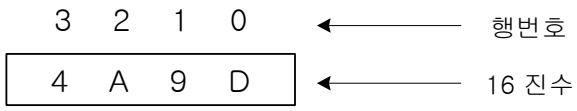
3) 16 진수 (Hexadecimal ····· HEX)

16 진수도 10 진수, 2 진수와 동일하게 생각하여 "0 ~ 9, A ~ F 의 종류의 기호를 사용하여 순서와 크기를 나타내는 수"를 말합니다.

그리고 0, 1, 2, ·····D, E, F 다음에 '10' 으로 자리올림을 하고 계속 진행됩니다.

10 진수	16 진수	2 진수
0	0	0
1	1	1
2	2	10
3	3	11
4	4	100
5	5	101
6	6	110
7	7	111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111
16	10	10000
17	11	10001
18	12	10010





= (4) × 163 + (A) × 162 + (9) × 161 + (D) × 160

= 4 × 4096 + 10 × 2568 + 9 × 16 + 13 × 1

= 19101

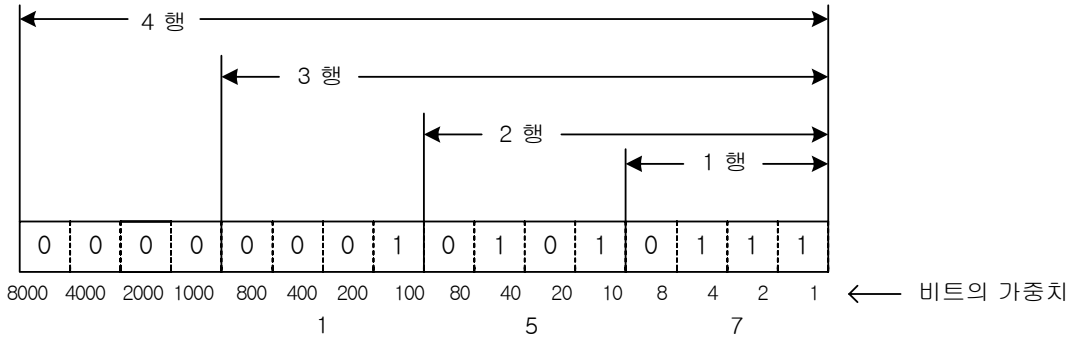
16 진수의 한자리는 2 진수의 4 비트로 대응됩니다.

4) 2 진화 10 진수 (Binary Coded Decimal ····· BCD)

2 진화 10 진수는 "10 진수의 각행의 숫자를 2 진수로 나타낸 수" 를 말합니다.

따라서, 2 진화 10 진수는 10 진수의 0 ~ 9,999 (4 행의 최대치)를 16 비트로 나타냅니다.

예를 들면, 10 진수의 157 는 다음과 같이 나타낼 수 있으며, 각 비트의 가중치는 다음과 같습니다.



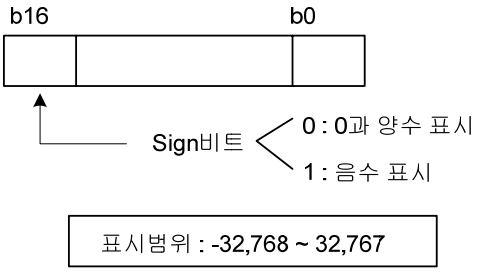
5) 수치 체계표

2 진화 10 진수 (Binary coded Decimal) BCD	2 진수 (Binary) BIN	10 진수 (Decimal)	16 진수 (Hexadecimal) H
00000000 00000000	00000000 00000000	0	0000
00000000 00000001	00000000 00000001	1	0001
00000000 00000010	00000000 00000010	2	0002
00000000 00000011	00000000 00000011	3	0003
00000000 00000100	00000000 00000100	4	0004
00000000 00000101	00000000 00000101	5	0005
00000000 00000100	00000000 00000100	6	0006
00000000 00000111	00000000 00000111	7	0007
00000000 00001000	00000000 00001000	8	0008
00000000 00001001	00000000 00001001	9	0009
00000000 00010000	00000000 00001010	10	000A
00000000 00010001	00000000 00001011	11	000B
00000000 00010010	00000000 00001100	12	000C
00000000 00010011	00000000 00001101	13	000D
00000000 00010100	00000000 00001110	14	000E
00000000 00010101	00000000 00001111	15	000F
00000000 00000110	00000000 00010000	16	0010
00000000 00000111	00000000 00010001	17	0011
00000000 00001000	00000000 00010010	18	0012
00000000 00001001	00000000 00010011	19	0013
00000000 00100000	00000000 00010100	20	0014
00000000 00100001	00000000 00010101	21	0015
00000000 00100010	00000000 00010110	22	0016
00000000 00100011	00000000 00010111	23	0017
00000001 00000000	00000000 01100100	100	0064
00000001 00100111	00000000 01111111	127	007F
00000010 01010101	00000000 11111111	255	00FF
00010000 00000000	00000000 11100000	1,000	03E8
00100000 01000111	00000000 11111111	2,047	07FF
01000000 10010101	00000000 11111111	4,095	0FFF
10011001 10011001	00000111 00001111	9,999	270F
	00100111 00010000	10,000	2710
	01111111 11111111	32,767	7FFF

부1.2. 정수표현

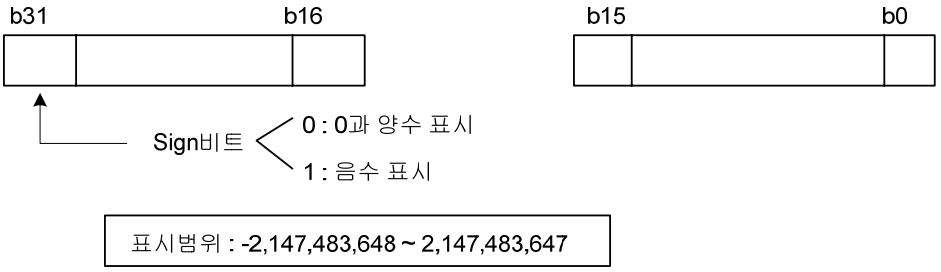
XGI/XGR 명령어에서는 음수체계연산(Signed)을 기본으로 합니다.
이때 정수표시는 최상위 비트(MSB)가 0이 되면 양수를 나타내고 1이면 음수로 나타나게 됩니다.
음수, 양수를 표시하는 최상위 비트를 Sign 비트라고 합니다.
16비트, 32비트에서는 MSB의 위치가 다르기 때문에 Sign비트 위치에 주의해야 합니다.

✸ 16비트 일 경우



표시범위 : -32,768 ~ 32,767

✸ 32비트 일 경우



표시범위 : -2,147,483,648 ~ 2,147,483,647

부1.3. 음수의 표현

예) - 0001을 표기하는 방법

(1) 음수번호를 땐 0001을 표기한다. (b15 = 1)

b15				b0
1	0	~	0	1

(2) (1) 의 결과를 반전시킨다. (b15 = 제외)

b15				b0
1	1	~	1	0

(3) (2) 의 결과에 +1을 한다.

b15							b0
1	1				~	1	1

-0001 = 16#FFFF

부록2 플래그 일람(XGI)

부2.1. 모드와 상태

예 약 변 수	데이터타입	내 용
_SYS_STATE	DWORD	PLC의 모드와 운전 상태를 표시합니다.
_RUN	BOOL	RUN 상태입니다.
_STOP	BOOL	STOP 상태입니다.
_ERROR	BOOL	ERROR 상태입니다.
_DEBUG	BOOL	DEBUG 상태입니다.
_LOCAL_CON	BOOL	로컬 컨트롤 모드입니다.
_REMOTE_CON	BOOL	리모트 컨트롤 모드입니다.
_RUN_EDIT_ST	BOOL	런중 수정 프로그램 다운로드 중입니다.
_RUN_EDIT_CHK	BOOL	런중 수정 내부 처리 중입니다.
_RUN_EDIT_DONE	BOOL	런중 수정 완료입니다.
_RUN_EDIT_NG	BOOL	런중 수정 비정상 완료
_CMOD_KEY	BOOL	키에 의해 운전모드가 변경 되었습니다.
_CMOD_LPADT	BOOL	로컬 PADT에 의해 운전모드가 변경 되었습니다.
_CMOD_RPADT	BOOL	리모트 PADT에 의해 운전모드가 변경 되었습니다.
_CMOD_RLINK	BOOL	리모트 통신 모듈에 의해 운전모드가 변경 되었습니다.
_FORCE_IN	BOOL	강제입력 상태입니다.
_FORCE_OUT	BOOL	강제출력 상태입니다.
_SKIP_ON	BOOL	입출력 SKIP이 실행 중입니다.
_EMASK_ON	BOOL	고장 마스크가 실행 중입니다.
_MON_ON	BOOL	모니터가 실행 중입니다.
_USTOP_ON	BOOL	STOP 펑션에 의해 STOP 되었습니다.
_ESTOP_ON	BOOL	ESTOP 펑션에 의해 STOP 되었습니다.
_INIT_RUN	BOOL	초기화 태스크가 수행 중입니다.
_PB1	BOOL	프로그램 코드1이 선택되었습니다.
_PB2	BOOL	프로그램 코드2가 선택되었습니다.
_RTC_WR	BOOL	RTC에 데이터 쓰고 읽어오기
_SCAN_WR	BOOL	스캔 값 초기화
_CHK_ANC_ERR	BOOL	외부기기에서 중고장 검출 요청
_CHK_ANC_WAR	BOOL	외부기기에서 경고장 검출 요청
_INIT_DONE	BOOL	초기화 태스크 수행 완료를 표시
_KEY	DWORD	로컬 키의 현재 상타를 나타냅니다.

부2.2. 시스템 에러

예 약 변 수	데이터타입	내 용
_CNF_ER	WORD	시스템의 중고장 상태를 보고합니다.
_AB_SD_ER	BOOL	비정상 운전으로 인하여 정지합니다.
_IO_TYER	BOOL	모듈 타입이 일치하지 않습니다.
_IO_DEER	BOOL	모듈이 착탈되었습니다.
_IO_TYER_N	WORD	모듈 타입 불일치 슬롯 넘버
_IO_DEER_N	WORD	모듈 착탈 슬롯 넘버
_FUSE_ER	BOOL	퓨즈 단선 에러
_FUSE_ER_N	WORD	퓨즈 단선 슬롯 번호
_FUSE_ERR	ARRAY[0..7] OF WORD	퓨즈 단선 에러 상세 정보(베이스 및 슬롯 번호)
_ANNUM_ER	BOOL	외부기기에 중고장이 검출되었습니다.
_BPRM_ER	BOOL	기본 파라미터에 이상이 있습니다.
_IOPRM_ER	BOOL	IO 구성 파라미터에 이상이 있습니다.
_SPPRM_ER	BOOL	특수 모듈 파라미터가 비정상입니다.
_CPPRM_ER	BOOL	통신 모듈 파라미터가 비정상입니다.
_PGM_ER	BOOL	프로그램에 에러가 있습니다.
_CODE_ER	BOOL	프로그램 코드에 에러가 있습니다.
_SWDT_ER	BOOL	시스템 워치독 타이머가 작동했습니다.
_BASE_POWER_ER	BOOL	베이스 전원에 이상이 있습니다.
_WDT_ER	BOOL	스캔 워치독 타이머가 작동했습니다.
_IO_TYERR	ARRAY[0..7] OF WORD	메인 및 증설 베이스 모듈 타입 에러
_IO_DEERR	ARRAY[0..7] OF WORD	메인 및 증설 베이스 모듈 착탈 에러

부2.3. 시스템 경고

예 약 변 수	데이터타입	내 용
_CNF_WAR	DWORD	시스템의 경고장 상태를 보고합니다.
_RTC_ER	BOOL	RTC 데이터에 이상이 있습니다.
_TASK_ER	BOOL	태스크가 충돌하고 있습니다.
_BAT_ER	BOOL	배터리 상태에 이상이 있습니다.
_ANNUM_WAR	BOOL	외부 기기의 경고장이 검출 되었습니다.
_BASE_INFO_ER	BOOL	베이스 정보 이상
_HS_WAR1	BOOL	고속 링크 - 파라미터 1 이상
_HS_WAR2	BOOL	고속 링크 - 파라미터 2 이상
_HS_WAR3	BOOL	고속 링크 - 파라미터 3 이상
_HS_WAR4	BOOL	고속 링크 - 파라미터 4 이상
_HS_WAR5	BOOL	고속 링크 - 파라미터 5 이상
_HS_WAR6	BOOL	고속 링크 - 파라미터 6 이상
_HS_WAR7	BOOL	고속 링크 - 파라미터 7 이상
_HS_WAR8	BOOL	고속 링크 - 파라미터 8 이상
_HS_WAR9	BOOL	고속 링크 - 파라미터 9 이상
_HS_WAR10	BOOL	고속 링크 - 파라미터 10 이상
_HS_WAR11	BOOL	고속 링크 - 파라미터 11 이상
_HS_WAR12	BOOL	고속 링크 - 파라미터 12 이상
_P2P_WAR1	BOOL	P2P - 파라미터 1 이상
_P2P_WAR2	BOOL	P2P - 파라미터 2 이상
_P2P_WAR3	BOOL	P2P - 파라미터 3 이상
_P2P_WAR4	BOOL	P2P - 파라미터 4 이상
_P2P_WAR5	BOOL	P2P - 파라미터 5 이상
_P2P_WAR6	BOOL	P2P - 파라미터 6 이상
_P2P_WAR7	BOOL	P2P - 파라미터 7 이상
_P2P_WAR8	BOOL	P2P - 파라미터 8 이상
_CONSTANT_ER	BOOL	고정주기 오류
_ANC_ERR	WORD	외부 기기의 중고장 정보를 표시
_ANC_WAR	WORD	외부 기기의 경고장 정보를 표시

부2.4. 사용자 플래그

예 약 변 수	데이터타입	내 용
_T20MS	BOOL	20ms 주기의 CLOCK 입니다.
_T100MS	BOOL	100ms 주기의 CLOCK 입니다.
_T200MS	BOOL	200ms 주기의 CLOCK 입니다.
_T1S	BOOL	1s 주기의 CLOCK 입니다.
_T2S	BOOL	2s 주기의 CLOCK 입니다.
_T10S	BOOL	10s 주기의 CLOCK 입니다.
_T20S	BOOL	20s 주기의 CLOCK 입니다.
_T60S	BOOL	60s 주기의 CLOCK 입니다.
_ON	BOOL	항상 On 상태인 비트입니다.
_OFF	BOOL	항상 Off 상태인 비트입니다.
_1ON	BOOL	첫 스캔만 On 상태인 비트입니다.
_1OFF	BOOL	첫 스캔만 Off 상태인 비트입니다.
_STOG	BOOL	매 스캔 반전됩니다.

부2.5. 연산 결과 플래그

예 약 변 수	데이터타입	내 용
_ERR	BOOL	연산 에러 플래그
_LER	BOOL	연산 에러시 1 스캔 동안 On
_ARY_IDX_ERR	BOOL	배열 인덱스 범위 초과 에러 플래그
_ARY_IDX_LER	BOOL	배열 인덱스 범위 초과 래치 에러 플래그
_ALL_OFF	BOOL	모든 출력이 Off 일 경우 On
_PUTGET_ERR	WORD	PUT/GET 에러
_PUTGET_NDR	WORD	PUT/GET 완료

부2.6. 시스템 운전 상태 정보

예 약 변 수	데이터타입	내 용
_CPU_TYPE	WORD	CPU 타입에 관한 정보를 알려줍니다.
_CPU_VER	WORD	CPU 버전을 표시합니다.

예 약 변 수	데이터타입	내 용
_OS_VER	DWORD	OS 버전을 표시합니다.
_OS_DATE	DWORD	OS 배포일을 표시합니다.
_SCAN_MAX	WORD	런 이래로 최대 스캔시간을 나타냅니다. 단위는 0.1ms 입니다.
_SCAN_MIN	WORD	런 이래로 최소 스캔시간을 나타냅니다. 단위는 0.1ms 입니다.
_SCAN_CUR	WORD	현재 스캔시간을 나타냅니다. 단위는 0.1ms 입니다.
_RTC_TIME	ARRARY[0..7] OF BYTE	PLC 의 현재시각 데이터입니다.
_RTC_TIME[0]	BYTE	현재시각의 [년도] 데이터입니다.
_RTC_TIME[1]	BYTE	현재시각의 [월] 데이터입니다.
_RTC_TIME[2]	BYTE	현재시각의 [일] 데이터입니다.
_RTC_TIME[3]	BYTE	현재시각의 [시] 데이터입니다.
_RTC_TIME[4]	BYTE	현재시각의 [분] 데이터입니다.
_RTC_TIME[5]	BYTE	현재시각의 [초] 데이터입니다.
_RTC_TIME[6]	BYTE	현재시각의 [요일] 데이터입니다.
_RTC_TIME[7]	BYTE	현재시각의 [년대] 데이터입니다.
_RTC_TIME_USER	ARRARY[0..7] OF BYTE	설정하고자 하는 시각 데이터입니다.
_RTC_TIME_USER[0]	BYTE	설정하고자 하는 시각 [년도] 데이터입니다.
_RTC_TIME_USER[1]	BYTE	설정하고자 하는 시각 [월] 데이터입니다.
_RTC_TIME_USER[2]	BYTE	설정하고자 하는 시각 [일] 데이터입니다.
_RTC_TIME_USER[3]	BYTE	설정하고자 하는 시각 [시] 데이터입니다.
_RTC_TIME_USER[4]	BYTE	설정하고자 하는 시각 [분] 데이터입니다.
_RTC_TIME_USER[5]	BYTE	설정하고자 하는 시각 [초] 데이터입니다.
_RTC_TIME_USER[6]	BYTE	설정하고자 하는 시각 [요일] 데이터입니다.
_RTC_TIME_USER[7]	BYTE	설정하고자 하는 시각 [년대] 데이터입니다.
_RTC_DATE	WORD	RTC 의 현재 날짜
_RTC_WEEK	WORD	RTC 의 현재 요일
_RTC_TOD	DWORD	RTC 의 현재 시간 (ms 단위)
_BASE_INFO	ARRARY[0..7] OF WORD	메인 및 증설 베이스 슬롯 정보
_RBANK_NUM	WORD	현재 사용중인 블록 번호
_AC_F_CNT	WORD	순시 정전 발생 횟수를 알려줍니다.
_FALS_NUM	WORD	FALS 의 번호를 표시합니다.

부2.7. 고속링크 플래그 (* = 0~12, *** = 000~127)

예 약 변 수	데이터타입	내 용
_HS*_RLINK	BOOL	고속 링크 *번의 모든 국 정상 동작
_HS*_LTRBL	BOOL	_HS*RLINK ON 이후 비정상 상태 표시
_HS*_STATE***	BOOL	고속링크 *번의 ***번 블록의 종합적 상태표시
_HS*_MOD***	BOOL	고속링크 *번 ***번 블록 국의 런 운전 모드
_HS*_TRX***	BOOL	고속링크 *번 ***번 블록 국과 정상 통신 표시
_HS*_ERR***	BOOL	고속링크 *번 ***번 블록 국의 운전 에러 모드
_HS*_SETBLOCK***	BOOL	고속링크 *번 ***번 블록 설정 표시

부2.8. P2P 플래그 (* = 0 ~ 8, ** = 0 ~ 63)

예 약 변 수	데이터타입	내 용
_P2P*_NDR**	BOOL	P2P *번 **번 블록 서비스 정상완료
_P2P*_ERR**	BOOL	P2P *번 **번 블록 서비스 비정상완료
_P2P*_STATUS**	WORD	P2P *번 **번 블록 서비스 비정상완료시 에러코드
_P2P*_SVCCNT**	DWORD	P2P *번 **번 블록 서비스 정상 수행 횟수
_P2P*_ERRCNT**	DWORD	P2P *번 **번 블록 서비스 비정상 수행 횟수

부2.9. PID 플래그 (* = 0 ~ 7, ** = 0 ~ 31)

예 약 변 수	데이터타입	내 용
_PID*_MAN	DWORD	PID 출력 선택(0:자동 ,1:수동) - 블록*
PID***MAN	BOOL	PID 출력 선택(0:자동 ,1:수동) - 블록* 루프**
_PID*_PAUSE	DWORD	PID 일시정지 (0:STOP/RUN ,1:PAUSE) - 블록*
PID***PAUSE	BOOL	PID 일시정지 (0:STOP/RUN ,1:PAUSE) - 블록* 루프**
_PID*_REV	DWORD	PID 동작 선택(0:정 ,1:역) - 블록*
PID***REV	BOOL	PID 동작 선택(0:정 ,1:역) - 블록* 루프**
_PID*_AW2D	DWORD	PID Anti Wind-up2 금지(0:동작 ,1:금지) - 블록*
PID***AW2D	BOOL	PID Anti Wind-up2 금지(0:동작 ,1:금지) - 블록* 루프**
_PID*_REM_RUN	DWORD	PID 리모트(HMI) 실행비트 (0:STOP ,1:RUN) - 블록*
PID***REM_RUN	DWORD	PID 리모트(HMI) 실행비트 (0:STOP ,1:RUN) - 블록* 루프**
_PID*_P_on_PV	DWORD	PID 비례(P) 계산 소스 선택 (0:ERR, 1:PV) - 블록*

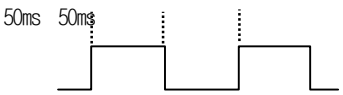
예 약 변 수	데이터타입	내 용
PID***P_on_PV	BOOL	PID 비례(P) 계산 소스 선택 (0:ERR, 1:PV) - 블록* 루프**
_PID*_D_on_ERR	DWORD	PID 미분(D) 계산 소스 선택 (0:PV, 1:ERR) - 블록*
PID***D_on_ERR	BOOL	PID 미분(D) 계산 소스 선택 (0:PV, 1:ERR) - 블록* 루프**
_PID*_AT_EN	DWORD	PID 오토튜닝 설정 (0:Disable, 1:Enable) - 블록*
PID***AT_EN	BOOL	PID 오토튜닝 설정 (0:Disable, 1:Enable) - 블록* 루프**
_PID*_MV_BMPL	DWORD	PID 모드 전환(A/M)시 MV 비충격 변환 설정 (0:Disable, 1:Enable) - 블록*
PID***MV_BMPL	BOOL	PID 모드 전환(A/M)시 MV 평활 설정 (0:Disable, 1:Enable) - 블록* 루프**
PID***SV	INT	PID 목표값 (SV) - 블록* 루프**
PID***T_s	WORD	PID 연산 주기 (T_s)[0.1msec] - 블록* 루프**
PID***K_p	REAL	PID P - 상수 (K_p) - 블록* 루프**
PID***T_i	REAL	PID I - 상수 (T_i)[sec] - 블록* 루프**
PID***T_d	REAL	PID D - 상수 (T_d)[sec] - 블록* 루프**
PID***d_PV_max	WORD	PID PV 변화량 제한 - 블록* 루프**
PID***d_MV_max	WORD	PID MV 변화량 제한 - 블록* 루프**
PID***MV_max	INT	PID MV 최대값 제한 - 블록* 루프**
PID***MV_min	INT	PID MV 최소값 제한 - 블록* 루프**
PID***MV_man	INT	PID 수동 출력 (MV_man) - 블록* 루프**
PID***STATE	WORD	PID State - 블록* 루프**
PID***ALARM0	BOOL	PID Alarm 0 (1:T_s 설정이 작음) - 블록* 루프**
PID***ALARM1	BOOL	PID Alarm 1 (1:K_p 가 0 임) - 블록* 루프**
PID***ALARM2	BOOL	PID Alarm 2 (1:PV 변화량 제한됨) - 블록* 루프**
PID***ALARM3	BOOL	PID Alarm 3 (1:MV 변화량 제한됨) - 블록* 루프**
PID***ALARM4	BOOL	PID Alarm 4 (1:MV 최대값 제한됨) - 블록* 루프**
PID***ALARM5	BOOL	PID Alarm 5 (1:MV 최소값 제한됨) - 블록* 루프**
PID***ALARM6	BOOL	PID Alarm 6 (1:AT 비정상 취소 상태) - 블록* 루프**
PID***ALARM7	BOOL	PID Alarm 7 - 블록* 루프**
PID***STATE0	BOOL	PID State 0 (0:PID_STOP, 1:PID_RUN) - 블록* 루프**
PID***STATE1	BOOL	PID State 1 (0:AT_STOP, 1:AT_RUN) - 블록* 루프**
PID***STATE2	BOOL	PID State 2 (0:AT_UNDONE, 1:DONE) - 블록* 루프**
PID***STATE3	BOOL	PID State 3 (0:REM_STOP, 1:REM_RUN) - 블록* 루프**
PID***STATE4	BOOL	PID State 4 (0:AUTO_OUT, 1:MAN_OUT) - 블록* 루프**
PID***STATE5	BOOL	PID State 5 (0:CAS_STOP, 1:CAS_RUN) - 블록* 루프**
PID***STATE6	BOOL	PID State 6 (0:SLV/SINGLE, 1:CAS_MST) - 블록* 루프**
PID***STATE7	BOOL	PID State 7 (0:AW_STOP, 1:AW_ACT) - 블록* 루프**

예 약 변 수	데이터타입	내 용
PID***PV	INT	PID 현재값 (PV) - 블록* 루프**
PID***PV_old	INT	PID 이전값 (PV_old) - 블록* 루프**
PID***MV	INT	PID 출력값 (MV) - 블록* 루프**
PID***MV_BMPL_val	INT	PID 비충격 동작 메모리 (사용자 설정 금지) - 블록* 루프**
PID***ERR	DINT	PID 제어 에러값 - 블록* 루프**
PID***MV_p	REAL	PID 출력값 P 성분 - 블록* 루프**
PID***MV_i	REAL	PID 출력값 I 성분 - 블록* 루프**
PID***MV_d	REAL	PID 출력값 D 성분 - 블록* 루프**
PID***DB_W	WORD	PID 데드밴드 설정 (안정화 후 동작) - 블록* 루프**
PID***Td_lag	WORD	PID 미분 함수 LAG 필터 - 블록* 루프**
PID***AT_HYS_val	WORD	PID 오토튜닝 히스테리시스 설정 - 블록* 루프**
PID***AT_SV	INT	PID 오토튜닝 시 SV 설정 - 블록* 루프**
PID***AT_step	WORD	PID 오토튜닝 상태 표시 (사용자 설정 금지) - 블록* 루프**
PID***INT_MEM	WORD	PID 내부 메모리 (사용자 설정 금지) - 블록* 루프**

부록 3. 플래그 일람

부3.1. 사용자 플래그

1. 사용자 플래그

주소	플래그명	타입	쓰기가능	내 용	설 명
%FX6144	_T20MS	BOOL	_	20ms 클럭	사용자 프로그램에서 사용할 수 있는 클럭신호로 반주기 마다 On/Off 반전됩니다. 스캔종료 후에 신호반전을 처리하므로, 프로그램수행 시간에 따라 클럭신호가 지연 또는 왜곡될 수 있으므로, 스캔시간보다 충분히 긴 클럭을 사용하여야 합니다. 클럭신호는 초기화 프로그램 시작 시, 스캔 프로그램 시작 시에 Off 에서 시작합니다. _T100ms 클럭 예 
%FX6145	_T100MS	BOOL	_	100ms 클럭	
%FX6146	_T200MS	BOOL	_	200ms 클럭	
%FX6147	_T1S	BOOL	_	1 초 클럭	
%FX6148	_T2S	BOOL	_	2 초 클럭	
%FX6149	_T10S	BOOL	_	10 초 클럭	
%FX6150	_T20S	BOOL	_	20 초 클럭	
%FX6151	_T60S	BOOL	_	60 초 클럭	
%FX6153	_ON	BOOL	_	상시 On	사용자 프로그램 작성시 사용할 수 있는 상시 On 플래그
%FX6154	_OFF	BOOL	_	상시 Off	사용자 프로그램 작성시 사용할 수 있는 상시 Off 플래그
%FX6155	_10N	BOOL	_	첫 스캔 On	운전시작 후 첫 스캔 동안만 On 되는 플래그
%FX6156	_10FF	BOOL	_	첫 스캔 Off	운전시작 후 첫 스캔 동안만 Off 되는 플래그
%FX6157	_STOG	BOOL	_	스캔 반전 (scan toggle)	사용자 프로그램 수행시 매 스캔마다 On/Off 반전되는 플래그(첫 스캔 On)
%FX6163	_ALL_OFF	BOOL	-	전 출력 Off	모든 출력이 off 일 경우 On
%FX30720	_RTC_WR	BOOL	가능	RTC 에 데이터 쓰기	RTC 에 데이터 쓰고 읽어오기
%FX30721	_SCAN_WR	BOOL	가능	스캔 값 초기화	스캔 값 초기화
%FX30722	_CHK_ANC_ERR	BOOL	가능	외부 기기 중고장 검출 요청	사용자 프로그램에 의해 외부기기 중고장(에러)을 검출 요청하는 플래그
%FX30723	_CHK_ANC_WAR	BOOL	가능	외부 기기 경고장 검출 요청	사용자 프로그램에 의해 외부기기 경고장(경고)을 검출 요청하는 플래그
%FX30724	_MASTER_CHG	BOOL	가능	마스터/스탠바이 전환	마스터/스탠바이를 전환하고자 할 때 사용할 수 있는 플래그
%FW3860	_RTC_TIME_USER	ARRAY[0..7] OF BYTE	가능	설정하고자 하는 시간	사용자가 시간을 설정하는 플래그 (년도,월,일,시,분,초,요일,년대 설정 가능)

부3.2. 시스템 에러 대표 플래그

1. 마스터 CPU 시스템 에러 대표 플래그

주소	플래그명	타입	BIT 위치	내 용	설 명
%FD65	_CNF_ER	DWORD	대표 키워드	시스템의 에러(중고장)	아래와 같은 운전중지 고장관련 에러 플래그들을 일괄 취급합니다.
%FX2081	_IO_TYER	BOOL	BIT 1	모듈 타입 불일치 에러	각 슬롯의 I/O 구성 파라미터와 실제 장착모듈의 구성이 서로 다른 경우 또는 특정 모듈이 장착될 수 없는 슬롯에 장착된 경우 이를 검출하여 표시하는 대표 플래그 (_IO_TYER_N, _IO_TYERR[n] 참조)
%FX2082	_IO_DEER	BOOL	BIT 2	모듈 착탈 에러	운전 중 각 슬롯의 모듈 구성이 달라질 경우 이를 검출하여 표시하는 대표 플래그 (_IO_DEER_N,_IO_DEERR[n] 참조]
%FX2083	_FUSE_ER	BOOL	BIT 3	퓨즈 단선 에러	각 슬롯의 모듈 중 Fuse 가 부착된 모듈의 퓨즈가 단선된 경우 이를 검출 하여 표시하는 대표 플래그 (_FUSE_ER_N, _FUSE_ERR[n] 참조)
%FX2086	_ANNUM_ER	BOOL	BIT 6	외부기기의 중고장 검출에러	사용자 프로그램에 의해 외부기기의 중고장을 검출하여 _ANC_ERR[n]에 기록한 경우 고장검출의 발생을 표시하는 대표 플래그
%FX2088	_BPRM_ER	BOOL	BIT 8	기본파라미터 이상	기본 파라미터가 CPU 기종과 맞지 않게 설정된 경우 발생
%FX2089	_IOPRM_ER	BOOL	BIT 9	I/O 파라미터 이상	IO 구성 파라미터에 이상이 있습니다.
%FX2090	_SPPRM_ER	BOOL	BIT 10	특수모듈 파라미터 이상	특수 모듈 파라미터가 비정상
%FX2091	_CPPRM_ER	BOOL	BIT 11	통신모듈 파라미터 이상	통신 모듈 파라미터가 비정상
%FX2092	_PGM_ER	BOOL	BIT 12	프로그램 에러	사용자가 작성한 프로그램에 이상이 발생한 경우
%FX2093	_CODE_ER	BOOL	BIT 13	프로그램 코드 에러	사용자 프로그램 수행 중 해독할 수 없는 명령을 만났을 때 발생하는 에러
%FX2094	_SWDT_ER	BOOL	BIT 14	CPU 비정상 종료	CPU 가 비정상 종료로 저장된 프로그램의 파괴된 경우 또는 프로그램 수행이 불가능한 에러
%FX2095	_BASE_POWER_ER	BOOL	BIT 15	베이스 전원이상	베이스 전원이 Off 상태 또는 전원모듈 불량일 때 발생
%FX2096	_WDT_ER	BOOL	BIT 16	스캔 워치독 에러	프로그램의 스캔 타임이 파라미터에 의해 지정한 스캔지연 감시시간(Scan Watchdog Time)을 초과했을 때 발생하는 에러
%FX2097	_BASE_INFO_ER	BOOL	BIT 17	베이스 정보 이상	기본베이스 정보가 비정상일 경우 발생
%FX2102	_BASE_DEER	BOOL	BIT 22	증설 베이스 착탈 에러	증설 베이스가 착탈 됐을 경우 발생
%FX2103	_DUPL_PRM_ER	BOOL	BIT 23	이중화 파라미터 이상 에러	이중화 파라미터가 비정상
%FX2104	_INSTALL_ER	BOOL	BIT 24	모듈 장착 위치 에러	메인 베이스에 장착할 수 없는 모듈을 장착했거나 증설 베이스에 장착할 수 없는 모듈을 장착 했을 때 발생
%FX2105	_BASE_ID_ER	BOOL	BIT 25	증설 베이스 중복 설정 에러	증설 베이스 번호가 중복설정 되었을 경우 발생
%FX2106	_DUPL_SYNC_ER	BOOL	BIT 26	이중화 운전 동기에러	마스터, 스탠바이 CPU 간에 이중화 동기가 정상적으로 이루어 지지 않을 때 발생합니다.

주소	플래그명	타입	BIT 위치	내 용	설 명
%FX2107	_AB_SIDEKEY_ER	BOOL	BIT 27	A/B SIDE 키 중복설정 에러	마스터,스탠바이 CPU 의 A,B 사이드 키가 동일하게 설정 되었을 경우 발생합니다. 마스터,스탠바이 CPU 의 A,B 사이드 키는 다르게 설정되어야 합니다.

2. 스탠바이 CPU 시스템 에러 대표 플래그

주소	플래그명	타입	BIT 위치	내 용	설 명
%FD129	_SB_CNF_ER	DWORD	대표 키워드	시스템의 에러 (중고장)	아래와 같은 운전중지 고장관련 에러 플래그들을 일괄 취급합니다.
%FX4129	_SB_IO_TYER	BOOL	BIT 1	모듈 타입 불일치 에러	각 슬롯의 I/O 구성 파라미터와 실제 장착모듈의 구성이 서로 다른 경우 또는 특정 모듈이 장착될 수 없는 슬롯에 장착된 경우 이를 검출하여 표시하는 대표 플래그 (_SB_IO_TYER_N, _SB_IO_TYERR 참조)
%FX4130	_SB_IO_DEER	BOOL	BIT 2	모듈 착탈 에러	운전 중 각 슬롯의 모듈 구성이 달라질 경우 이를 검출하여 표시하는 대표 플래그 (_SB_IO_DEER_N,_SB_IO_DEERR 참조]
%FX4131	_SB_FUSE_ER	BOOL	BIT 3	퓨즈 단선 에러	각 슬롯의 모듈 중 Fuse 가 부착된 모듈의 퓨즈가 단선된 경우 이를 검출 하여 표시하는 대표 플래그
%FX4134	_SB_ANNUM_ER	BOOL	BIT 6	외부기기의 중고장 검출에러	사용자 프로그램에 의해 외부기기의 중고장을 검출하여 _ANC_ERR[n]에 기록한 경우 고장검출의 발생을 표시하는 대표 플래그
%FX4136	_SB_BPRM_ER	BOOL	BIT 8	기본파라미터 이상	기본 파라미터가 CPU 기종과 맞지 않게 설정된 경우 발생
%FX4137	_SB_IOPRM_ER	BOOL	BIT 9	I/O 파라미터 이상	IO 구성 파라미터에 이상이 있습니다.
%FX4138	_SB_SPPRM_ER	BOOL	BIT 10	특수모듈 파라미터 이상	특수 모듈 파라미터가 비정상
%FX4139	_SB_CPPRM_ER	BOOL	BIT 11	통신모듈 파라미터 이상	통신 모듈 파라미터가 비정상
%FX4141	_SB_CODE_ER	BOOL	BIT 13	프로그램 코드 에러	사용자 프로그램 수행 중 해독할 수 없는 명령을 만났을 때 발생하는 에러
%FX4142	_SB_SWDT_ER	BOOL	BIT 14	CPU 비정상 종료	CPU 가 비정상 종료로 저장된 프로그램의 파괴된 경우 또는 프로그램 수행이 불가능한 에러
%FX4143	_SB_BASE_POWER_ER	BOOL	BIT 15	베이스 전원이상	베이스 전원이 Off 상태 또는 전원모듈 불량일 때 발생
%FX4144	_SB_WDT_ER	BOOL	BIT 16	스캔 워치독 에러	프로그램의 스캔 타임이 파라미터에 의해 지정한 스캔지연 감시시간 (Scan Watchdog Time)을 초과했을 때 발생하는 에러
%FX4145	_SB_BASE_INFO_ER	BOOL	BIT 17	베이스 정보 이상	기본베이스 정보가 비정상일 경우 발생
%FX4150	_SB_BASE_DEER	BOOL	BIT 22	증설 베이스 착탈 에러	증설 베이스가 착탈 됐을 경우 발생
%FX4151	_SB_DUPL_PRM_ER	BOOL	BIT 23	이중화 파라미터 이상 에러	이중화 파라미터가 비정상
%FX4152	_SB_INSTALL_ER	BOOL	BIT 24	모듈 장착 위치 에러	메인 베이스에 장착할 수 없는 모듈을 장착했거나 증설 베이스에 장착할 수 없는 모듈을 장착 했을 때 발생

주소	플래그명	타입	BIT 위치	내 용	설 명
%FX4153	_SB_BASE_ID_ER	BOOL	BIT 25	증설 베이스 중복 설정 에러	증설 베이스 번호가 중복설정 되었을 경우 발생
%FX4154	_SB_DUPL_SYNC_ER	BOOL	BIT 26	이중화 운전 동기에러	마스터, 스탠바이 CPU 간에 이중화 동기가 정상적으로 이루어 지지 않을 때 발생합니다.
%FX4156	_SB_CPU_RUN_ER	BOOL	BIT 28	스탠바이 CPU 런 에러	마스터 CPU 의 에러로 인해 스탠바이 CPU 가 이중화 운전 참여에 실패했을 경우 발생

부3.3. 시스템 에러 상세 플래그

1. 마스터 CPU 시스템 에러 상세 플래그

주소	플래그명	타입	쓰기 가능	내 용	설 명
%FW424	_IO_TYERR	ARRAY[0..31] OF WORD	-	모듈 타입 불일치 에러	모듈 타입 불일치 에러가 발생한 해당 베이스 및 슬롯을 표시
%FW456	_IO_DEERR	ARRAY[0..31] OF WORD	-	모듈 착탈 에러	모듈 착탈 에러가 발생한 해당 베이스 및 슬롯을 표시
%FW488	_FUSE_ERR	ARRAY[0..31] OF WORD	-	퓨즈 단선 에러	퓨즈 단선 에러가 발생한 해당 베이스 및 슬롯을 표시
%FD83	_BASE_DEERR	DWORD	-	증설 베이스 착탈 에러	증설 베이스 탈락 에러가 발생한 해당 베이스를 표시
%FD574	_BASE_POWER_FAIL	DWORD	-	전원 모듈 에러가 발생한 베이스 정보	전원 모듈 에러가 발생한 해당 베이스를 표시
%FW416	_IO_TYER_N	WORD	-	모듈 타입 불일치 슬롯 넘버	모듈 타입 불일치 에러가 발생한 슬롯의 넘버를 표시. 두 개 이상 중복 발생했을 경우는 가장 번호가 앞선 슬롯을 표시.
%FW417	_IO_DEER_N	WORD	-	모듈 착탈 슬롯 넘버	모듈 착탈 에러가 발생한 슬롯의 넘버를 표시. 두 개 이상 중복 발생했을 경우는 가장 번호가 앞선 슬롯을 표시.
%FW418	_FUSE_ER_N	WORD	-	퓨즈 단선 슬롯 넘버	퓨즈 단선 에러가 발생한 슬롯의 넘버를 표시. 두 개 이상 중복 발생했을 경우는 가장 번호가 앞선 슬롯을 표시.
%FW1922	_ANC_ERR	WORD	가능	외부 기기의 중고장 정보	사용자가 정의한 에러의 종류를 구분하여 0 을 제외한 값을 쓰고 외부 기기 중고장 검출 요청을 하면 외부기기 중고장 검출 에러를 발생 시킬 수 있습니다. 이 플래그를 직접 모니터링 함으로서 중고장 원인을 알 수 있습니다.

2. 스탠바이 CPU 시스템 에러 상세 플래그

주소	플래그명	타입	쓰기 가능	내 용	설 명
%FD147	_SB_BASE_DEERR	DWORD	-	증설 베이스 착탈 에러	증설 베이스 탈락 에러가 발생한 해당 베이스를 표시
%FW588	_SB_IO_TYERR	WORD	-	모듈 타입 불일치 에러	모듈 타입 불일치 에러가 발생한 해당 베이스 및 슬롯을 표시
%FW589	_SB_IO_DEERR	WORD	-	모듈 착탈 에러	모듈 착탈 에러가 발생한 해당 베이스 및 슬롯을 표시

부3.4. 시스템 경고 대표 플래그

1. 마스터 CPU 시스템 경고 대표 플래그

주소	플래그명	타입	BIT 위치	내 용	설 명
%FD66	_CNF_WAR	DWORD	대표 키워드	시스템 경고	시스템의 경고장 상태 대표 플래그
%FX2112	_RTC_ER	BOOL	BIT 0	RTC 이상	RTC 데이터에 이상이 발생한 경우
%FX2114	_BASE_EXIST_WAR	BOOL	BIT 2	미참가 베이스 존재	운전에 참여하지 않는 베이스 존재 경고
%FX2115	_AB_SD_ER	BOOL	BIT 3	운전 이상 정지	비정상 운전으로 인하여 정지합니다.
%FX2116	_TASK_ER	BOOL	BIT 4	태스크 충돌	태스크가 충돌하고 있습니다.
%FX2117	_BAT_ER	BOOL	BIT 5	배터리 이상	배터리 상태에 이상이 있습니다.
%FX2118	_ANNUM_WAR	BOOL	BIT 6	외부기기 고장	외부 기기의 경고장이 검출 되었습니다.
%FX2120	_HS_WAR	BOOL	BIT 8	고속 링크	고속 링크 파라미터 이상
%FX2121	_REDUN_WAR	BOOL	BIT 9	이중화 구성 경고	단독 CPU 운전 모드 설정을 하지 않고 이중화 구성을 하지 않았을 때 발생
%FX2122	_OS_VER_WAR	BOOL	BIT 10	O/S 버전 불일치 경고	CPU,증설 매니저,증설 드라이브 모듈 간에 O/S 버전이 상이할 때 발생
%FX2123	_RING_WAR	BOOL	BIT 11	링 토폴로지 구성 경고	증설 케이블 연결을 링 토폴로지로 구성해 주시기 바랍니다.
%FX2132	_P2P_WAR	BOOL	BIT 20	P2P 파라미터	P2P 파라미터 이상
%FX2140	_CONSTANT_ER	BOOL	BIT 28	고정주기 오류	고정주기 오류
%FX2141	_BASE_POWER_WAR	BOOL	BIT 29	전원 모듈 이상 경고	두 개 전원모듈 중 한 개 모듈이 이상이 있거나 꺼져있을 때 경고 발생
%FX2142	_BASE_SKIP_WAR	BOOL	BIT 30	베이스 스킵 해제 경고	베이스 스킵을 해제 시 IO 파라미터의 모듈 설정 상태와 베이스 스킵 해제를 한 베이스의 모듈 장착 상태가 다른 경우 발생

주소	플래그명	타입	BIT 위치	내 용	설 명
%FX2143	_BASE_NUM_OVER_WAR	BOOL	BIT 31	베이스 번호 설정 경고	증설 드라이브 모듈의 베이스 번호 설정이 1~31 번 이외의 번호를 설정했을 경우 발생

2. 스탠바이 CPU 시스템 경고 대표 플래그

주소	플래그명	타입	BIT 위치	내 용	설 명
%FD130	_SB_CNF_WAR	DWORD	대표 키워드	시스템 경고	시스템의 경고장 상태 대표 플래그
%FX4160	_SB_RTC_ER	BOOL	BIT 0	RTC 이상	RTC 데이터에 이상이 발생한 경우
%FX4162	_SB_BASE_EXIST_WAR	BOOL	BIT 2	미참가 베이스 존재	운전에 참여하지 않는 베이스 존재 경고
%FX4163	_SB_AB_SD_ER	BOOL	BIT 3	운전 이상 정지	비정상 운전으로 인하여 정지합니다.
%FX4164	_SB_TASK_ER	BOOL	BIT 4	태스크 충돌	태스크가 충돌하고 있습니다.
%FX4165	_SB_BAT_ER	BOOL	BIT 5	배터리 이상	배터리 상태에 이상이 있습니다.
%FX4166	_SB_ANNUM_WAR	BOOL	BIT 6	외부기기 고장	외부 기기의 경고장이 검출 되었습니다.
%FX4168	_SB_HS_WAR	BOOL	BIT 8	고속 링크	고속 링크 파라미터 이상
%FX4170	_SB_OS_VER_WAR	BOOL	BIT 10	O/S 버전 불일치 경고	CPU,증설 매니저,증설 드라이브 모듈 간에 O/S 버전이 상이할 때 발생
%FX4171	_SB_RING_WAR	BOOL	BIT 11	링 토폴로지 구성 경고	증설 케이블 연결을 링 토폴로지로 구성해 주시기 바랍니다.
%FX4180	_SB_P2P_WAR	BOOL	BIT 20	P2P 파라미터	P2P 파라미터 이상
%FX4188	_SB_CONSTANT_ER	BOOL	BIT 28	고정주기 오류	고정주기 오류
%FX4189	_SB_BASE_POWER_WAR	BOOL	BIT 29	전원 모듈 이상 경고	두 개 전원모듈 중 한 개 모듈이 이상이 있거나 꺼져있을 때 경고 발생
%FX4190	_SB_BASE_SKIP_WAR	BOOL	BIT 30	베이스 스킵 해제 경고	베이스 스킵을 해제 시 IO 파라미터의 모듈 설정 상태와 베이스 스킵 해제를 한 베이스의 모듈 장착 상태가 다른 경우 발생
%FX4191	_SB_BASE_NUM_OVER_WAR	BOOL	BIT 31	베이스 번호 설정 경고	증설 드라이브 모듈의 베이스 번호 설정이 1~31 번 이외의 번호를 설정했을 경우 발생

부3.5. 시스템 경고 상세 플래그

1. 마스터 CPU 시스템 경고 상세 플래그

주소	플래그명	타입	쓰기가능	내 용	설 명
%FX2624	_HS_WARN	ARRAY[0..11] OF BOOL	-	고속링크 파라미터 이상	고속링크 파라미터가 비정상적인 경우 해당 플래그 ON
%FX2640	_P2P_WARN	ARRAY[0..7] OF BOOL	-	P2P 파라미터 이상	P2P 파라미터가 비정상적인 경우 해당 플래그 ON
%FD587	_BASE_ACPF_WAR	DWORD	-	순시 정전 발생 경고 정보	순시 정전이 발생한 베이스 표시
%FW164	_HS_WAR_W	WORD	-	고속링크 파라미터 이상	고속링크 파라미터가 비정상인 고속링크 번호를 비트 별로 표시
%FW165	_P2P_WAR_W	WORD	-	P2P 파라미터 이상	P2P 파라미터가 비정상적인 P2P 번호를 비트 별로 표시
%FW1923	_ANC_WAR	WORD	-	외부 기기의 경고장 정보	사용자가 정의한 경고의 종류를 구분하여 0 을 제외한 값을 쓰고 외부 기기 경고장 검출 요청을 하면 외부기기 경고장 검출 경고를 발생 시킬 수 있습니다. 이 플래그를 직접 모니터링 함으로서 경고장 원인을 알 수 있습니다.

2. 스탠바이 CPU 시스템 경고 상세 플래그

주소	플래그명	타입	쓰기가능	내 용	설 명
%FX4672	_SB_HS_WARN	ARRAY[0..11] OF BOOL	-	고속링크 파라미터 이상	고속링크 파라미터가 비정상적인 경우 해당 플래그 On
%FX4688	_SB_P2P_WARN	ARRAY[0..7] OF BOOL	-	P2P 파라미터 이상	P2P 파라미터가 비정상적인 경우 해당 플래그 On
%FW292	_SB_HS_WAR_W	WORD	-	고속링크 파라미터 이상	고속링크 파라미터가 비정상인 고속링크 번호를 비트 별로 표시
%FW293	_SB_P2P_WAR_W	WORD	-	P2P 파라미터 이상	P2P 파라미터가 비정상적인 P2P 번호를 비트 별로 표시

부3.6. 시스템 운전상태 정보 플래그

1. 마스터 CPU 시스템 운전상태 정보 플래그

주소	플래그명	타입	BIT 위치	내 용	설 명
%FD64	_SYS_STATE	DWORD	대표 키워드	PLC 모드와 운전 상태	아래와 같이 시스템 상태를 일괄 취급합니다.
%FX2048	_RUN	BOOL	BIT 0	RUN (CPU 운전상태)	CPU 의 운전 상태를 표시합니다.
%FX2049	_STOP	BOOL	BIT 1	STOP (CPU 운전상태)	
%FX2050	_ERROR	BOOL	BIT 2	ERROR (CPU 운전상태)	
%FX2051	_DEBUG	BOOL	BIT 3	DEBUG (CPU 운전상태)	
%FX2052	_LOCAL_CON	BOOL	BIT 4	로컬 컨트롤	로컬 컨트롤 모드입니다.
%FX2054	_REMOTE_CON	BOOL	BIT 6	리모트 모드 ON	리모트 컨트롤 모드입니다.
%FX2058	_RUN_EDIT_DONE	BOOL	BIT 10	런 중 수정 완료	런 중 수정을 완료 하면 표시합니다.
%FX2059	_RUN_EDIT_NG	BOOL	BIT 11	런 중 수정 비정상 완료	런 중 수정을 비정상 적으로 완료 하면 표시합니다.
%FX2060	_CMOD_KEY	BOOL	BIT 12	키에 의한 운전모드 변경	키에 의한 운전모드 변경을 표시 합니다.
%FX2061	_CMOD_LPADT	BOOL	BIT 13	로컬 PADT 에 의한 운전 모드 변경	로컬 PADT 에 의한 운전모드 변경을 표시합니다.
%FX2062	_CMOD_RPADT	BOOL	BIT 14	리모트 PADT 에 의한 운전모드 변경	리모트 PADT 에 의한 운전모드 변경을 표시합니다.
%FX2063	_CMOD_RLINK	BOOL	BIT 15	리모트 통신 모듈에 의한 운전 모드 변경	리모트 통신 모듈에 의한 운전 모드 변경을 표시합니다.
%FX2064	_FORCE_IN	BOOL	BIT 16	강제 입력	입력접점에 대한 강제 ON/OFF 실행 중 표시합니다.
%FX2065	_FORCE_OUT	BOOL	BIT 17	강제 출력	출력접점에 대한 강제 ON/OFF 실행 중 표시합니다.
%FX2066	_SKIP_ON	BOOL	BIT 18	입출력 스킵 실행 중	스킵이 설정 되어 있을 때 표시합니다.
%FX2067	_EMASK_ON	BOOL	BIT 19	고장 마스크 실행 중	고장 마스크가 설정 되어 있을 때 표시합니다.
%FX2069	_USTOP_ON	BOOL	BIT 21	STOP 펑션에 의한 STOP	RUN 모드 운전 중 STOP 펑션에 의해 정지되었음을 표시합니다.
%FX2070	_ESTOP_ON	BOOL	BIT 22	ESTOP 펑션에 의한 STOP	RUN 모드 운전 중 ESTOP 펑션에 의해 즉시 정지되었음을 표시합니다.
%FW192	_SL_OS_VER	ARRAY[0..31] OF WORD	-	증설 드라이브 모듈 O/S 버전	장착된 증설 드라이브 모듈의 O/S 버전을 표시합니다.
%FW600	_BASE_INFO	ARRAY[0..31] OF WORD	-	베이스 정보	장착된 각 베이스의 슬롯 개수를 나타냅니다.
%FB12	_RTC_TIME	ARRAY[0..7] OF	-	현재시각	현재 시각을 표시합니다.

주소	플래그명	타입	BIT 위치	내 용	설 명
		BYTE			
%FX2072	_INIT_RUN	BOOL	-	초기화 태스크 수행 중	초기화 태스크가 수행 중입니다.
%FX2074	_AB_SIDE	BOOL	-	CPU 장착 위치	CPU 장착 위치(A-SIDE: ON, B-SIDE: OFF)
%FX2076	_PB1	BOOL	-	프로그램 코드 1	프로그램 코드 1 이 선택되었습니다.
%FX2077	_PB2	BOOL	-	프로그램 코드 2	프로그램 코드 2 가 선택되었습니다.
%FX30736	_INIT_DONE	BOOL	가능	초기화 태스크 수행 완료	초기화 태스크 수행 완료를 표시합니다.
%FW584	_RTC_DATE	DATE	-	RTC 의 현재 날짜	RTC 의 현재 날짜를 표시합니다.
%FD67	_OS_VER	DWORD	-	O/S 버전 번호	CPU 의 O/S 버전을 표시합니다.
%FD68	_OS_DATE	DWORD	-	O/S 날짜	CPU 의 O/S 날짜을 표시합니다.
%FD69	_CP_OS_VER	DWORD	-	증설 매니저 O/S 버전	증설 매니저의 O/S 버전을 표시합니다.
%FD573	_OS_TYPE	DWORD	-	PLC 구분용.	타 사업부 제공 여부
%FW1081	_FALS_NUM	INT	-	FALS 번호	FALS 의 번호를 표시합니다.
%FD293	_RTC_TOD	TIME_OF_DAY	-	RTC 의 현재 시간	RTC 의 현재 시간을 표시합니다. (ms 단위)
%FD582	_RUN_EDIT_CNT	UDINT	-	런 중 수정한 횟수	런 중 수정한 횟수를 표시합니다.
%FW140	_AC_F_CNT	UINT	-	순시 정전 발생횟수	순시 정전이 발생한 횟수를 표시합니다.
%FW158	_POWER_OFF_CNT	UINT	-	전원 차단 횟수	전원이 차단된 횟수를 표시합니다.
%FW386	_SCAN_MAX	UINT	가능	최대 스캔 시간	런 이래로 최대 스캔시간을 나타냅니다. 단위는 0.1ms 입니다.
%FW387	_SCAN_MIN	UINT	가능	최소 스캔 시간	런 이래로 최소 스캔시간을 나타냅니다. 단위는 0.1ms 입니다.
%FW388	_SCAN_CUR	UINT	가능	현재 스캔 시간	현재 스캔시간을 나타냅니다. 단위는 0.1ms 입니다.
%FW585	_RTC_WEEK	UINT	-	RTC 의 현재 요일	RTC 의 현재 요일을 표시합니다.
%FW141	_CPU_TYPE	WORD	-	CPU ID (XGR - 0xA801)	CPU 타입을 표시합니다.
%FW633	_RBANK_NUM	WORD	-	현재 사용중인 블록 번호	현재 사용중인 블록 번호를 표시합니다.

2. 스탠바이 CPU 시스템 운전상태 정보 플래그

주소	플래그명	타입	BIT 위치	내 용	설 명
%FD128	_SB_SYS_STATE	DWORD	대표 키워드	시스템 정보	아래와 같이 시스템 상태를 일괄 취급합니다.
%FX4096	_SB_RUN	BOOL	BIT 0	RUN (CPU 운전상태)	CPU 의 운전 상태를 표시합니다.
%FX4097	_SB_STOP	BOOL	BIT 1	STOP (CPU 운전상태)	
%FX4098	_SB_ERROR	BOOL	BIT 2	ERROR (CPU 운전상태)	

주소	플래그명	타입	BIT 위치	내 용	설 명
%FX4100	_SB_LOCAL_CON	BOOL	BIT 4	로컬 컨트롤	로컬 컨트롤 모드입니다.
%FX4102	_SB_REMOTE_CON	BOOL	BIT 6	리모트 모드 On	리모트 컨트롤 모드입니다.
%FX4106	_SB_RUN_EDIT_DONE	BOOL	BIT 10	런 중 수정 완료	런 중 수정을 완료 하면 표시합니다.
%FX4107	_SB_RUN_EDIT_NG	BOOL	BIT 11	런 중 수정 비정상 완료	런 중 수정을 비정상 적으로 완료 하면 표시합니다.
%FX4108	_SB_CMOD_KEY	BOOL	BIT 12	키에 의한 운전모드 변경	키에 의한 운전모드 변경을 표시 합니다.
%FX4109	_SB_CMOD_LPADT	BOOL	BIT 13	로컬 PADT 에 의한 운전모드 변경	로컬 PADT 에 의한 운전모드 변경을 표시합니다.
%FX4110	_SB_CMOD_RPADT	BOOL	BIT 14	리모트 PADT 에 의한 운전모드 변경	리모트 PADT 에 의한 운전모드 변경을 표시합니다.
%FX4111	_SB_CMOD_RLINK	BOOL	BIT 15	리모트 통신 모듈에 의한 운전 모드 변경	리모트 통신 모듈에 의한 운전 모드 변경을 표시합니다.
%FX4112	_SB_FORCE_IN	BOOL	BIT 16	강제 입력	입력접점에 대한 강제 ON/OFF 실행 중 표시합니다.
%FX4113	_SB_FORCE_OUT	BOOL	BIT 17	강제 출력	출력접점에 대한 강제 ON/OFF 실행 중 표시합니다.
%FX4114	_SB_SKIP_ON	BOOL	BIT 18	입출력 스킵 실행 중	스킵이 설정 되어 있을 때 표시합니다.
%FX4115	_SB_EMASK_ON	BOOL	BIT 19	고장 마스크 실행 중	고장 마스크가 설정 되어 있을 때 표시합니다.
%FX4117	_SB_USTOP_ON	BOOL	-	STOP 펑션에 의한 STOP	RUN 모드 운전 중 STOP 펑션에 의해 정지되었음을 표시합니다.
%FX4118	_SB_ESTOP_ON	BOOL	-	ESTOP 펑션에 의한 STOP	RUN 모드 운전 중 ESTOP 펑션에 의해 즉시 정지되었음을 표시합니다.
%FD131	_SB_OS_VER	DWORD	-	O/S 버전	CPU 의 O/S 버전을 표시합니다.
%FD132	_SB_OS_DATE	DWORD	-	O/S 날짜	CPU 의 O/S 날짜를 표시합니다.
%FD133	_SB_CP_OS_VER	DWORD	-	증설 매니저 O/S 버전	증설 매니저의 O/S 버전을 표시합니다.
%FW286	_SB_POWER_OFF_CNT	UINT	-	전원 차단 횟수	전원이 차단된 횟수를 표시합니다.
%FW269	_SB_CPU_TYPE	WORD	-	CPU ID (XGR - 0xA801)	CPU 타입을 표시합니다.
%FW632	_SB_BASE_INFO	WORD	-	베이스 정보	스탠바이에 장착된 베이스의 슬롯 개수를 나타냅니다.

부3.7. 이중화 운전모드 정보 플래그

1. 이중화 운전모드 정보

주소	플래그명	타입	BIT 위치	내 용	설 명
%FD0	_REDUN_STATE	DWORD	대표 키워드	이중화 운전 정보	이중화 시스템의 운전상태를 표시하는 대표 플래그
%FX0	_DUAL_RUN	BOOL	BIT 0	이중화 운전 중	이중화 운전 중으로 CPU A, CPU B가 정상 운전
%FX1	_RING_TOPOLOGY	BOOL	BIT 1	링 토폴로지 상태	증설베이스가 링으로 연결됨

주소	플래그명	타입	BIT 위치	내 용	설 명
%FX2	_LINE_TOPOLOGY	BOOL	BIT 2	라인 토폴로지 상태	증설베이스가 라인으로 연결됨
%FX4	_SINGLE_RUN_A	BOOL	BIT 4	A-SIDE 단독 런 모드	이중화 시스템에서 CPU A 만 단독으로 운전 중 임을 표시
%FX5	_SINGLE_RUN_B	BOOL	BIT 5	B-SIDE 단독 런 모드	이중화 시스템에서 CPU B 만 단독으로 운전 중 임을 표시
%FX6	_MASTER_RUN_A	BOOL	BIT 6	A-SIDE 가 마스터이고 런 모드인 상태(스탠바이 CPU 가 존재할 경우 해당)	이중화 운전 중으로 CPU A 가 마스터로 운전 중 임을 표시
%FX7	_MASTER_RUN_B	BOOL	BIT 7	B-SIDE 가 마스터이고 런 모드인 상태(스탠바이 CPU 가 존재할 경우 해당)	이중화 운전 중으로 CPU B 가 마스터로 운전 중 임을 표시

부3.8.연산 결과 정보 플래그

1. 연산 결과 정보 플래그

주소	플래그명	타입	쓰기가능	내 용	설 명
%FX672	_ARY_IDX_ERR	BOOL	가능	배열 사용시 인덱스 범위 초과 에러	배열 사용시 인덱스의 범위가 설정한 값보다 초과 할 경우 발생합니다.
%FX704	_ARY_IDX_LER	BOOL	가능	배열 사용시 인덱스 범위 초과 에러 래치	배열 사용시 인덱스의 범위가 설정한 값보다 초과 할 경우 발생된 에러는 해당 프로그램 블록이 끝날 때까지 유지되며 프로그램에 의해서 지우는 것이 가능합니다.
%FX6160	_ERR	BOOL	가능	연산 에러 플래그	연산 펑션(FN) 또는 펑션 블록(FB) 단위의 연산 에러 플래그로 연산이 수행될 때 마다 갱신됩니다.
%FX6165	_LER	BOOL	가능	연산 에러 래치 플래그	프로그램 블록(PB) 단위의 연산 에러 래치 플래그입니다. 프로그램 블록 수행 중 발생한 에러표시는 해당 프로그램 블록이 끝날 때까지 유지되며 프로그램에 의해서 지우는 것이 가능합니다.

부3.9. 운전 모드 키 상태 정보 플래그

1. 운전모드 키 상태 정보 플래그

주소	플래그명	타입	쓰기가능	내 용	설 명
%FX291	_REMOTE_KEY	BOOL	-	리모트 키 스위치 상태 정보	CPU 키 스위치 상태 정보 - (리모트 상태일 때: OFF, 리모트 상태가 아닐 때: ON)
%FX294	_STOP_KEY	BOOL	-	스톱 키 스위치 상태 정보	CPU 키 스위치 상태 정보 - (스톱 상태일 때:OFF, 스톱 상태가 아닐 때:ON)
%FX295	_RUN_KEY	BOOL	-	런 키 스위치 상태 정보	CPU 키 스위치 상태 정보 - (런 상태일 때:OFF,런 상태가 아닐 때:ON)

부3.10. 링크 플래그(L) 일람

데이터 링크용 플래그(L)에 대하여 설명합니다.

[표 1.10.1] 고속링크 번호에 따른 링크릴레이 일람 고속링크 번호(n) 1 ~ 12

항목	키워드	타입	내 용	내 용 설 명
고속링크	_HSn_RLINK	비트	고속링크 파라미터 n 번의 모든 국 정상 동작	고속 링크에서 설정된 파라미터 대로 모든 국이 정상적으로 동작하고 있음을 표시하며, 아래와 같은 조건에서 On 됩니다. 1.파라미터에 설정된 모든 국이 RUN 모드이고,에러가 없고 2.파라미터에 설정된 모든 데이터 블록이 정상적으로 통신되며 3.파라미터에 설정된 각국 자체에 설정된 파라미터가 정상적으로 통신되는 경우 런_링크는 한번 On 되면 링크 디스에이블에 의해 중단 시키지 않는 한 계속 On 을 유지합니다.
	_HSn_LTRBL	비트	_HSnRLINK ON 이후 비정상 상태 표시	_HSmRLINK 플래그가 On 된 상태에서 파라미터에 설정된 국과 데이터 블록의 통신 상태가 다음과 같을 때 이 플래그는 On 됩니다. 1.파라미터에 설정된 국이 RUN 모드가 아니거나 2.파라미터에 설정된 국에 에러가 있거나 3.파라미터에 설정된 데이터 블록의 통신 상태가 원활하지 못한 경우 링크 트러블은 위 1,2,3 의 조건이 발생하면 On 되고,그 조건이 정상적을 돌아가면 다시 Off 됩니다.
	_HSn_STATE[k] (k=000~127)	비트 Array	고속링크 파라미터 n 번 k 번 블록의 종합적 상태 표시	설정된 파라미터의 각 데이터 블록에 대한 통신 정보의 종합적 상태를 표시합니다 HS1STATE[k]=HS1MOD[k]&_HS1TRX[k]&(~_HSmERR[k])
	_HSn_MOD[k] (k=000~127)	비트 Array	고속링크 파라미터 n 번 k 번 블록 국의 런 운전 모드	파라미터의 k 데이터 블록에 설정된 국의 동작 모드를 표시합니다
	_HSn_TRX[k] (k=000~127)	비트 Array	고속링크 파라미터 n 번 k 번 블록 국과 정상 통신 표시	파라미터의 k 데이터 블록의 통신 상태가 설정된 대로 원활히 통신되고 있는지를 표시합니다
	_HSn_ERR[k] (k=000~127)	비트 Array	고속링크 파라미터 n 번 k 번 블록 국의 운전 에러 모드	파라미터의 k 데이터 블록의 통신 상태에 에러가 발생했는지를 표시합니다
	_HSn_SETBLOCK[k]	비트 Array	고속링크 파라미터 n 번 K 번 블록 설정 표시	파라미터의 k 데이터 블록 설정 여부를 표시합니다

알아두기

고속링크 번호	L 영역 번지수	비 고
1	L000000~L00049F	[표 1]의 고속링크 1 일 때와 비교하여 다른 고속링크 국번의 플래그 번지수는 간단한 계산식에 의해 다음과 같습니다. *계산식:L 영역 번지수 = L000000 + 500 x (고속링크 번호 - 1) 프로그램 및 모니터링을 위하여 고속링크 플래그를 이용하고자 할 경우에는 XG5000 에 등록된 플래그 맵을 이용하시면 편리하시게 이용하실 수 있습니다.
2	L000500~L00099F	
3	L001000~L00149F	
4	L001500~L00199F	
5	L002000~L00249F	
6	L002500~L00299F	
7	L003000~L00349F	
8	L003500~L00399F	
9	L004000~L00449F	
10	L004500~L00499F	
11	L005000~L00549F	

k 는 블록 번호로 000~127 까지 128 개의 블록에 대한 정보를 1 워드에 16 개씩 8 워드에 거쳐 나타납니다.
예를 들면 모드 정보 (_HS1MOD)는 L00010 에 블록 0 부터 블록 15 까지 L00011, L00012, L00013, L00014, L00015, L00016, L00017 에 블록 16~31, 32~47, 48~63, 64~79, 80~95, 96~111, 112~127 의 정보가 나타납니다. 따라서 블록 번호 55 의 모드정보는 L000137 에 나타납니다.

[표 1.10.2] P2P 서비스 설정에 따른 링크 플래그 일람 P2P 파라미터 번호(n) : 1~8, P2P 블록(xx) : 0~63

항목	키워드	타입	내 용	내 용 설 명
P2P	_P2Pn_NDRxx	비트	P2P 파라미터 n 번 xx 번 블록 서비스 정상 완료	P2P 파라미터 n 번 xx 번 블록 서비스 정상 완료
	_P2Pn_ERRxx	비트	P2P 파라미터 n 번 xx 번 블록 서비스 비정상 완료	P2P 파라미터 n 번 xx 번 블록 서비스 비정상 완료
	_P2Pn_STATUSxx	워드	P2P 파라미터 n 번 xx 번 블록 서비스 비정상 완료 시 에러 코드	P2P 파라미터 n 번 xx 번 블록 서비스 비정상 완료 시 에러 코드를 표시합니다.
	_P2Pn_SVCCNTxx	더블워드	P2P 파라미터 n 번 xx 번 블록 서비스 정상 수행 횟수	P2P 파라미터 n 번 xx 번 블록 서비스 정상 수행 횟수를 표시합니다.
	_P2Pn_ERRCNTxx	더블워드	P2P 파라미터 n 번 xx 번 블록 서비스 비정상 수행 횟수	P2P 파라미터 n 번 xx 번 블록 서비스 비정상 수행 횟수를 표시합니다.

부3.11. 통신 플래그(P2P) 일람

[표 1.11.1] P2P 번호에 따른 통신 레지스터 일람 P2P 파라미터 번호(n) : 1~8, P2P 블록(xx) : 0~63

번호	플래그	타입	내 용	내 용 설 명
N00000	_PnBxxSN	워드	P2P 파라미터 n 번 xx 번 블록 상대 국번	P2P 파라미터 n 번 xx 번 블록의 상대 국번을 저장합니다. XG-PD 에서 상대 국번을 이용할 경우에는 P2PSN 명령어를 이용하여 런중에 수정 가능합니다.
N00001 ~ N00004	_PnBxxRD1	디바이스 구조체	P2P 파라미터 n 번 xx 번 블록 읽을 영역 디바이스 1	P2P 파라미터 n 번 xx 번 블록 읽을 영역 디바이스 1 을 저장합니다.
N00005	_PnBxxRS1	워드	P2P 파라미터 n 번 xx 번 블록 읽을 영역 사이즈 1	P2P 파라미터 n 번 xx 번 블록 읽을 영역 사이즈 1 을 저장합니다.
N00006 ~ N00009	_PnBxxRD2	디바이스 구조체	P2P 파라미터 n 번 xx 번 블록 읽을 영역 디바이스 2	P2P 파라미터 n 번 xx 번 블록 읽을 영역 디바이스 2 를 저장합니다.
N00010	_PnBxxRS2	워드	P2P 파라미터 n 번 xx 번 블록 읽을 영역 사이즈 2	P2P 파라미터 n 번 xx 번 블록 읽을 영역 사이즈 2 를 저장합니다.
N00011 ~ N00014	_PnBxxRD3	디바이스 구조체	P2P 파라미터 n 번 xx 번 블록 읽을 영역 디바이스 3	P2P 파라미터 n 번 xx 번 블록 읽을 영역 디바이스 3 을 저장합니다.
N00015	_PnBxxRS3	워드	P2P 파라미터 n 번 xx 번 블록 읽을 영역 사이즈 3	P2P 파라미터 n 번 xx 번 블록 읽을 영역 사이즈 3 을 저장합니다.
N00016 ~ N00019	_PnBxxRD4	디바이스 구조체	P2P 파라미터 n 번 xx 번 블록 읽을 영역 디바이스 4	P2P 파라미터 n 번 xx 번 블록 읽을 디바이스 영역 4 를 저장합니다.
N00020	_PnBxxRS4	워드	P2P 파라미터 n 번 xx 번 블록 읽을 영역 사이즈 4	P2P 파라미터 n 번 xx 번 블록 읽을 영역 사이즈 4 를 저장합니다.
N00021 ~ N00024	_PnBxxWD1	디바이스 구조체	P2P 파라미터 n 번 xx 번 블록 저장 영역 디바이스 1	P2P 파라미터 n 번 xx 번 블록 저장 영역 디바이스 1 을 저장합니다.
N00025	_PnBxxWS1	워드	P2P 파라미터 n 번 xx 번 블록 저장 영역 사이즈 1	P2P 파라미터 n 번 xx 번 블록 저장 영역 사이즈 1 을 저장합니다.
N00026 ~ N00029	_PnBxxWD2	디바이스 구조체	P2P 파라미터 n 번 xx 번 블록 저장 영역 디바이스 2	P2P 파라미터 n 번 xx 번 블록 저장 영역 디바이스 2 를 저장합니다.
N00030	_PnBxxWS2	워드	P2P 파라미터 n 번 xx 번 블록 저장 영역 사이즈 2	P2P 파라미터 n 번 xx 번 블록 저장 영역 사이즈 2 를 저장합니다.
N00031 ~ N00034	_PnBxxWD3	디바이스 구조체	P2P 파라미터 n 번 xx 번 블록 저장 영역 디바이스 3	P2P 파라미터 n 번 xx 번 블록 저장 영역 디바이스 3 을 저장합니다.
N00035	_PnBxxWS3	워드	P2P 파라미터 n 번 xx 번 블록 저장 영역 사이즈 3	P2P 파라미터 n 번 xx 번 블록 저장 영역 사이즈 3 을 저장합니다.
N00036 ~ N00039	_PnBxxWD4	디바이스 구조체	P2P 파라미터 n 번 xx 번 블록 저장 영역 디바이스 4	P2P 파라미터 n 번 xx 번 블록 저장 영역 디바이스 4 를 저장합니다.
N00040	_PnBxxWS4	워드	P2P 파라미터 n 번 xx 번 블록 저장 영역 사이즈 4	P2P 파라미터 n 번 xx 번 블록 저장 영역 사이즈 4 를 저장합니다.

알아두기	
통신 플래그는 XG-PD 를 이용하여 P2P 파라미터를 설정할 경우 자동으로 설정되며 P2P 전용 명령을 이용하여 런중에 수정할 수도 있습니다. 통신 플래그는 P2P 파라미터 설정 번호, 블록 인덱스 번호에 따라 사용되는 번지수가 구분되므로 P2P 서비스로 이용하지 않는 영역은 내부 디바이스로 사용 가능합니다.	

부3.12. 예약어

예약어는 시스템에서 사용하기 위해 미리 정의한 단어들입니다. 따라서 식별자로 이 예약어를 사용할 수는 없습니다.

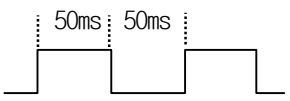
예 약 어
ACTION ... END_ACTION
ARRAY ... OF
AT
CASE ... OF ... ELSE ... END_CASE CONFIGURATION ... END_CONFIGURATION
데이터 타입 이름
DATE#, D# DATE_AND_TIME#, DT#
EXIT
FOR ... TO ... BY ... DO ... END_FOR
FUNCTION ... END_FUNCTION
FUNCTION_BLOCK ... END_FUNCTION_BLOCK
펑션 블록의 이름들
IF ... THEN ... ELSIF ... ELSE ... END_IF
OK
연산자 (IL 언어) 연산자 (ST 언어)
PROGRAM
PROGRAM ... END_PROGRAM
REPEAT ... UNTIL ... END_REPEAT
RESOURCE ... END_RESOURCE
RETAIN
RETURN
STEP ... END_STEP
STRUCTURE ... END_STRUCTURE
T#
TASK ... WITH
TIME_OF_DAY#, TOD#
TRANSITION ... FROM... TO ... END_TRANSITION
TYPE ... END_TYPE

예 약 어
VAR ... END_VAR VAR_INPUT ... END_VAR VAR_OUTPUT ... END_VAR
VAR_IN_OUT ... END_VAR
VAR_EXTERNAL ... END_VAR
VAR_ACCESS ... END_VAR
VAR_GLOBAL ... END_VAR
WHILE ... DO ... END_WHILE
WITH

부록4 플래그 일람(XEC)

부4.1. 특수 릴레이(F)일람

예 약 변 수	데이터타입	내 용
_SYS_STATE	모드와 상태	PLC 의 모드와 운전 상태를 표시합니다.
_RUN	RUN	RUN 상태입니다.
_STOP	STOP	STOP 상태입니다.
_ERROR	ERROR	ERROR 상태입니다.
_DEBUG	DEBUG	DEBUG 상태입니다.
_LOCAL_CON	로컬 컨트롤	로컬 컨트롤 모드입니다.
_REMOTE_CON	리모트 모드	리모트 컨트롤 모드입니다.
_RUN_EDIT_ST	런 중 수정	런중 수정 프로그램 다운로드 중입니다.
_RUN_EDIT_CHK		런중 수정 내부 처리 중입니다.
_RUN_EDIT_DONE		런중 수정 완료
_RUN_EDIT_NG		런중 수정이 비정상 종료
_CMOD_KEY	운전 모드 변경	키에 의해 운전 모드가 변경
_CMOD_LPADT		로컬 PADT 에 의해 운전 모드 변경
_CMOD_RPADT		리모트 PADT 에 의해 운전 모드 변경
_CMOD_RLINK		리모트 통신 모듈에 의해 운전 모드 변경
_FORCE_IN	강제 입력	입력접점에 대한 강제 On/Off 실행 중임을 표시합니다
_FORCE_OUT	강제 출력	출력접점에 대한 강제 On/Off 실행 중임을 표시합니다.
_MON_ON	모니터링	모니터가 실행 중입니다.
_USTOP_ON	STOP 펑션에 의해 STOP 되었습니다.	RUN 모드 운전 중 STOP 펑션에 의해 스캔 종료 후 정지합니다
_ESTOP_ON	ESTOP 펑션에 의해 STOP 되었습니다.	RUN 모드 운전 중 ESTOP 펑션에 의해 즉시 정지
_INIT_RUN	초기화중	초기화 태스크가 수행 중입니다.
_PB1	프로그램 코드 1	프로그램 코드 1 이 선택되었습니다.
_PB2	프로그램 코드 2	프로그램 코드 2 가 선택되었습니다.
_CB1	컴파일 코드 1	컴파일 코드 1 이 선택되었습니다.
_CB2	컴파일 코드 2	컴파일 코드 2 가 선택되었습니다.
_CNF_ER	시스템 에러	시스템의 중고장 상태를 보고합니다.
_IO_TYER	모듈 타입 에러	모듈 타입이 일치하지 않습니다.
_IO_DEER	모듈 착탈 에러	모듈이 착탈 되었습니다.
_IO_RWER	모듈 입출력 에러	모듈 입출력에 문제가 발생했습니다.
_IP_IFER	모듈 인터페이스 에러	특수 / 통신 모듈 인터페이스에 문제가 발생했습니다.

예 약 변 수	데이터타입	내 용
_ANNUM_ER	외부 기기 고장	외부 기기에 중고장이 검출되었습니다.
_BPRM_ER	기본 파라미터	기본 파라미터에 이상이 있습니다.
_IOPRM_ER	IO 파라미터	IO 구성 파라미터에 이상이 있습니다.
_SPPRM_ER	특수 모듈 파라미터	특수 모듈 파라미터가 비정상입니다.
_CPPRM_ER	통신 모듈 파라미터	통신 모듈 파라미터가 비정상입니다.
_PGM_ER	프로그램 에러	사용자가 작성한 프로그램의 체크섬 등의 이상이 발생한 경우
_CODE_ER	프로그램 코드 에러	사용자 프로그램 수행 중 해독할 수 없는 명령을 만났을 때 발생하는 에러
_SWDT_ER	CPU 비정상 종료 또는 고장	CPU 비정상 종료로 저장된 프로그램이 파괴된 경우 또는 프로그램 수행이 불가능한 에러
_WDT_ER	스캔 워치독	스캔 워치독이 작동했습니다.
_CNF_WAR	시스템 경고	시스템의 경고장 상태를 보고합니다.
_RTC_ER	RTC 데이터 이상	RTC 데이터에 이상이 발생했습니다.
_DBCK_ER	데이터 백업 이상	데이터 벡업에 문제가 발생했습니다.
_HBCK_ER	리스타트 이상	핫 리스타트가 불가능합니다.
_ABSD_ER	운전 이상 정지	비정상 운전으로 인하여 정지합니다.
_TASK_ER	태스크 충돌	태스크가 충돌하고 있습니다.
_BAT_ER	배터리 이상	배터리 상태에 이상이 있습니다.
_ANNUM_WAR	외부 기기 경고장 검출	외부 기기의 경고장이 검출 되었습니다.
_HS_WAR1	고속 링크 1	고속 링크 - 파라미터 1 이상
_HS_WAR2	고속 링크 2	고속 링크 - 파라미터 2 이상
_P2P_WAR1	P2P 파라미터 1	P2P - 파라미터 1 이상
_P2P_WAR2	P2P 파라미터 2	P2P - 파라미터 2 이상
_P2P_WAR3	P2P 파라미터 3	P2P - 파라미터 3 이상
_CONSTANT_ER	고정주기 오류	고정주기 오류
_USER_F	유저 접점	사용자가 사용할 수 있는 타이머입니다.
_T20MS	20ms	사용자 프로그램에서 사용할 수 있는 클럭신호로 반주기 마다 On/Off 반전됩니다. 스캔종료 후에 신호반전을 처리하므로, 프로그램수행 시간에 따라 클록신호가 지연 또는 왜곡될 수 있으므로, 스캔시간보다 충분히 긴 클록을 사용하여야 합니다. 클록신호는 초기화 프로그램 시작시, 스캔 프로그램 시작시에 Off 에서 시작합니다. _T100ms 클럭 예 
_T100MS	100ms	
_T200MS	200ms	
_T1S	1 초 Clock	
_T2S	2 초 Clock	
_T10S	10 초 Clock	
_T20S	20 초 Clock	
_T60S	60 초 Clock	

예 약 변 수	데이터타입	내 용
_On	항시 On	항상 On 상태인 비트입니다.
_Off	항시 Off	항상 Off 상태인 비트입니다.
_10n	1 스캔 On	첫 스캔만 On 상태인 비트입니다.
_10ff	1 스캔 Off	첫 스캔만 Off 상태인 비트입니다.
_STOG	반전	매 스캔 반전됩니다.
_USER_CLK	유저 Clock	사용자가 설정 가능한 Clock 입니다.
_USR_CLK0	지정 스캔 반복	지정된 스캔만큼 On/Off Clock 0
_USR_CLK1	지정 스캔 반복	지정된 스캔만큼 On/Off Clock 1
_USR_CLK2	지정 스캔 반복	지정된 스캔만큼 On/Off Clock 2
_USR_CLK3	지정 스캔 반복	지정된 스캔만큼 On/Off Clock 3
_USR_CLK4	지정 스캔 반복	지정된 스캔만큼 On/Off Clock 4
_USR_CLK5	지정 스캔 반복	지정된 스캔만큼 On/Off Clock 5
_USR_CLK6	지정 스캔 반복	지정된 스캔만큼 On/Off Clock 6
_USR_CLK7	지정 스캔 반복	지정된 스캔만큼 On/Off Clock 7
_LOGIC_RESULT	로직 결과	로직 결과를 표시합니다.
_ERR	연산 에러	연산 에러시 1 스캔동안 On
_LER	연산 에러 래치	연산 에러시 계속 On 유지
_FALS_NUM	FALS 번호	FALS 의 번호를 표시합니다.
_PUTGET_ERR0	PUT/GET 에러 0	메인 베이스 PUT / GET 에러
_PUTGET_NDR0	PUT/GET 완료 0	메인 베이스 PUT / GET 완료
_CPU_TYPE	CPU 타입	CPU 타입에 관한 정보를 알려줍니다.
_CPU_VER	CPU 버전	CPU 버전을 표시합니다.
_OS_VER	OS 버전	OS 버전을 표시합니다.
_OS_DATE	OS 날짜	OS 배포일을 표시합니다.
_SCAN_MAX	최대 스캔시간	최대 스캔시간을 나타냅니다.
_SCAN_MIN	최소 스캔시간	최소 스캔시간을 나타냅니다.
_SCAN_CUR	현재스캔시간	현재 스캔시간을 나타냅니다.
_MON_YEAR	월 / 년	PLC 의 월, 년 데이터입니다.
_TIME_DAY	시 / 일	PLC 의 시, 일 데이터입니다.
_SEC_MIN	초 / 분	PLC 의 초, 분 데이터입니다.
_HUND_WK	백년 / 요일	PLC 의 백년, 요일 데이터입니다.
_REF_COUNT	리프레시 카운트	모듈 리프레시 수행시 증가
_REF_OK_CNT	리프레시 정상 카운트	모듈 리프레시가 정상일 때 증가
_REF_NG_CNT	리프레시 비정상 카운트	모듈 리프레시가 비정상일 때 증가

예 약 변 수	데이터타입	내 용
_REF_LIM_CNT	리프레시 Limit	모듈 리프레시가 비정상일 때 증가 (TIME OUT)
_REF_ERR_CNT	리프레시 Error	모듈 리프레시가 비정상일 때 증가
_BUF_FULL_CNT	버퍼 Full	CPU 내부 버퍼 FULL 일 경우 증가
_PUT_CNT	PUT 카운트	PUT 수행 시 증가합니다.
_GET_CNT	GET 카운트	GET 수행 시 증가합니다.
_KEY	현재 키	로컬 키의 현재 상태를 나타냅니다.
_KEY_PREV	이전 키	로컬 키의 이전 상태를 나타냅니다.
_IO_TYER_N	모듈 타입 불일치 슬롯	모듈 타입 불일치 슬롯 번호 표시
_IO_DEER_N	모듈 착탈 슬롯	모듈 착탈이 일어난 슬롯 번호 표시
_IO_RWER_N	입출력모듈 R/W 에러 슬롯	모듈 읽기/쓰기 에러 슬롯 번호 표시
_IP_IFER_N	특수/통신 IF 에러 슬롯	모듈 인터페이스 에러 슬롯 번호 표시
_IO_TYER0	모듈타입 0 에러	메인 베이스 모듈 타입 에러
_IO_DEER0	모듈착탈 0 에러	메인 베이스 모듈 착탈 에러
_IO_RWER0	모듈 RW 0 에러	메인 베이스 모듈 읽기/쓰기 에러
_IO_IFER_0	모듈 IF 0 에러	메인 베이스 모듈 인터페이스 에러
_AC_FAIL_CNT	전원 차단 횟수	전원이 차단 된 횟수를 저장합니다.
_ERR_HIS_CNT	에러 발생 횟수	에러가 발생한 횟수를 저장합니다.
_MOD_HIS_CNT	모드 전환 횟수	모드가 전환된 횟수를 저장합니다.
_SYS_HIS_CNT	이력 발생 횟수	시스템 이력 발생 횟수를 저장합니다.
_LOG_ROTATE	로그 로테이트	로그 로테이트 정보를 저장합니다.
_BASE_INF00	슬롯 정보 0	메인 베이스 슬롯 정보
_RBANK_NUM	현재 사용중인 블록 번호	현재 사용중인 블록 번호를 표시합니다.
_RBLOCK_STATE	현재 사용중인 블록 상태	현재 사용중인 블록 번호의 상태 (읽기/쓰기/에러)를 표시합니다
_RBLOCK_RD_FLAG	플래시 N 블록 읽기	플래시 N 블록의 데이터 읽을 때 On 됩니다.
_RBLOCK_WR_FLAG	플래시 N 블록 쓰기	플래시 N 블록의 데이터 쓸 때 On 됩니다.
_RBLOCK_ER_FLAG	플래시 N 블록 에러	플래시 N 블록 서비스중 에러 발생을 표시합니다.
_USER_WRITE_F	사용가능 접점	프로그램에서 사용 가능한 접점
_RTC_WR	RTC RW	RTC 에 데이터 쓰고 읽어오기
_SCAN_WR	스캔 RW	스캔 값 초기화
_CHK_ANC_ERR	외부 중고장 요청	외부기기에서 중고장 검출 요청
_CHK_ANC_WAR	외부 경고장 요청	외부기기에서 경고장 검출 요청
_USER_STAUS_F	사용자 접점	사용자 접점
_INIT_DONE	초기화 완료	초기화 태스크 수행 완료를 표시
_ANC_ERR	외부 중고장 정보	외부 기기의 중고장 정보를 표시

예 약 변 수	데이터타입	내 용
_ANC_WAR	외부 경고장 경보	외부 기기의 경고장 정보를 표시
_MON_YEAR_DT	월 / 년	시계 정보 데이터 (월 / 년)
_TIME_DAY_DT	시 / 일	시계 정보 데이터 (시 / 일)
_SEC_MIN_DT	초 / 분	시계 정보 데이터 (초 / 분)
_HUND_WK_DT	백년 / 요일	시계 정보 데이터 (백년 / 요일)
_ARY_IDX_ERR	배열 인덱스 범위 초과 에러 플래그	설정된 배열 개수를 초과 하였을 시 에러 플래그가 표시됩니다.
_ARY_IDX_LER	배열 인덱스 범위 초과 래치 에러 플래그	설정된 배열 개수를 초과 하였을 시 에러 래치 플래그가 표시됩니다.

.

부4.2. 고속링크 플래그 (* = 1~2, *** = 000~063)

예 약 변 수	데이터타입	내 용
_HS*_RLINK	BOOL	고속 링크 *번의 모든 국 정상 동작
_HS*_LTRBL	BOOL	_HS*RLINK ON 이후 비정상 상태 표시
_HS*_STATE***	BOOL	고속링크 *번의 ***번 블록의 종합적 상태표시
_HS*_MOD***	BOOL	고속링크 *번 ***번 블록 국의 런 운전 모드
_HS*_TRX***	BOOL	고속링크 *번 ***번 블록 국과 정상 통신 표시
_HS*_ERR***	BOOL	고속링크 *번 ***번 블록 국의 운전 에러 모드
_HS*_SETBLOCK***	BOOL	고속링크 *번 ***번 블록 설정 표시

부4.3. P2P 플래그 (* = 0 ~ 8, ** = 0 ~ 63)

예 약 변 수	데이터타입	내 용
_P2P*_NDR**	BOOL	P2P *번 **번 블록 서비스 정상완료
_P2P*_ERR**	BOOL	P2P *번 **번 블록 서비스 비정상완료
_P2P*_STATUS**	WORD	P2P *번 **번 블록 서비스 비정상완료시 에러코드
_P2P*_SVCCNT**	DWORD	P2P *번 **번 블록 서비스 정상 수행 횟수
_P2P*_ERRCNT**	DWORD	P2P *번 **번 블록 서비스 비정상 수행 횟수

부4.4. PID 플래그 (* = 0 ~ 15, ** = 0 ~ 15)

예 약 변 수	데이터타입	내 용
_PID_MAN	WORD	PID 출력 선택(0:자동 ,1:수동)
_PID*_MAN	BOOL	PID 출력 선택(0:자동 ,1:수동) - 루프**

예 약 변 수	데이터타입	내 용
_PID_PAUSE	WORD	PID 일시정지 (0:STOP/RUN ,1:PAUSE)
_PID*_PAUSE	BOOL	PID 일시정지 (0:STOP/RUN ,1:PAUSE) - 루프**
_PID_REV	WORD	PID 동작 선택(0:정 ,1:역)
_PID*_REV	BOOL	PID 동작 선택(0:정 ,1:역) - 루프**
_PID_AW2D	WORD	PID Anti Wind-up2 금지(0:동작 ,1:금지)
_PID*_AW2D	BOOL	PID Anti Wind-up2 금지(0:동작 ,1:금지) - 루프**
_PID_REM_RUN	WORD	PID 리모트(HMI) 실행비트 (0:STOP ,1:RUN)
_PID*_REM_RUN	BOOL	PID 리모트(HMI) 실행비트 (0:STOP ,1:RUN) - 루프**
_PID_P_on_PV	WORD	PID 비례(P) 계산 소스 선택 (0:ERR, 1:PV)
_PID*_P_on_PV	BOOL	PID 비례(P) 계산 소스 선택 (0:ERR, 1:PV) - 루프**
_PID_D_on_ERR	WORD	PID 미분(D) 계산 소스 선택 (0:PV, 1:ERR)
_PID*_D_on_ERR	BOOL	PID 미분(D) 계산 소스 선택 (0:PV, 1:ERR) - 루프**
_PID_AT_EN	WORD	PID 오토튜닝 설정 (0:Disable, 1:Enable)
_PID*_AT_EN	BOOL	PID 오토튜닝 설정 (0:Disable, 1:Enable) - 루프**
_PID_PWM_EN	WORD	PID PWM 운전 허용 (0:금지, 1:허용)
_PID*_PWM_EN	BOOL	PID PWM 운전 허용 (0:금지, 1:허용) - 루프**
_PID_STD	WORD	PID 동작 상태 표시 (0:Stop, 1:Run)
_PID*_STD	WORD	PID 동작 상태 표시 (0:Stop, 1:Run) - 루프00**
_PID_ALARM	BOOL	PID P - 상수 (K_p) - 블록* 루프**
_PID*_ALARM	REAL	PID I - 상수 (T_i)[sec] - 루프**
_PID_ERROR	WORD	PID 에러 발생 표시 (0: 정상 1: 에러발생)
_PID*_ERROR	BOOL	PID 에러 발생 표시 (0: 정상 1: 에러발생) - 루프01
_PID*_SV	INT	PID 목표값 (SV) - 루프**
_PID*_T_s	WORD	PID 연산 주기 (T_s)[0.1msec] - 루프**
_PID*_K_p	REAL	PID P - 상수 (K_p) - 루프**
_PID*_T_i	REAL	PID I - 상수 (T_i)[sec] - 루프**
_PID*_T_d	REAL	PID D - 상수 (T_d)[sec] - 루프**
_PID*_d_PV_max	WORD	PID PV 변화량 제한 - 루프**
_PID*_d_MV_max	WORD	PID MV 변화량 제한 - 루프**
_PID*_MV_max	INT	PID MV 최대값 제한 - 루프**
_PID*_MV_min	INT	PID MV 최소값 제한 - 루프**
_PID*_MV_man	INT	PID 수동 출력 (MV_man) - 루프**
_PID*_PV	INT	PID 현재값 (PV) - 루프**
_PID*_PV_old	INT	PID 이전 현재값 (PV_old) - 루프**
_PID*_MV	INT	PID 출력값 (MV) - 루프**

예 약 변 수	데이터타입	내 용
_PID*_ERR	DINT	PID 제어 에러값 - 루프**
_PID*_MV_p	REAL	PID 출력값 P 성분 - 루프**
_PID*_Mv_i	REAL	PID 출력값 I 성분 - 루프**
_PID*_MV_d	REAL	PID 출력값 D 성분 - 루프**
_PID*_DB_W	WORD	PID 데드밴드 설정 (안정화 후 동작) - 루프**
_PID*_Td_lag	WORD	PID 미분 함수 LAG 필터 - 루프**
_PID*_PWM	WORD	PID PWM 접점설정값 - 루프**
_PID*_PWM_Prd	WORD	PID PWM 출력 주기 - 루프**
_PID*_SV_RAMP	WORD	PID 목표값 균배 설정값 - 루프**
_PID*_PV_Track	WORD	PID 현재값 추종 설정값 - 루프**
_PID*_PV_MIN	INT	PID 현재값 입력 최소값 제한 - 루프**
_PID*_PV_MAX	INT	PID 현재값 입력 최대값 제한 - 루프**
_PID*_ALM_CODE	WORD	PID 경보 코드 - 루프**
_PID*_ERR_CODE	WORD	PID 에러 코드 - 루프**
_PIDOO_CUR_SV	INT	PID 현재 목표값 (SV) - 루프**
_AT_REV	WORD	AT 동작 선택(0:정, 1:역)
_AT*_REV	BOOL	AT 동작 선택(0:정, 1:역) - 루프**
_AT_PWM_EN	WORD	AT PWM 운전 허용 (0:금지, 1:허용)
_AT*_PWM_EN	BOOL	AT PWM 운전 허용 (0:금지, 1:허용) - 루프**
_AT_ERROR	WORD	AT 에러발생 표시(0:정상, 1:에러발생)
_AT*_ERROR	BOOL	AT 에러발생 표시(0:정상, 1:에러발생) - 루프**
_AT*_SV	INT	AT 목표값 (SV) - 루프**
_AT*_T_s	WORD	AT 연산 주기 (T_s)[0.1msec] - 루프**
_AT*_MV_max	INT	AT MV 최대값 제한 - 루프**
_ATOO_MV_min	INT	AT MV 최소값 제한 - 루프**
_AT*_PWM	WORD	AT PWM 접점설정값 - 루프**
_AT*_PWM_Prd	WORD	AT PWM 출력 주기 - 루프 **
_AT*_HYS_val	WOPD	AT 히스테리시스 설정 - 루프**
_AT*_STATUS	WORD	AT 오토튜닝 상태 표시 (사용자 설정 금지) - 루프**
_AT*_ERR_CODE	WORD	AT 에러코드 - (사용자 설정 금지) - 루프**
_AT*_K_p	REAL	AT 결과 P - 상수 (K_p) - 루프**
_AT*_T_i	REAL	AT 결과 I - 상수 (T_i)[sec] - 루프**
_AT*_T_d	REAL	AT 결과 D - 상수 (T_d)[sec] - 루프 OO
_AT*_PV	INT	AT 현재값 - 루프**
_AT*_MV	INT	AT 출력값 - 루프**

부4.5. 고속카운터 플래그 (* = 0 ~ 7, ** = 0 ~ 7)

예 약 변 수	데이터타입	내 용
_HSC*_Cnt_En	BOOL	채널** 카운터 사용
_HSC*_IntPrs_En	BOOL	채널** 카운터 내부 프리셋 사용
_HSC*_DecCnt_En	BOOL	채널** 감산카운터 지정
_HSC*_Cmp0_En	BOOL	채널** 비교출력 0 출력 허용
_HSC*_Rpu_En	BOOL	채널** 단위시간당 회전수 사용
_HSC*_Latch_En	BOOL	채널** 래치 카운터 사용
_HSC*_Cmp1_En	BOOL	채널** 비교출력 1 출력 허용
_HSC*_Carry	BOOL	채널** Carry 신호
_HSC*_Borrow	BOOL	채널** Borrow 신호
_HSC*_CmpOut0	BOOL	채널** 비교출력 0 출력 신호
_HSC*_CmpOut1	BOOL	채널** 비교출력 1 출력 신호
_HSC*_CurCnt	DINT	채널** 현재 카운트값
_HSC*_CurRpu	DINT	채널** 단위시간당 회전수
_HSC*_ErrCode	DINT	채널** 에러코드
_HSC*_CntMode	INT	채널** 카운터 모드
_HSC*_PlsMode	INT	채널** 펄스입력 모드
_HSC*_CmpMode0	WORD	채널** 비교출력 0 종류
_HSC*_CmpMode1	WORD	채널** 비교출력 1 종류
_HSC*_IntPrs_Val	DINT	채널** 내부 프리셋 설정값
_HSC*_ExtPrs_Val	DINT	채널** 외부 프리셋 설정값
_HSC*_RingMin_Val	DINT	채널** 링 카운터 최소 설정값
_HSC*_RingMax_Val	DINT	채널** 링 카운터 최대 설정값
_HSC*_CmpMin_Val0	DINT	채널** 비교출력 0 최소 설정값
_HSC*_CmpMax_Val0	DINT	채널** 비교출력 0 최대 설정값
_HSC*_CmpMin_Val1	DINT	채널** 비교출력 1 최소 설정값
_HSC*_CmpMax_Val1	DINT	채널** 비교출력 1 최대 설정값
_HSC*_CmpContact0	WORD	채널** 비교출력 0 출력접점 지정
_HSC*_CmpContact1	WORD	채널** 비교출력 1 출력접점 지정
_HSC*_UnitTime	WORD	채널** 단위시간 설정값
_HSC*_PlsPerRev	INT	채널** 1회전당 펄스 수

부4.6. 위치결정 플래그 (* = 0 ~ 80, ** = 0 ~ 80)

예 약 변 수	데이터타입	내 용
_POS_X_Busy	BOOL	X 축 BUSY
_POS_Y_Busy	BOOL	Y 축 BUSY
_POS_X_Err	BOOL	X 축 에러
_POS_Y_Err	BOOL	Y 축 에러
_POS_X_Done	BOOL	X 축 위치결정완료
_POS_Y_Done	BOOL	Y 축 위치결정완료
_POS_X_McodeOn	BOOL	X 축 M 코드 On
_POS_Y_McodeOn	BOOL	Y 축 M 코드 On
_POS_X_OriginFix	BOOL	X 축 원점결정
_POS_Y_OriginFix	BOOL	Y 축 원점결정
_POS_X_OutInhibit	BOOL	X 축 출력금지
_POS_Y_OutInhibit	BOOL	Y 축 출력금지
_POS_X_Stop	BOOL	X 축 정지상태
_POS_Y_Stop	BOOL	Y 축 정지상태
_POS_X_ULimit	BOOL	X 축 상한검출
_POS_Y_ULimit	BOOL	Y 축 상한검출
_POS_X_LLimit	BOOL	X 축 하한검출
_POS_Y_LLimit	BOOL	Y 축 하한검출
_POS_X_Estop	BOOL	X 축 비상정지
_POS_Y_Estop	BOOL	Y 축 비상정지
_POS_X_Dir	BOOL	X 축 정/역회전
_POS_Y_Dir	BOOL	Y 축 정/역회전
_POS_X_Acc	BOOL	X 축 운전상태(가속중)
_POS_Y_Acc	BOOL	Y 축 운전상태(가속중)
_POS_X_Const	BOOL	X 축 운전상태(정속중)
_POS_Y_Const	BOOL	Y 축 운전상태(정속중)
_POS_X_Dec	BOOL	X 축 운전상태(감속중)
_POS_Y_Dec	BOOL	Y 축 운전상태(감속중)
_POS_X_Dwell	BOOL	X 축 운전상태(드웰중)
_POS_Y_Dwell	BOOL	Y 축 운전상태(드웰중)
_POS_X_Position	BOOL	X 축 운전제어형태(위치제어중)
_POS_Y_Position	BOOL	Y 축 운전제어형태(위치제어중)
_POS_X_Speed	BOOL	X 축 운전제어형태(속도제어중)
_POS_Y_Speed	BOOL	Y 축 운전제어형태(속도제어중)

예 약 변 수	데이터타입	내 용
_POS_X_LinearInt	BOOL	X 축 운전제어형태(직선보간중)
_POS_Y_LinearInt	BOOL	Y 축 운전제어형태(직선보간중)
_POS_X_Home	BOOL	X 축 원점복귀
_POS_Y_Home	BOOL	Y 축 원점복귀
_POS_X_PosSync	BOOL	X 축 위치동기
_POS_Y_PosSync	BOOL	Y 축 위치동기
_POS_X_SpdSync	BOOL	X 축 속도동기
_POS_Y_SpdSync	BOOL	Y 축 속도동기
_POS_X_JogLow	BOOL	X 축 조그저속
_POS_Y_JogLow	BOOL	Y 축 조그저속
_POS_X_JogHigh	BOOL	X 축 조그고속
_POS_Y_JogHigh	BOOL	Y 축 조그고속
_POS_X_Inching	BOOL	X 축 인칭운전
_POS_Y_Inching	BOOL	Y 축 인칭운전
_POS_X_CurPos	DWORD	X 축 현재위치
_POS_Y_CurPos	DWORD	Y 축 현재위치
_POS_X_CurSpd	DWORD	X 축 현재속도
_POS_Y_CurSpd	DWORD	Y 축 현재속도
_POS_X_CurStep	WORD	X 축 스텝번호
_POS_Y_CurStep	WORD	Y 축 스텝번호
_POS_X_ErrCode	WORD	X 축 에러코드
_POS_Y_ErrCode	WORD	Y 축 에러코드
_POS_X_Mcode	WORD	X 축 M 코드
_POS_Y_Mcode	WORD	Y 축 M 코드
_POS_X_Start	BOOL	X 축 기동
_POS_Y_Start	BOOL	Y 축 기동
_POS_X_CwJogStart	BOOL	X 축 정방향 조그 기동
_POS_Y_CwJogStart	BOOL	Y 축 정방향 조그 기동
_POS_X_CcwJogStart	BOOL	X 축 역방향 조그 기동
_POS_Y_CcwJogStart	BOOL	Y 축 역방향 조그 기동
_POS_X_JogLowHigh	BOOL	X 축 조그 저속/고속
_POS_Y_JogLowHigh	BOOL	Y 축 조그 저속/고속
_POS_X_BiasSpd	DWORD	X 축 바이어스 속도
_POS_Y_BiasSpd	DWORD	Y 축 바이어스 속도
_POS_X_SpdLimit	DWORD	X 축 속도 제한치

예 약 변 수	데이터타입	내 용
_POS_Y_SpdLimit	DWORD	Y 축 속도 제한치
_POS_X_AccTime1	WORD	X 축 가속 시간 1
_POS_Y_AccTime1	WORD	Y 축 가속 시간 1
_POS_X_DecTime1	WORD	X 축 감속 시간 1
_POS_Y_DecTime1	WORD	Y 축 감속 시간 1
_POS_X_AccTime2	WORD	X 축 가속 시간 2
_POS_Y_AccTime2	WORD	Y 축 가속 시간 2
_POS_X_DecTime2	WORD	X 축 감속 시간 2
_POS_Y_DecTime2	WORD	Y 축 감속 시간 2
_POS_X_AccTime3	WORD	X 축 가속 시간 3
_POS_Y_AccTime3	WORD	Y 축 가속 시간 3
_POS_X_DecTime3	WORD	X 축 감속 시간 3
_POS_Y_DecTime3	WORD	Y 축 감속 시간 3
_POS_X_AccTime4	WORD	X 축 가속 시간 4
_POS_Y_AccTime4	WORD	Y 축 가속 시간 4
_POS_X_DecTime4	WORD	X 축 감속 시간 4
_POS_Y_DecTime4	WORD	Y 축 감속 시간 4
_POS_X_SwULimit	DWORD	X 축 소프트 상한
_POS_Y_SwULimit	DWORD	Y 축 소프트 상한
_POS_X_SwLLimit	DWORD	X 축 소프트 하한
_POS_Y_SwLLimit	DWORD	Y 축 소프트 하한
_POS_X_Backlash	WORD	X 축백래쉬 보정량
_POS_Y_Backlash	WORD	Y 축백래쉬 보정량
_POS_X_McodeMode_L	BOOL	X 축 M 코드 출력모드(Low 비트)
_POS_Y_McodeMode_L	BOOL	Y 축 M 코드 출력모드(Low 비트)
_POS_X_McodeMode_H	BOOL	X 축 M 코드 출력모드(High 비트)
_POS_Y_McodeMode_H	BOOL	Y 축 M 코드 출력모드(High 비트)
_POS_X_LimitDetect	BOOL	X 축 등속 운전중 스프트 상하한 검출
_POS_Y_LimitDetect	BOOL	Y 축 등속 운전중 스프트 상하한 검출
_POS_X_HomeAddr	DWORD	X 축 원점 어드레스
_POS_Y_HomeAddr	DWORD	Y 축 원점 어드레스
_POS_X_HomeHSpd	DWORD	X 축 원점 복귀 고속속도
_POS_Y_HomeHSpd	DWORD	Y 축 원점 복귀 고속속도
_POS_X_HomeLSpd	DWORD	X 축 원점 복귀 저속속도
_POS_Y_HomeLSpd	DWORD	Y 축 원점 복귀 저속속도

예 약 변 수	데이터타입	내 용
_POS_X_HomeAccTime	WORD	X 축 원점 복귀 가속 시간
_POS_Y_HomeAccTime	WORD	Y 축 원점 복귀 가속 시간
_POS_X_HomeDccTime	WORD	X 축 원점 복귀 감속 시간
_POS_Y_HomeDccTime	WORD	Y 축 원점 복귀 감속 시간
_POS_X_HomeDwlTime	WORD	X 축 원점 복귀 드웰 시간
_POS_Y_HomeDwlTime	WORD	Y 축 원점 복귀 드웰 시간
_POS_X_HomeMethod_L	BOOL	X 축 원점 복귀 방법(Low 비트)
_POS_Y_HomeMethod_L	BOOL	Y 축 원점 복귀 방법(Low 비트)
_POS_X_HomeMethod_H	BOOL	X 축 원점 복귀 방법(High 비트)
_POS_Y_HomeMethod_H	BOOL	Y 축 원점 복귀 방법(High 비트)
_POS_X_HomeDir	BOOL	X 축 원점 복귀 방향
_POS_Y_HomeDir	BOOL	Y 축 원점 복귀 방향
_POS_X_JogHSpd	DWORD	X 축 조그 고속 속도
_POS_Y_JogHSpd	DWORD	Y 축 조그 고속 속도
_POS_X_JogLSpd	DWORD	X 축 조그 저속 속도
_POS_Y_JogLSpd	DWORD	Y 축 조그 저속 속도
_POS_X_JogAccTime	WORD	X 축 조그 가속 시간
_POS_Y_JogAccTime	WORD	Y 축 조그 가속 시간
_POS_X_JogDecTime	WORD	X 축 조그 감속 시간
_POS_Y_JogDecTime	WORD	Y 축 조그 감속 시간
_POS_X_JogInchSpd	WORD	X 축 인칭속도
_POS_Y_JogInchSpd	WORD	Y 축 인칭속도
_POS_X_Position_En	BOOL	X 축 위치결정 사용
_POS_Y_Position_En	BOOL	Y 축 위치결정 사용
_POS_X_OutLevel	BOOL	X 축 펄스 출력 레벨
_POS_Y_OutLevel	BOOL	Y 축 펄스 출력 레벨
_POS_X_Limit_En	BOOL	X 축 상하한 리미트 사용
_POS_Y_Limit_En	BOOL	Y 축 상하한 리미트 사용
_POS_X_OutMode	BOOL	X 축 펄스 출력 모드
_POS_Y_OutMode	BOOL	Y 축 펄스 출력 모드
_POS_X_ST*_Addr	DWORD	X 축 스텝** 목표위치
_POS_Y_ST*_Speed	DWORD	Y 축 스텝** 운전속도
_POS_X_ST*_Dwell	WORD	X 축 스텝** 드웰시간
_POS_Y_ST*_Dwell	WORD	Y 축 스텝** 드웰시간
_POS_X_ST*_Mcode	WORD	X 축 스텝** M 코드 번호

예 약 변 수	데이터타입	내 용
_POS_Y_ST*_Mcode	WORD	Y 축 스텝** M 코드 번호
_POS_X_ST*_Method	BOOL	X 축 스텝 01 운전방식
_POS_Y_ST*_Method	BOOL	Y 축 스텝 01 운전방식
_POS_X_ST*_Control	BOOL	X 축 스텝** 제어방식
_POS_Y_ST*_Control	BOOL	Y 축 스텝** 제어방식
_POS_X_ST*_Pattern_L	BOOL	X 축 스텝** 운전패턴(Low 비트)
_POS_Y_ST*_Pattern_L	BOOL	Y 축 스텝** 운전패턴(Low 비트)
_POS_X_ST*_Pattern_H	BOOL	X 축 스텝** 운전패턴(High 비트)
_POS_Y_ST*_Pattern_H	BOOL	Y 축 스텝** 운전패턴(High 비트)
_POS_X_ST*_Cordi	BOOL	X 축 스텝** 좌표
_POS_Y_ST*_Cordi	BOOL	Y 축 스텝** 좌표
_POS_X_ST*_AccDecN_L	BOOL	X 축 스텝** 가감속번호(Low 비트)
_POS_Y_ST*_AccDecN_L	BOOL	Y 축 스텝** 가감속번호(Low 비트)
_POS_X_ST*_AccDecN_H	BOOL	X 축 스텝** 가감속번호(High 비트)
_POS_Y_ST*_AccDecN_H	BOOL	Y 축 스텝** 가감속번호(High 비트)
_POS_X_ST01_RptStep	BOOL	X 축 스텝** 반복스텝
_POS_Y_ST01_RptStep	BOOL	Y 축 스텝** 반복스텝

보증 내용

1. 보증 기간
   구입하신 제품의 보증 기간은 제조일로부터 18 개월입니다.

2. 보증 범위
   위의 보증 기간 중에 발생한 고장에 대해서는 부분적인 교환 또는 수리를 받으실 수 있습니다. 다만, 아래에 해당하는 경우에는 그 보증 범위에서 제외하오니 양지하여 주시기 바랍니다.

   (1) 사용설명서에 명기된 이외의 부적당한 조건 · 환경 · 취급으로 발생한 경우
   (2) 고장의 원인이 당사의 제품 이외의 것으로 발생한 경우
   (3) 당사 및 당사가 정한 지정점 이외의 장소에서 개조 및 수리를 한 경우
   (4) 제품 본래의 사용 방법이 아닌 경우
   (5) 당사에서 출하 시 과학 · 기술의 수준에서는 예상이 불가능한 사유에 의한 경우
   (6) 기타 천재 · 화재 등 당사측에 책임이 없는 경우

3. 위의 보증은 PLC 단위체만의 보증을 의미하므로 시스템 구성이나 제품응용 시에는 안전성을 고려하여 사용하여 주십시오.

환경 방침

LS 산전은 다음과 같이 환경 방침을 준수하고 있습니다.

환 경 경 영

LS산전은 환경보전을
경영의 우선과제로 하며,
전 임직원은 쾌적한 지구환경보전을
위해 최선을 다한다.

제품 폐기에 대한 안내

LS산전 PLC는 환경을 보호할 수
있도록 설계된 제품입니다.
제품을 폐기할 경우 알루미늄, 철
합성수지(커버)류로 분리하여 재활용
할 수 있습니다.

한번 맺은 인연을 가장 소중히 여깁니다!

품질과 더불어 고객 서비스를 최우선으로 여기는 LS 산전은
소비자를 위한 소비자에 의한 기업임을 굳게 다짐하며
고객 여러분의 만족을 위해 최선을 다하겠습니다.

www.lsis.com

LS산전주식회사

10310000739

■ 전국영업망 전화번호
서울 : 경기도 안양시 동안구 호계동 1026-6번지 LS타워(4F~10F)
(우)431-848 http://www.lsis.com

■ 구입 문의

서울 영업	TEL:(02)2034-4620~34	FAX:(02)2034-4622
부산 영업	TEL:(051)310-6855~60	FAX:(051)310-6851
대구 영업	TEL:(053)603-7740~7	FAX:(053)603-7788
서부 영업(광주)	TEL:(062)510-1885~91	FAX:(062)526-3262
서부 영업(대전)	TEL:(042)820-4240~42	FAX:(042)820-4298
서부 영업(전주)	TEL:(063)271-4012	FAX:(063)271-2613

■ A/S 문의

고객지원팀	TEL:(031)689-7112	FAX:(031)689-7113
천안 고객지원	TEL:(041)550-8308~9	FAX:(041)554-3949
부산 고객지원	TEL:(051)310-6922~3	FAX:(051)310-6851
대구 고객지원	TEL:(053)603-7751~4 TEL:(053)383-2083	FAX:(053)603-7788
광주 고객지원	TEL:(062)510-1883,1892	FAX:(062)526-3262

서비스 신고요령 LS산전의 PLC를 사용 중 이상이 생겼거나
의문이 있으면 서비스 대표 전화로 연락 하십시오.

서비스 대표전화 (전국 어디서나)1544-2080

■ 기술 문의

고객상담센터	TEL: 080-777-2080 (수신자부담) TEL : 1544-2080 FAX : (041)550-8600

■ 기술 지정점

동현 산전(안양)	TEL:(031)479-4785~6	FAX:(031)456-4524
신광 ENG(부산)	TEL:(051)319-1051	FAX:(051)319-1052
에이엔디시스템(부산)	TEL:(051)319-4939	FAX:(051)319-4938
LS-WILL(구미)	TEL:(054)454-7909	FAX:(054)473-3909
나노오토메이션(대전)	TEL:(042)636-8015	FAX:(042)636-8016

■ 교육 문의

LS산전 연수원	TEL:(043)268-2631~2	FAX:(043)268-2633~4
서울/경기교육장	TEL : (031)689-7101	FAX:(031)689-7113
부산 교육장	TEL : (051)310-6860	FAX:(051)310-6851
대구 교육장	TEL : (053)603-7744	FAX:(053)603-7788

■ 서비스 지정점

명 산전(서울)	TEL:(02)462-3053	FAX : (02)462-3054
TPI시스템(서울)	TEL : (02)895-4803~4	FAX : (02)6264-3054
우진 산전(의정부)	TEL : (031)877-8273	FAX : (031)878-8279
신진시스템(안산)	TEL : (031)495-9606	FAX : (031)494-9606
태영시스템(대전)	TEL : (042)670-7363	FAX : (042)670-7364
서진 산전 (울산)	TEL : (052)227-0335	FAX : (052)227-0337
동남 산전 (창원)	TEL : (055)265-0371	FAX : (055)265-0373
대명시스템(대구)	TEL : (053)564-4370	FAX : (053)564-4371
정석시스템(광주)	TEL : (062)526-4151	FAX : (062)526-4152
코리아산전(익산)	TEL : (063)835-2411	FAX : (063)831-1411
파란자동화(천안)	TEL : (041)579-8308	FAX : (041)579-8309

● 본 설명서에 기재된 제품은 예고 없이 단종이나 제품에 변동이 있을 수 있으므로 구입시 반드시 확인 바랍니다.
● 제품 사용 중 이상이 생겼거나 불편한 점은 LS산전으로 문의 바랍니다.

2014. 4