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.

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 에러로부터 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 100㎜ 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

Version	일자	주요 변경 내용	수정 Page
V 1.0	'06.12	초판 발행	-

※ 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 Download받으실 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.	10310000511
XGK 명령어집	XGK CPU를 장착한 PLC시스템에서 사용하는 전체적인 명령어에 대해서 사용방법을 설명한 프로그래밍하기 위한 사용설명서입니다.	10310000509

제 1 장 개요

이 언어 설명서는 XGI PLC를 지원하는 언어에 대한 설명서입니다.
XGI 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

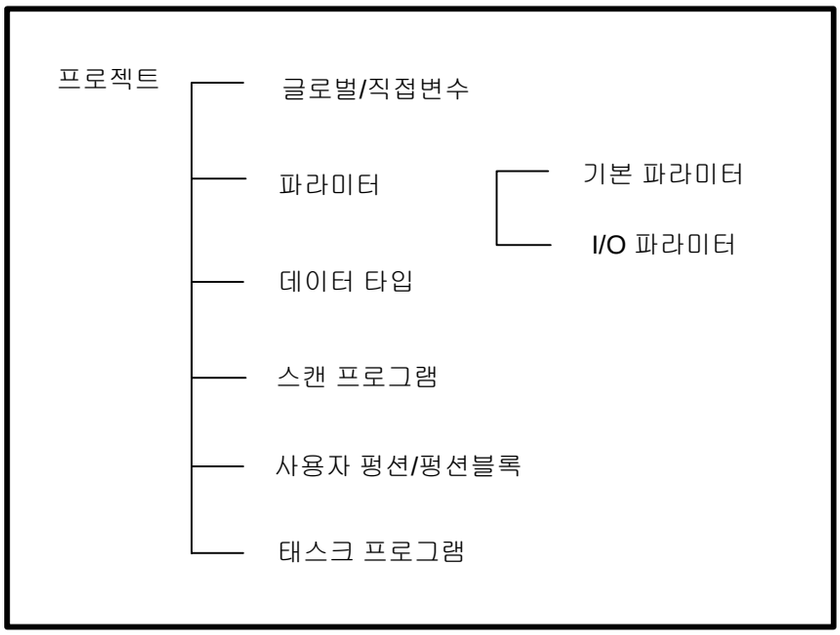
IEC에서 표준화한 PLC용 언어는 두 개의 도식 언어와 두 개의 문자식 언어, 그리고 SFC로 이루어져 있습니다.

- ▷ 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)

제 2 장 소프트웨어 구조

2.1 개요

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



2.2 프로젝트(Project)

▷ XGI 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.

제 3 장 공통 요소

XGI PLC 의 프로그램 구성 요소(프로그램, 펑션, 펑션 블록)는 LD, SFC 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 PLC 에서 데이터로 사용하는 것은 숫자(Numeric Literals)와 문자열(Character String), 시간 문자(Time Literals) 등 입니다.

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 붙이지 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	TOD#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.

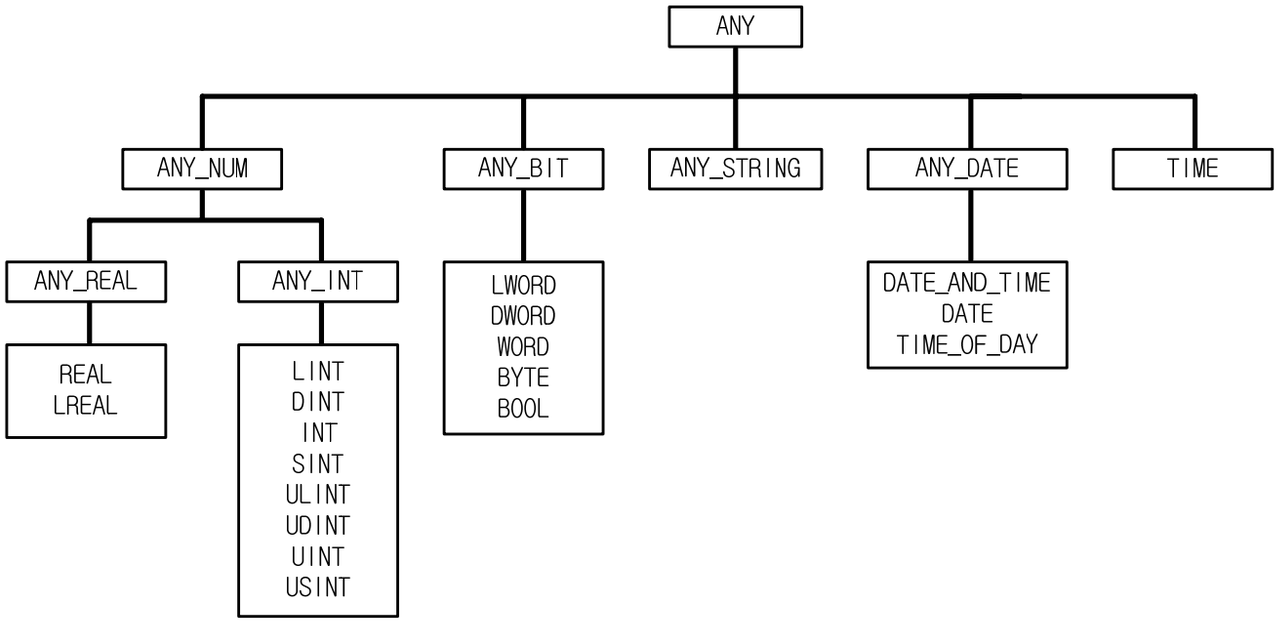
3.2.1 기본 데이터 타입

XGI 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	TOD#00:00:00 ~ TOD#23:59:59.999
14	DATE_AND_TIME	Date And Time Of Day	64	DT#1984-01-01-00:00:00 ~ DT#2163-12-31-23:59:59.999
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 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

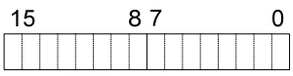
BOOL

1 비트, 범위: 0, 1

USINT

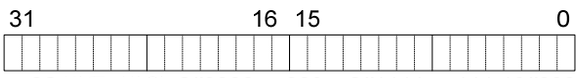
8 비트 , 범위: 2#0000_0000 ~ 2#1111_1111, 16#00 ~ 16#FF

WORD



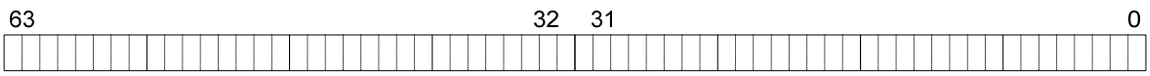
16 비트, 범위: 2#0000_0000 _0000_0000 ~ 2#1111_1111_1111_1111 16#0000 ~ 16#FFFF

DWORD



32 비트, 범위: 2#0000_...000 ~ 2#1111_...111 16#00000000 ~ 16#FFFFFFFF

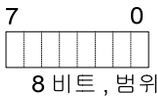
LWORD



64 비트, 범위: 2#0000_...000 ~ 2#1111_...111, 16#0000000000000000 ~ 16#FFFFFFFFFFFFFFFF

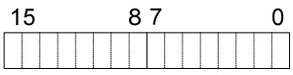
 # Unsigned Integer

USINT



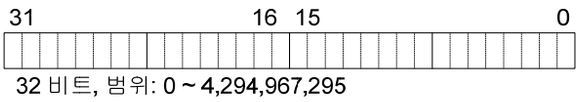
8 비트 , 범위: 0 ~ 255

UINT



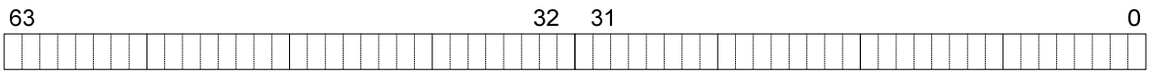
16 비트, 범위: 0 ~ 65,535

UDINT



32 비트, 범위: 0 ~ 4,294,967,295

ULINT



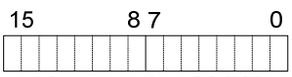
64 비트, 범위: 0 ~ $2^{64}-1$

 # Integer (음수는 2' Complement 표현)

SINT

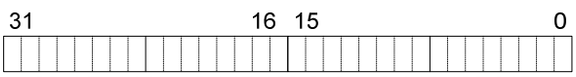
8비트, 범위: −128 ~ 127

INT



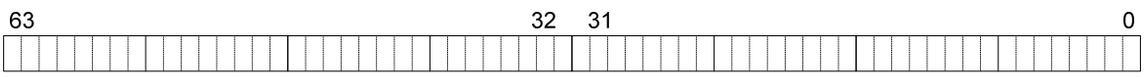
16비트, 범위: -32,768 ~ 32,767

DINT



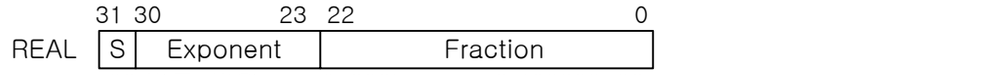
32비트. 범위: -2,147,483,648 ~ 2,147,483,647

LINT

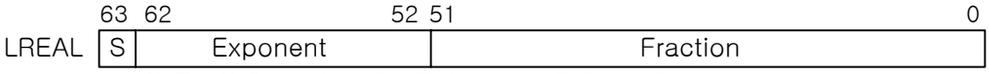


64비트, 범위: -2^{63} ~ $2^{63}-1$

Real (IEEE Standard 754-1984 기준)



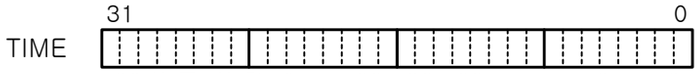
32 비트, 범위: 1.401298E-45 ~ 3.402823E38



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

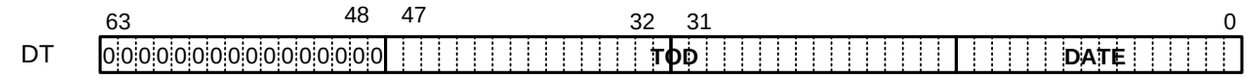
- S: 부호표시 (0일 때 양수, 1일 때 음수)
- Exponent: 2의 승수부 (2^{e-127}: $e=b_{30}b_{29}...b_{23}$, $e=b_{62}b_{61}...b_{52}$)
- Fraction: 소수점 이하 값 (Fraction: $f=b_{22}b_{21}...b_0$, $f=b_{51}b_{52}...b_0$)

Time

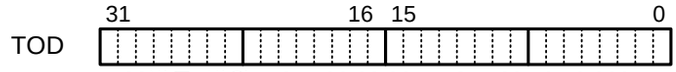


32 비트, 범위: 0 ~ 4,294,967,295ms ,T#49d17h2m47s295ms

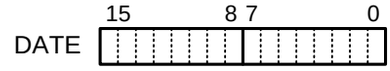
Date



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

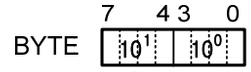


32비트, 범위: TOD#00:00:00 ~ TOD#23:59:59.999

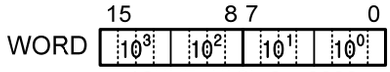


16비트, 범위: D#1984-01-01 ~ D#2163-6-6

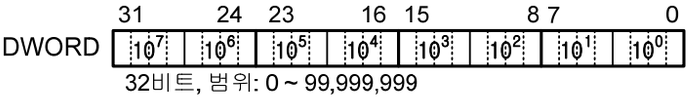
#BCD



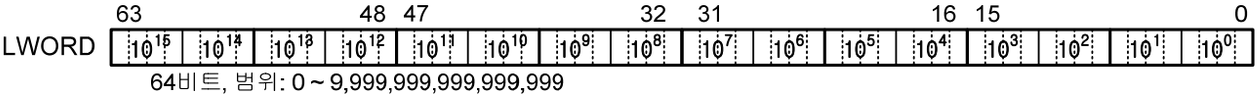
8비트, 범위: 0 ~ 99



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
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

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.

예

B 라는 사용자 펑션이 있는데 이 사용자 펑션의 내용이
출력 1 <= IN1 + IN2 + Val
Val <= 출력 1 (여기서 Val 은 내부 변수)
이라면 Val 이라는 내부 변수가 있기 때문에 이것은 사용자 펑션이 아닙니다. 내부 변수가 있다는 것은 입력이 같아도 출력이 다를 수 있다는 말과 같습니다. 위의 예에서 출력 1 값은 IN1 과 IN2 값이 일정하더라도 Val 변수 때문에 달라질 수 있습니다. 위의 A 라는 사용자 펑션과 비교하면 A 라는 사용자 펑션에서 만약 IN1 이 20, IN2 가 30 이라면 출력 1 값은 언제나 150 입니다. 입력이 일정하면 출력 값도 일정하다는 것을 알 수 있습니다.

▷ 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.

예

사용자 펑션 이름이 WEIGH 이고 그 사용자 펑션의 결과값의 데이터 타입이 WORD 라면 사용자 펑션 내부에 이름이 WEIGH 이고 데이터 타입이 WORD 인 변수가 하나 자동적으로 생성됩니다. 사용자는 사용자 펑션의 연산 결과를 이 WEIGH 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 인스

턴스를 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.

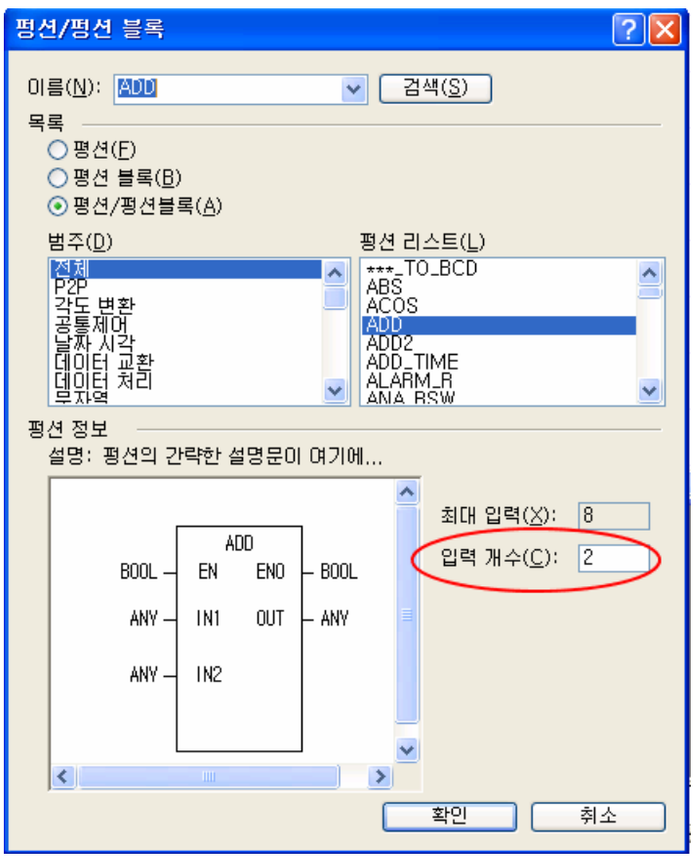
▷ 프로그램에는 입/출력변수가 없습니다.

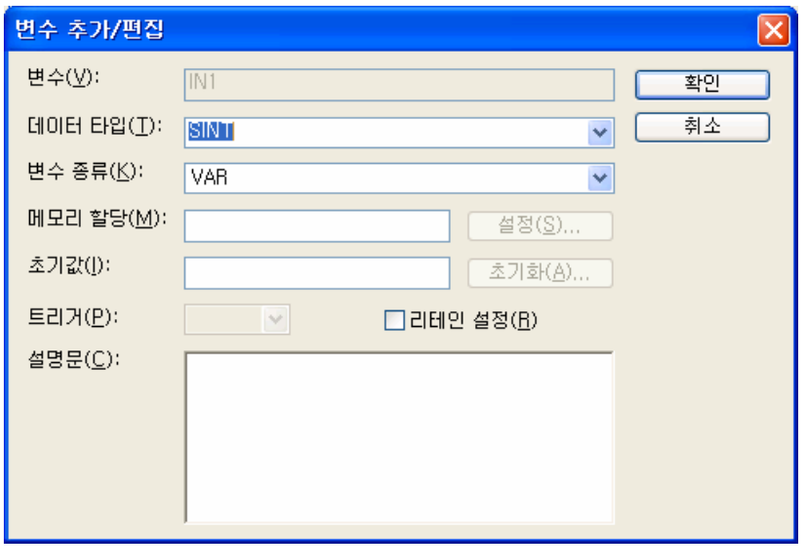
▷ 프로그램에는 사용자 펑션 및 사용자 펑션 블록을 호출할 수 있습니다.

3.5 명령어 선정

3.5.1 내부적으로 결정되는 명령어

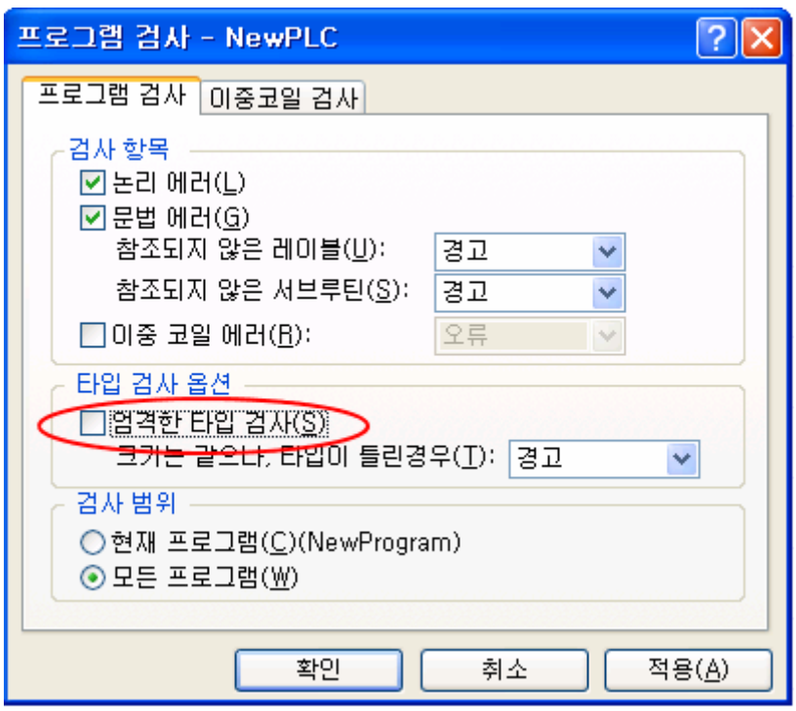
▷ 펑션의 경우 하나의 이름을 갖지만 여러 종류의 변수타입을 입력할 수 있는 명령어는 사용되는 변수에 따라서 내부적으로 여러 명령어로 구분됩니다. 예를 들어 ADD 의 경우, 설정한 입력개수 및 입력/출력 변수 타입에 따라서 여러 종류로 구분되어 처리됩니다. 아래와 같이 선택한 경우에는 래더 프로그램 상에서 보이는 명령어는 ADD 이지만, 내부적으로 ADD2_SINT 명령어를 수행하게 되는 것입니다.





▷ 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.

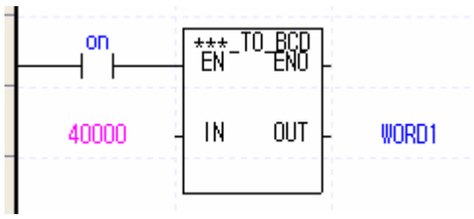
▷ IEC 에서는 같은 타입끼리의 연산만을 허용하고 있으나, XG5000 에서는 검사옵션에 엄격한 검사 옵션을 두어 변수의 크기(BYTE, WORD, DWORD, LWORD)만 같으면 연산을 허용하는 옵션이 있습니다.



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.

제 4 장 SFC(Sequential Function Chart)

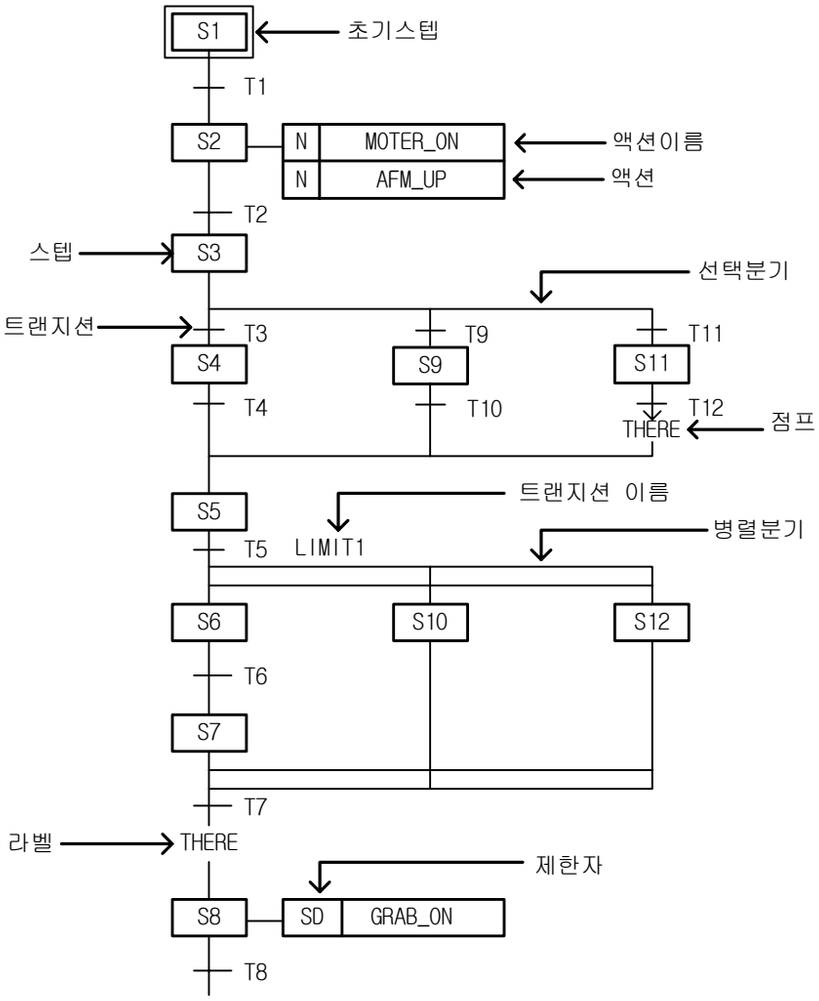
4.1 개요

▷ 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.

▷ SFC 는 상태 정보를 가지고 있어야 하기 때문에, 프로그램 종류 중 단지 프로그램과 펑션 블록만이 이 SFC 를 이용할 수 있습니다.

▷ 형태



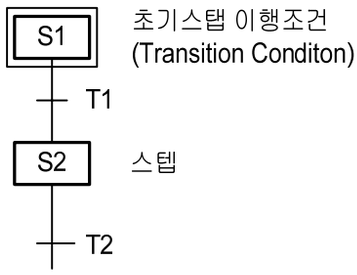
4.2 SFC 구조

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 트랜지션

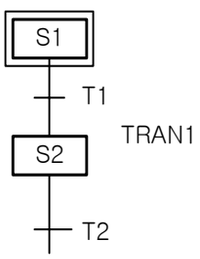
▷ 트랜지션은 스텝간의 실행 처리 이행 조건을 나타냅니다.

▷ 이행 조건은 PLC 언어인 LD 로 표현되어야 합니다.

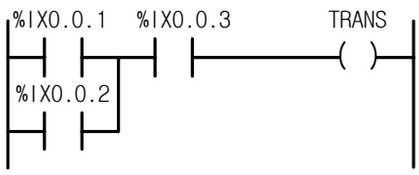
▷ 이행 조건의 결과는 항상 BOOL 로 되어야 하며, 그 변수의 이름은 어느 트랜지션이나 TRANS 가 됩니다.

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

▷ 스텝과 스텝 사이에는 반드시 트랜지션이 있어야 합니다.



TRAN1 의 내용



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.

4.2.3 액션

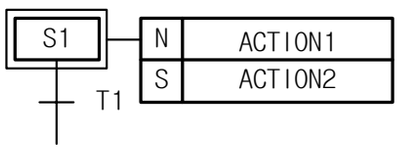
▷ 각 스텝에는 액션을 2 개까지 연결할 수 있습니다.

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

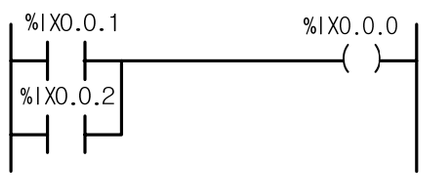
▷ 액션은 PLC 언어인 LD 로 구성되고, 스텝이 활성화될 동안 액션의 내용이 실행됩니다.

▷ 액션 제한자가 액션을 제어하는 데 사용됩니다.

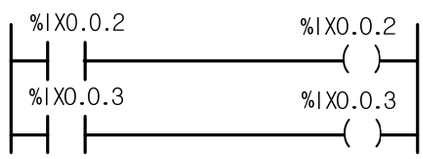
▷ 액션이 활성화되었다가 비활성화 상태로 될 때 액션에서 실행된 접점 출력은 0 으로 됩니다.
단, S, R, 펑션, 펑션 블록 출력은 비활성화되기 전의 상태를 유지합니다.



ACTION1 의 내용



ACTION2 의 내용



- 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.

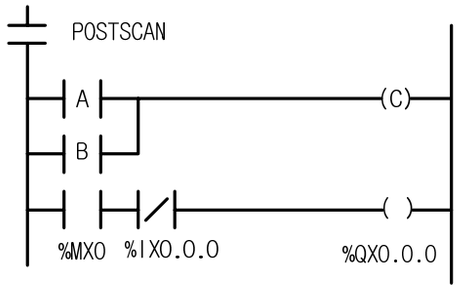
참고

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.



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

4.2.4 액션 제한자(Action Qualifier)

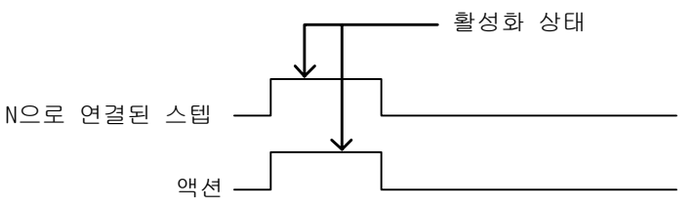
▷ 액션이 사용될 때마다 액션 제한자가 사용됩니다.

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

▷ 액션 제한자의 종류는 다음과 같습니다.

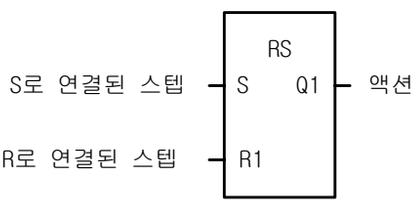
1) N(Non-Stored)

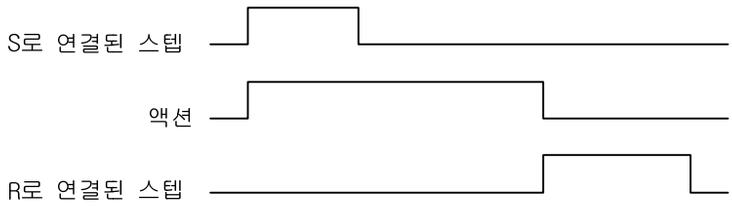
스텝이 활성화된 동안만 액션이 실행됩니다.



2) S(Set)

스텝이 활성화되면 R 제한자가 실행될 때까지 액션이 실행됩니다.



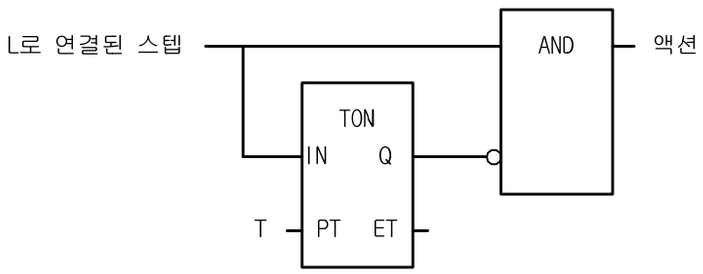


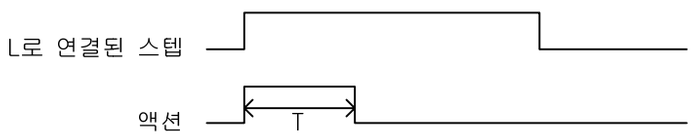
3) R(Overriding Reset)

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

4) L(Time Limited)

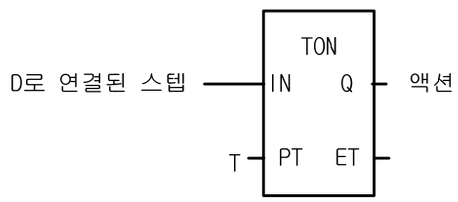
스텝이 활성화된 후 지정된 시간까지, 또는 스텝이 비활성화될 때까지 액션이 실행됩니다.

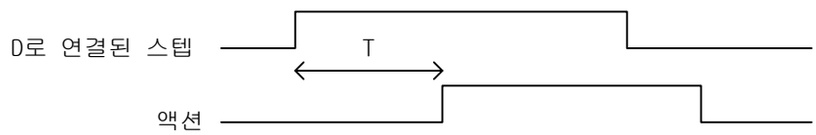




5) D(Time Delayed)

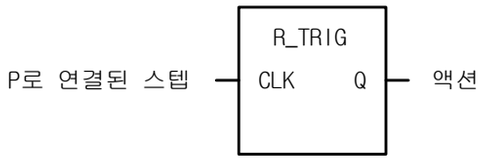
스텝이 활성화된 후 지정된 시간이 경과한 후부터 비활성화될 때까지 액션이 실행됩니다.

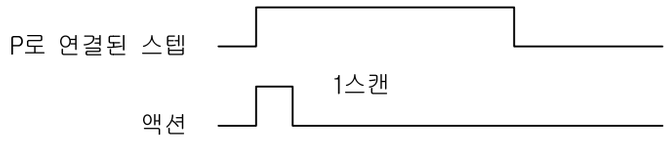




6) P(Pulse)

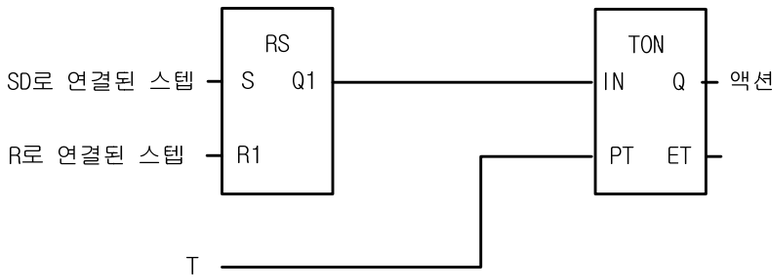
스텝이 활성화된 순간에만 액션이 실행됩니다.

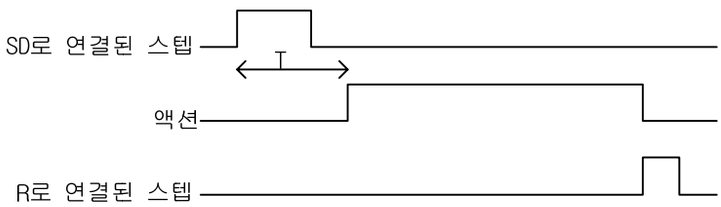




7) SD(Stored & Time Delayed)

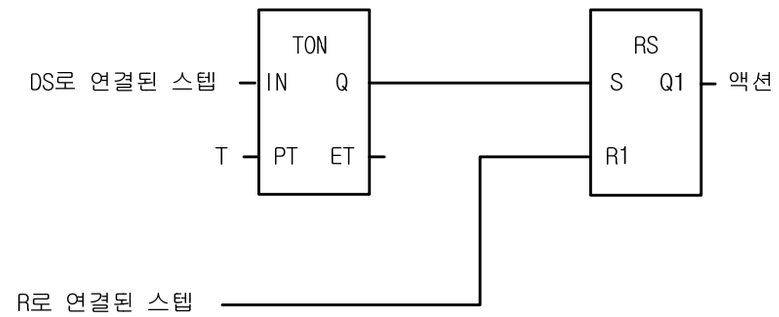
스텝이 활성화 된 후 지정된 시간이 경과한 후부터 R 제한자가 실행될 때까지 액션이 실행됩니다. 단, 시간이 경과하기 전에 R 제한자가 실행되면 액션은 실행되지 않습니다.

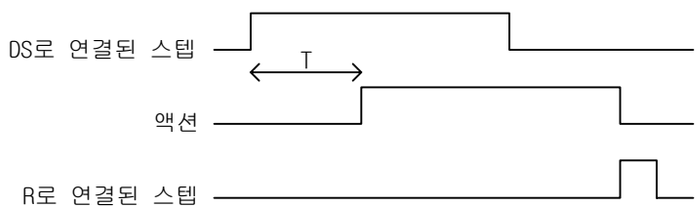




8) DS(Delayed & Stored)

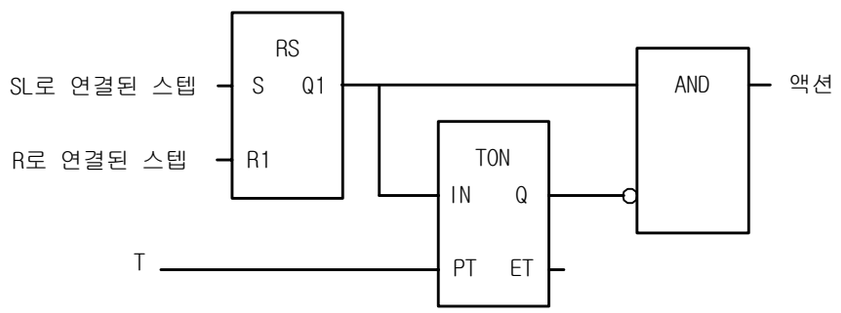
스텝이 활성화 된 후 지정된 시간이 경과한 후부터 R 제한자가 실행될 때까지 액션이 실행됩니다. 단, 시간이 경과하기 전에 스텝이 비활성화되거나 R 제한자가 실행되면 액션은 실행되지 않습니다.

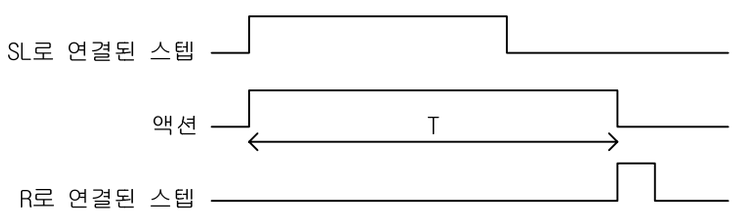




9) SL(Stored & Timed Limited)

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



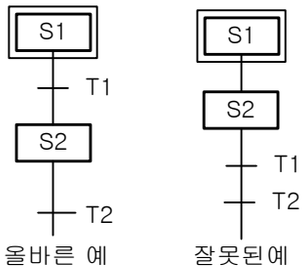


4.3 전개 규칙

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.

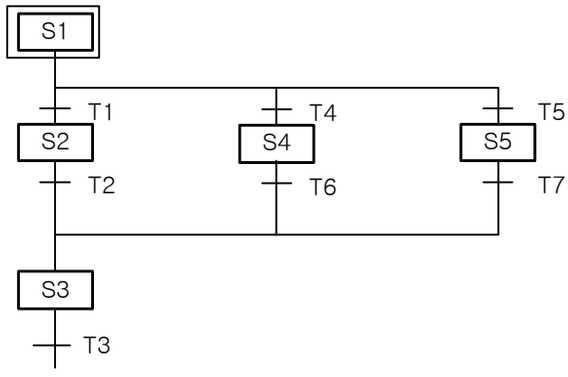


▷ 직렬로 연결되어 있는 스텝간의 이행은 상위 스텝이 활성화된 상태에서 다음에 연결된 트랜지션의 이행 조건이 1 로 되면 하위 스텝이 활성화 상태가 됩니다.

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.

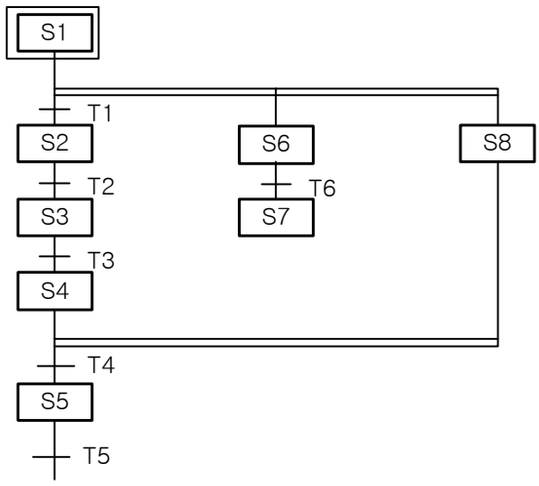
* T4, T5 의 이행 조건이 동시에 1 이 되었을 경우 S1 -> S4 -> S3 순으로 활성화 상태가 됩니다.

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.

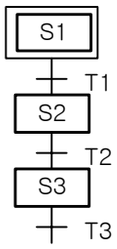
* Active 순서

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

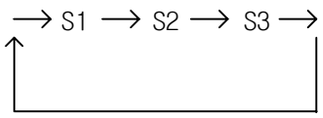
4.3.4 점프

▷ 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

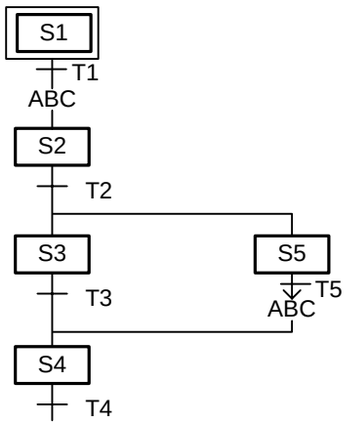


▷ 점프를 사용하면 원하는 곳으로 전개를 이어 나갈 수 있습니다.

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

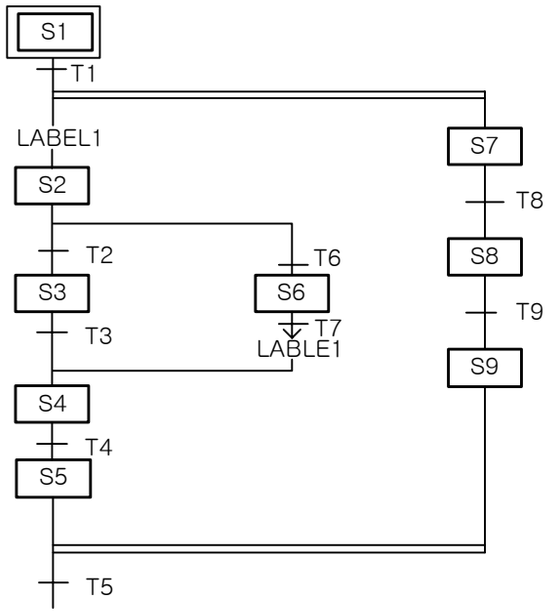
예

1) 선택분기 끝에서의 점프

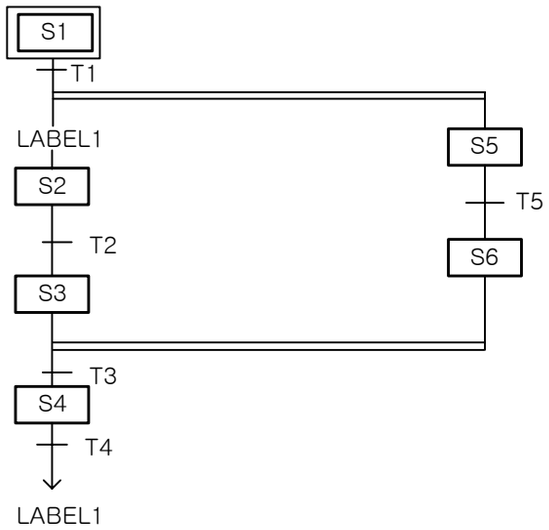


- S5 다음에는 S2 이 활성화됩니다.

2) 병렬 분기 안에서의 점프



3) 병렬 분기 안으로는 점프할 수 없습니다.

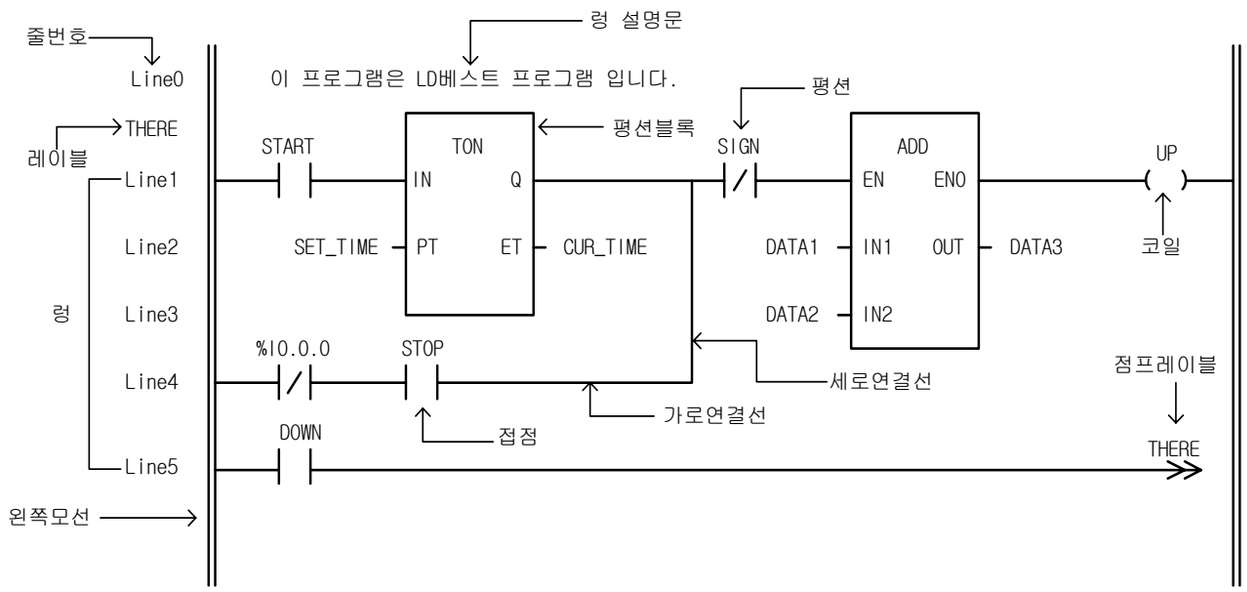


제 5 장 LD(Ladder Diagram)

5.1 개요

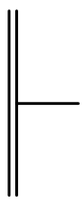
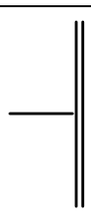
▷ LD 프로그램은 릴레이 로직 다이어그램에서 많이 사용하는 코일이나 접점 등의 그래픽 기호를 통하여 PLC 의 프로그램을 표현하는 것입니다.

▷ 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

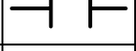
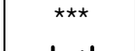
▷ 왼쪽 모선의 BOOL 1 값은 작성한 도면에 따라 오른쪽으로 전달됩니다. 그 전달되는 값을 가진 선을 전원 흐름선 또는 연결선이라고 하며, 접점이나 코일에 연결되어 있는 선입니다. 전원 흐름선은 언제나 BOOL 값을 가지고 있으며, 한 렁(Rung)에서 하나만 존재합니다. 여기서 렁이란 LD 의 처음부터 밑으로 내려가는 선이 없는 줄까지를 말합니다.

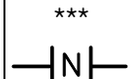
▷ LD 의 각 요소를 연결하는 연결선에는 가로 연결선과 세로 연결선이 있습니다.

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.

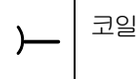
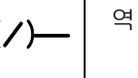
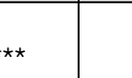
정적 접점			
No	기호	이름	설명
1	*** 	평상시 열린 접점 (Normally Open Contact)	BOOL 변수("***" 로 표시된 것)의 상태가 On 일 때에는 왼쪽의 연결선 상태는 오른쪽의 연결선으로 복사됩니다. 그렇지 않을 경우에는 오른쪽의 연결선 상 태가 Off 입니다.
2	*** 	평상시 닫힌 접점 (Normally Closed Contact)	BOOL 변수("***" 로 표시된 것)의 상태가 Off 일 때에는 왼쪽의 연결선 상태는 오른쪽의 연결선으로 복사됩니다. 그렇지 않을 경우에는 오른쪽의 연결선 상태가 Off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

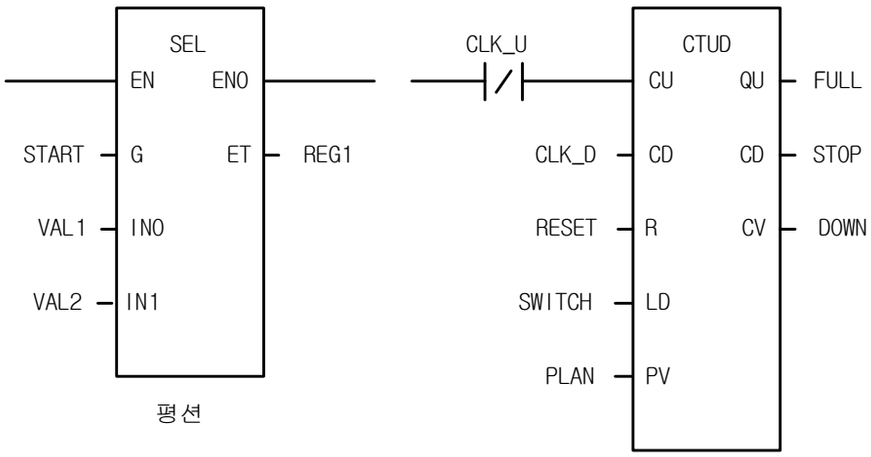
임시 코일(Momentary Coils)			
No.	기호	이름	설명
1	*** 	코일(Coil)	왼쪽에 있는 연결선의 상태를 관련된 BOOL 변수("***" 로 표시된 것)에 넣습니다.
2	*** 	역 코일(Negated Coil)	왼쪽에 있는 연결선 상태의 역(Negated)값을 관련된 BOOL 변수("***" 로 표시된 것)에 넣습니다. 즉, 왼쪽 연결선 상태 Off 이면 관련된 변수를 On 시키고, 왼쪽 연결선 상태가 On 이면 관련된 변수를 Off 시킵니다.
래치 코일(Latched Coils)			
No.	기호	이름	설명
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

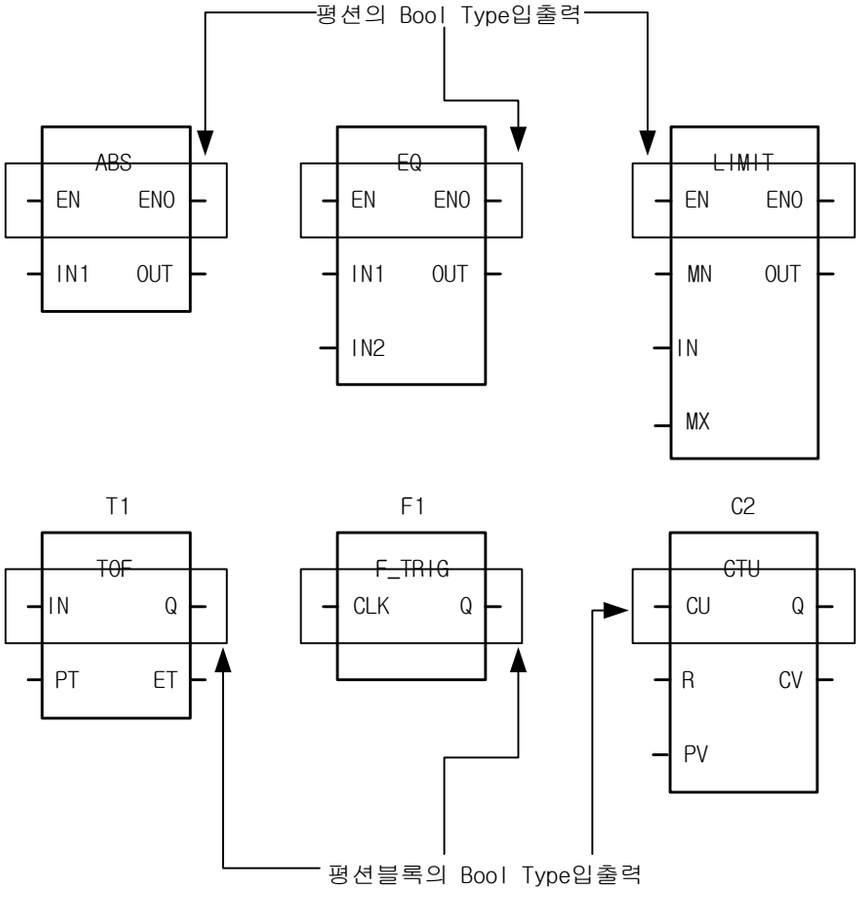


펑션

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.

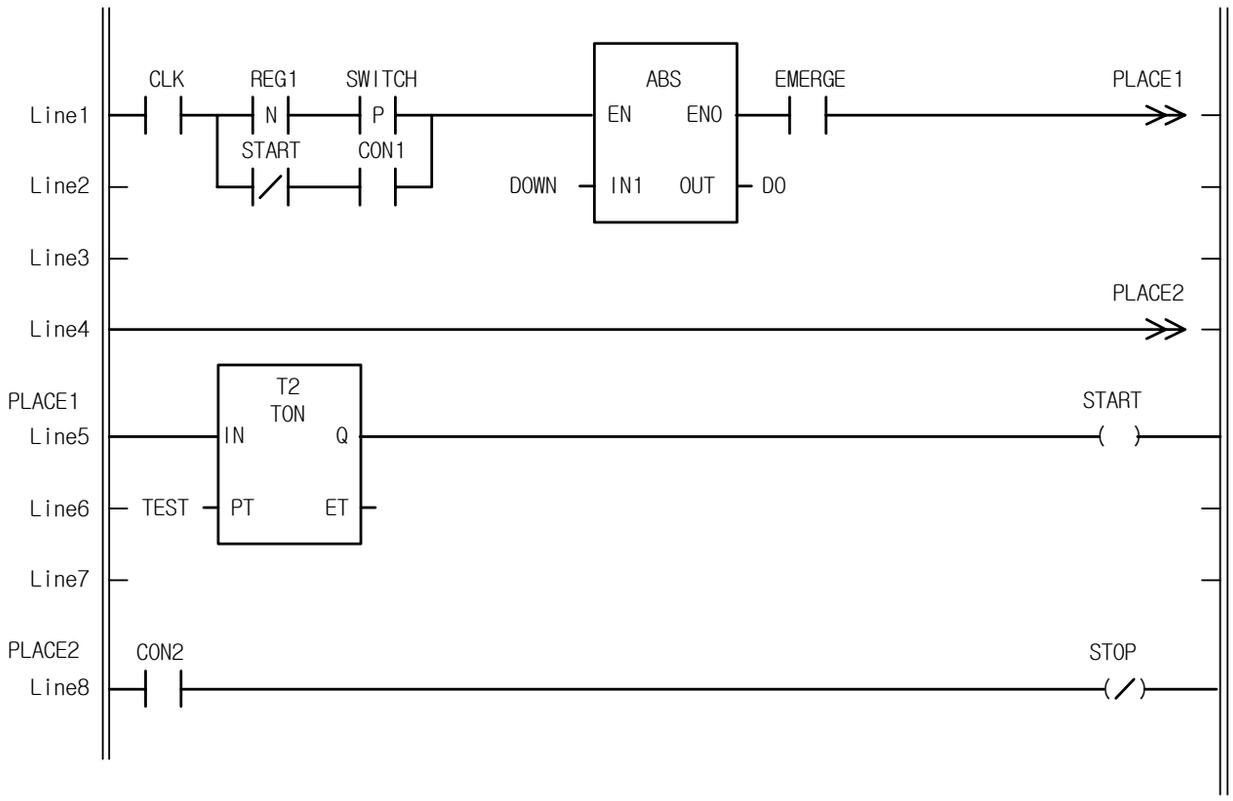
예



▷ 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.

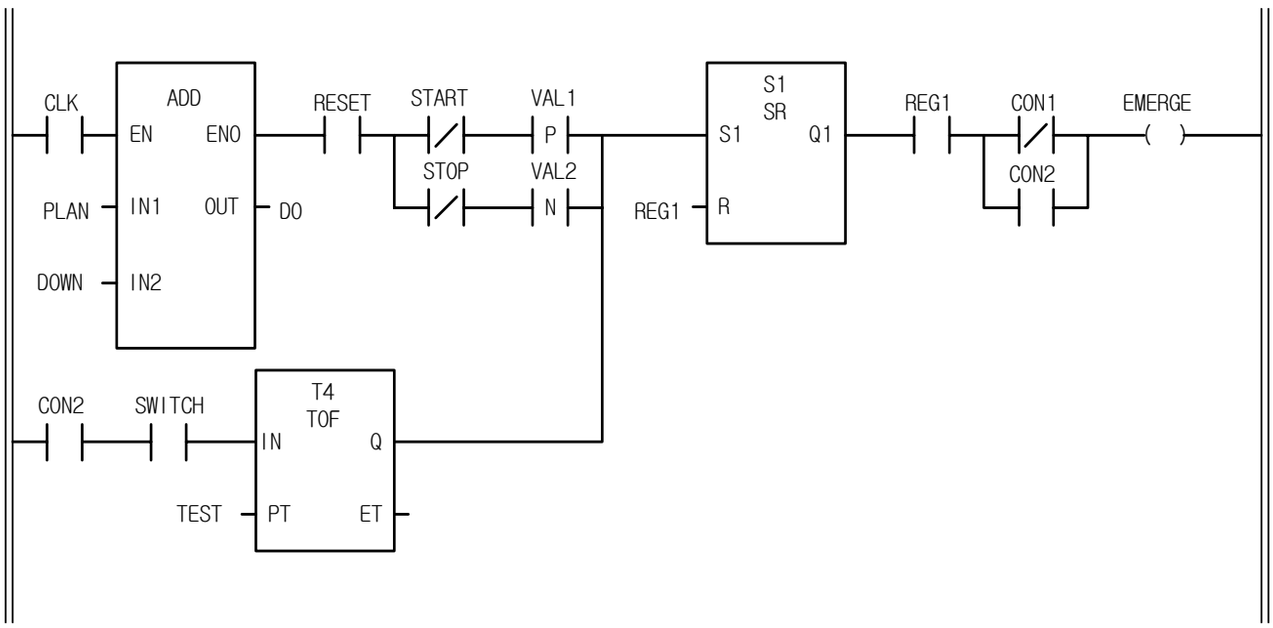
▷ LD 펑션 블록의 입력도 펑션과 같은 방법으로 지정합니다. 펑션 블록의 출력은 그 인스턴스 안에 저장되어 있으므로 변수를 지정하지 않더라도 상관없습니다. 펑션 블록에는 EN, ENO 입출력이 없으므로 펑션 블록을 만나면 항시 수행합니다. 따라서 어떤 로직 결과에 따라 펑션 블록의 수행 여부를 결정하기 위해서는 점프(—>>)를 사용하여야 합니다. 펑션 블록에 전원 흐름선을 연결할 때는 역시 데이터 타입이 BOOL 인 입출력에 연결하여야 합니다.

예



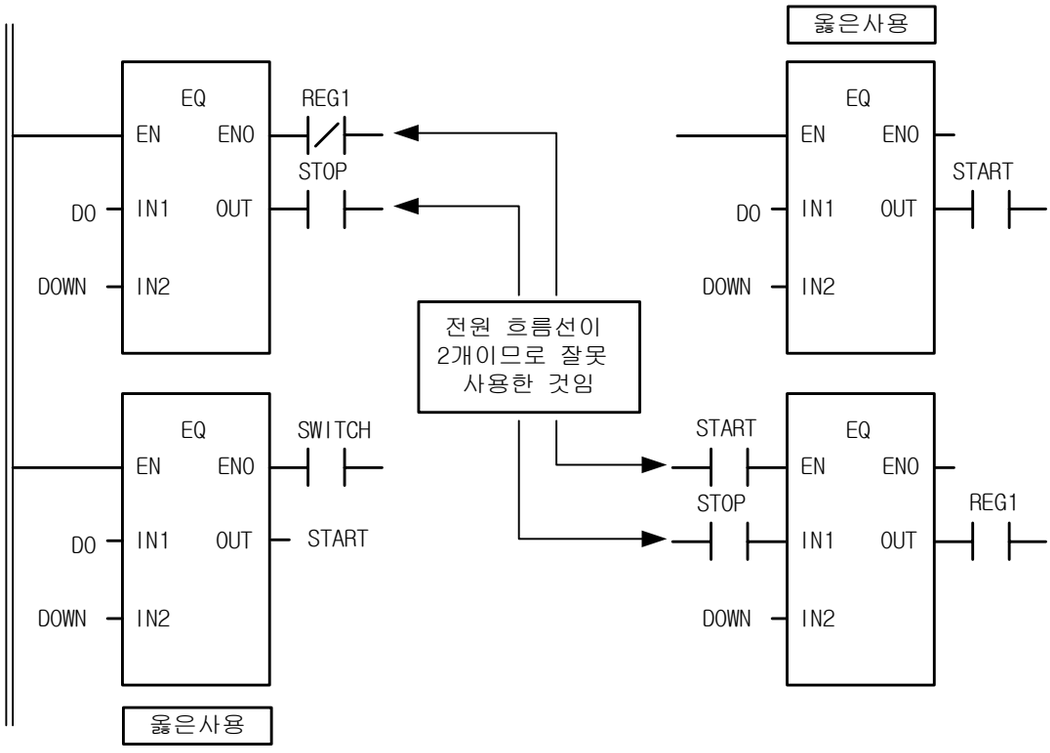
▷ LD 에서 펑션과 펑션 블록은 어느 곳에라도 올 수 있습니다. 펑션과 펑션 블록의 출력에 전원 흐름선을 연결하고 거기에 접점 등을 연결하여 로직 연산을 계속할 수도 있습니다.

예



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

예



제 6 장 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	-
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	-

No.	펑션 이름	기 능	비 고
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	-
1	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	-

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)	-

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

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_POR	위치 오버라이드	-
15	APM_SOR	속도 오버라이드	-
16	APM_PSO	위치 지정 속도 오버라이드	-
17	APM_NMV	연속 운전	-
18	APM_INC	인칭 기동	-
19	APM_RTP	수동 운전 이전 위치로 복귀 기동	-
20	APM_SNS	기동 스텝 번호 변경	-
21	APM_SRS	반복 스텝 번호 변경	-
22	APM_MOF	M 코드 해제	-
23	APM_PRS	현재 위치 프리셋	-
24	APM_ZONE	Zone 출력 허용/금지	-
25	APM_EPRE	엔코더값 프리셋	-
26	APM_TEA	단독 티칭(ROM, RAM)	-
27	APM_ATEA	복수 티칭(ROM, RAM)	-
28	APM_SBP	기본 파라미터 설정	-
29	APM_SEP	확장 파라미터 설정	-
30	APM_SHP	원점 복귀 파라미터 설정	-
31	APM_SMP	수동 운전 파라미터 설정	-
32	APM_SIP	입력 신호 파라미터 설정	-
33	APM_SCP	공통 파라미터 설정	-

No.	펑션 블록 이름	기 능	비 고
34	APM_SMD	운전 데이터 설정	-
35	APM_EMG	비상 정지	-
36	APM_RST	에러 리셋 / 출력 금지 해제	-
37	APM_PST	포인트 운전	-
38	APM_WRT	파라미터/운전 데이터 저장	-
39	APM_CRD	운전 정보 읽기	-
40	APM_SRD	운전 정보 읽기	-
41	APM_ENCRD	엔코더값 읽기	-
42	APM_JOG	조그 기동	-

제 7 장 기본 펑션

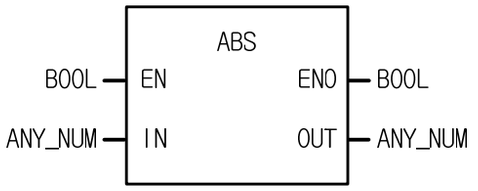
1. 각각의 기본 펑션에 대한 설명입니다.
2. 기본 펑션을 사용하시기 전에 3.4.1 의 펑션에 대한 일반사항을 이해 하신 후 프로그램에 적용하시면, 프로그램 작성이 용이합니다.

ABS

절대값 연산

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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						○	○	○	○	○	○	○	○	○	○					

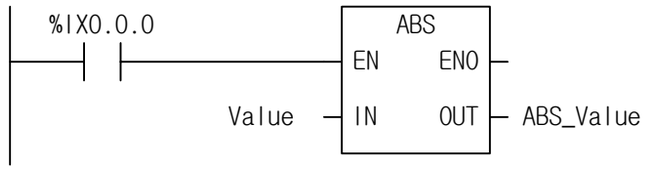
■ 기능

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.

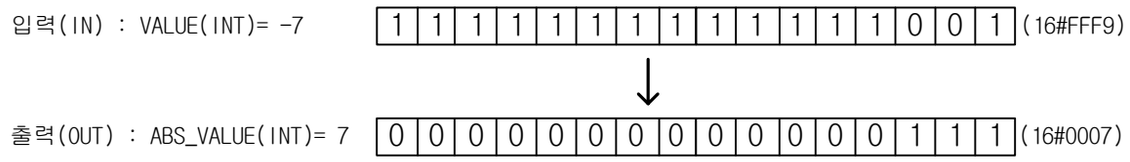
■ 플래그

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) 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)

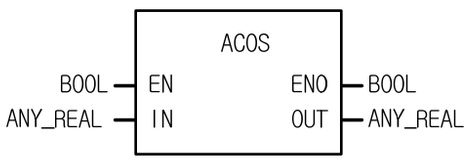


ACOS

Arc Cosine 연산

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

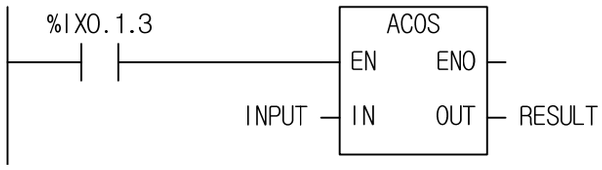
1. IN 의 Arc Cosine 값을 구해 OUT 으로 출력시킵니다. 출력 값은 0 에서 π 사이의 값이 됩니다.

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

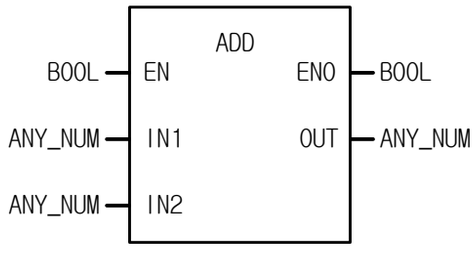
(2) INPUT 으로 선언된 입력 변수의 값이 0.8660... ($\sqrt{3}/2$) 일 때 출력 변수로 선언된 RESULT 은 0.5235...
($\pi/6$ rad = 30□) 입니다.

ADD

더하기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_ERR, _LER

펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 IN1 : 더할 값 IN2 : 더할 값 입력은 8 개까지 확장 가능 **출력** ENO : 에러 없이 실행되면 1 을 출력 OUT : 더한 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 더해서 OUT 으로 출력시킵니다.

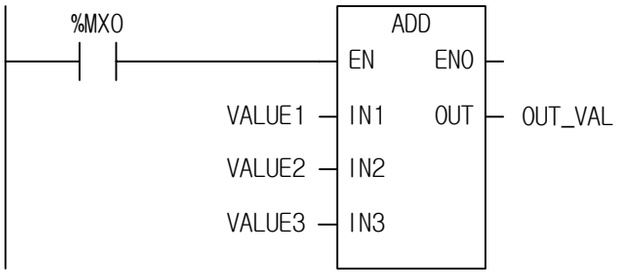
   OUT = IN1 + IN2 + ... + INn

■ 플래그

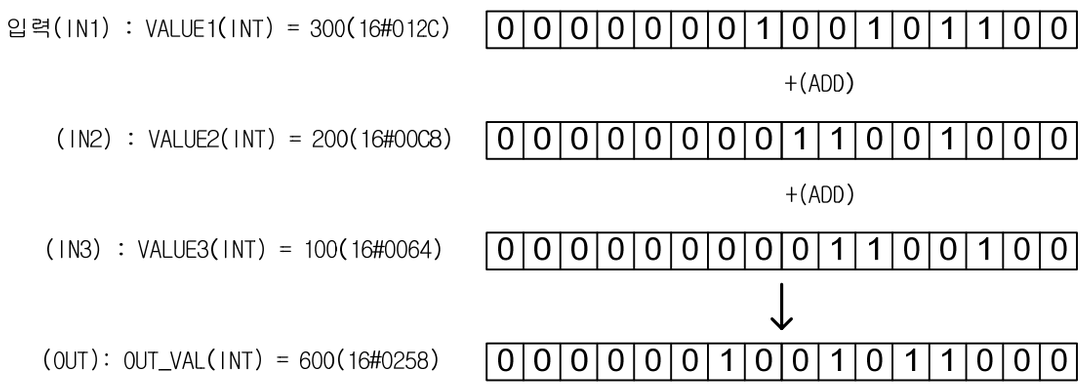
플래그	설명
_ERR	출력 값이 해당 데이터 타입의 범위를 벗어날 경우 _ERR, _LER 플래그가 셋(Set)됩니다

☆ 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.#SNAN000000000000e+000, 1.#QNAN000000000000e+000)으로 표시됩니다.

■ 프로그램 예



(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.

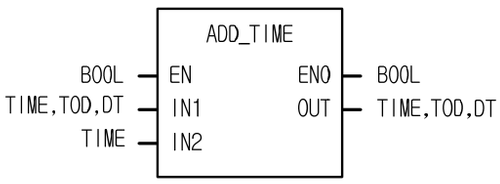


ADD_TIME

시간 더하기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_ERR, _LER

펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 IN1 : 기준시각 또는 시간 IN2 : 더할 시간 **출력** ENO : 에러 없이 실행되면 1 을 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																○		○	○	

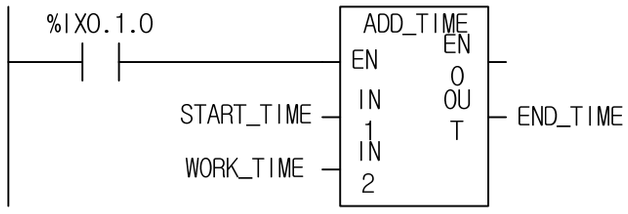
■ 기능

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) 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

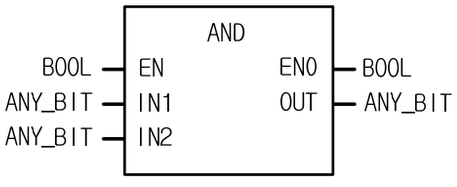
↓

출력 (OUT) : END_TIME(TOD) = TOD#10:40:20.5

AND

논리곱

CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** N : 1일 때 펑션 실행 IN1 : AND 될 값 IN2 : AND 될 값 입력이 8 개까지 확장 가능 **출력** ENO : EN 값이 그대로 출력 OUT : AND 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 AND 해서 OUT 으로 출력시킵니다.

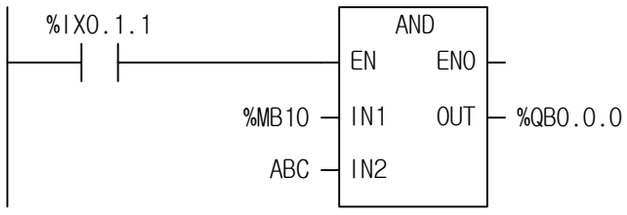
IN1 1111 0000

&

IN2 1010 1010

OUT 1010 0000

■ 프로그램 예



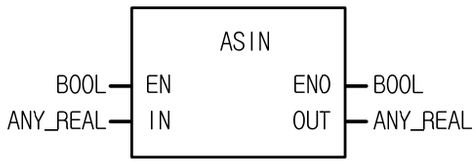
(1) 실행조건(%IX0.1.1)이 On 하면 펑션 AND 가 실행됩니다.
(2) IN1= %MB10 과 IN2 = ABC 값을 AND 시킨 결과가 OUT = %QB0.0.0 에 출력됩니다.

ASIN

Arc Sine연산

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : Arc Sine 연산의 입력 값 **출력** ENO : EN값이 그대로 출력 OUT : Arc Sine 연산의 결과 값 (Radian) 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

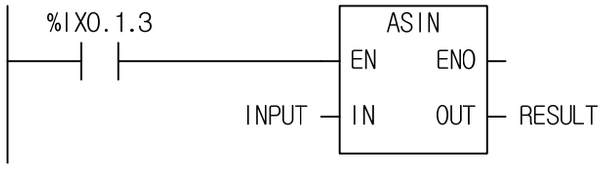
1. IN의 Arc Sine 값을 구해 OUT으로 출력시킵니다. 출력 값은 -π/2에서 π/2 사이의 값이 됩니다.

   OUT = ASIN (IN)

■ 에러

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

■ 프로그램 예



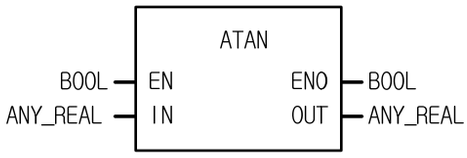
(1) 실행조건(%IX0.1.3)이 On 하면 펑션 ASIN 이 실행됩니다.

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

ATAN

Arc Tangent 연산

CPU 명	XGI	XGR
적용 가능	●	●

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

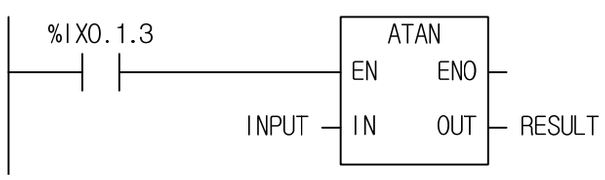
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

■ 기능

1. IN의 Arc Tangent 값을 구해 OUT으로 출력시킵니다. 출력 값은 -π/2에서 π/2 사이의 값이 됩니다.

   OUT = ATAN (IN)

■ 프로그램 예



(1) 실행조건(%IX0.1.3)이 On하면 펑션 Arc Tangent 연산 펑션 ATAN이 실행됩니다.

(2) INPUT으로 선언된 입력 변수의 값이 1.0일 때 출력 변수로 선언된 RESULT은 0.7853...

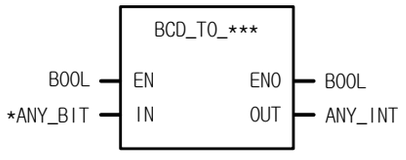
(π/4 rad = 45°) 입니다.

BCD_TO_***

BCD타입을 정수로 변환

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

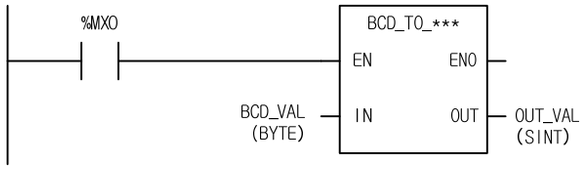
1. IN 을 타입 변환해서 OUT 으로 출력시킵니다.

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	

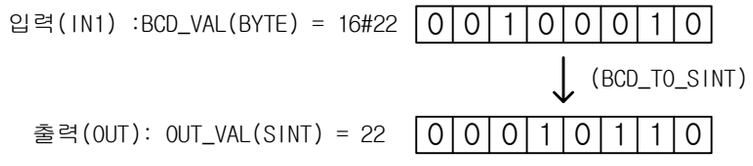
■ 플래그

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

■ 프로그램 예



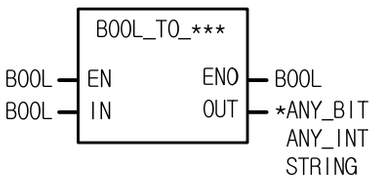
(1) 실행조건(%MX0)이 On 하면 펑션 BCD_TO_*** 이 실행됩니다.
(2) BCD_VAL(BYTE 타입) = 16#22(2#0010_0010)이면, 펑션의 출력 변수로 선언된 OUT_VAL(SINT 타입) = 22 (2#0001_0110)가 출력됩니다.



BOOL_TO_***

BOOL타입 변환

CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 비트(1 비트) **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		○	○	○	○	○	○	○	○	○	○	○	○							○

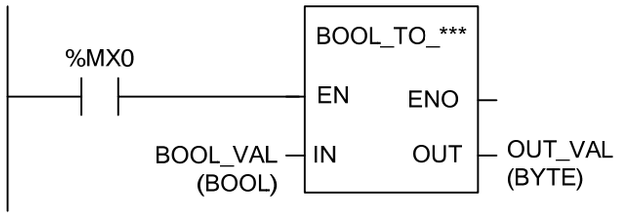
*ANY_BIT: ANY_BIT 타입 중 BOOL 제외

■ 기능

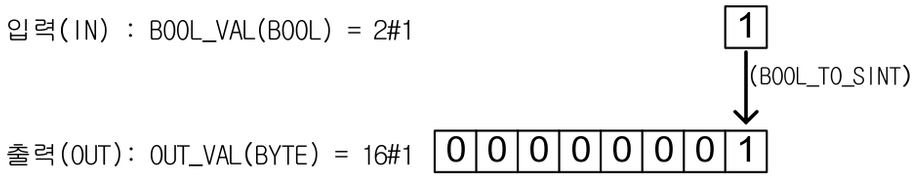
1. IN을 타입 변환해서 OUT으로 출력시킵니다.

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



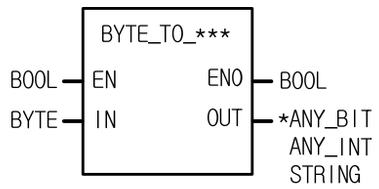
(1) 실행조건(%MX0)이 On 하면 펑션 BOOL_TO_***이 실행됩니다.
(2) 입력 변수로 선언된 BOOL_VAL(BOOL 타입) = 2#1 이면, 출력 변수로 선언된 OUT_VAL(BYTE 타입) = 2#0000_0001)이 출력 됩니다.



BYTE_TO_***

BYTE타입 변환

CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 비트(1 비트) **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		○	○	○	○	○	○	○	○	○	○	○	○							○

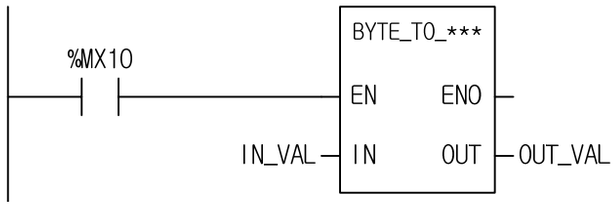
*ANY_BIT : ANY_BIT 타입중 BOOL 제외

■ 기능

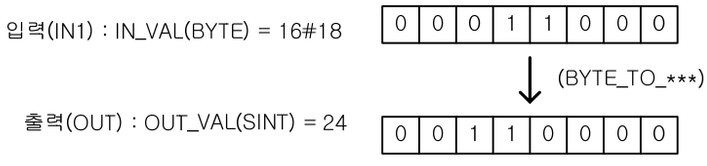
1. IN을 타입 변환해서 OUT으로 출력시킵니다.

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

■ 프로그램 예



(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

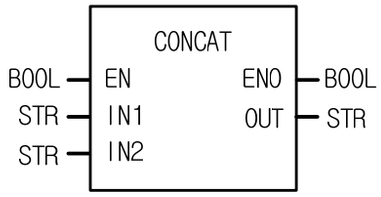


CONCAT

문자열 연결하기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_ERR, _LER

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

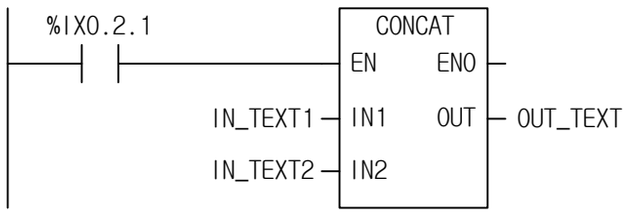
■ 기능

1. 입력 문자열 IN1, IN2, IN3,...., INn(n 은 입력 개수)을 순서대로 붙여서 출력 문자열 OUT 에 출력시킵니다.

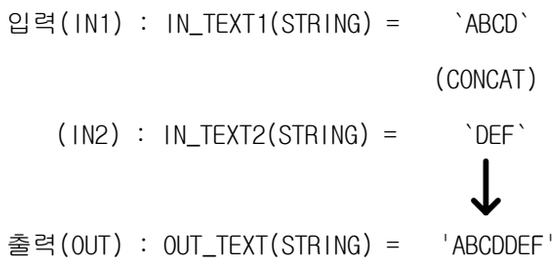
■ 플래그

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

■ 프로그램 예



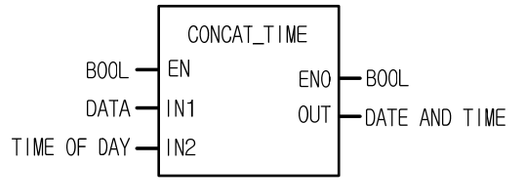
(1) 실행조건(%IX0.2.1)이 On 하면 펑션 CONCAT 이 실행됩니다.
(2) 펑션의 입력변수로 선언된 IN_TEXT1=`ABCD`, IN_TEXT2=`DEF`이면, 출력 변수로 선 OUT_TEXT=`ABCDDEF`가 됩니다.


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

CONCAT_TIME

날짜와 시각 연결하기

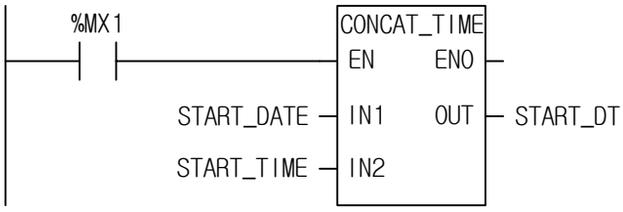
CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 날짜 데이터 입력 IN2 : 시각 데이터 입력 **출력** ENO : EN값이 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) 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.

입력(IN1) : START_DATE(DATE) = D#1995-12-06

(CONCAT_TIME)

입력(IN2) : START_TIME(TOD) = TOD#08:30:00

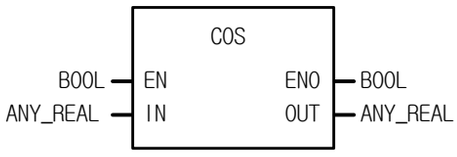
↓

출력(OUT) : START_DT(DT) = DT#1995-12-06-08:30:00

COS

Cosin 연산

CPU 명	XGI	XGR
적용 가능	●	●

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

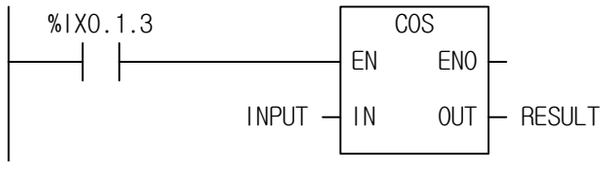
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

■ 기능

1. IN 의 Cosine 값을 구해 OUT 으로 출력시킵니다.

   OUT = COS (IN)

■ 프로그램 예



(1) 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

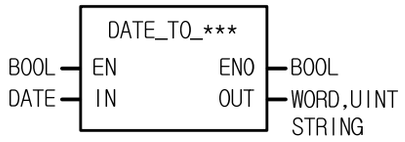
↓ (COS)

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

DATE_TO_***

DATE 타입변환

CPU 명	XGI	XGR
적용 가능	●	●

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

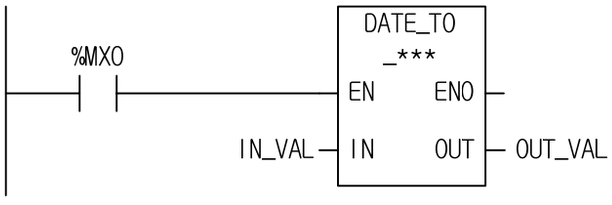
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN			○								○									○

■ 기능

1. IN을 타입 변환해서 OUT으로 출력시킵니다.

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

■ 프로그램 예



(1) 실행조건(%MX0)이 On하면 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)

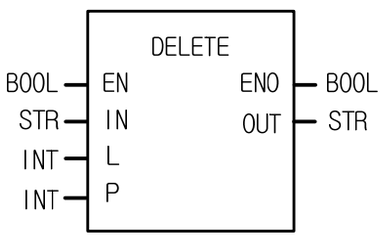
출력(OUT) : OUT_VAL(STRING) = 'D#1995-12-01'

DELETE

문자열을 삭제하기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

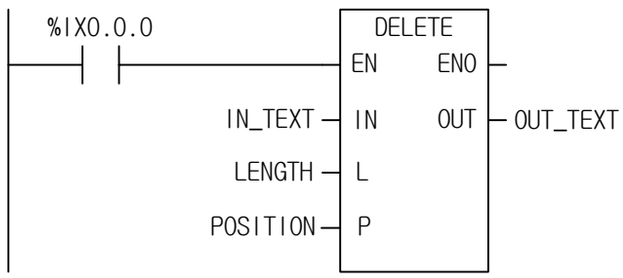
■ 기능

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

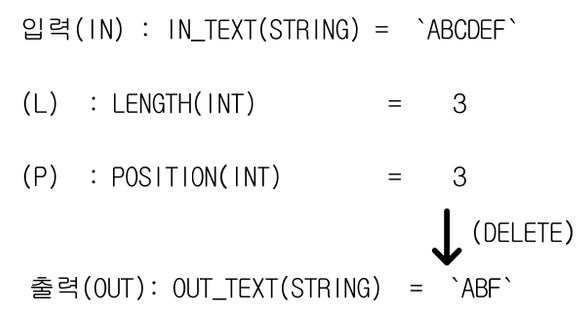
■ 플래그

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

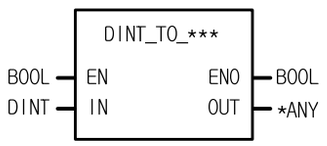

입력(IN) : IN_TEXT(STRING) = `ABCDEF`
(L) : LENGTH(INT) = 3
(P) : POSITION(INT) = 3
(DELETE)
출력(OUT): OUT_TEXT(STRING) = `ABF`

DINT_TO_***

DINT 타입 변환

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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.

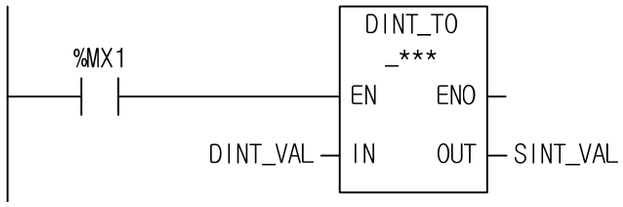
펑션	출력타입	동작 설명
DINT_TO_SINT	SINT	입력이 -128 ~127일 경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
DINT_TO_INT	INT	입력이 -32,768~32,767 일 경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
DINT_TO_LINT	LINT	LINT 타입으로 정상 변환합니다.
DINT_TO_USINT	USINT	입력이 0~255일 경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
DINT_TO_UINT	UINT	입력이 0~65,535일 경우 정상 변화되나, 그 외 값은 에러가 발생합니다.
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



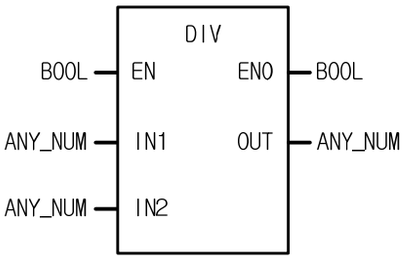
(1) 실행조건(%MX1)이 On 하면 데이터 타입 변환 펑션 DINT_TO_***가 실행됩니다.
(2) INI = DINT_VAL(DINT 타입) = -77 이면, SINT_VAL(SINT 타입) = -77 이 됩니다.

DIV

나누기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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.

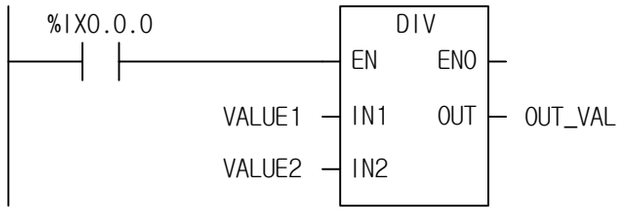
   OUT = IN1/IN2

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

■ 플래그

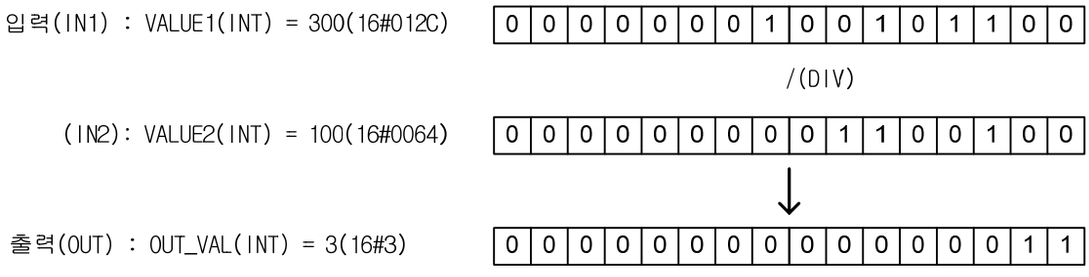
플래그	설명
_ERR	나눌 값(제수)이 '0' 인 경우, 나눈 결과값이 각 타입의 최대값을 넘을 경우 _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예



(1) 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.

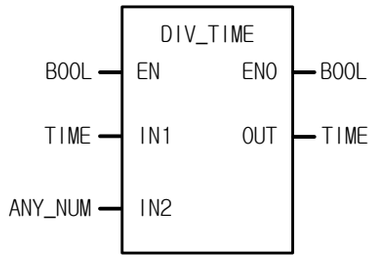


DIV_TIME

시간 나누기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

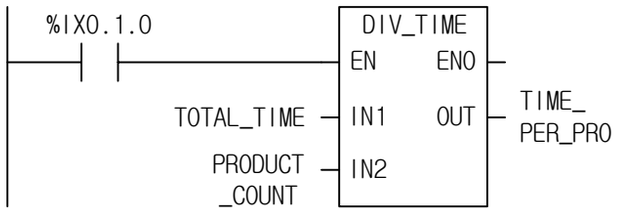
1. IN1(시간)을 IN2(숫자)로 나누어서 나누어진 시간을 OUT으로 출력시킵니다.

■ 플래그

플래그	설명
_ERR	제수(IN2)가 0인 경우, 또는 0보다 작은 경우 _ERR, _LER 플래그가 셋(Set)됩니다. IN2에 음수 값 입력 시 _ERR, _LER 플래그가 ON되고 결과에 0을 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) 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

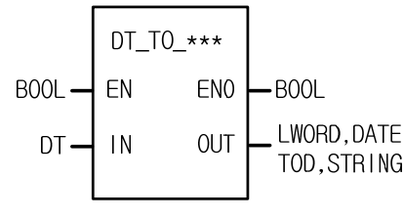
↓

출력(OUT) : TIME_PER_PRO(TIME) = T#1H2M2S

DT_TO_***

DT 타입변환

CPU 명	XGI	XGR
적용 가능	●	●

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

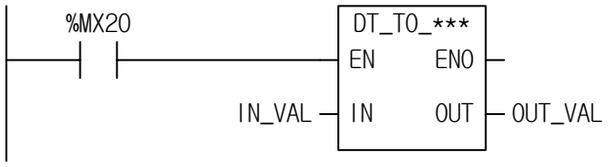
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT					○												○	○		○

■ 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) 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

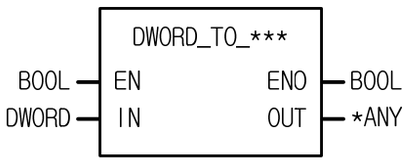
↓ (DT_TO_DATE)

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

DWORD_TO_***

DWORD 타입변환

CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 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

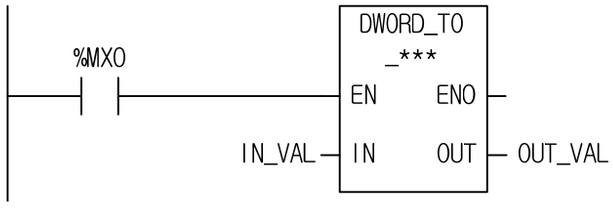
■ 기능

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.
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)

펑션	출력타입	동작 설명
		하고 TOD 범위 내에서 순환하여 변환합니다.
DWORD_TO_STRING	STRING	입력 값을 Decimal 로 바꾸어 STRING 타입으로 변환합니다.

■ 프로그램 예



(1) 실행조건(%MX0)이 On 하면 타입 변환 펑션 DWORD_TO_TOD 가 실행됩니다.

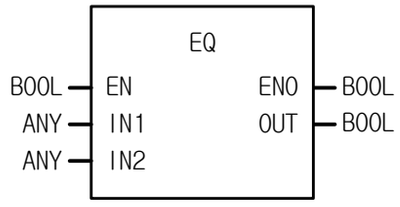
(2) IN_VAL(DWORD 타입) = 16#3E8(1000)이면, OUT_VAL(TOD 타입) = TOD#1S 가 됩니다.

(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

'같다' 비교

CPU 명	XGI	XGR
적용 가능	●	●

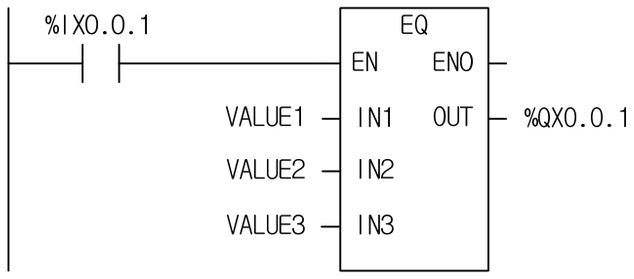
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 비교할 값 IN2 : 비교할 값 입력은 8개까지 확장 가능 IN1, IN2, ...는 모두 같은 타입이어야 함. **출력** ENO : EN값이 그대로 출력 OUT : 비교 결과값

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

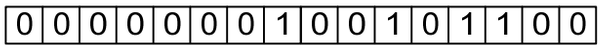
1. IN1=IN2=IN3...=INn(n 은 입력개수)이면 OUT 으로 1 이 출력됩니다.
2. 다른 경우에는 OUT 으로 0 이 출력됩니다.

■ 프로그램 예



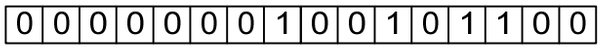
(1) 실행조건(%IX0.0.1)이 On 하면 비교 펑션 'EQ' 가 실행됩니다.
(2) VALUE1 = 300, VALUE2 = 300, VALUE3 = 300 이면, 비교결과 VALUE1 = VALUE2 = VALUE3 이므로 출력 값 %QX0.0.1 = 1 이 됩니다.

입력(IN1) : VALUE1(INT)= 300(16#012C)



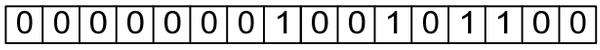
=(EQ)

(IN2) : VALUE2(INT)= 300(16#012C)



=(EQ)

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



↓

출력 (OUT) : %QX0.0.1(BOOL)= 1(16#1)

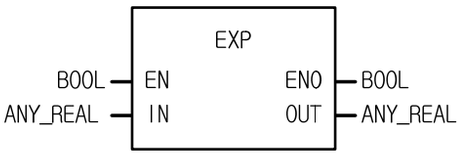
1

EXP

EXP 연산

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

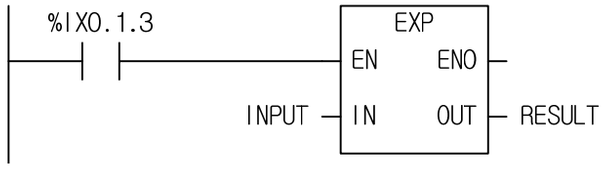
1. IN의 지수연산 값을 구해 OUT으로 출력시킵니다.

$$OUT = e^{IN}$$

■ 에러

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

■ 프로그램 예



(1) 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.

$RESULT = e^{INPUT}$

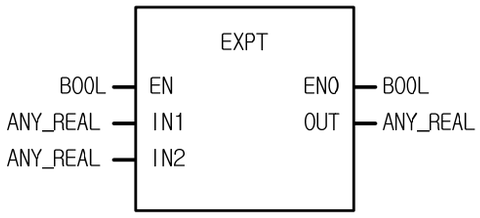
INPUT = 2.0, RESULT = 7.3890...

EXPT

지수 연산

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 진수 IN2 : 지수 **출력** ENO : EN값이 그대로 출력 OUT : 연산결과 값 IN1, 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
	IN1														○	○					
	IN2														○	○					
	OUT														○	○					

■ 기능

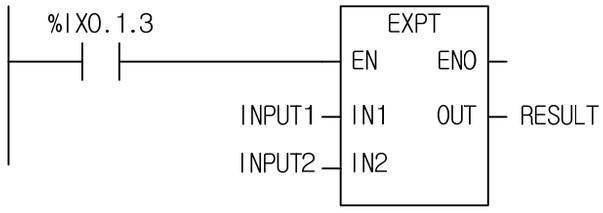
1. IN1에 IN2를 지수승하여 OUT으로 출력시킵니다.

$$OUT = IN1^{IN2}$$

■ 에러

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

■ 프로그램 예



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

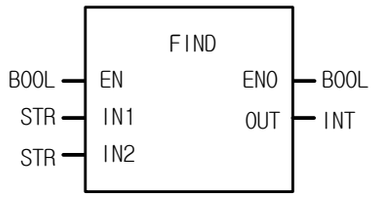
(2) INPUT 으로 선언된 입력 변수의 값 INPUT1 = 1.5, INPUT2 = 3 이면 출력 변수로 선언된 RESULT 는 3.375 가 됩니다.

$$3.375 = 1.5^{3}$$

FIND

문자열 찾기

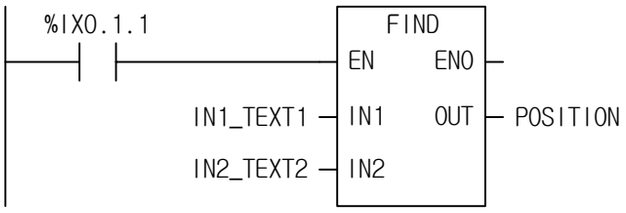
CPU 명	XGI	XGR
적용 가능	●	●

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) 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'

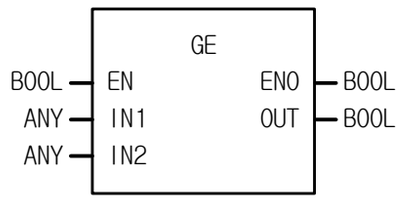
↓ (FIND)

출력(OUT) : POSITION(INT) = 2

GE

'크거나 같다' 비교

CPU 명	XGI	XGR
적용 가능	●	●

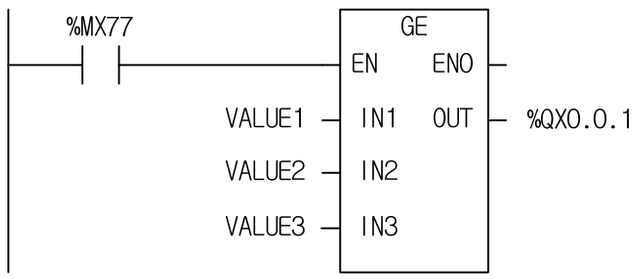
펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 IN1 : 비교할 값 IN2 : 비교할 값 입력은 8 개까지 확장 가능 IN1, IN2,...는 모두 같은 타입이어야 함. **출력** ENO : EN 값이 그대로 출력 OUT : 비교 결과값

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

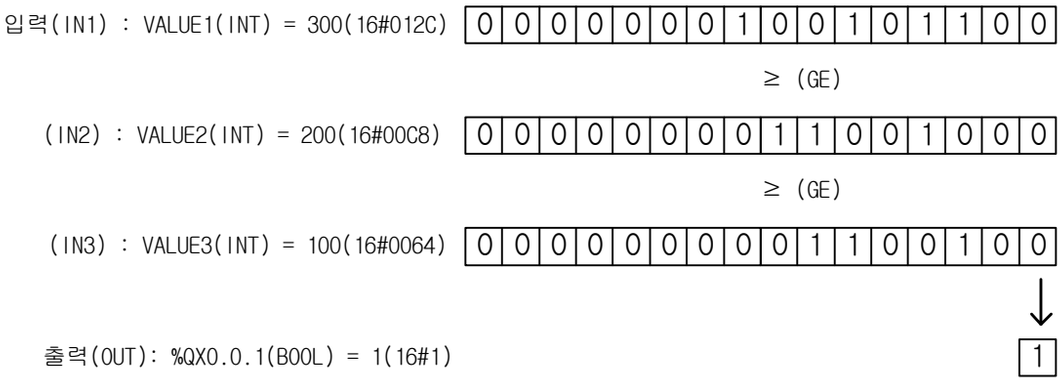
1. IN1≥IN2≥IN3...≥INn(n 은 입력개수)이면 OUT 으로 1 이 출력됩니다.
2. 다른 경우에는 OUT 으로 0 이 출력됩니다.

■ 프로그램 예



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

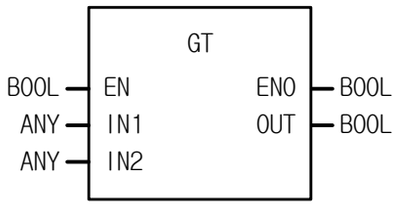
(2) 입력변수로 선언된 VALUE1=300, VALUE2=200, VALUE3=100 이면, 비교 결과 VALUE1 ≥ VALUE2 ≥ VALUE3 이므로, 출력 결과 값 %QX0.0.1=1 이 됩니다.


입력(IN1) : VALUE1(INT) = 300(16#012C)
0 0 0 0 0 0 0 1 0 0 1 0 1 1 0 0
≥ (GE)
(IN2) : VALUE2(INT) = 200(16#00C8)
0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0
≥ (GE)
(IN3) : VALUE3(INT) = 100(16#0064)
0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0
출력(OUT): %QX0.0.1(BOOL) = 1(16#1)
1

GT

'크다' 비교

CPU 명	XGI	XGR
적용 가능	●	●

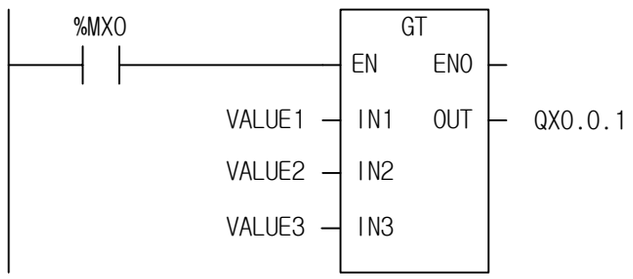
펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 IN1 : 비교될 값 IN2 : 비교할 값 입력은 8 개까지 확장 가능 IN1, IN2, ...는 모두 같은 타입이어야 함. **출력** ENO : EN 값이 그대로 출력 OUT : 비교 결과 값

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

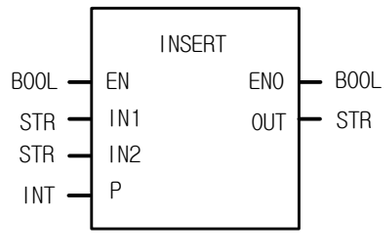
(2) 입력변수로 선언된 VALUE1 = 300, VALUE2 = 200, VALUE3 = 100 이면, 비교결과 VALUE1 > VALUE2 > VALUE3 이므로 출력결과값 %QX0.0.1 = 1 이 됩니다.

INSERT

문자열 삽입하기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

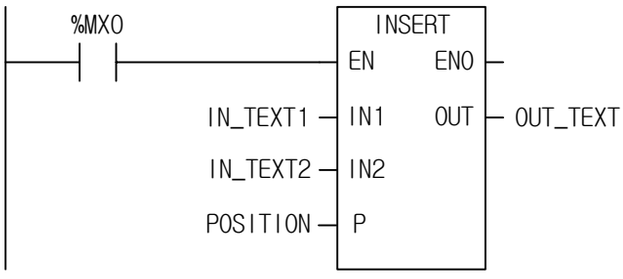
■ 기능

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) 실행조건(%MX0)이 On하면, INSERT(문자열 삽입하기) 펑션이 실행됩니다.

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

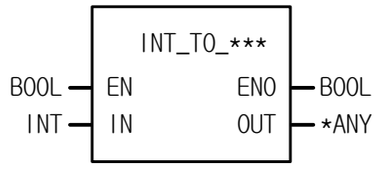
```
입력(IN1) : IN_TEXT1(STRING) = 'ABCD'
    (IN2) : IN_TEXT2(STRING) =  'XY'
    (P)   : POSITION(INT)    =   2
                                 ↓ (FIND)
 출력(OUT): OUT_TEXT         = `ABXYCD`
```

INT_TO_***

INT 타입변환

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

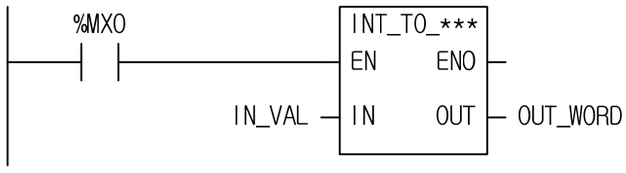
1. IN 을 타입 변환해서 OUT 으로 출력시킵니다.

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

■ 플래그

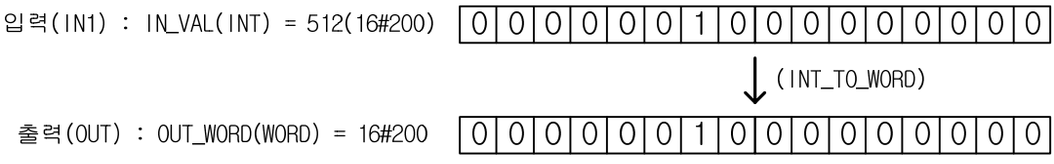
플래그	설명
_ERR	변환에러 발생시 _ERR, _LER 플래그가 셋(Set)됩니다. 에러 발생시 출력 타입의 비트 수 만큼 하위 비트를 취해 내부 비트 배열의 변환 없이 출력 시킵니다

■ 프로그램 예



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

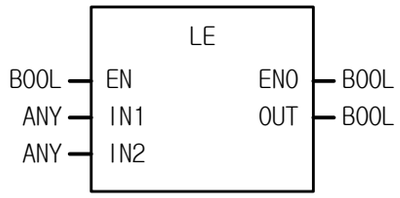
(2) 입력 변수로 선언된 IN_VAL(INT 타입) = 512(16#200)이면, 출력 변수로 선언된 OUT_WORD(WORD 타입) =16#200 이 됩니다.



LE

'작거나 같다' 비교

CPU 명	XGI	XGR
적용 가능	●	●

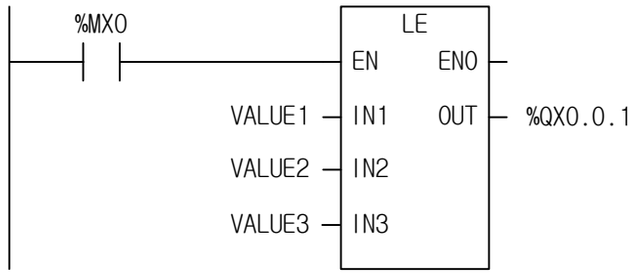
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 비교할 값 IN2 : 비교할 값 입력은 8개까지 확장 가능 IN1, IN2, ...는 모두 같은 타입이어야 함. **출력** ENO : EN값이 그대로 출력 OUT : 비교 결과

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

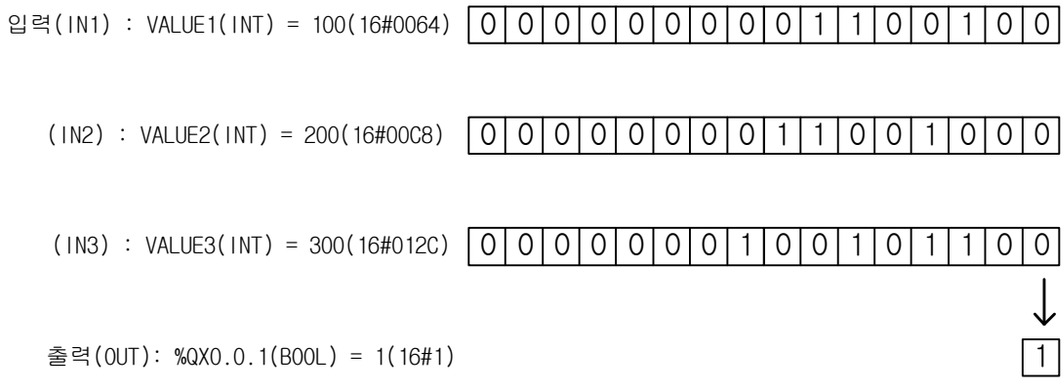
1. IN1 ≤ IN2 ≤ IN3... ≤ INn(n 은 입력 개수)이면 OUT 으로 1 이 출력됩니다.
2. 다른 경우에는 OUT 으로 0 이 출력됩니다.

■ 프로그램 예



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

(2) 입력 변수로 선언된 VALUE1 = 150, VALUE2 = 200, VALUE3 = 250 이면, 비교결과 VALUE1 ≤ VALUE2 ≤ VALUE3 이므로, 출력 결과값 %QX0.0.1 = 1 이 됩니다.

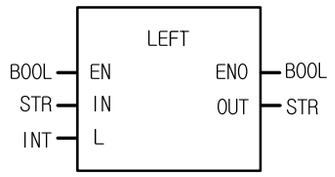

입력(IN1) : VALUE1(INT) = 100(16#0064)
0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 0
(IN2) : VALUE2(INT) = 200(16#00C8)
0 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0
(IN3) : VALUE3(INT) = 300(16#012C)
0 0 0 0 0 0 0 1 0 0 1 0 1 1 0 0
출력(OUT): %QX0.0.1(BOOL) = 1(16#1)
1

LEFT

문자열의 왼쪽을 취하기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

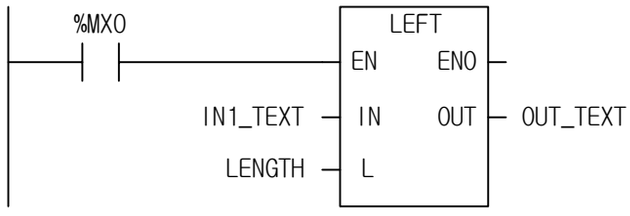
■ 기능

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

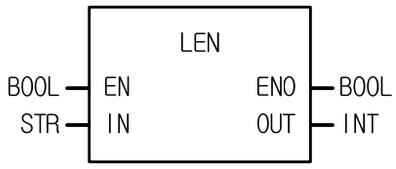
(2) 입력 변수로 선언된 문자열이 IN_TEXT = `ABCDEFG`이고, 출력할 문자열의 길이 LENGTH = 3 이면, 출력 문자열 변수로 지정된 OUT_TEXT = `ABC`가 됩니다.

```
입력(IN1) : IN_TEXT(STRING) =  'ABCDEFG'
    (IN2) : LENGTH(INT)     =     3
                                  ↓(LEFT)
출력(OUT) : OUT_TEXT(STRING) =   `ABC`
```

LEN

문자열 길이 구하기

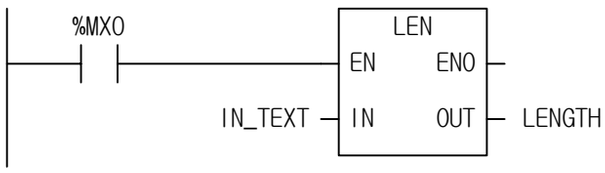
CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 입력 문자열 **출력** ENO : EN값이 그대로 출력 OUT : 문자열 길이

■ 기능

1. 입력 문자열(IN)의 길이(문자 수)가 OUT으로 출력됩니다.

■ 프로그램 예



(1) 실행조건(%MX0)이 On 하면 LEN(문자열 길이 구하기) 펑션이 실행됩니다.

(2) 입력 변수로 선언된 IN_TEXT = `ABCD`이면, 출력 변수로 선언된 LENGTH = 4가 됩니다.

입력 (IN) : IN_TEXT(STRING) = ' ABCD'

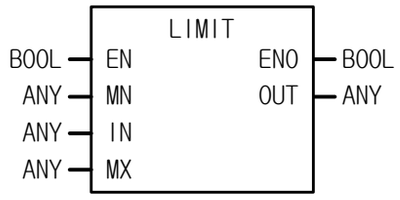
↓

출력 (OUT) : LENGTH(INT) = 4

LIMIT

상하한 제한

CPU 명	XGI	XGR
적용 가능	●	●

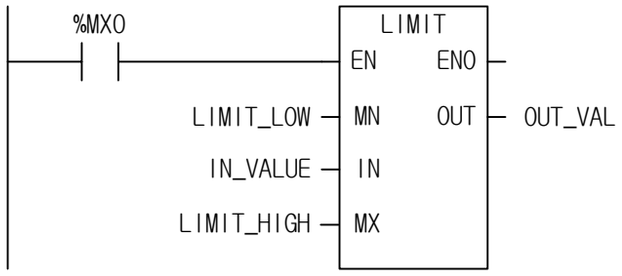
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 MN : 최소값 IN : 제한될 값 MX : 최대값 **출력** ENO : EN 값이 그대로 출력 OUT : 범위 안에 든 값 MN, IN, MX, OUT 은 데이터 타입이 모두 같아야 함

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) 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)

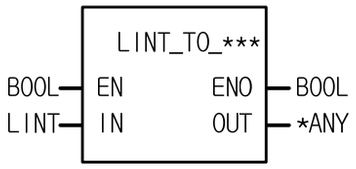
출력(OUT) : OUT_VAL (INT) = 3000

LINT_TO_***

LINT 타입변환

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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.

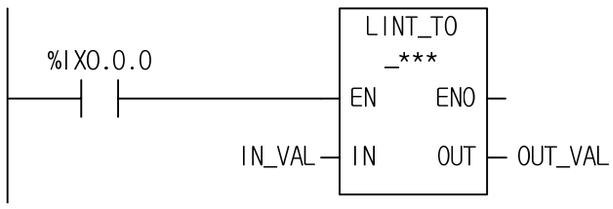
펑션	출력타입	동작 설명
LINT_TO_SINT	SINT	입력이 -128 ~ 127일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
LINT_TO_INT	INT	입력이 -32,768~32,767 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
LINT_TO_DINT	DINT	입력이 -2^{31} ~ $2^{31}-1$일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
LINT_TO_USINT	USINT	입력이 0~ 255일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
LINT_TO_UINT	UINT	입력이 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) 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.

입력 (IN) : IN_VAL(LINT) = 123,456,789(16#00000000075BCD15)

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

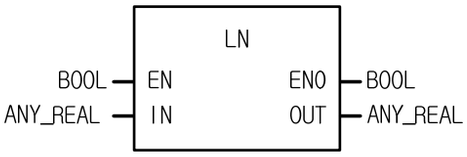
출력(OUT) : OUT_VAL(DINT) = 123,456,789 (16#075BCD15)

LN

LN연산

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

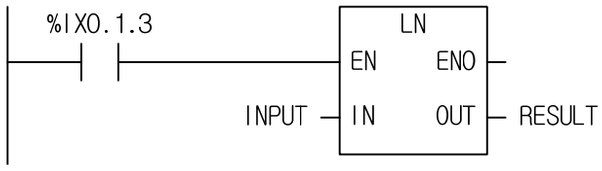
1. IN의 자연대수 값을 구해 OUT으로 출력시킵니다.

   OUT = ln (IN)

■ 에러

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

■ 프로그램 예



(1) 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.

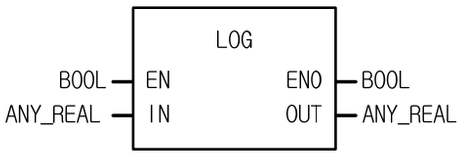
ln(2.0) = 0.6931...

LOG

LOG연산

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

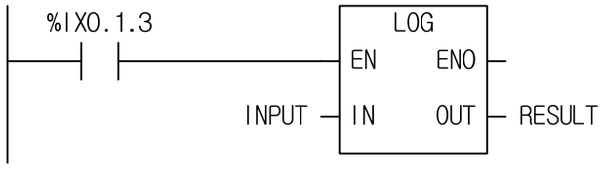
1. IN의 상용대수 값을 구해 OUT으로 출력시킵니다.

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

■ 에러

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

■ 프로그램 예



(1) 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.

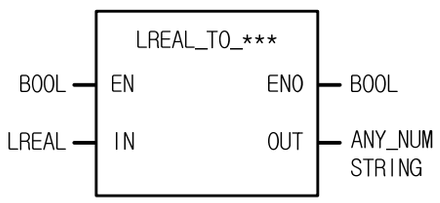
$Log_{10}(2.0) = 0.3010...$

LREAL_TO_***

LREAL 타입변환

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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.

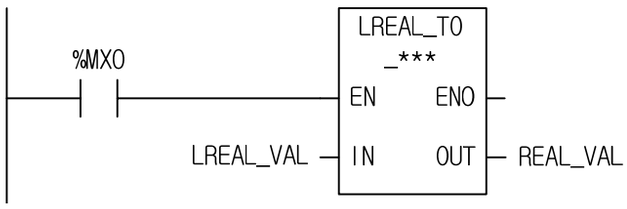
펑션	출력타입	동작 설명
LREAL_TO_SINT	SINT	입력의 정수 부분이 -128 ~ 127 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다. (소수점 이하는 반올림)
LREAL_TO_INT	INT	입력의 정수 부분이 -32,768 ~ 32,767 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다. (소수점 이하는 반올림)
LREAL_TO_DINT	DINT	입력의 정수 부분이 -2^{31} ~ $2^{31}-1$ 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다. (소수점 이하는 반올림)
LREAL_TO_LINT	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)

펑션	출력타입	동작 설명
LREAL_TO_LWORD	LWORD	내부 비트 배열의 변화 없이 LWORD 타입으로 변환합니다.
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



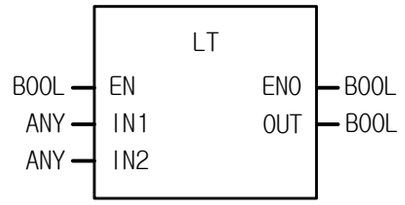
(1) 입력조건(%MX0)이 On 하면, LREAL_TO_*** 펑션이 실행됩니다.
(2) 입력변수로 선언된 LREAL_VAL(LREAL 타입) = -1.34E-12 이면,
출력변수로 선언된 REAL_VAL(REAL 타입) = -1.34E-12 가 됩니다.

입력(IN) : LREAL_VAL (LREAL) = -1.34E-12
↓ (LREAL_TO_REAL)
출력(OUT) : REAL_VAL (REAL) = -1.34E-12

LT

'작다' 비교

CPU 명	XGI	XGR
적용 가능	●	●

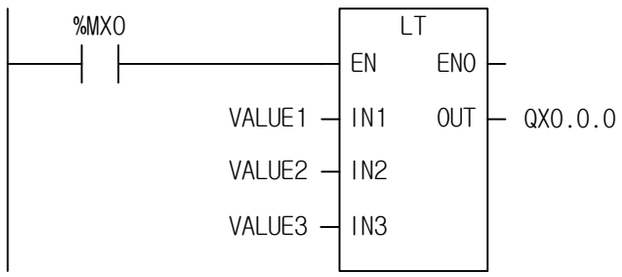
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 비교할 값 IN2 : 비교할 값 입력은 8개까지 확장 가능 IN1, IN2, ...는 모두 같은 타입이어야 함. **출력** ENO : EN값이 그대로 출력 OUT : 비교 결과값

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.

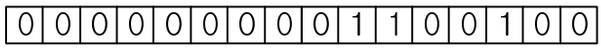
■ 프로그램 예



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

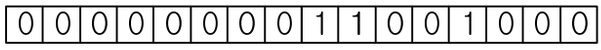
(2) 입력변수로 선언된 VALUE1 = 100, VALUE2 = 200, VALUE3 = 300 이면, 비교결과 VALUE1 < VALUE2 < VALUIE3 이므로 출력 결과값 %QX0.0.0 = 1 이 됩니다.

입력(IN1) : VALUE1(INT) = 100(16#0064)



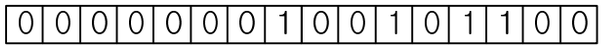
<(LT)

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



<(LT)

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



↓

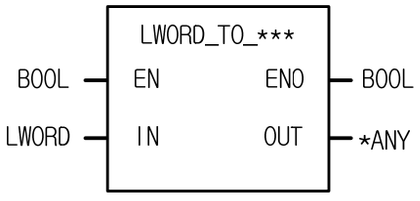
출력(OUT) : %QX0.0.0(BOOL) = 1(16#0001)

1

LWORD_TO_***

LWORD 타입변환

CPU 명	XGI	XGR
적용 가능	●	●

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

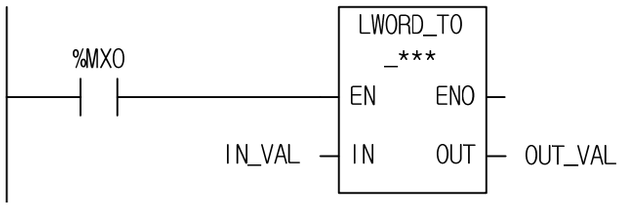
■ 기능

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 (DT#2163-12-31-23:59:59:999)를 벗어난 값 입력시 _ERR, _LER 플래그

펑션	출력타입	동작 설명
		를 SET 하고 DT 범위 내에서 순환하여 변환합니다.
LWORD_TO_STRING	STRING	입력 값을 STRING 타입으로 변환합니다.

■ 프로그램 예



(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

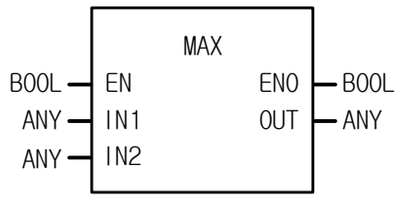
↓ (LWORD_TO_LINT)

출력(OUT) : OUT_VAL(LINT) = -1

MAX

최대값

CPU 명	XGI	XGR
적용 가능	●	●

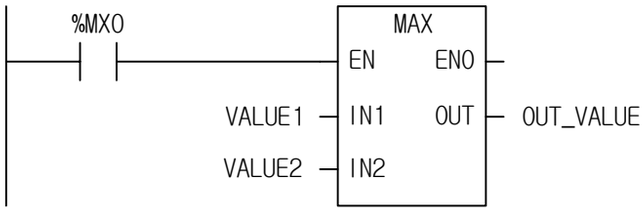
펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 IN1 : 비교될 값 IN2 : 비교될 값 입력 8 개까지 확장 가능 **출력** ENO : EN 값이 그대로 출력 OUT : 입력 값 중 최대값 IN1, IN2, ..., OUT 은 모두 같은 타입이어야 함.

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. 입력 IN1, IN2, ..., INn(n 은 입력 개수)중에서 최대값을 OUT 으로 출력시킵니다.

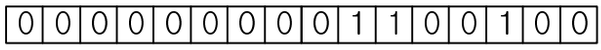
■ 프로그램 예



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

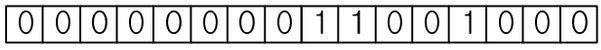
(2) 입력변수로 선언된 VALUE1 = 100, VALUE2 = 200 을 비교결과 최대값이 200 이므로 출력변수로 선언된 OUT_VALUE = 200 으로 출력됩니다.

입력(IN1) : VALUE1(INT) = 100(16#0064)

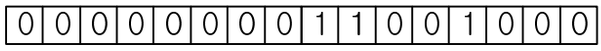


(MAX)

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



출력(OUT): OUT_VALUE(INT) = 200(16#00C8)

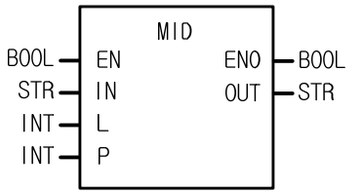


MID

문자열의 중간을 취하기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

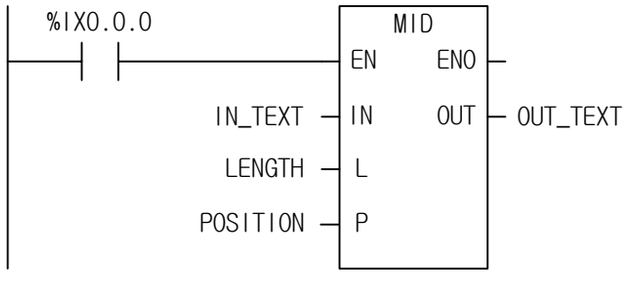
■ 기능

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

■ 플래그

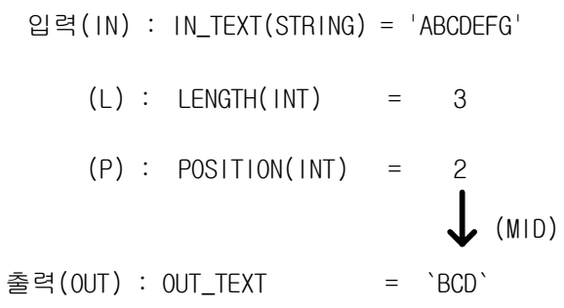
플래그	설명
_ERR	(변수 IN 의 문자 수) < P 인 경우, 또는 P <= 0 및 L < 0 인 경우 _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예



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

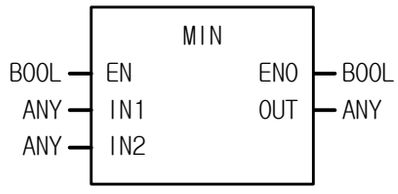
(2) 입력된 문자열 IN_TEXT = `ABCDEFG`이고, 출력할 문자열의 길이 LENGTH = 3, 출력할 문자열의 시작위치 POSITION = 2 이면, 출력 문자열 변수로 선언된 OUT_TEXT = `BCD`가 됩니다.


입력(IN) : IN_TEXT(STRING) = 'ABCDEFG'
(L) : LENGTH(INT) = 3
(P) : POSITION(INT) = 2
(MID)
출력(OUT) : OUT_TEXT = `BCD`

MIN

최소값

CPU 명	XGI	XGR
적용 가능	●	●

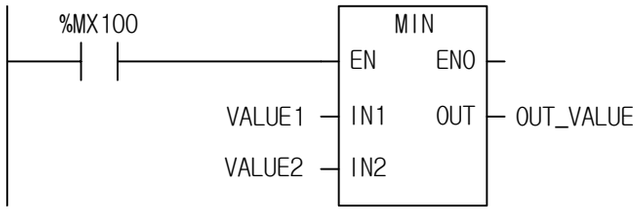
펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 IN1 : 비교될 값 IN2 : 비교될 값 입력 8 개까지 확장 가능 **출력** ENO : EN 값이 그대로 출력 OUT : 입력 값 중 최소값 IN1, IN2, ..., OUT 은 모두 같은 타입이어야 함.

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

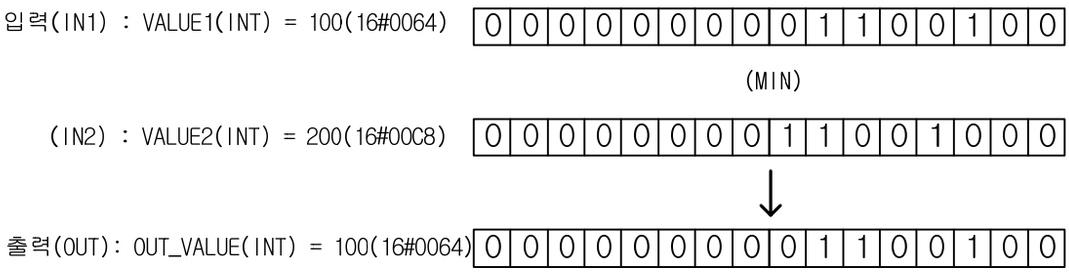
1. 입력 IN1, IN2, ..., INn(n 은 입력 개수) 중에서 최소값을 OUT 으로 출력시킵니다.

■ 프로그램 예



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

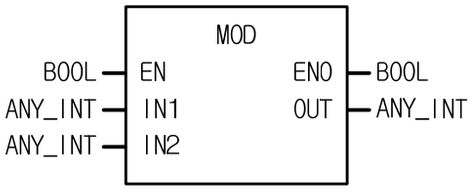
(2) 입력변수로 선언된 VALUE1 = 100, VALUE2 = 200 을 비교결과 최소값이 100 이므로 출력변수로 선언된 OUT_VALUE = 100 이 출력됩니다.


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

MOD

나머지 구하기

CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 나누어 질 값(피제수) IN2 : 나눌 값(제수) **출력** ENO : EN값이 그대로 출력 OUT : 나눈 결과값(나머지) 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						○	○	○	○	○	○	○	○							

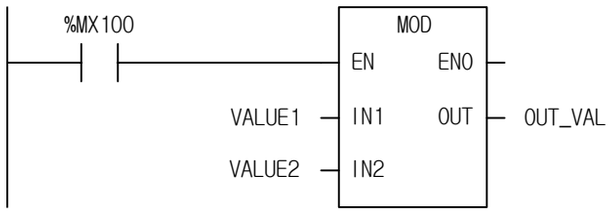
■ 기능

1. IN1을 IN2로 나눠서 그 나머지를 OUT으로 출력시킵니다.

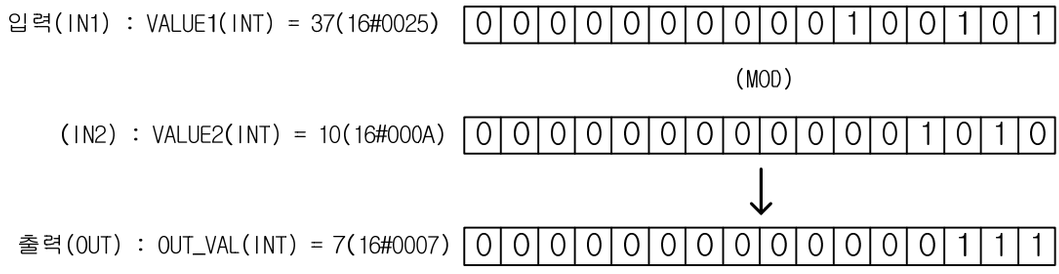
   OUT = IN1 - (IN1/IN2) × IN2 (단, IN2 = 0 이면 OUT = 0)

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

■ 프로그램 예



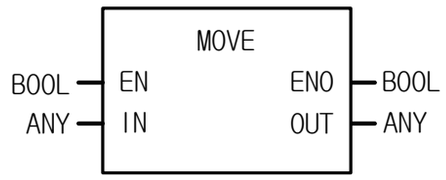
(1) 실행조건(%MX100)이 On 하면 MOD(나머지 구하기) 펑션이 실행합니다.
(2) 입력변수 중 나누어질 값 VALUE1 = 37 이고, 나눌 값 VALUE2 = 10 이면, 출력변수로 선언된 OUT_VAL 의 값은 37 을 10 으로 나눈 나머지 7 이 됩니다.



MOVE

데이터 복사

CPU 명	XGI	XGR
적용 가능	●	●

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

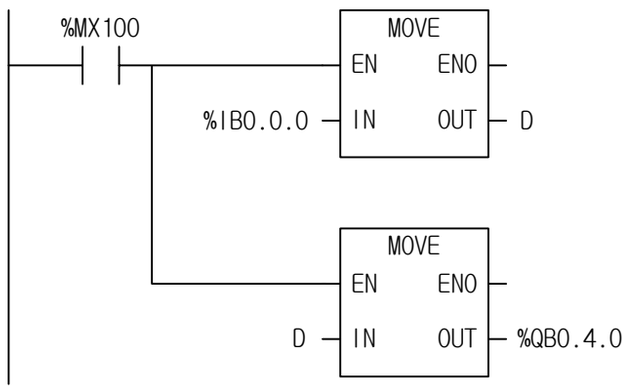
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ 기능

1. IN의 값을 OUT으로 이동합니다.

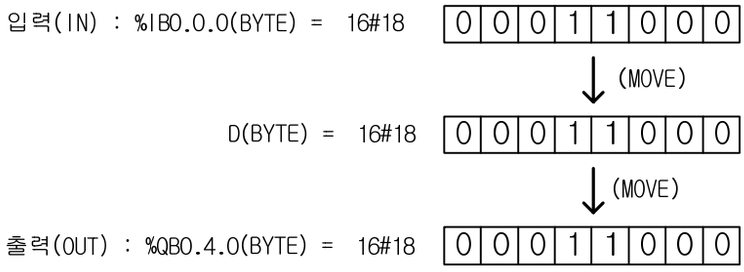
■ 프로그램 예

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) 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.

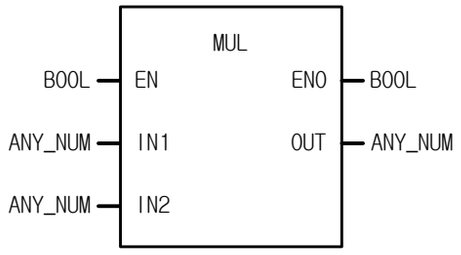


MUL

곱하기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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.

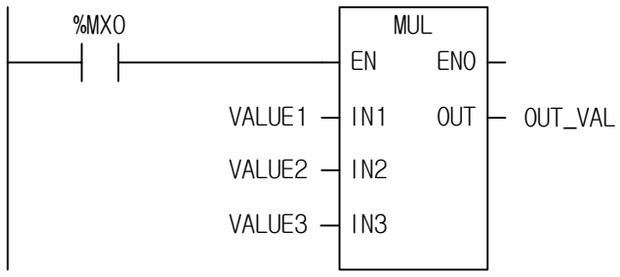
   OUT = IN1 × IN2 × ... × INn

■ 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.#SNAN000000000000e+000, 1.#QNAN000000000000e+000)으로 표시됩니다.

■ 프로그램 예



(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 |

↓

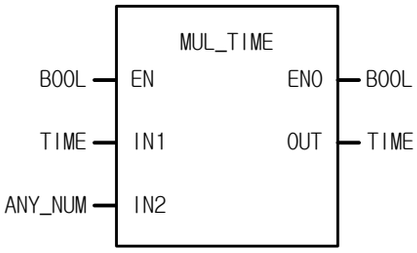
출력(OUT) : OUT_VAL(INT) = 6,000(16#1770) | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

MUL_TIME

시간 곱하기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

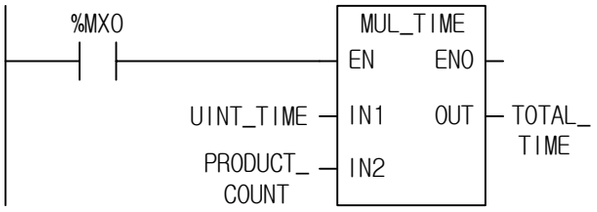
1. IN1(시간)을 IN2(숫자)로 곱해서 결과 시간을 OUT 으로 출력시킵니다.

■ 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) 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.

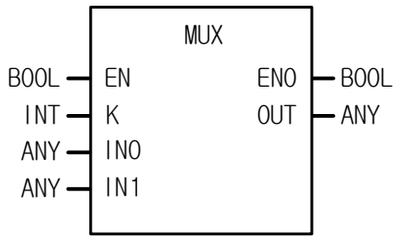
입력(IN1): UNIT_TIME(TIME) = T#20MS2S
(MUL_TIME)
(IN2): PRODUCT_COUNT(INT) = 16#18
↓
출력(OUT): TOTAL_TIME(TIME) = T#6H40M40S

MUX

여러 개 중 선택

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_ERR, _LER

펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 K : 선택 IN0 : 선택될 값 IN1 : 선택될 값 입력은 7 개까지 확장 가능(IN0, IN1,..., IN6) **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	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

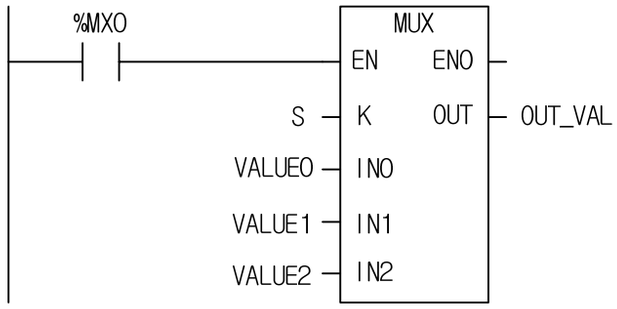
■ 기능

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

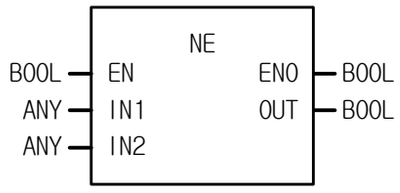
(2) 입력변수로 설정된 VALUE0, 1, 2 중 선택변수 S 의 값에 따라 선택되어 출력변수로 선언된 OUT 으로 출력시킵니다.

```
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

'같지 않다' 비교

CPU 명	XGI	XGR
적용 가능	●	●

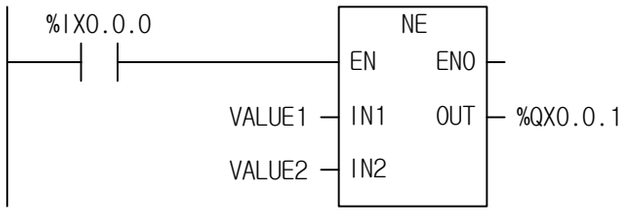
펑 션	설 명
	**입력** EN : 실행 허용 IN1 : 비교될 값 IN2 : 비교될 값 IN1, IN2 는 같은 타입이어야 함. **출력** ENO : EN 값이 그대로 출력 OUT : 비교 결과값

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

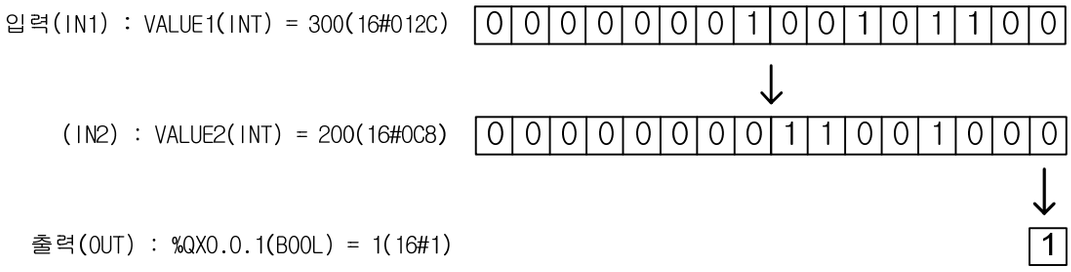
1. IN1 이 IN2 와 같지 않으면 OUT 으로 1 이 출력됩니다.
2. 같으면 OUT 으로 0 이 출력됩니다.

■ 프로그램 예



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

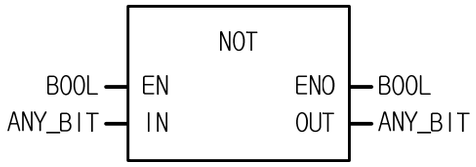
(2) 입력변수로 선언된 VALUE1 = 300, VALUE2 = 200 이면, 비교결과 VALUE1 과 VALUE2 가 다르므로 출력결과값 %QX0.0.1 = 1 이 됩니다


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

논리 반전

CPU 명	XGI	XGR
적용 가능	●	●

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	○	○	○	○	○															

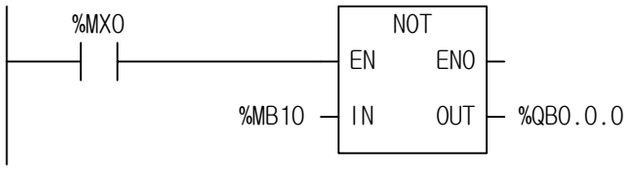
■ 기능

1. IN 을 비트별로 NOT(반전)해서 OUT 으로 출력시킵니다.

   IN 1100 1010

   OUT 0011 0101

■ 프로그램 예



(1) 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 |

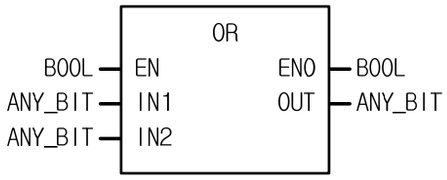
↓ (NOT)

출력(OUT) : %QB0.0.0(BYTE) = 16#33 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |

OR

논리합

CPU 명	XGI	XGR
적용 가능	●	●

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.

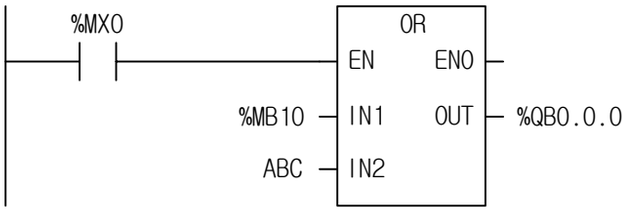
   IN1 1111 0000

   OR

   IN2 1010 1010

   OUT 1111 1010

■ 프로그램 예



(1) 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.

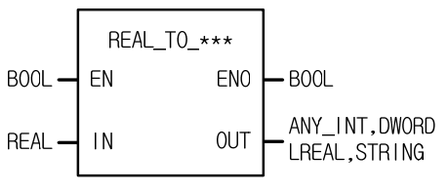
입력(IN1) : %IB0.0.0(BYTE) = 16#CC	1	1	0	0	1	1	0	0
& (OR)								
(IN2) : ABC(BYTE) = 16#F0	1	1	1	1	0	0	0	0
↓								
출력(OUT) : %QB0.0.0(BYTE) = 16#FC	1	1	1	1	1	1	0	0

REAL_TO_***

REAL 타입 변환

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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.

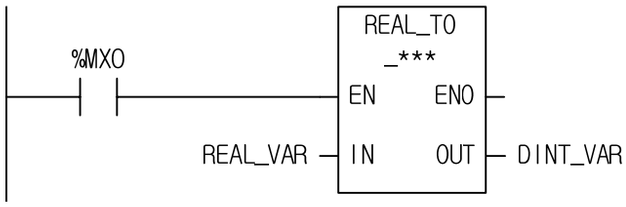
펑션	출력 타입	동작 설명
REAL_TO_SINT	SINT	입력의 정수 부분이 -128 ~ 127 일 경우 정상 변환되나, 그 외 값은 에러가 발생 합니다. (소수점 이하는 반올림)
REAL_TO_INT	INT	입력의 정수 부분이 -32768 ~ 32767 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다. (소수점 이하는 반올림)
REAL_TO_DINT	DINT	입력의 정수 부분이 -2^{31} ~ $2^{31}-1$ 일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다. (소수점 이하는 반올림)
REAL_TO_LINT	LINT	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) 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)

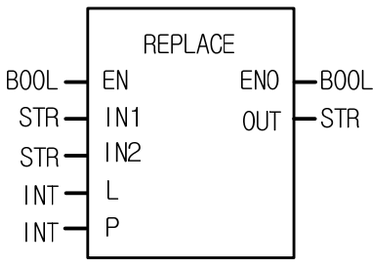
출력(OUT) : DINT_VAL(DINT) = 12,340

REPLACE

문자열 대체하기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

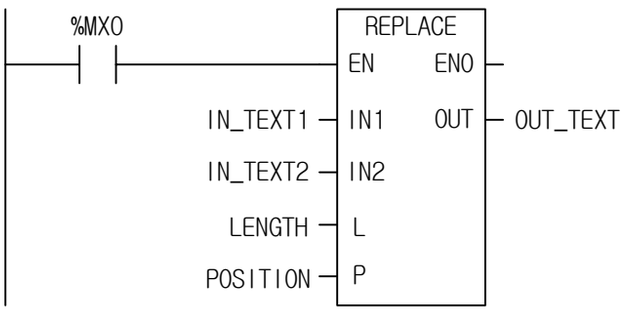
■ 기능

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) 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 `AXET`가 출력됩니다.

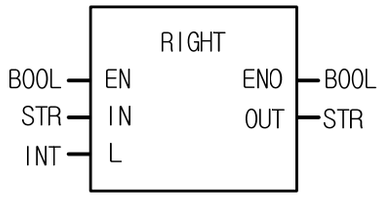
```
입력 (IN1) : IN_TEXT1(STRING) = `ABCDEF`
     (IN2) : IN_TEXT2(STRING) =  ` X`
       (L) : LENGTH(INT) =         3
       (P) : POSITION(INT) =       2
                                   ↓
출력(OUT)  : OUT_TEXT(STRING) =  `AXET`
```

RIGHT

문자열의 오른쪽을 취하기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

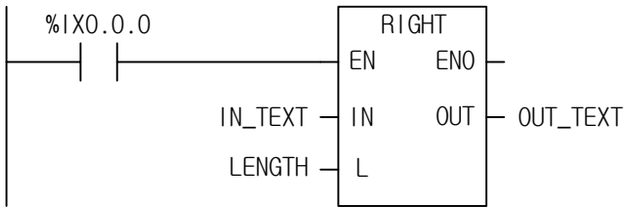
■ 기능

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) 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

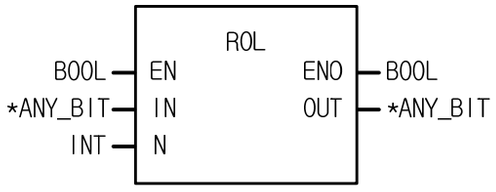
↓ (RIGHT)

출력(OUT) : OUT_TEXT(STRING) = `EFG`

ROL

왼쪽으로 회전 (Rotate Left)

CPU 명	XGI	XGR
적용 가능	●	●

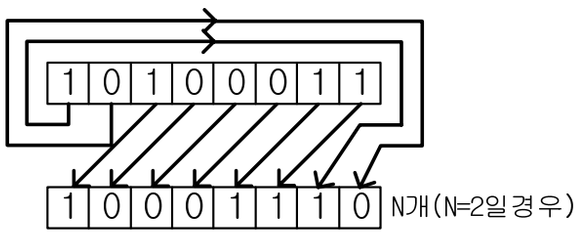
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 회전될 값 N : 회전할 비트 수 **출력** ENO : EN값이 그대로 출력 OUT : 회전된 값

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

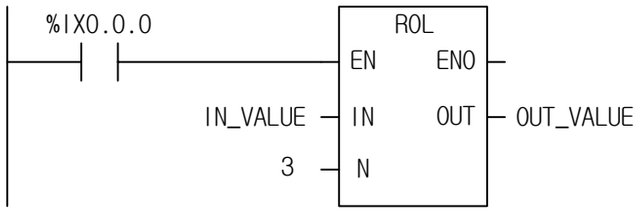
■ 기능

1. 입력 IN을 N 비트 수만큼 왼쪽으로 회전시킵니다.



■ 프로그램 예

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

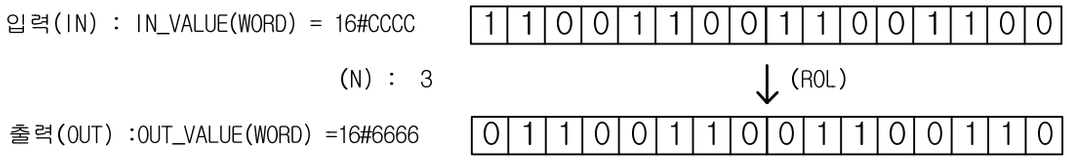


(1) 회전할 데이터 값을 입력한 변수 IN_VALUE 로 설정한다.

(2) 좌회전할 비트 수 3 을 회전할 비트 수 지정 입력(N)에 쓴다.

(3) 회전된 데이터 값을 출력할 출력변수를 OUT_VALUE 로 설정한다.

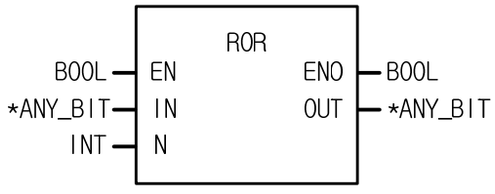
(4) 실행조건 %IX0.0.0 이 On 하면 ROL(왼쪽으로 회전) 펑션이 실행되어 입력변수로 설정된 데이터 비트를 좌로 3 비트 회전하여 출력변수로 선언된 OUT_VALVE 값에 출력된다.



ROR

오른쪽으로 회전 (Rotate Right)

CPU 명	XGI	XGR
적용 가능	●	●

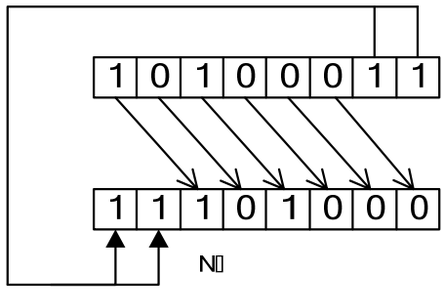
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 회전될 값 N : 회전할 비트 수 **출력** ENO : EN값이 그대로 출력 OUT : 회전된 값

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 타입 중 BOOL 제외

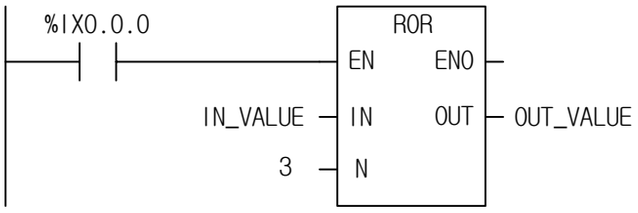
■ 기능

1. 입력 IN을 N 비트 수만큼 오른쪽으로 회전시킵니다.



■ 프로그램 예

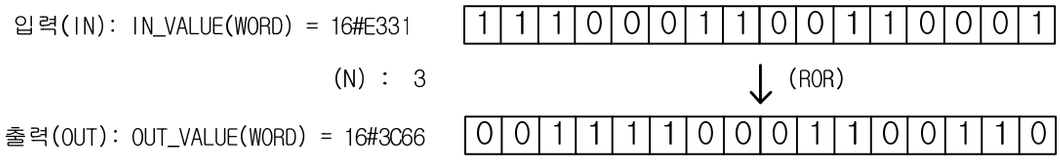
1. 입력 %IX0.0.0 이 On 하면 입력 데이터 값(2#1110_0011_0011_0001: 16#E331)을 우로 3 비트 만큼 회전시키는 프로그램



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

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

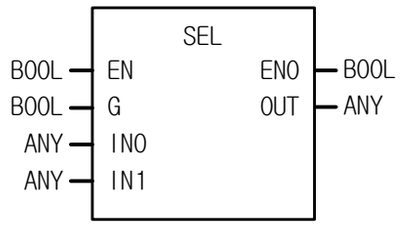
(3) 실행조건 %IX0.0.0 이 On 하면 ROR(오른쪽으로 회전) 펑션이 실행되어 입력변수로 설정된 데이터 비트가 우로 3 비트만큼 회전되어 출력변수로 선언된 OUT_VALUE 값에 출력된다.



SEL

둘 중 선택

CPU 명	XGI	XGR
적용 가능	●	●

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

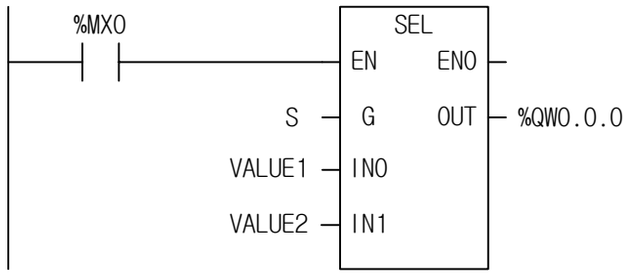
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	INO	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	IN1	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

■ 기능

1. G가 0 이면 INO 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) 실행조건(%MX0)이 On 하면 SEL(둘 중 선택) 펑션이 실행됩니다.

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

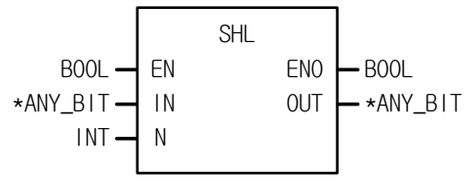
입력(G) : S = 1
(IN0) : VALUE1(WORD) = 16#1110
(IN1) : VALUE2(WORD) = 16#FF00

출력(OUT) : %QW0.0.0(WORD) = 16#FF00

SHL

왼쪽으로 이동 (Shift Left)

CPU 명	XGI	XGR
적용 가능	●	●

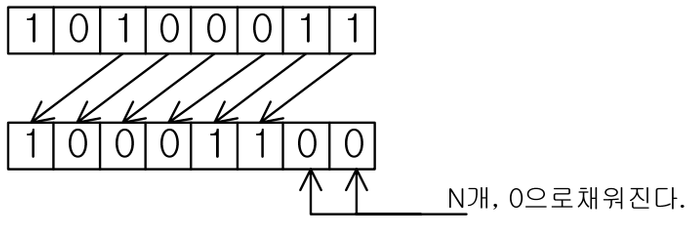
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 이동될 비트 열 N : 이동할 비트 수 **출력** ENO : EN값이 그대로 출력 OUT : 이동된 값

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

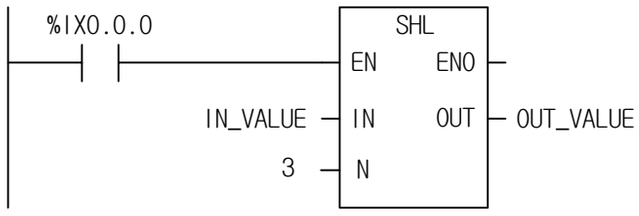
■ 기능

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



■ 프로그램 예

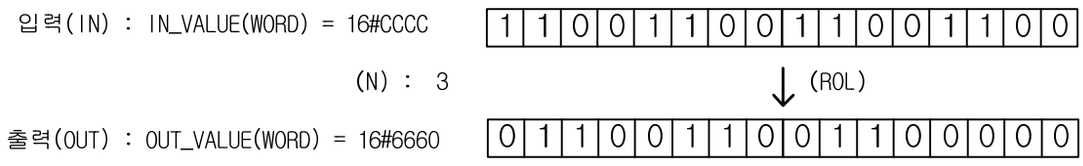
1. 입력 %IX0.0.0 이 On 하면 입력 데이터 값(2#1100_1100_1100_1100: 16#CCCC)을 좌로 3 비트 만큼 이동시키는 프로그램



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

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

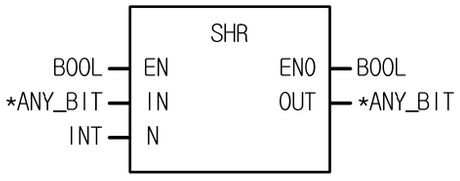
(3) 실행조건(%IX0.0.0)이 On 하면 SHL(왼쪽으로 이동) 펑션이 실행되어 입력변수로 설정된 데이터 비트가 좌로 3 비트 이동하여, 출력변수로 선언된 OUT_VALUE 에 출력됩니다.



SHR

오른쪽으로 이동 (Shift Right)

CPU 명	XGI	XGR
적용 가능	●	●

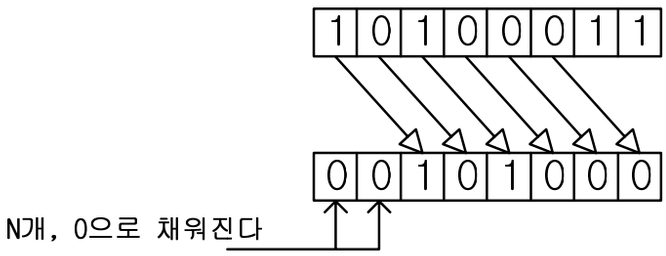
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 이동될 비트 열 N : 이동할 비트 수 **출력** ENO : EN값이 그대로 출력 OUT : 이동된 값

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		○	○	○	○															

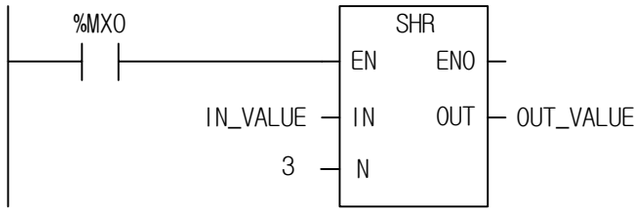
*ANY_BIT: ANY_BIT 중 BOOL 제외

■ 기능

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

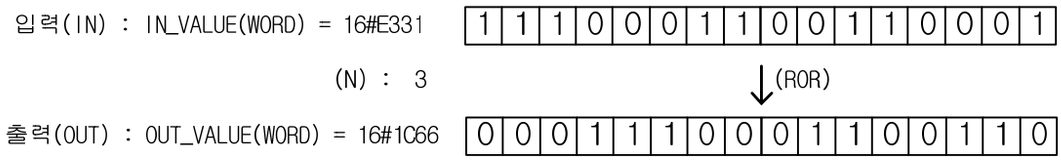


■ 프로그램 예



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

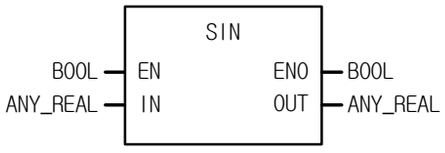
(2) 입력변수로 설정된 데이터 비트가 우로 3 비트 이동하여, 출력변수로 선언된 OUT_VALUE 에 출력됩니다.



SIN

Sine 연산

CPU 명	XGI	XGR
적용 가능	●	●

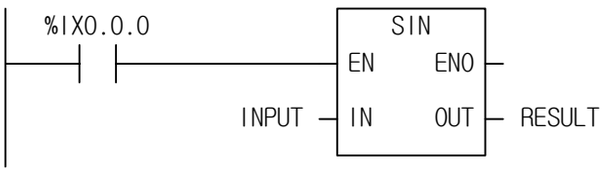
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : Sine 연산의 각도 입력 값(Radian) **출력** ENO : EN값이 그대로 출력 OUT : Sine 연산결과 값 IN, OUT은 같은 데이터 타입이어야 함.

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 Sine 값을 구해 OUT으로 출력시킵니다.
   OUT = SIN (IN)

■ 프로그램 예



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

(2) INPUT으로 선언된 입력변수의 값이 1.0471 .... ($\pi/3$ rad = 60°) 일 때 출력변수로 선언된 RESULT는 0.8660 ....

($\sqrt{3}/2$)이 출력됩니다. $SIN(\pi/3) = \sqrt{3}/2 = 0.8660$

입력(IN) : INPUT(REAL) = 1.0471

↓(SIN)

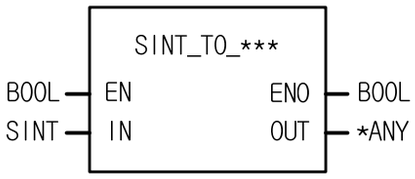
출력(OUT) : RESULT(REAL) = 8.65976572E-01

SINT_TO_***

SINT 타입 변환

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

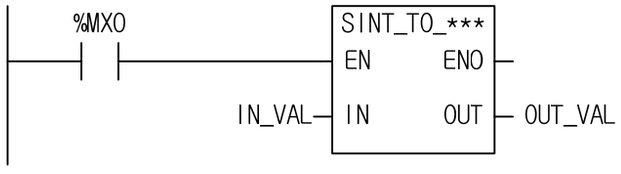
1. IN을 타입 변환해서 OUT으로 출력시킵니다.

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 1비트를 취해 BOOL 타입으로 변환합니다.
SINT_TO_BYTE	BYTE	내부 비트 배열의 변화 없이 BYTE 타입으로 변환합니다.
SINT_TO_WORD	WORD	상위 비트들을 0으로 채운 WORD 타입으로 변환합니다.
SINT_TO_DWORD	DWORD	상위 비트들을 0으로 채운 DWORD 타입으로 변환합니다.
SINT_TO_LWORD	LWORD	상위 비트들을 0으로 채운 LWORD 타입으로 변환합니다.
SINT_TO_REAL	REAL	SINT를 REAL 타입으로 정상 변환합니다.
SINT_TO_LREAL	LREAL	SINT를 LREAL 타입으로 정상 변환합니다.
SINT_TO_STRING	STRING	SINT를 STRING 타입으로 정상 변환합니다.

■ 플래그

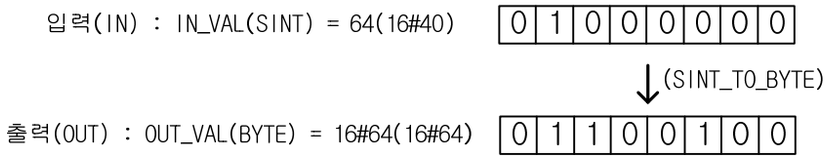
플래그	설명
_ERR	변환에러 발생시 _ERR, _LER 플래그가 셋(Set)됩니다. 에러 발생시 출력 타입의 비트 수만큼 하위 비트를 취해 내부 비트 배열의 변환 없이 출력시킵니다.

■ 프로그램 예



(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.

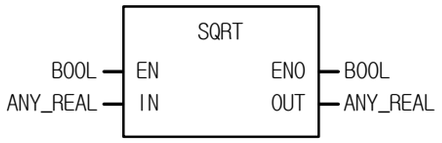


SQRT

제곱근 연산

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

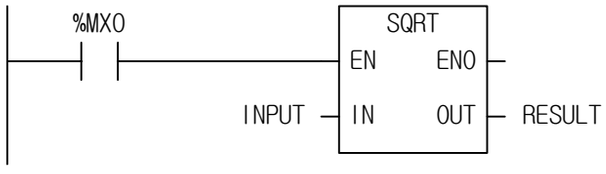
1. IN의 제곱근 값을 구해 OUT으로 출력시킵니다.

   $OUT = \sqrt{IN}$

■ 플래그

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

■ 프로그램 예

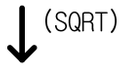


(1) 실행조건(%MX0)이 On 하면 SQRT(제곱근 연산) 펑션이 실행됩니다.

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

$\sqrt{9.0} = 3.0$

입력(IN) : INPUT(REAL) = 9.0



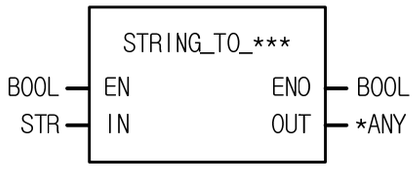
출력(OUT) : RESULT(REAL) = 3.0

STRING_TO_***

STRING 타입 변환

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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.

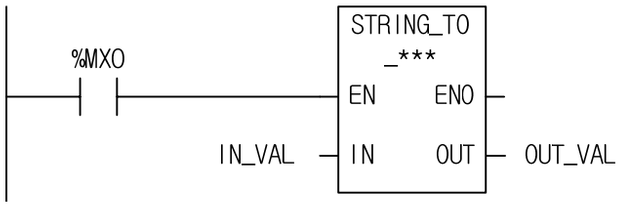
펑 션	출력 타입	동작 설명
STRING_TO_SINT	SINT	STRING을 SINT 타입으로 변환합니다.
STRING_TO_INT	INT	STRING을 INT 타입으로 변환합니다.
STRING_TO_DINT	DINT	STRING을 DINT 타입으로 변환합니다.
STRING_TO_LINT	LINT	STRING을 LINT 타입으로 변환합니다.
STRING_TO_USINT	USINT	STRING을 USINT 타입으로 변환합니다.
STRING_TO_UINT	UINT	STRING을 UINT 타입으로 변환합니다.
STRING_TO_UDINT	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.
STRING_TO_TOD	TOD	STRING을 TOD 타입으로 변환합니다.

펑 션	출력 타입	동작 설명
STRING_TO_TIME	TIME	STRING 을 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) 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)

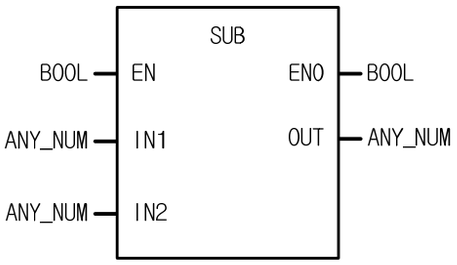
출력(OUT) : OUT_VAL(REAL) = -1.34E12

SUB

빼기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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.

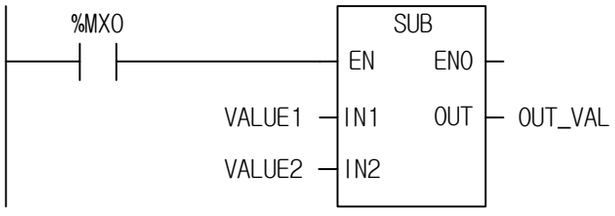
   OUT = IN1 − IN2

■ 플래그

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

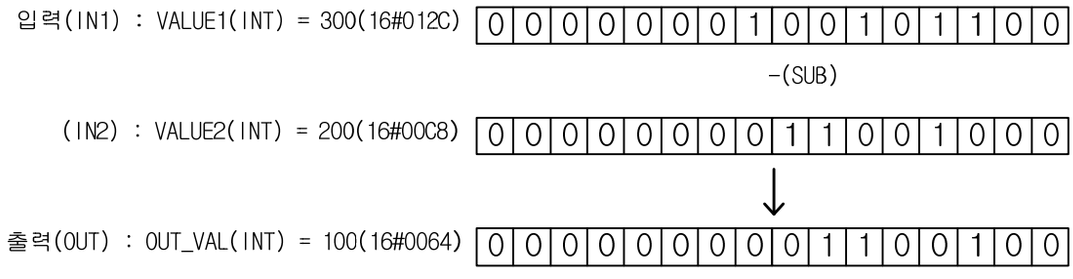
☆ REAL, LREAL 타입 연산에서는 IN1 에서 IN8 로 순차적으로 연산을 하기 때문에 중간 연산과정에서 이미 REAL, LREAL 의 최대값이나 최소값을 넘게 되면 _ERR, _LER 플래그가 셋(Set)되고 결과값은 무한대 값 또는 비정상적인 값 (1.#INF000000000000e+000, 1.#SNAN000000000000e+000, 1.#QNAN000000000000e+000)으로 표시됩니다.

■ 프로그램 예



(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.

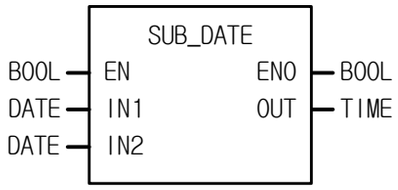


SUB_DATE

날짜 빼기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

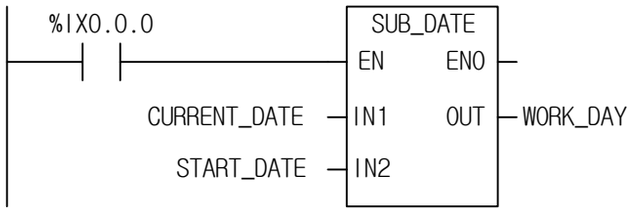
■ 기능

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) 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

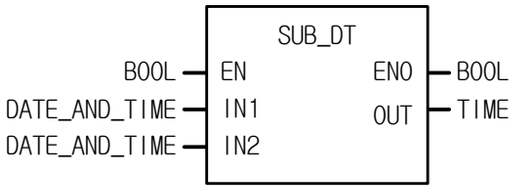
출력(OUT) : WORK_DAY(TIME) = T#44D

SUB_DT

시간과 날짜 빼기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

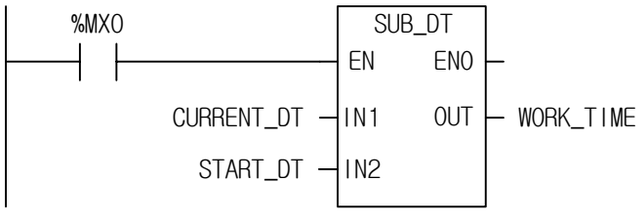
■ 기능

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) 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

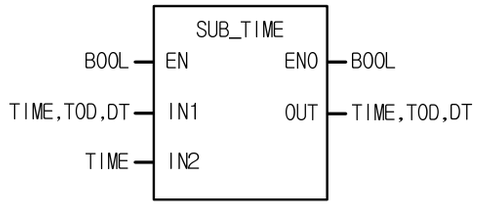
출력(OUT) : WORK_TIME(TIME) = T#2D2H30M

SUB_TIME

시간과 빼기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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																○		○	○	

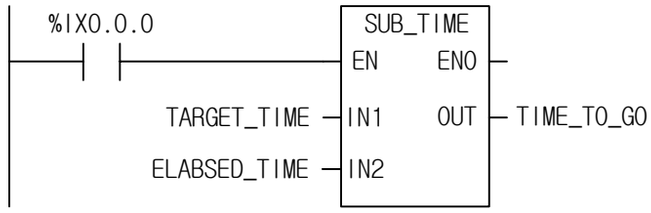
■ 기능

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) 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

↓

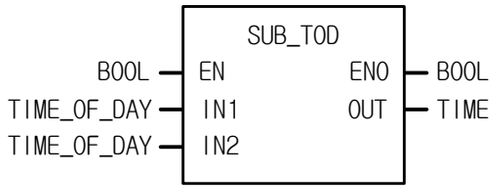
출력(OUT) : TIME_TO_GO(TIME) = T#1H19M29S700MS

SUB_TOD

시각에서 시각빼기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

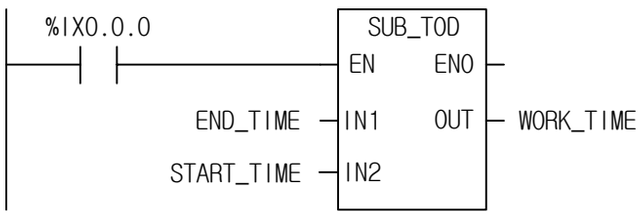
■ 기능

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



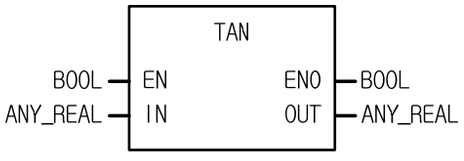
(1) 실행조건(%IX0.0.0)이 On 하면 SUB_TOD(시각에서 시각빼기) 펑션이 실행됩니다.
(2) 입력변수로 선언된 END_TIME 이 TOD#14:20:30.500 이고 작업을 시작한 START_TIME 이 TOD#12:00:00 이면, 출력변수로 선언된 작업에 소요된 시간 WORK_TIME 에는 T#2H20M30S500MS 가 출력됩니다.

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

Tangent 연산

CPU 명	XGI	XGR
적용 가능	●	●

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.

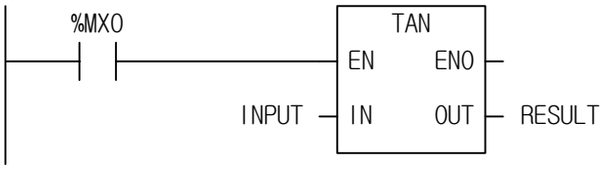
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

■ 기능

1. IN의 Tangent 값을 구해 OUT으로 출력시킵니다.

   OUT = TAN(IN)

■ 프로그램 예



(1) 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

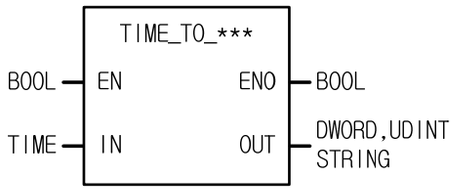
↓ (TAN)

출력(OUT) : RESULT(REAL) = 9.99803722E-01

TIME_TO_***

TIME 타입변환

CPU 명	XGI	XGR
적용 가능	●	●

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

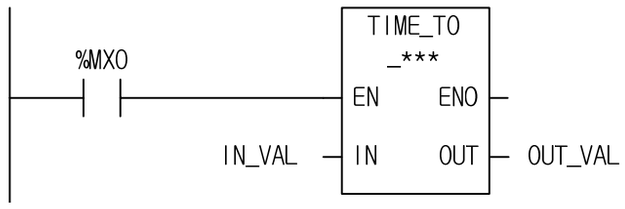
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT				○								○								○

■ 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) 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)

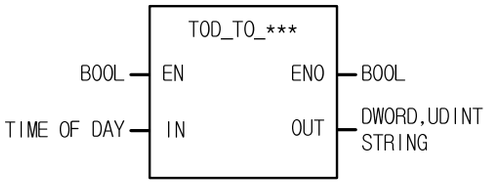
출력(OUT) : OUT_VAL(UDINT) = 120(16#78)

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

TOD_TO_***

TOD 타입변환

CPU 명	XGI	XGR
적용 가능	●	●

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

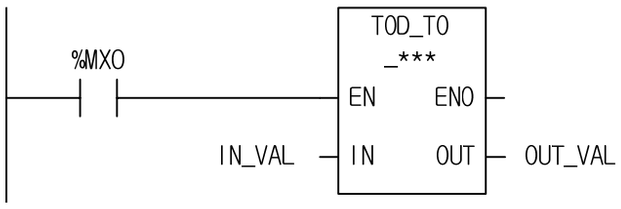
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT				○								○								○

■ 기능

1. IN 을 타입 변환해서 OUT 으로 출력시킵니다.

펑 션	출력 타입	동작 설명
TOD_TO_UDINT	UDINT	TOD 를 UDINT 타입으로 변환합니다. 데이터(내부 비트 배열 상태)의 변화 없이 데이터 타입만 변환합니다.
TOD_TO_DWORD	DWORD	TOD 를 DWORD 타입으로 변환합니다. 데이터(내부 비트 배열 상태)의 변화 없이 데이터 타입만 변환합니다.
TOD_TO_STRING	STRING	TOD 를 STRING 타입으로 변환합니다.

■ 프로그램 예



(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)

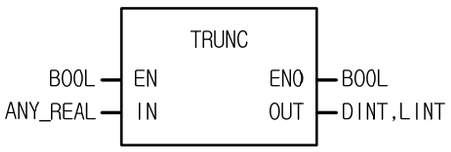
출력(OUT) : OUT_VAL(STRING) = 'TOD#12:00:00'

TRUNC

소수점 이하 값을 버리고 정수로 변환

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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								○	○											

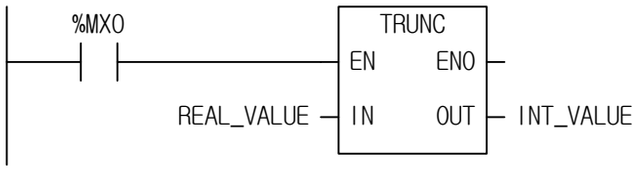
■ 기능

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) 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)

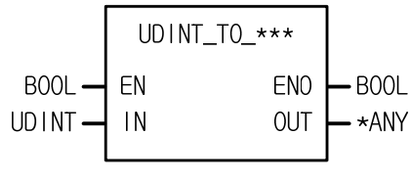
출력(OUT) : INT_VALUE(INT) = 1

UDINT_TO_***

UDINT 타입 변환

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_ERR, _LER

펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 IN : 타입 변환할 Unsigned Double 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 UDINT, DATE, DT 제외

■ 기능

1. IN 을 타입 변환해서 OUT 으로 출력시킵니다.

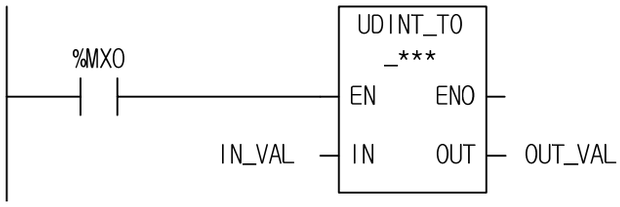
펑 션	출력 타입	동작 설명
UDINT_TO_SINT	SINT	입력이 0~127 일 경우 정상 변환되나 그 외 값은 에러가 발생합니다.
UDINT_TO_INT	INT	입력이 0~32,767 일 경우 정상 변환되나 그 외 값은 에러가 발생합니다.
UDINT_TO_DINT	DINT	입력이 0~2,147,483,647 일 경우 정상 변환되나 그 외 값은 에러가 발생합니다.
UDINT_TO_LINT	LINT	UDINT 를 LINT 타입으로 정상 변환합니다.
UDINT_TO_USINT	USINT	입력이 0~255 일 경우 정상 변환되나 그 외 값은 에러가 발생합니다.
UDINT_TO_UINT	UINT	입력이 0~65,535 일 경우 정상 변환되나 그 외 값은 에러가 발생합니다.
UDINT_TO_ULINT	ULINT	UDINT 를 ULINT 타입으로 정상 변환합니다.
UDINT_TO_BOOL	BOOL	하위 1 비트를 취해 BOOL 타입으로 변환합니다.
UDINT_TO_BYTE	BYTE	하위 8 비트를 취해 BYTE 타입으로 변환합니다.
UDINT_TO_WORD	WORD	하위 16 비트를 취해 WORD 타입으로 변환됩니다.
UDINT_TO_DWORD	DWORD	내부 비트 배열의 변환 없이 DWORD 타입으로 변환합니다.
UDINT_TO_LWORD	LWORD	상위 비트를 0 으로 채운 LWORD 타입으로 변환합니다.
UDINT_TO_REAL	REAL	UDINT 를 REAL 타입으로 변환합니다. 변환 중 정밀도에 따른 오차가 발생할 수 있습니다.
UDINT_TO_LREAL	LREAL	UDINT 를 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) 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

↓

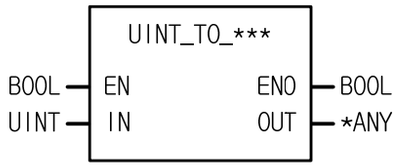
출력(OUT) : OUT_VAL(TIME) = T#123MS

UINT_TO_***

UINT 타입 변환

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 Unsigned Integer 값 **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 UINT, TIME, TOD, DT 제외

■ 기능

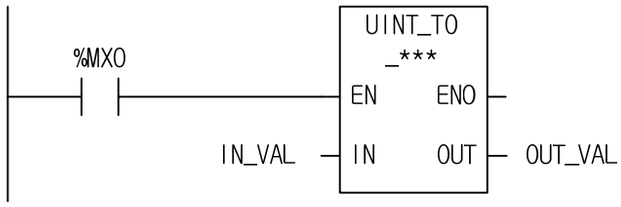
1. IN을 타입 변환해서 OUT으로 출력시킵니다.

펑 션	출력타입	동작 설명
UINT_TO_SINT	SINT	입력이 0~127일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
UINT_TO_INT	INT	입력이 0~32,767일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
UINT_TO_DINT	DINT	UINT를 UDINT 타입으로 정상 변환합니다.
UINT_TO_LINT	LINT	UINT를 ULINT 타입으로 정상 변환합니다.
UINT_TO_USINT	USINT	입력이 0~255일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
UINT_TO_UDINT	UDINT	UINT를 UDINT 타입으로 정상 변환합니다.
UINT_TO_ULINT	ULINT	UINT를 ULINT 타입으로 정상 변환합니다.
UINT_TO_BOOL	BOOL	하위 1비트를 취해 BOOL 타입으로 변환합니다.
UINT_TO_BYTE	BYTE	하위 8비트를 취해 BYTE 타입으로 변환합니다.
UINT_TO_WORD	WORD	내부 비트 배열의 변환 없이 WORD 타입으로 변환합니다.
UINT_TO_DWORD	DWORD	상위 비트를 0으로 채운 DWORD 타입으로 변환합니다.
UINT_TO_LWORD	LWORD	상위 비트를 0으로 채운 LWORD 타입으로 변환합니다.
UINT_TO_REAL	REAL	UINT를 REAL 타입으로 변환합니다.
UINT_TO_LREAL	LREAL	UINT를 LREAL 타입으로 변환합니다.
UINT_TO_DATE	DATE	내부 비트 배열의 변환 없이 DATE 타입으로 변환합니다.
UINT_TO_STRING	STRING	UINT를 STRING 타입으로 변환합니다.

■ 플래그

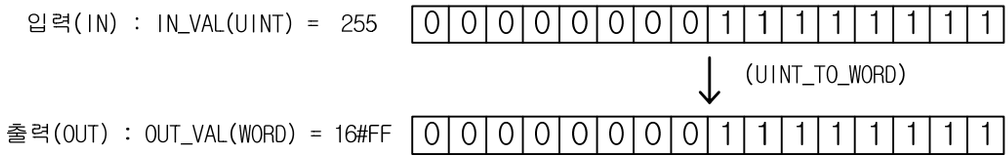
플래그	설명
_ERR	변환에러 발생시 _ERR, _LER 플래그가 셋(Set)됩니다. 에러 발생시 출력 타입의 비트 수만큼 하위 비트를 취해 내부 비트 배열의 변환 없이 출력시킵니다.

■ 프로그램 예



(1) 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.

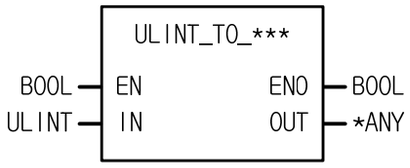


ULINT_TO_***

ULINT 타입 변환

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 타입 변환할 Unsigned Long Integer 값 **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 UINT, TIME, TOD, DT 제외

■ 기능

1. IN을 타입 변환해서 OUT으로 출력시킵니다.

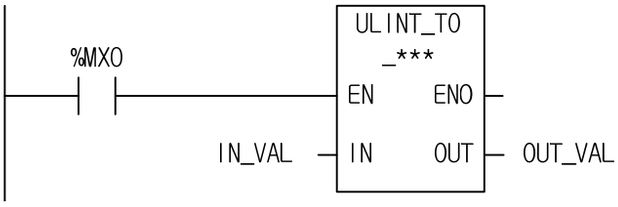
펑 션	출력 타입	동작 설명
ULINT_TO_SINT	SINT	입력이 0~127일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
ULINT_TO_INT	INT	입력이 0~32,767일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
ULINT_TO_DINT	DINT	입력이 0~2^{31}-1일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
ULINT_TO_LINT	LINT	입력이 0~2^{63}-1일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
ULINT_TO_USINT	USINT	입력이 0~255일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
ULINT_TO_UINT	UINT	입력이 0~65,535일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
ULINT_TO_UDINT	UDINT	입력이 0~2^{32}-1일 경우 정상 변환되나, 그 외 값은 에러가 발생합니다.
ULINT_TO_BOOL	BOOL	하위 1비트를 취해 BOOL 타입으로 변환합니다.
ULINT_TO_BYTE	BYTE	하위 8비트를 취해 BYTE 타입으로 변환합니다.
ULINT_TO_WORD	WORD	하위 16비트를 취해 WORD 타입으로 변환합니다.
ULINT_TO_DWORD	DWORD	하위 32비트를 취해 DWORD 타입으로 변환합니다.
ULINT_TO_LWORD	LWORD	내부 비트 배열의 변환 없이 LWORD 타입으로 변환합니다.
ULINT_TO_REAL	REAL	ULINT를 REAL 타입으로 변환합니다. 변환 중 정밀도에 따른 오차가 발생할 수 있습니다.
ULINT_TO_LREAL	LREAL	ULINT를 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) 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)

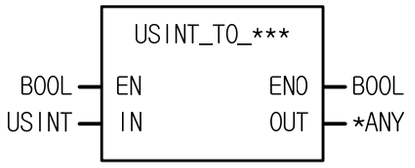
출력(OUT) : OUT_VAL(LINT) = 123,567,899

USINT_TO_***

USINT 타입 변환

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

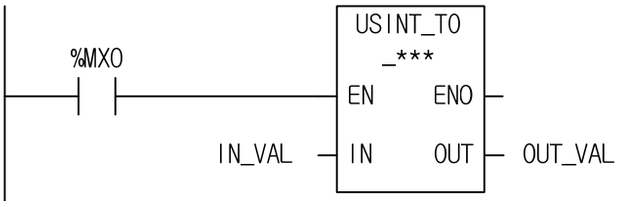
1. IN 을 타입 변환해서 OUT 으로 출력시킵니다.

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

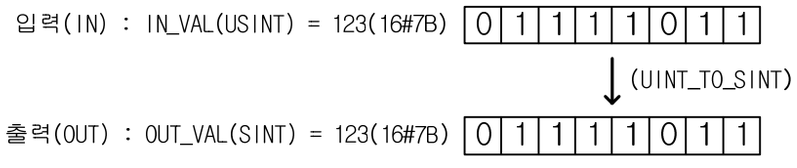
플래그	설명
_ERR	변환에러 발생시 _ERR, _LER 플래그가 셋(Set)됩니다. 에러 발생시 출력 타입의 비트 수만큼 하위 비트를 취해 내부 비트 배열의 변환 없이 출력 시킵니다.

■ 프로그램 예



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

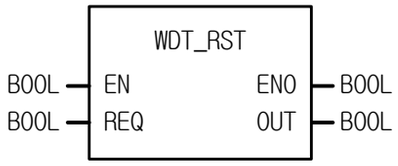
(2) 입력변수로 선언된 IN_VAL(USINT 타입) = 123 이면, 출력변수로 선언된 OUT_VAL(SINT 타입) = 123 이 됩니다.



WDT_RST

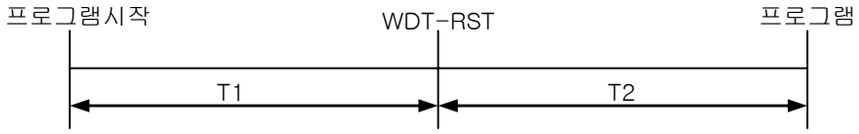
Watch_Dog 타이머 초기화

CPU 명	XGI	XGR
적용 가능	●	●

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 1 출력

■ 기능

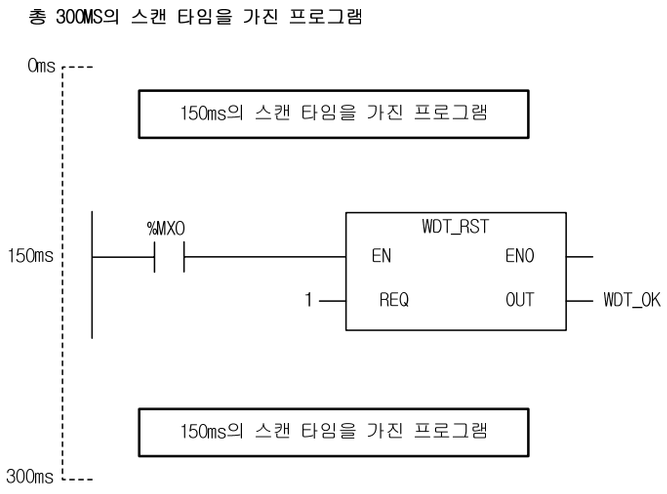
1. 프로그램 중에서 Watch-Dog Timer 를 Reset 시킵니다.
2. 프로그램에서 스캔 타임이 조건에 의해 설정된 Watch-Dog Time 시간을 넘는 경우 사용합니다.
3. 스캔 타임이 매번 설정된 스캔 Watch-Dog Time 을 넘는 경우는 스캔 워치 독 타임의 설정 값으로 변경해 주십시오.
4. 프로그램의 0 Line 부터 WDT_RST 펑션까지의 시간(T1)과 WDT_RST 펑션부터 프로그램 끝까지의 시간(T2) 어느 쪽도 스캔 워치 독 타임 설정 값을 넘지 않도록 설정해야 합니다.



5. 프로그램의 WDT_RST 펑션은 1 스캔 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.



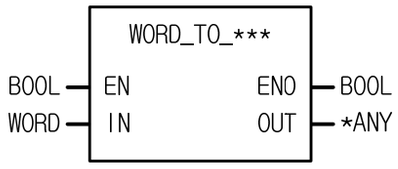
(1) 실행조건(%MX0)이 On 하면 WDT_RST(Watch Dog 타이머 초기화) 펑션이 실행합니다.

(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_***

WORD 타입변환

CPU 명	XGI	XGR
적용 가능	●	●

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	○	○		○	○	○	○	○	○	○	○	○	○				○			○

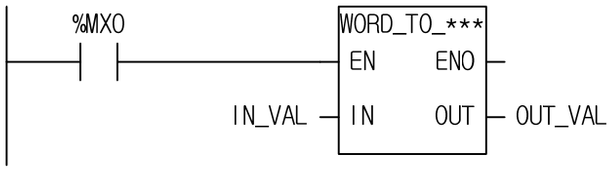
*ANY: ANY 타입 중 WORD, REAL, LREAL, TIME, TOD, DT 제외

■ 기능

1. IN을 타입 변환해서 OUT으로 출력시킵니다.

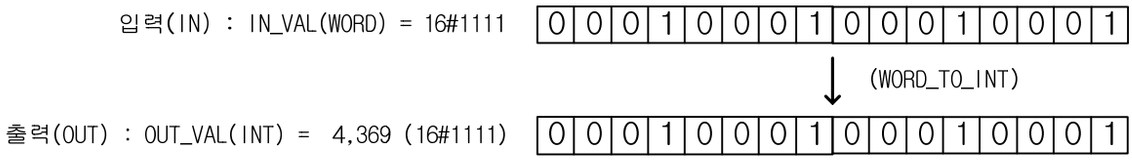
펑 션	출력 타입	동작 설명
WORD _TO_SINT	SINT	하위 8비트를 취해 SINT 타입으로 변환합니다.
WORD _TO_INT	INT	내부 비트 배열의 변환 없이 INT 타입으로 변환합니다.
WORD _TO_DINT	DINT	상위 비트를 0으로 채운 DINT 타입으로 변환합니다.
WORD _TO_LINT	LINT	상위 비트를 0으로 채운 LINT 타입으로 변환합니다.
WORD _TO_USINT	USINT	하위 8비트를 취해 SINT 타입으로 변환합니다.
WORD _TO_UINT	UINT	내부 비트 배열의 변환 없이 INT 타입으로 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) 입력조건(%MX0)이 On 하면 WORD_TO_*** 펑션이 실행됩니다.

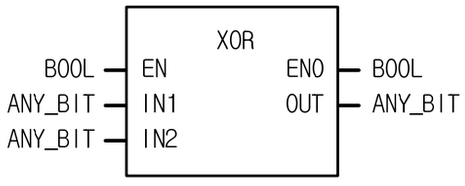
(2) 입력변수로 선언된 IN_VAL(WORD 타입) = 2#0001_0001_0001_0001 이면, 출력변수로 선언된 OUT_VAL(INT 타입) = 4,096 + 256 + 16 + 1 = 4,369 가 됩니다.



XOR

배타적 논리합

CPU 명	XGI	XGR
적용 가능	●	●

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.

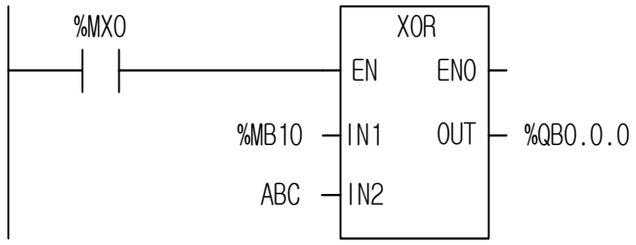
   IN1 1111 0000

   XOR

   IN2 1010 1010

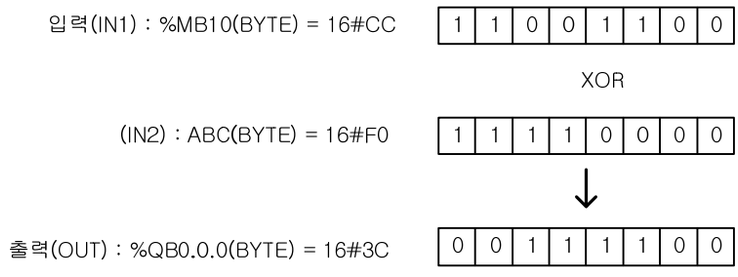
   OUT 0101 1010

■ 프로그램 예



(1) 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.

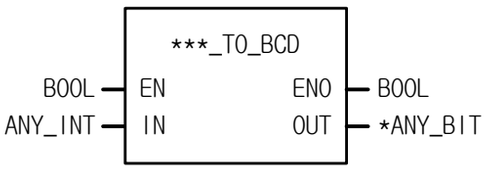


***_TO_BCD

ANY 타입을 BCD 타입 변환

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

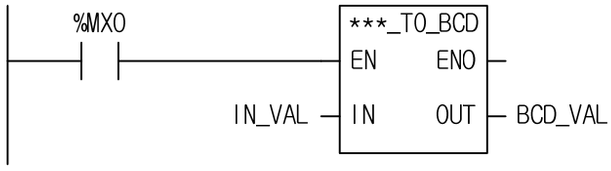
1. IN 을 타입 변환해서 OUT 으로 출력시킵니다.

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	

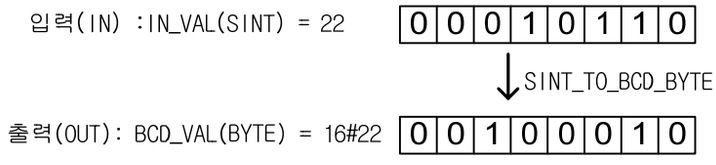
■ 플래그

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

■ 프로그램 예



(1) 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.



제 8 장 응용 펑션

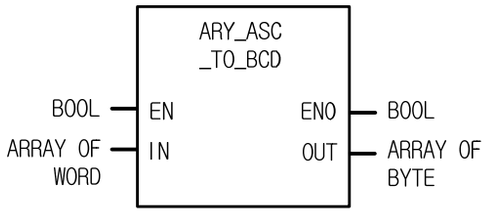
7 장에서 설명한 기본 펑션과는 다른 응용 펑션에 대하여 설명합니다.

ARY_ASC_TO_BCD

입력 ASCII Array, 출력 BCD Array

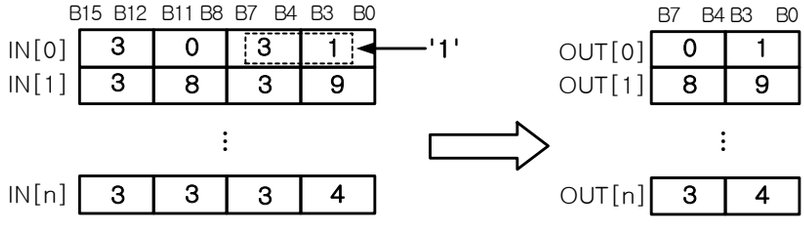
CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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.

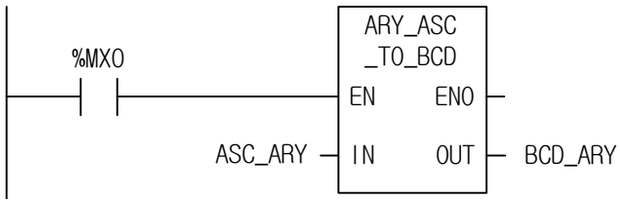


■ 플래그

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 Array의 개수가 서로 다르면 _ERR, _LER 플래그가 발생하는데 출력단 Array 변수 생략시 어레이의 개수를 0으로 보아 _ERR, _LER 플래그가 발생합니다.

■ 프로그램 예



(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.

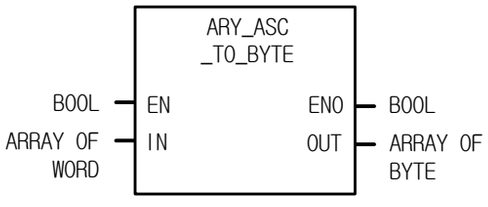
BYTE_ARY[0]	01
BYTE_ARY[1]	89
BYTE_ARY[2]	34

ARY_ASC_TO_BYTE

입력 ASCII Array, 출력 BYTE Array

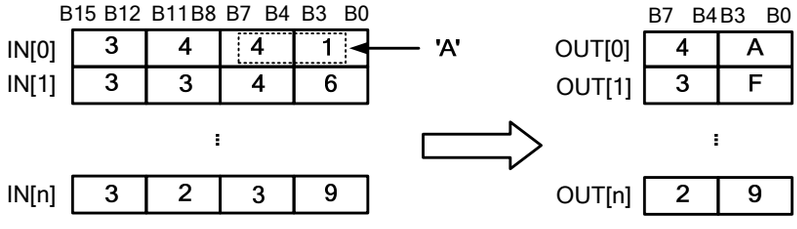
CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_ERR, _LER

펑 션	설 명
	**입력** EN : 1 일 때 펑션 실행 IN1 : ASCII Array 입력 **출력** ENO : 에러 없이 실행되면 1 을 출력 OUT : BYTE Array 출력

■ 기능

1. ASCII 데이터를 지니고 있는 WORD Array 를 입력 받아서, 16 진수(HEX) 값인 BYTE Array 로 변환합니다.

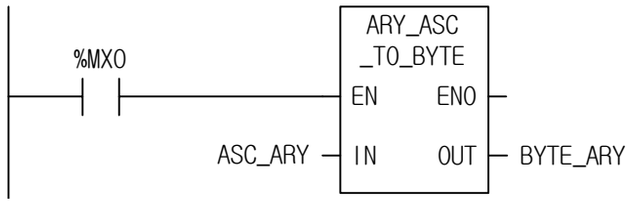


■ 플래그

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, OUT 에 해당하는 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) 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.

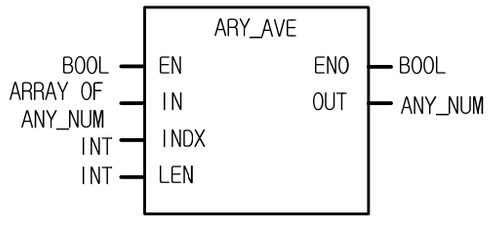
BYTE_ARY[0]	4A
BYTE_ARY[1]	3F
BYTE_ARY[2]	29

ARY_AVE

Array 평균값 구하기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

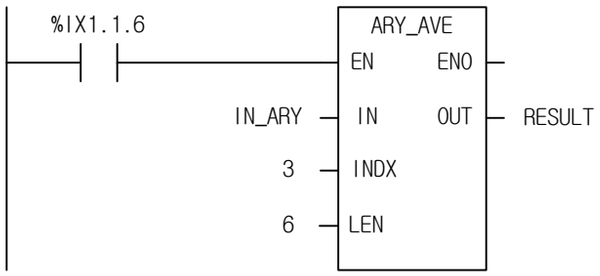
1. ARY_AVE 펑션은 Array 내부의 지정된 영역에 대하여 평균값을 구합니다.
2. 출력 타입은 입력 Array 타입과 동일하게 설정되어 있습니다.
3. LEN 이 마이너스일 경우에는 Array 인덱스부터 (Array 인덱스 - |LEN|) 사이의 원소들에 대한 평균값을 구합니다. 평균값은 반올림하여 출력합니다.

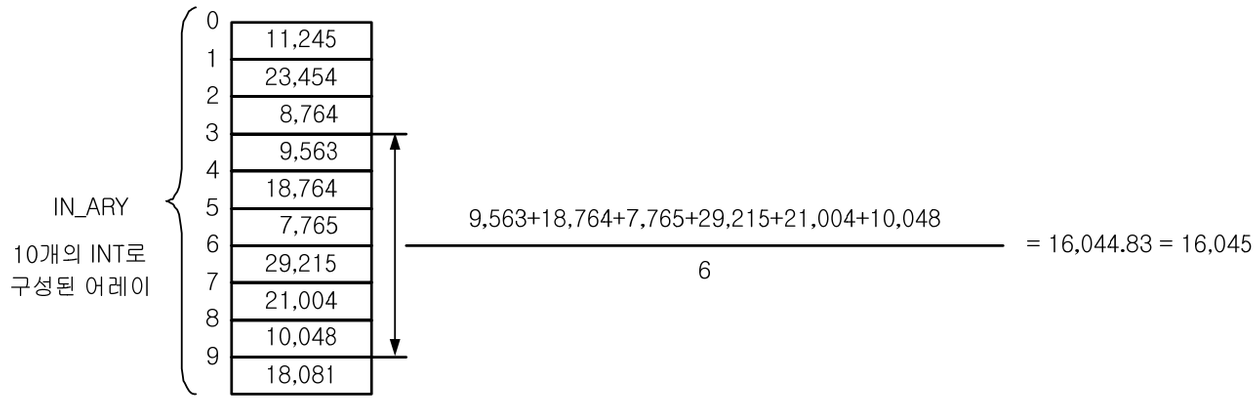
펑션	출력 타입	동작 설명
ARY_AVE	SINT	SINT 값들의 평균값을 구합니다. (소수점 이하는 반올림)
ARY_AVE	INT	INT 값들의 평균값을 구합니다. (소수점 이하는 반올림)
ARY_AVE	DINT	DINT 값들의 평균값을 구합니다. (소수점 이하는 반올림)
ARY_AVE	LINT	LINT 값들의 평균값을 구합니다. (소수점 이하는 반올림)
ARY_AVE	USINT	USINT 값들의 평균값을 구합니다. (소수점 이하는 반올림)
ARY_AVE	UINT	UINT 값들의 평균값을 구합니다. (소수점 이하는 반올림)
ARY_AVE	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





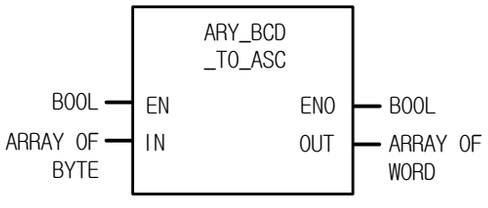
(1) 입력조건 (%IX1.1.6)이 On 되면, ARY_AVE 펑션의 INT 타입이 실행됩니다.
(2) ARRAY 내의 값이 위의 그림과 같을 경우 Array 인덱스 3 번째로부터 6 개의 원소에 대한 평균값을 구합니다
(3) 평균값이 16,044.8 이지만 출력 타입이 INT 이므로 반올림을 수행하여 16,045 를 출력합니다.

ARY_BCD_TO_ASC

입력 BCD Array, 출력 ASCII Array

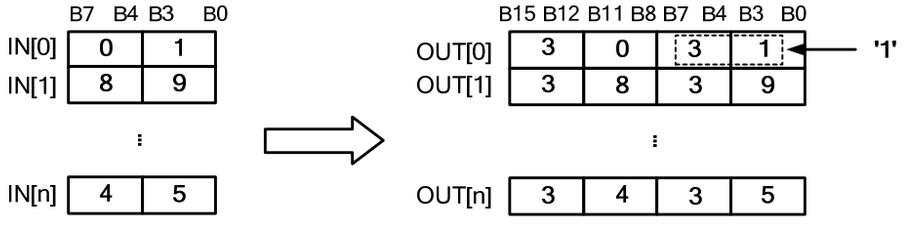
CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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.

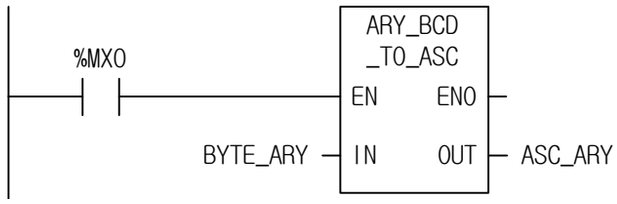


■ 플래그

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) 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.

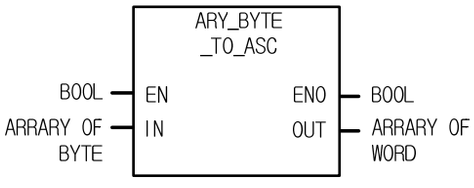
ASC_ARY[0]	3031
ASC_ARY[1]	3839
ASC_ARY[2]	3435

ARY_BYTE_TO_ASC

입력 BYTE Array, 출력 ASCII Array

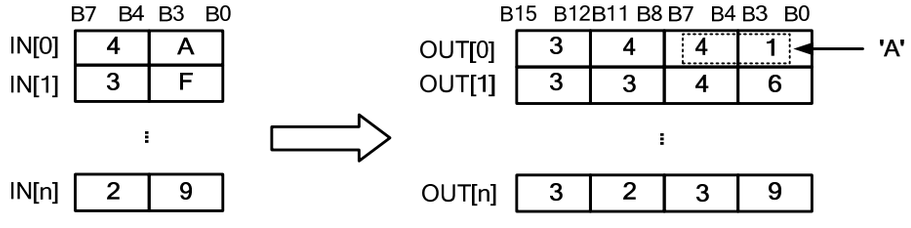
CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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.

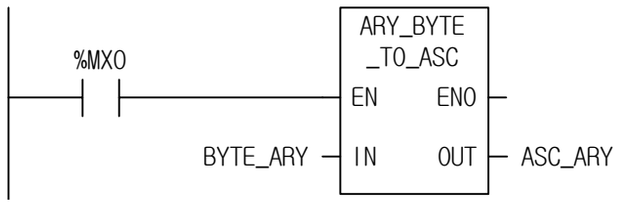


■ 플래그

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) 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.

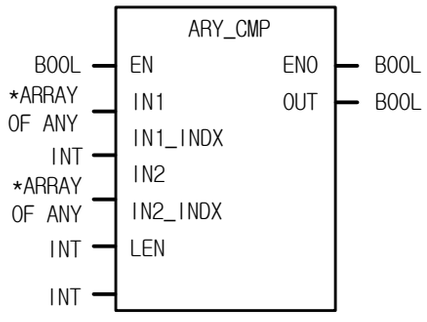
ASC_ARY[0]	3441
ASC_ARY[1]	3346
ASC_ARY[2]	3239

ARY_CMP

Array 비교

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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.

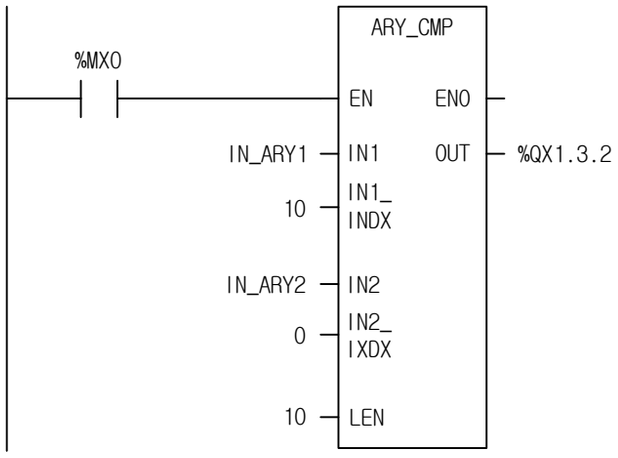
펑션	입력 Array 타입	동작 설명
ARY_CMP	BOOL	2개의 BOOL Array를 서로 비교합니다.
ARY_CMP	BYTE	2개의 BYTE Array를 서로 비교합니다.
ARY_CMP	WORD	2개의 WORD Array를 서로 비교합니다.
ARY_CMP	DWORD	2개의 DWORD Array를 서로 비교합니다.
ARY_CMP	LWORD	2개의 LWORD Array를 서로 비교합니다.
ARY_CMP	SINT	2개의 SINT Array를 서로 비교합니다.
ARY_CMP	INT	2개의 INT Array를 서로 비교합니다.
ARY_CMP	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) 입력조건(%MX0)이 On 되면 ARY_CMP 펑션이 실행됩니다.

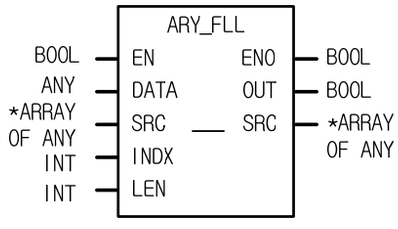
(2) IN_ARY1 이 100 개의 원소를 지닌 TIME Array 이고, IN_ARY2 가 10 개의 원소를 지닌 TIME Array 일 경우 IN_ARY1 의 11 번째 원소로부터 20 번째까지 10 개의 원소를 IN_ARY 2 의 첫번째 원소로부터 10 번째까지 10 개의 원소와 각각 비교하여 모두 동일하면 접점 출력 %QX1.3.2 가 On 됩니다.

ARY_FLL

Array 내부 영역 채우기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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.

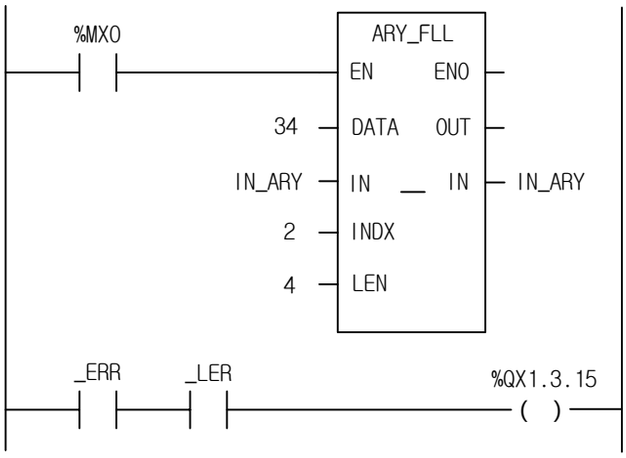
펑션	입출력 Array 타입	동작 설명
ARY_FLL	BOOL	BOOL Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	BYTE	BYTE Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	WORD	WORD Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	DWORD	DWORD Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	LWORD	LWORD Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	SINT	SINT Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	INT	INT Array 내부를 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.
ARY_FLL	UDINT	UDINT Array 내부를 지정된 값으로 채웁니다.

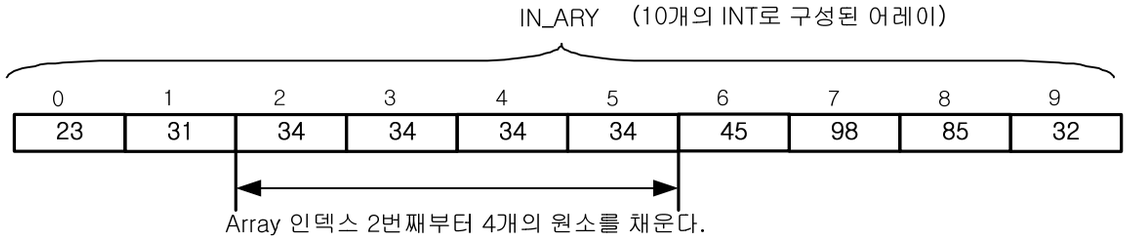
펑션	입출력 Array 타입	동작 설명
ARY_FLL	ULINT	ULINT Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	REAL	REAL Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	LREAL	LREAL Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	TIME	TIME Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	DATE	DATE Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	TOD	TOD Array 내부를 지정된 값으로 채웁니다.
ARY_FLL	DT	DT Array 내부를 지정된 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) 입력조건(%MX0)이 On 되면, ARY_FLL 펑션이 실행됩니다.

(2) Array 인덱스 2 번째부터 4 개의 원소를 지정된 값 34 로 채웁니다.

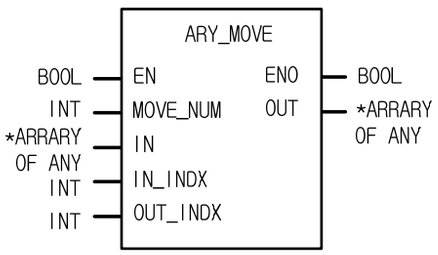
(3) 만약 LEN 을 9 로 대체하면 Array 의 전체 개수를 초과하므로 에러가 발생하여 _ERR 과 _LER 플래그가 On 되므로 출력 접점 %QX1.13.15 가 On 됩니다.

ARY_MOVE

Array 복사

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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 사용 불가) IN_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 사용 불가)

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

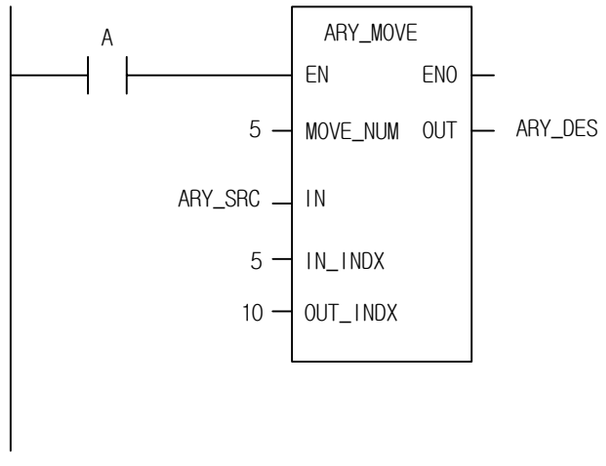
■ 플래그

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) 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
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

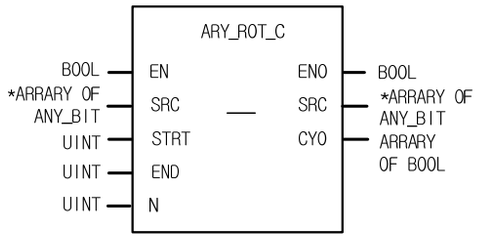
실행전				실행후			
ARY_SRC[9]	99	ARY_DES[9]	16#9	ARY_SRC[9]	99	ARY_DES[9]	16#9
-	-	ARY_DES[10]	16#A	-	-	ARY_DES[10]	16#37
-	-	ARY_DES[11]	16#B	-	-	ARY_DES[11]	16#42
-	-	ARY_DES[12]	16#C	-	-	ARY_DES[12]	16#4D
-	-	ARY_DES[13]	16#D	-	-	ARY_DES[13]	16#58
-	-	ARY_DES[14]	16#E	-	-	ARY_DES[14]	16#63

ARY_ROT_C

Array의 Bit Rotate with Carry

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_ERR, _LER

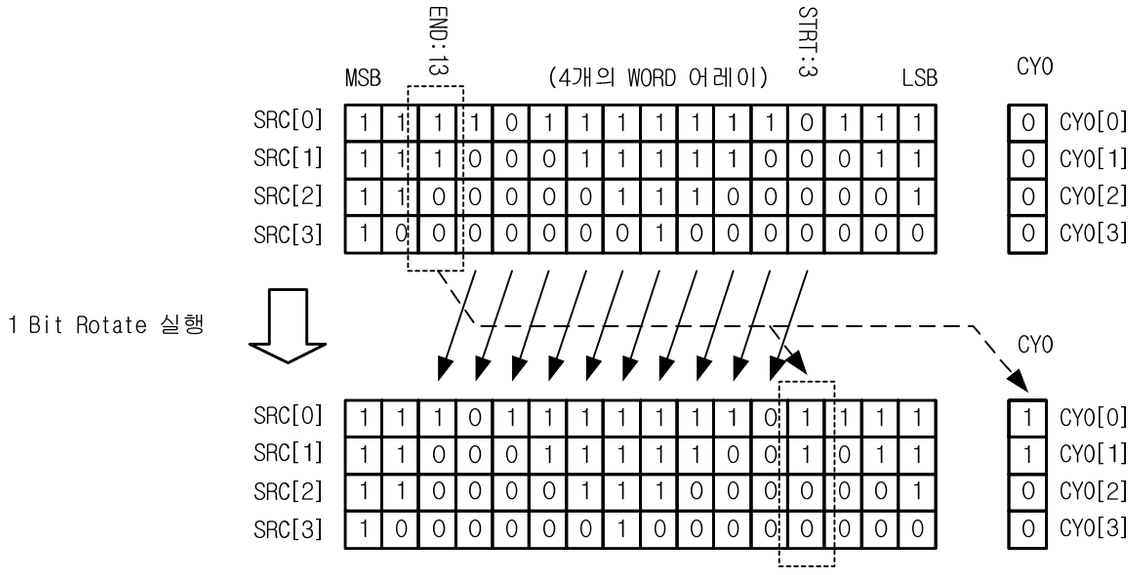
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 STRT : Rotate할 시작위치 END : Rotate할 종료위치 N : Rotate 시킬 개수 **출력** ENO : 에러 없이 실행되면 1을 출력 CYO : Rotate후에 출력되는 Carry bit Array **입출력** SRC : Rotate 시킬 Source Array

ANY 타입 변수명	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.



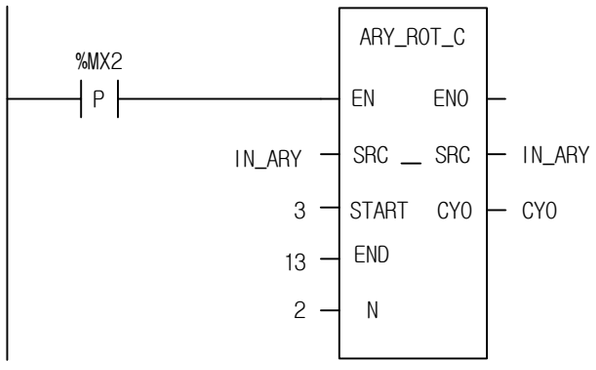
펑션	입출력 Array 타입	동작 설명
ARY_ROT_C	BYTE	각 타입 Array 원소들의 비트들을 지정된 범위 내에서 지정된 비트 수만큼 Rotate 시킵니다.
ARY_ROT_C	WORD	
ARY_ROT_C	DWORD	
ARY_ROT_C	LWORD	

■ 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.

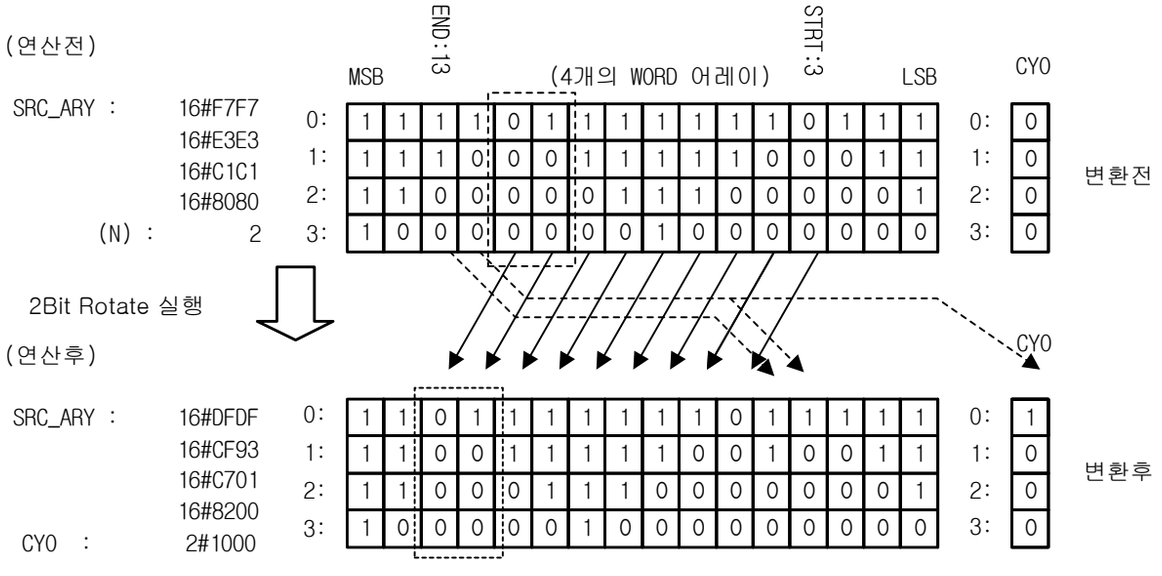
■ 프로그램 예



(1) 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.

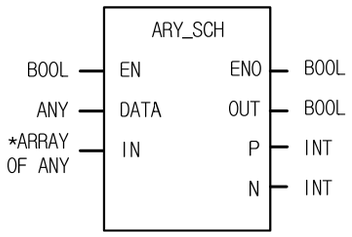
(3) 회전된 결과는 다시 IN_ARY 에 저장되며 이때 출력되는 캐리 비트열은 CYO Array 로 출력됩니다.



ARY_SCH

Array 탐색(search)

CPU 명	XGI	XGR
적용 가능	●	●

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

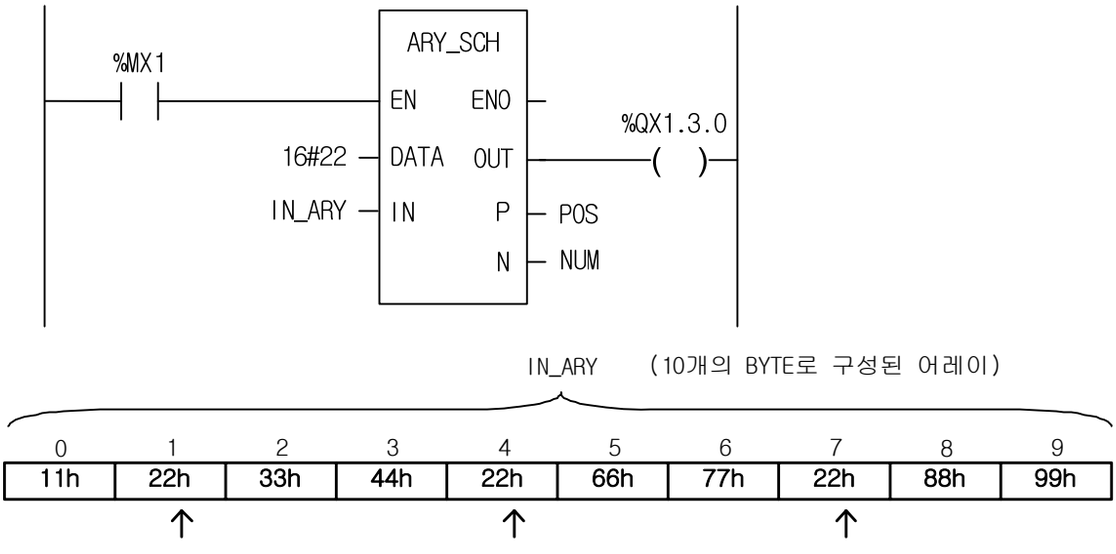
■ 기능

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.
ARY_SCH	ULINT	ULINT Array 내에서 탐색합니다.

펑션	입력 Array 타입	동작 설명
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) 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.

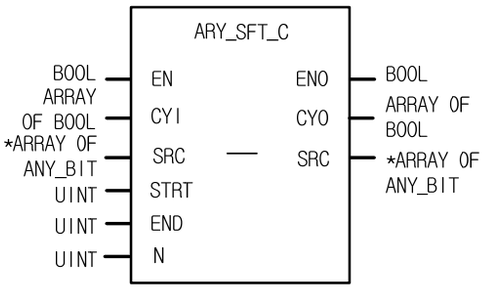
ARY_SFT_C

Array의 Bit Shift Left with Carry

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

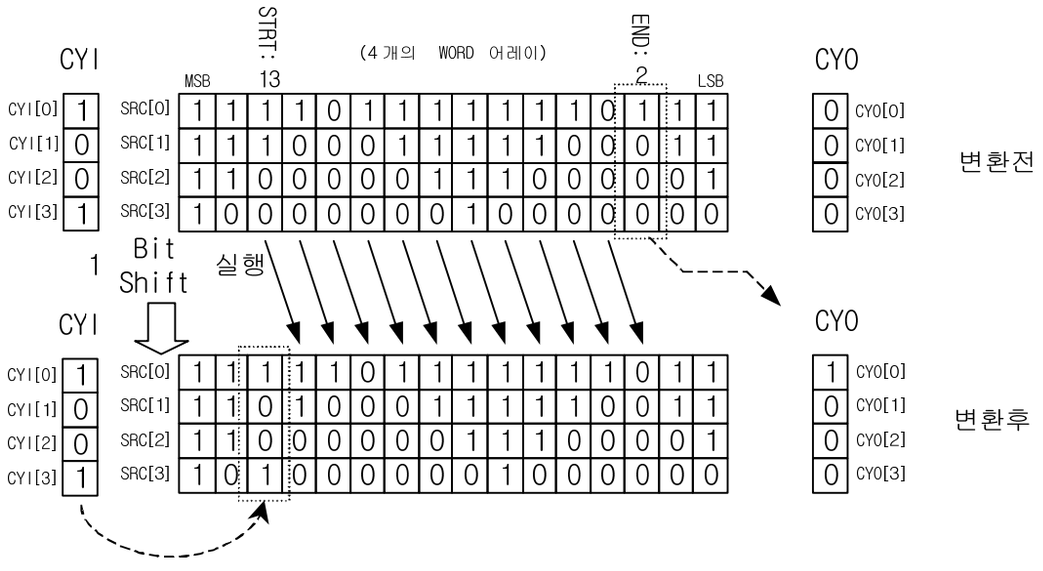
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	SRC		○	○	○	○															

*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: STRT 와 END 로 이동할 비트의 범위를 지정됩니다.
   - 이동 방향 및 횟수: STRT 에서 END 방향으로 지정된 횟수(N)만큼 이동합니다.
   - 입력 데이터 지정: Shift 한 후에 비워지는 위치를 입력 데이터(CYI)로 채웁니다.
   - 출력: 데이터 조작의 결과는 SRC 에 지정된 어레이에 저장되며, END 위치에서 Shift 동작으로 밀려 나온 비트 열 데이터는 CYO 로 출력됩니다.



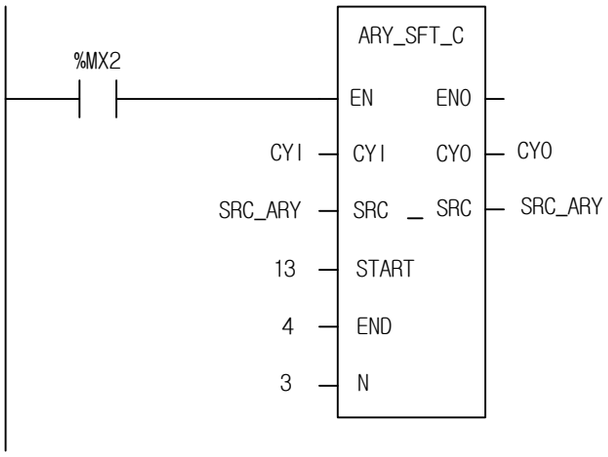
펑션	입출력 Array 타입	동작 설명
ARY_SFT_C	BYTE	각 타입 Array 원소들의 비트들을 지정된 범위 내에서 지정된 비트 수만큼 Shift 시킵니다.
ARY_SFT_C	WORD	
ARY_SFT_C	DWORD	
ARY_SFT_C	LWORD	

■ 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

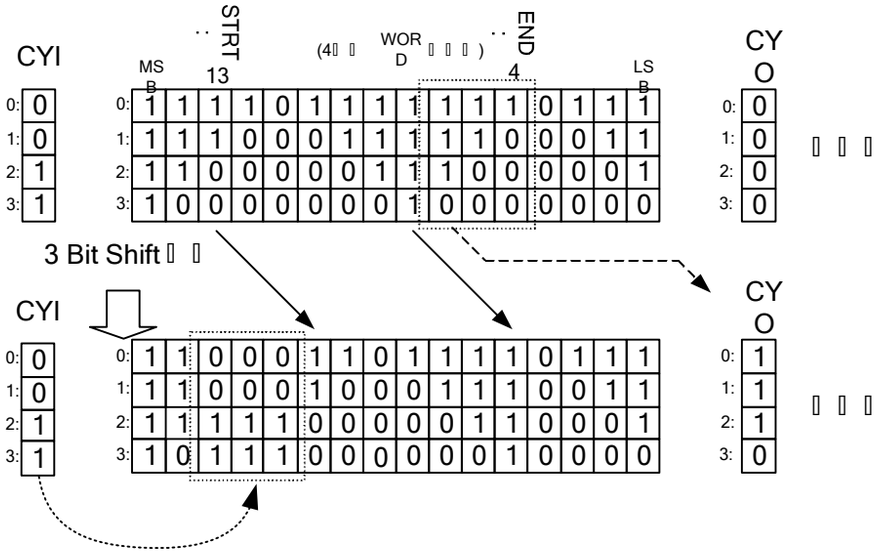


(1) 입력조건(%MX2)이 On 되면, ARY_SFT_C 펑션이 실행됩니다.

(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.

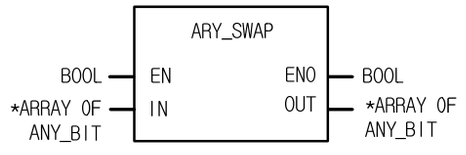


ARY_SWAP

Array 원소 데이터의 상위 하위 바꾸기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

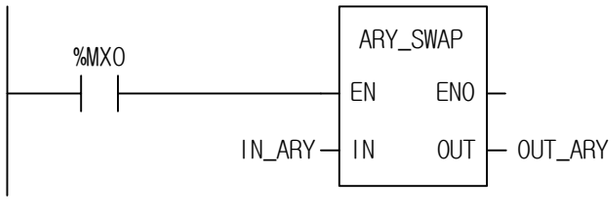
1. 입력된 Array 원소를 2개의 크기로 구분하여 상위와 하위를 서로 교환합니다.

■ 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 Array의 개수가 서로 다르면 _ERR, _LER 플래그가 발생하는데 출력단 Array 변수생략시 어레이의 개수를 0으로 보아 _ERR, _LER 플래그가 발생합니다.

■ 프로그램 예



(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.

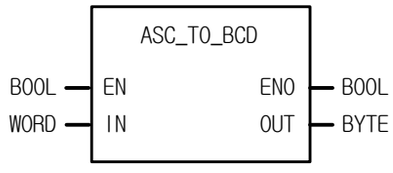
OUT_ARY[0]	AB12
OUT_ARY[1]	BC23
OUT_ARY[2]	CD34

ASC_TO_BCD

ASCII를 BCD로 변환

CPU 명	XGI	XGR
적용 가능		

발생플래그	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : ASCII 입력 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : BCD 출력

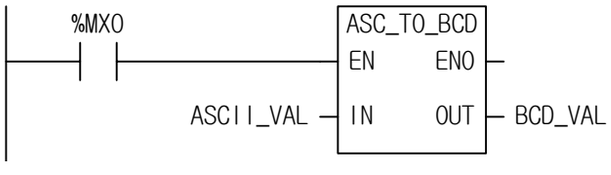
■ 기능

1. 2개의 ASCII 값을 받아서 2자리의 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) 조건(%MX0)이 On 되면, ASC_TO_BCD 펑션이 실행됩니다.

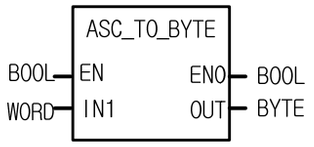
(2) 펑션 입력으로 선언된 ASCII_VAL(WORD 타입) = 16#3732 = '72' 일 경우, 펑션의 출력 변수로 선언된 BCD_VAL(BYTE 타입) = 16#72 가 출력됩니다.

ASC_TO_BYTE

ASCII를 BYTE로 변환

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : ASCII 입력 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : BYTE 출력

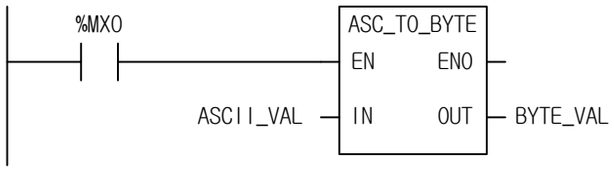
■ 기능

1. 2개의 ASCII 데이터를 입력을 받아서 2자리의 16진(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) 실행조건(%MX0)이 On되면, ASC_TO_BYTE 펑션이 실행됩니다.

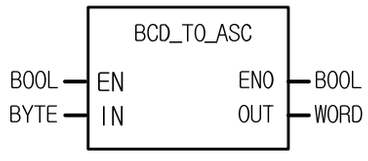
(2) 펑션의 입력변수로 선언된 ASCII_VAL(WORD 타입)=16#4339일 경우, 펑션의 출력 변수로 선언된 BYTE_VAL(BYTE 타입) = 16#C9가 출력됩니다.

BCD_TO_ASC

BCD를 ASCII로 변환

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : BCD 입력 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : ASCII 출력

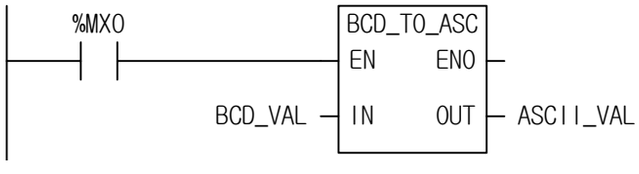
■ 기능

1. 2자리의 BCD(Binary Coded Decimal) 값을 받아서 2개의 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

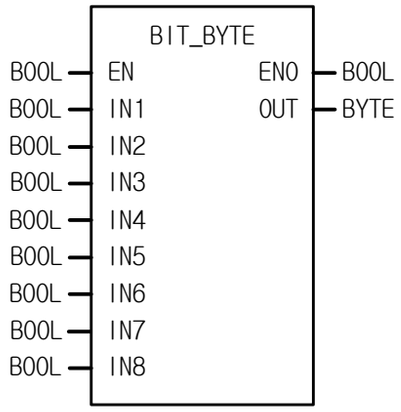


(1) 실행조건(%MX0)가 On 되면 BCD_TO_ASC 펑션이 동작합니다.
(2) 입력 BCD_VAL(BYTE 타입) = 16#85일 경우, 펑션의 출력 변수로 선언된 ASCII_VAL(WORD 타입) = 16#3835 = “85” 가 출력됩니다.

BIT_BYTE

8개 BIT들을 BYTE로 묶음

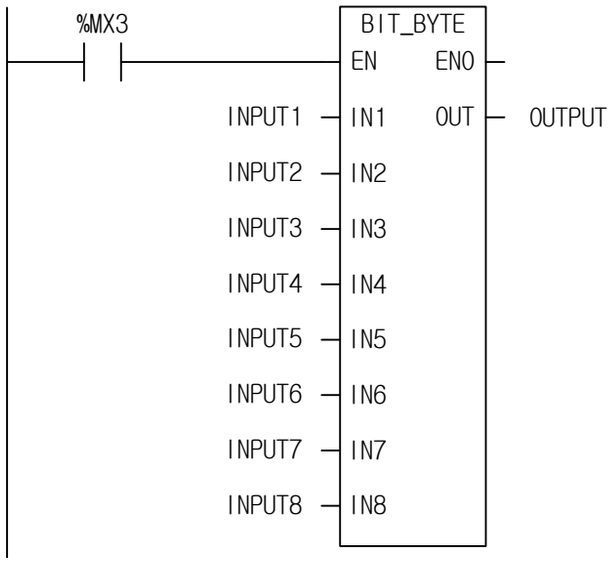
CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 ~ IN8 : Bit 입력 **출력** ENO : EN값을 그대로 출력 OUT : Byte 출력

■ 기능

1. 8개의 비트를 하나의 바이트로 조합합니다.
   IN8: 최상의 비트(MSB), IN1: 최하의 비트(LSB)

■ 프로그램 예



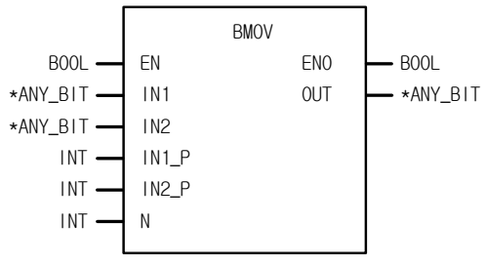
(1) 실행조건(%MX3)이 On 되면 BIT_BYTE 펑션이 실행됩니다.
(2) 8개의 입력이 IN1~ IN8까지 {0,1,1,0,1,1,0,0}이면 출력변수로 선언된 OUTPUT(BYTE타입) = 2#0110_1100 입니다.

BMOV

비트 스트링의 일부분을 복사, 이동

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

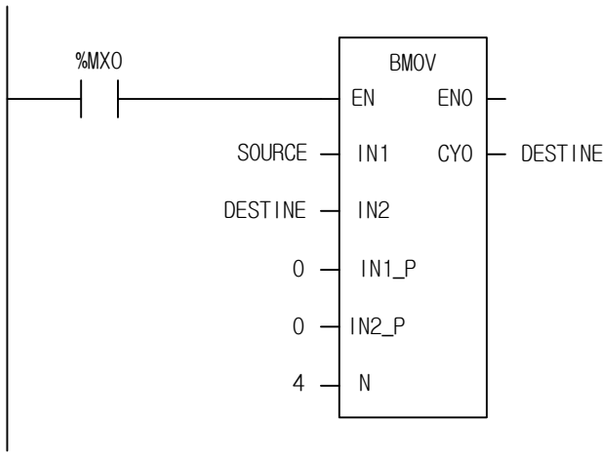
■ 기능

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

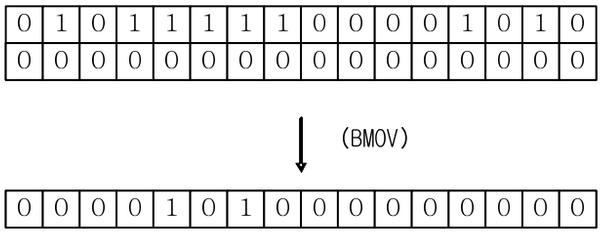


(1) 실행조건(%MX0)이 On 되면 BMOV 펑션이 실행됩니다.

(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

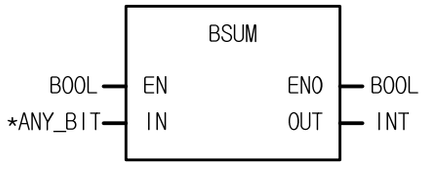
출력(OUT) : DESTINE(WORD) = 16#0A00



BSUM

On 된 비트 개수를 숫자로 출력

CPU 명	XGI	XGR
적용 가능	●	●

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		○	○	○	○															

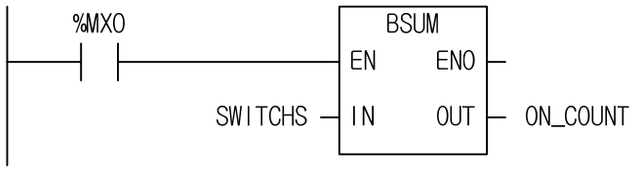
*ANY_BIT: ANY_BIT 타입 중 BOOL 제외

■ 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



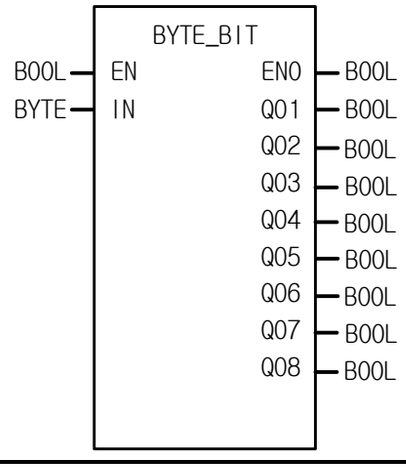
(1) 실행조건(%MX0)이 On 되면 BSUM 펑션이 실행됩니다.

(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

BYTE를 8개 BIT들로 나눔

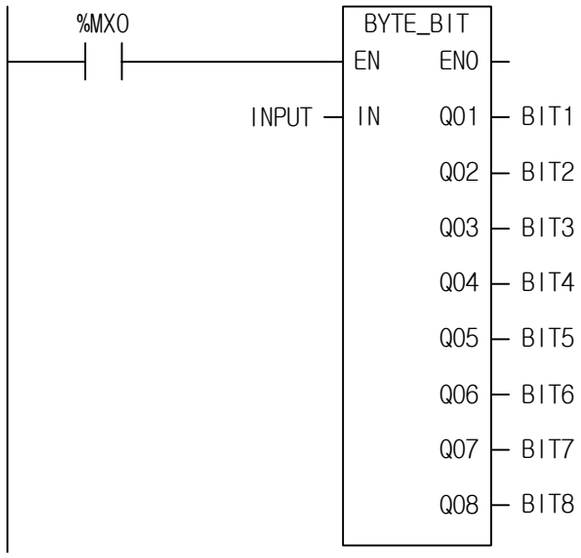
CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : Byte 입력 **출력** ENO : EN 값을 그대로 출력 Q01~8 : Bit 출력

■ 기능

1. 1개의 바이트를 출력 변수로 선언된 8개의 비트(Q01~Q02)로 분산합니다.
2. Q08: 최상위 비트(MSB), Q01: 최하위 비트(LSB)

■ 프로그램 예

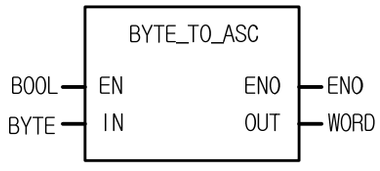


(1) 실행조건(%MX0)가 On되면 BYTE_BIT 펑션이 실행 됩니다.
(2) 입력으로 선언된 INPUT = 16#AC = 2#1010_1100 이면 출력 변수로 선언된 Q01 ~ Q08까지 Q01부터 순서대로 2#{0,0,1,1,0,1,0,1}이 저장됩니다.

BYTE_TO_ASC

BYTE를 ASCII로 변환

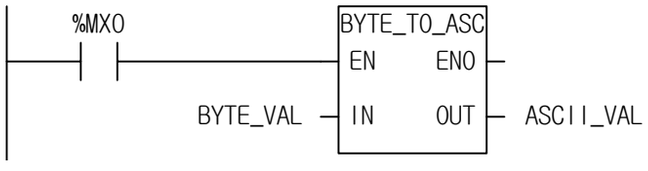
CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : BYTE 입력 **출력** ENO : EN값을 그대로 출력 OUT : ASCII 출력

■ 기능

1. 2자리의 16진(HEX) 데이터를 입력 받아서 2개의 아스키 값으로 출력합니다.
   예) 16#12 -> 3132
2. 16#A~F 는 대문자의 아스키 값으로 출력합니다.

■ 프로그램 예

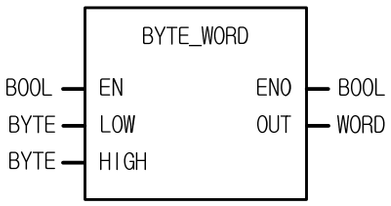


(1) 실행조건(%MX0)가 On 되면 BYTE_TO_ASC 펑션이 동작합니다.
(2) 펑션의 입력 변수로 선언된 BYTE_VAL(BYTE 타입) = 16#3A 일 경우, 펑션의 출력 변수로 선언된 ASCII_VAL(WORD 타입) = 16#3341 = '3', 'A' 가 출력됩니다.

BYTE_WORD

2개의 BYTE를 WORD로 모음

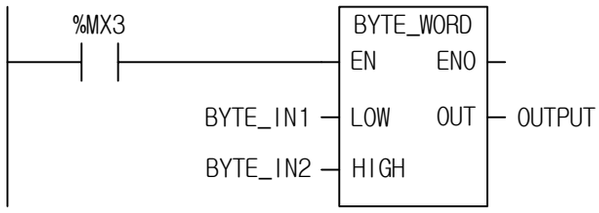
CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑 션 실행 LOW : 하위 BYTE 입력 HIGH : 상위 BYTE 입력 **출력** ENO : EN 값을 그대로 출력 OUT : WORD 출력

■ 기능

1. 2개의 바이트를 하나의 워드로 조합합니다.
   LOW: 하위 바이트 입력, HIGH: 상위 바이트 입력

■ 프로그램 예

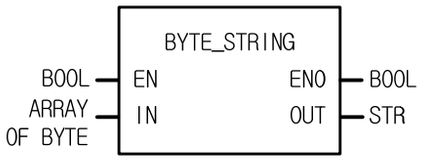


(1) 실행조건(%MX3)이 On 되면 BYTE_WORD 펑션이 실행됩니다.
(2) 입력변수로 선언된 BYTE_IN1 = 16#56 이고 BYTE_IN2 = 16#AD 이면 출력변수로 선언된 OUTPUT = 16#AD56 입니다.

BYTE_STRING

Byte Array를 문자열변환

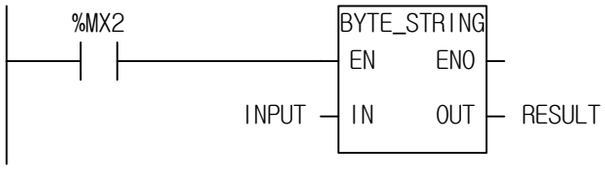
CPU 명	XGI	XGR
적용 가능	●	●

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

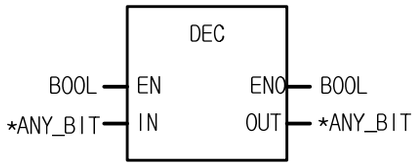


(1) 실행조건(%MX2)이 On 되면 BYTE_STRING 펑션이 실행됩니다.
(2) 입력 INPUT 어레이 변수를 3 개를 설정했을 때 INPUT[0] = 16#41, INPUT[1] = 16#31, INPUT[2] = 16#35 를 입력하면 출력 RESULT = 'A15' 입니다.

DEC

데이터를 하나 감소

CPU 명	XGI	XGR
적용 가능	●	●

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		○	○	○	○															

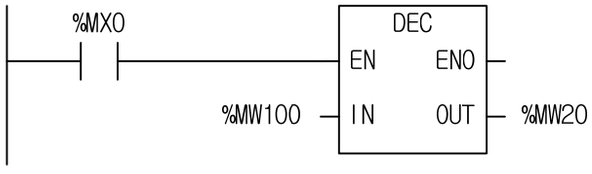
*ANY_BIT: ANY_BIT 타입 중 BOOL 제외

■ 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) 실행조건(%MX0)이 On 되면 DEC 펑션이 실행됩니다.

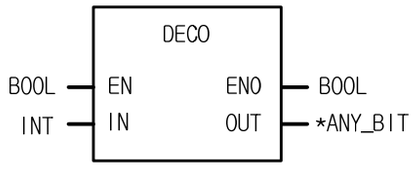
(2) 입력 변수로 선언된 %MW100 = 16#0007(2#0000_0000_0000_0111) 이라면, 연산이 실행된 후에는 %MW20 = 16#0006(2#0000_0000_0000_0110)이 됩니다.

DECO

지정된 비트 위치를 On

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

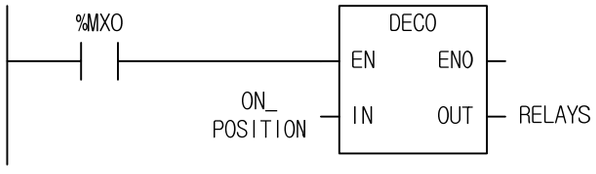
1. EN 이 1 이면, IN 의 값 즉 비트 위치지정 데이터에 따라서 출력의 비트 스트링 데이터 중 지정된 위치의 비트만 1로 하여 출력합니다.
2. 출력에는 BYTE, WORD, DWORD, LWORD 타입의 데이터가 접속 가능합니다.

FUNCTION	OUT 변수 타입	동작 설명
DECO	BYTE	출력 데이터에 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



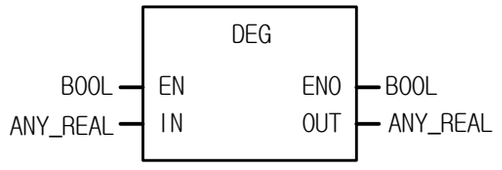
(1) 실행조건(%MX0)이 On 되면 DECO 펑션이 실행됩니다.

(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

Radian값을 각도로 변환

CPU 명	XGI	XGR
적용 가능	●	●

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

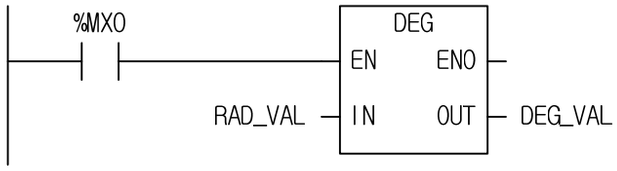
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

■ 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

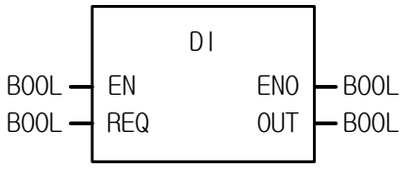


(1) 실행조건(%MX0)이 On 되면 DEG 펑션이 실행됩니다.
(2) 펑션의 입력변수로 선언된 RAD_VAL = 1.0일 경우 펑션의 출력변수 DEG_VAL = 5.72957795130785 50E+001가 됩니다.

DI

태스크 프로그램 기동 불허

CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 REQ : 태스크 프로그램 기동 불허 요구 **출력** ENO : EN 값이 그대로 출력 OUT : DI 동작이 실행되면 1 출력

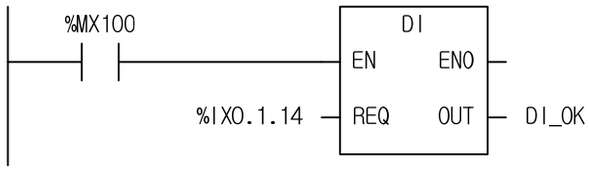
■ 기능

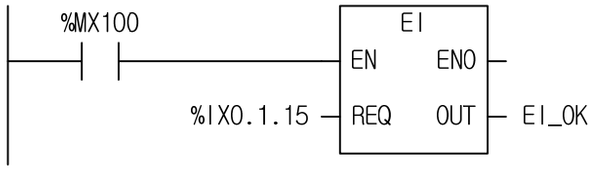
1. EN 이 1 이고 REQ 에 1 의 값이 들어오면 사용자에 의해서 작성된 태스크 프로그램(싱글, 인터벌, 인터럽트)의 기동을 막습니다.
2. 한번 'DI'명령이 수행되면 REQ 입력이 0 이 되어도 태스크 프로그램은 기동되지 않습니다.
3. 태스크 프로그램이 정상적으로 기동하도록 할 때는 'EI' 펑션을 사용하여 주십시오.
   프로그램의 수행 도중에 타 태스크 프로그램의 수행으로 연산 처리의 연속성을 잃을 경우 문제가 되는 부분에 대하여 부분적으로 태스크 프로그램의 수행을 막고자 할 때 사용할 수 있습니다.
4. 태스크 프로그램의 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

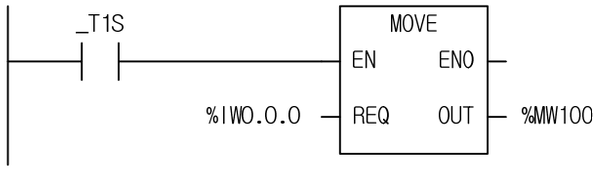
1 초마다 값을 증가하는 태스크 프로그램을 태스크 프로그램 기동 불허 펑션 DI 와 태스크 프로그램 기동허가 펑션 EI 를 이용하여 제어하는 프로그램

(1) Scan 프로그램(TASK 프로그램 제어)





(2) 1 초마다 실행하여 증가하는 태스크 프로그램

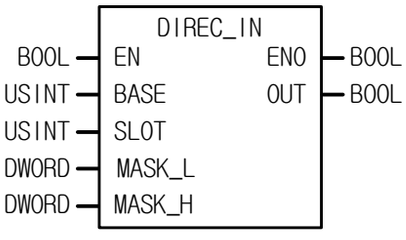


(3) DI(태스크 프로그램 기동불허 펑션)의 기동불허 요구인 REQ (직접변수 %IX0.1.14 로 지정)가 On 되면 펑션 DI 가 실행되어 출력 변수로 설정된 DI_OK 의 값이 1 이 됩니다.
(4) 펑션 DI 가 실행되면 1 초마다 실행되던 태스크 프로그램의 실행이 정지됩니다.
(5) EI (태스크 프로그램 기동허가 펑션)의 기동허가 요구인 REQ (직접변수 %IX0.1.15 로 지정)가 On 되면 펑션 EI 가 실행되어 출력 변수로 설정한 EI_OK 의 값이 1 이 됩니다.
(6) 펑션 EI 가 실행하면 펑션 DI 로 정지되었던 태스크 프로그램이 재실행 됩니다.

DIREC_IN

입력데이터 즉시갱신

CPU 명	XGI	XGR
적용 가능	●	●

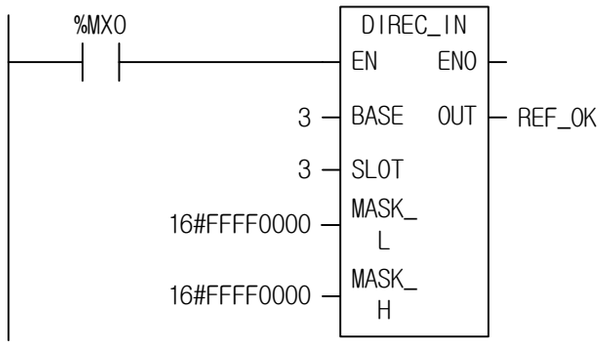
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 BASE : 입력모듈이 장착된 베이스의 위치번호 SLOT : 입력모듈이 장착된 슬롯의 위치번호 MASK_L : 입력하위 32 비트 데이터 중 갱신하지 않을 비트 지정 MASK_H : 입력상위 32 비트 데이터 중 갱신하지 않을 비트 지정 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 입력데이터 갱신이 완료되면 1출력

■ 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

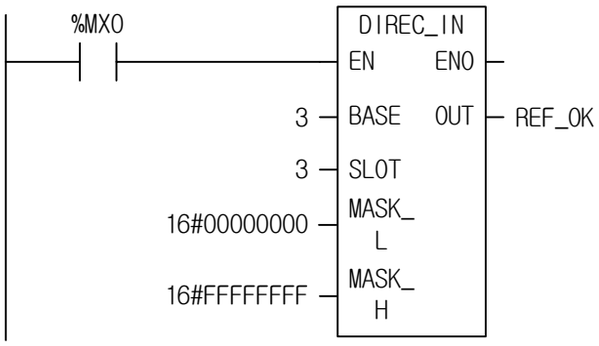
■ 프로그램 예

1. 3번째 증설 Base, 3번 Slot 에 꽂힌 모듈이 16 점 모듈이고, 입력 데이터가 2#1010_1010_1110_1011 로 즉시 갱신하는 프로그램



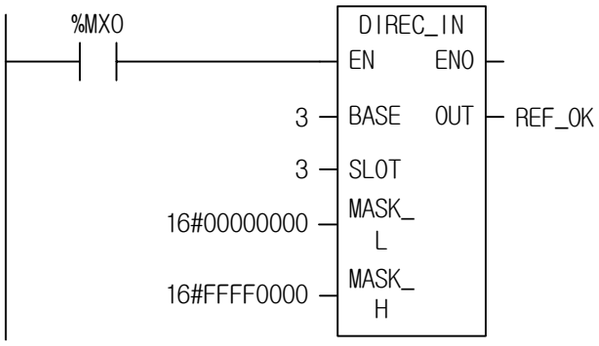
(1) 입력조건(%MX0)가 On 되면 DIREC_IN(입력데이터 즉시 갱신) 펑션이 실행합니다.
(2) 장착된 모듈이 16 점 모듈이므로 갱신대상 이미지 영역은 %IW3.3.0 이 되고 갱신하지 않을 비트의 지정변수 MASK_L(입력하위 32 비트)의 값에서 하위 16Bit 가 갱신허용으로 설정되어 있으므로 %IW3.3.0 은 스캔 도중 2#1010_1010_1110_1011 로 갱신됩니다.
(3) 비트의 지정 변수는 MASK_H(입력 상위 32 비트)의 설정 값은 현재 설정된 베이스, 슬롯에 16 점 모듈이 꽂혀 있으므로 무시됩니다.

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) 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.



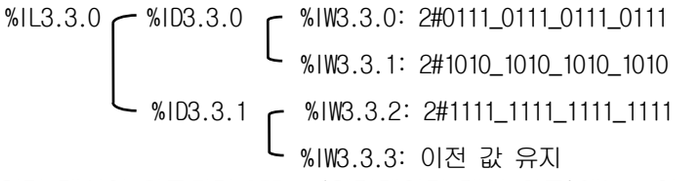
입력조건(%MX0)가 On 되면 DIREC_IN(입력데이터 즉시 갱신) 펑션이 실행합니다.

(1) 장착된 모듈이 64 점이므로 갱신 이미지 영역은 %IL3.3.0 즉 %ID3.3.0 과 %ID3.3.1 이 됩니다.

(2) 하위 32 비트(MASK_L) 모두는 갱신 허용으로 되어있으므로 %ID3.3.0 은 모두 갱신됩니다.

(3) 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.

(4) 따라서 이미지 영역의 데이터의 갱신은 아래와 같습니다.

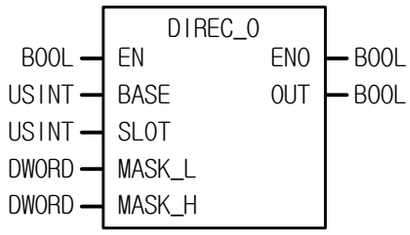


(5) 입력 갱신이 완료되면 REF_OK(입력데이터 갱신 완료)에는 1 이 출력됩니다.

DIREC_O

출력모듈 데이터 즉시갱신

CPU 명	XGI	XGR
적용 가능	●	●

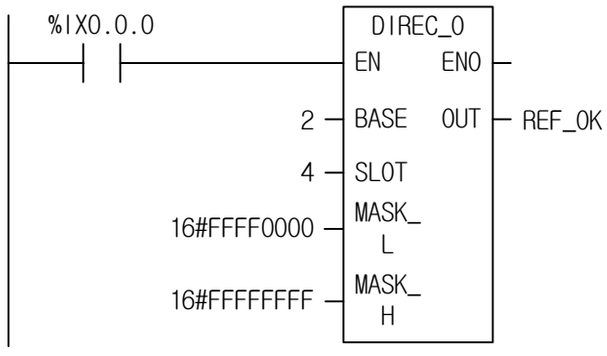
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 BASE : 출력모듈이 장착된 베이스의 위치번호 SLOT : 출력모듈이 장착된 슬롯의 위치번호 MASK_L : 출력하위 32 비트 데이터 중 갱신하지 않을 비트 지정 MASK_H : 출력상위 32 비트 데이터 중 갱신하지 않을 비트 지정 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : 출력데이터 갱신이 완료되면 1출력

■ 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 1출력)

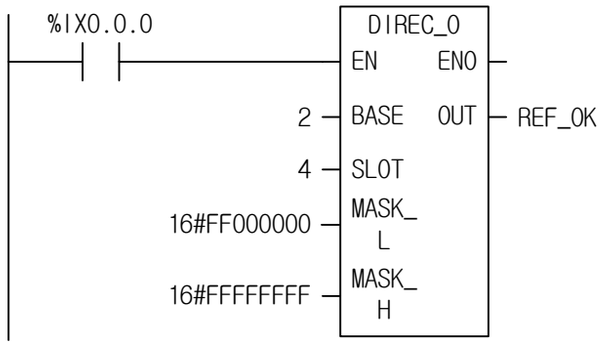
■ 프로그램 예

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) 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) 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_0(출력모듈 데이터 즉시 갱신) 펑션이 실행되어, 스캔 도중에 출력 모듈의 데이터가 2#□□□□_□□□□_1111_0000_1111_0000_1111_0000 로 출력됩니다.

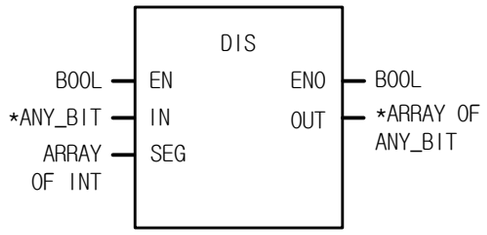
이전 값 유지

DIS

데이터 분산(Distribution)

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

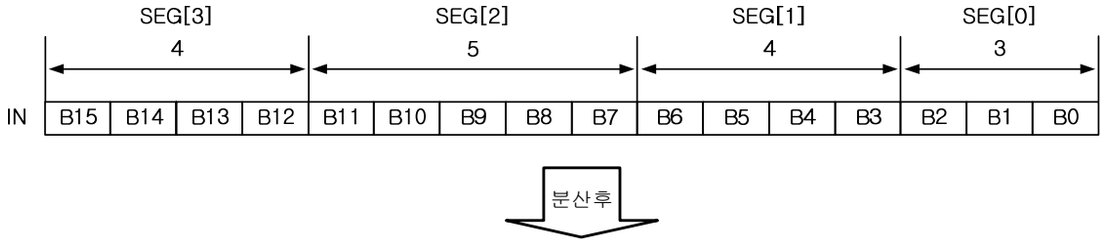
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

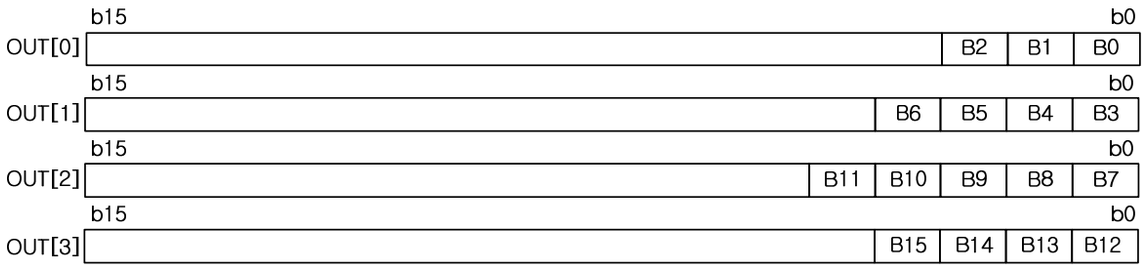
*ANY_BIT: ANY_BIT 타입 중 BOOL 제외

■ 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	



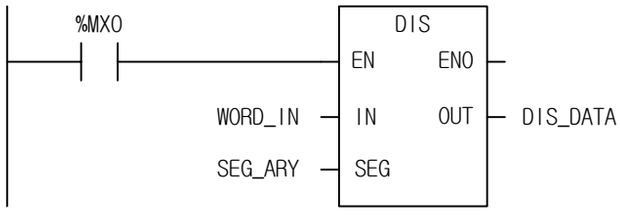


■ 플래그

플래그	설명
_ERR	SEG로 지정된 값들의 합이 입력 변수 타입의 비트 수를 초과할 경우 _ERR, _LER 에러 플래그가 셋(SET)됩니다.

☆ 출력단 Array 생략시 어레이의 개수를 0 으로 보아 _ERR, _LER 플래그가 발생합니다.

■ 프로그램 예



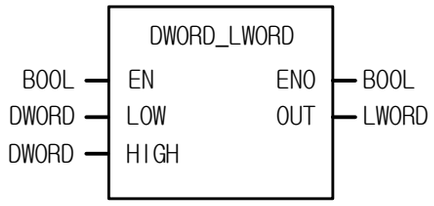
(1) 실행조건(%MX0)이 On 되면, DIS 펑션이 실행됩니다.

(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

2개의 DWORD를 LWORD로 모음

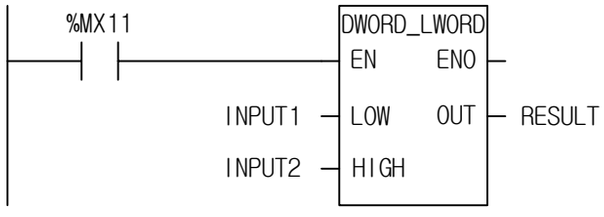
CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 LOW : 하위 DWORD 입력 HIGH : 상위 DWORD 입력 **출력** ENO : EN값을 그대로 출력 OUT : LWORD 출력

■ 기능

1. 2개의 DWORD를 하나의 LWORD로 조합합니다.
   LOW: 하위 더블워드 입력, HIGH: 상위 더블워드 입력

■ 프로그램 예

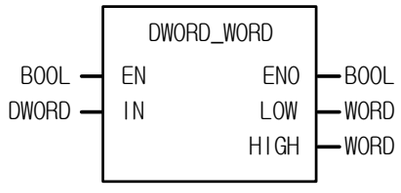


(1) 실행조건(%MX11)이 On 되면 DWORD_LWORD 펑션이 실행됩니다.
(2) 입력변수로 선언된 INPUT1=16#1A2A_3A4A_5A6A_7A8A이고, INPUT2=16#8C7C_6C5C_4C3C_2C1C 일 때 출력변수로 선언된 RESULT = 16#8C7C_6C5C_4C3C_2C1C_1A2A_3A4A_5A6A_7A8A가 출력됩니다.

DWORD_WORD

DWORD를 2개의 WORD로 나눔

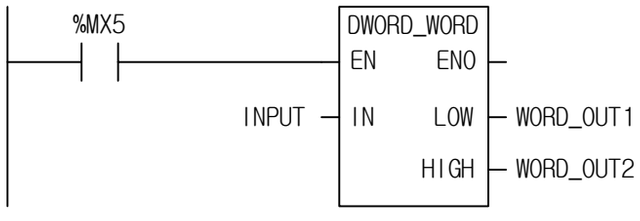
CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 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



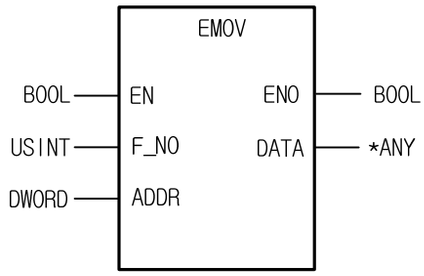
(1) 실행조건(%MX5)이 On 되면 DWORD_WORD 펑션이 실행됩니다.
(2) 입력변수로 선언된 INPUT=16#1122_3344_AABB_CCDD 일 때 입출력 변수로 선언된 WORD_OUT1 = 16#AABB_CCDD,
WORD_OUT2 = 16#1122_3344이 저장됩니다.

EMOV

설정한 플래쉬 영역으로부터 데이터 읽기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_ERR, _LER

펑 션	설 명
	**입력** REQ : 1 일 때 펑션 실행 F_NO : Move 할 데이터가 있는 블록 NO(0~31). ADDR : B_NO 로 설정된 블록의 바이트 주소 **출력** ENO : 에러 없이 수행되면 1 을 출력 DATA : 데이터 저장 영역 (BOOL 과 STRING 을 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
	IN		○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	OUT		○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ANY: ANY 타입 중 BOOL, STRING 제외

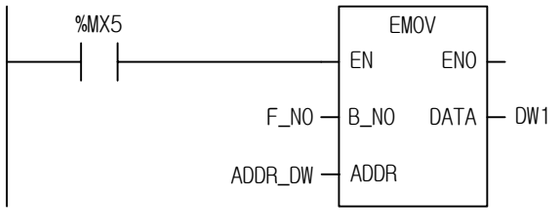
■ 기능

1. 플래시 메모리에 있는 32 개의 블록의 데이터 중 원하는 데이터 하나를 MOVE 하는 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. DATA 로 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 DATA 에 선언했다면, ADDR 변수도 4BYTE 형 변수를 사용해야 합니다.
4. F_NO 이 31 이상이거나, ADDR 값이 65,535 이상일 경우, _ERR, _LER 이 SET 됩니다.

■ 플래그

플래그	설명
_ERR	F_NO값이 31 이상이거나, ADDR 값이 65,535 이상일 경우

■ 프로그램 예

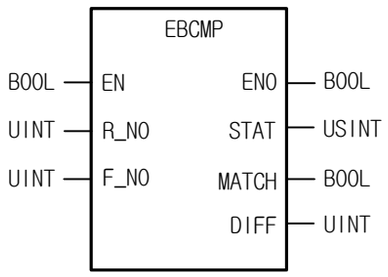


(1) 실행조건(%MX5)이 On 되면 EMOV 펑션이 실행됩니다.
(2) F_NO = 1, ADDR_DW(DWORD 타입) = 4로 설정할 때 1번 플래시 블록의 4BYTE OFFSET 위치의 DWORD DATA를 DW1 (DWORD)에 이동합니다.

EBCMP

내용 비교 후 일치 여부 확인

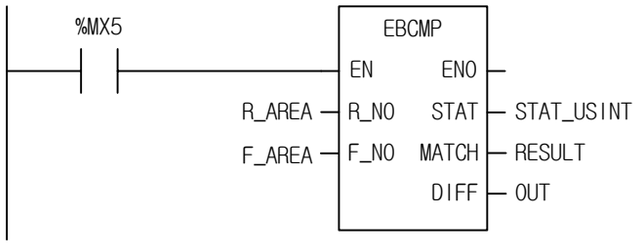
CPU 명	XGI	XGR
적용 가능	●	●

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

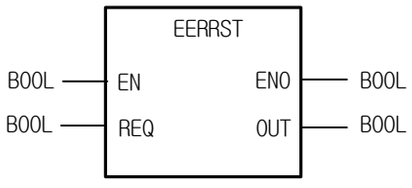


(1) 실행조건(%MX5)이 On되면 EBCMP 펑션이 실행됩니다.
(2) R_AREA = 0, F_AREA = 1로 설정할 때 R 디바이스 블록번호 0번 내용과 플래시 블록번호 1번 내용이 동일하면 RESULT(BOOL)는 On되며 OUT(불일치 개수) = 0을 나타냅니다.

EERRST

플래시 메모리관련 에러플래그 클리어

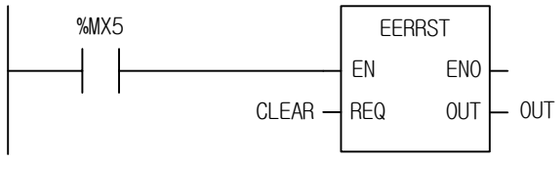
CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : On 일 때 펑션 실행 REQ : On 일 때 클리어 동작 수행 **출력** ENO : EN 값을 그대로 출력 OUT : 에러플래그 클리어 완료시

■ 기능

1. 입력 접점이 On 되고 REQ 가 On 되면, _RBLOCK_ER_FLAG 를 0 으로 클리어 시킵니다.
2. _RBLOCK_ER_FLAG 는 동시에 플래시 메모리 블록을 읽기(EBREAD) 또는 쓰기(EBWRITE) 시 발생됩니다.

■ 프로그램 예



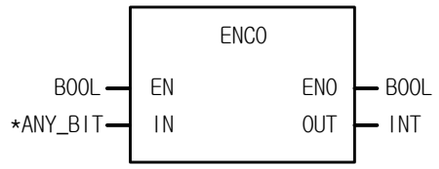
(1) 실행조건(%MX5)이 On되면 EERRST 펑션이 실행됩니다.
(2) 플래시 메모리 관련 에러 플래그(_RBLOCK_ER_FLAG)가 SET되어 있다면 0으로 클리어 됩니다.

ENCO

On된 비트 위치를 숫자로 출력

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : Encoding할 입력 데이터 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : Encoding한 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

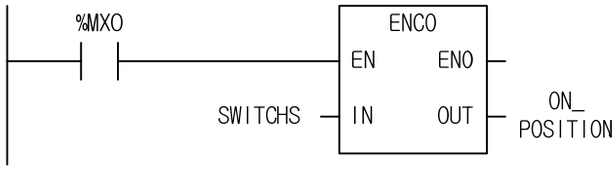
1. EN이 1이면, IN의 비트 스트링 데이터 중, 1로 되어있는 비트 중 최상위 비트의 위치를 OUT으로 출력합니다.
2. 입력에는 B(BYTE), W(WORD), D(DWORD), L(LWORD) 타입의 데이터가 접속 가능합니다.

FUNCTION	IN변수 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 1이 되어있지 않은 경우는 OUT은 -1이 되고, _ERR, _LER 플래그가 셋(Set)됩니다.

■ 프로그램 예



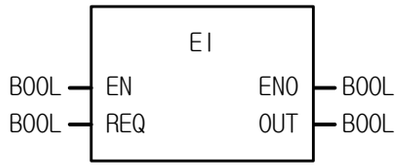
(1) 실행조건(%MX0)이 On 되면 ENCO 펑션이 실행됩니다.

(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

태스크 프로그램 기동허가 (DI의 해제)

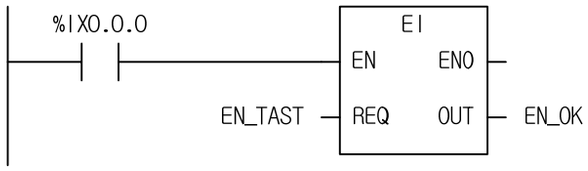
CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 REQ : 태스크 프로그램 기동 허가 요구 **출력** ENO : EN 값이 그대로 출력 OUT : EI 동작이 실행되면 1출력

■ 기능

1. EN이 1이고 REQ에 1의 값이 들어오면 'DI'펑션에 의해 막혀있던 태스크 프로그램이 정상적으로 기동합니다.
2. 한번 'EI'펑션이 수행되면 REQ입력이 0이 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) EN_TASK가 1이 되면 태스크 프로그램이 정상 기동 됩니다.
(2) 'EI'펑션에 의해서 태스크 실행 허가 상태가 되면 EN_OK에는 1이 출력됩니다.

ESTOP

프로그램에 의한 비상 운전정지

CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 REQ : 프로그램에 의한 비상 운전정지 요구 **출력** ENO : EN값이 그대로 출력. 기능 1번 참조 OUT : ESTOP 동작이 실행되면 1출력

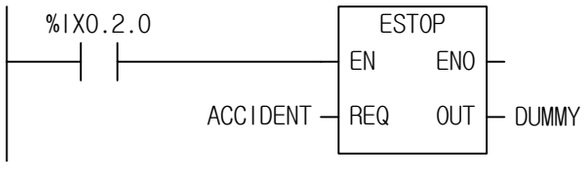
■ 기능

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) 실행조건(%IX0.2.0)이 On 되면 프로그램에 의한 비상 운전정지 펑션 'ESTOP' 이 실행됩니다.

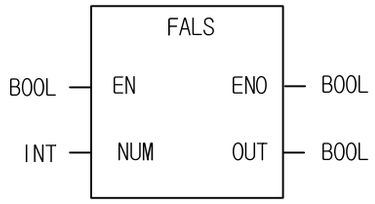
(2) 펑션 'ESTOP' 의 ACCIDENT 가 1 이 되면 실행중인 프로그램을 즉시 중지하고 STOP 모드로 됩니다.

※ 비상 사태 발생시 기계적 인터럽트와 함께 이중 안전장치로 사용할 수 있습니다.

FALS

사용자가 정한 상수(N)를 F영역의 지정된 주소에 저장(_FALS_NUM)

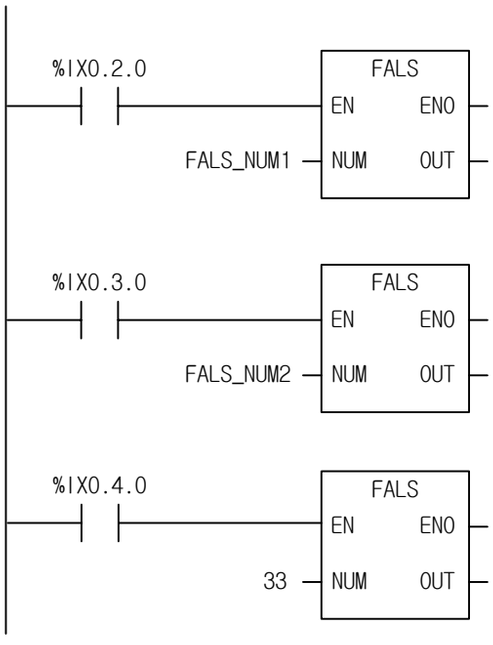
CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 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) 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.

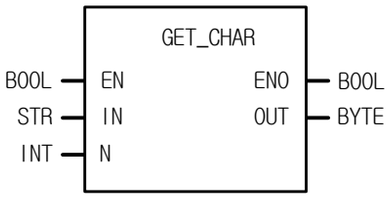
(4) FALS 펑션은 특수 상황에 따른 해당되는 값을 설정 함으로서 프로그램 실행 후 _FALS_NUM 플래그를 확인하여 해당되는 부분의 상태를 보기에 편리합니다.

GET_CHAR

문자열로부터 한 문자(CHAR)추출

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : STRING 입력 N : STRING내의 위치 지정 **출력** ENO : EN값을 그대로 출력 OUT : BYTE 출력

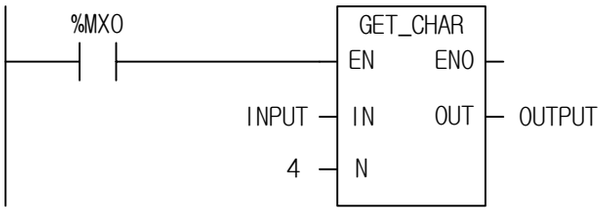
■ 기능

1. STRING의 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



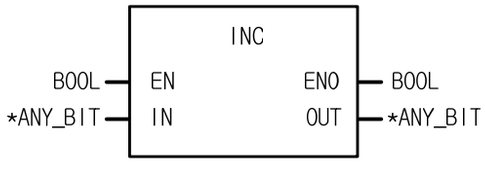
(1) 실행조건(%MX0)이 On되면, GET_CHAR 펑션이 실행됩니다.

(2) 입력 변수로 선언된 INPUT(STRING 타입) = “LS XGI PLC” 일 때 이 String의 4번째 문자를 추출하게 되면 출력 변수로 선언된 OUTPUT으로 16#58(“X”)가 출력됩니다.

INC

데이터를 하나 증가

CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 증가시킬 입력 데이터 **출력** ENO : EN값이 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		○	○	○	○															

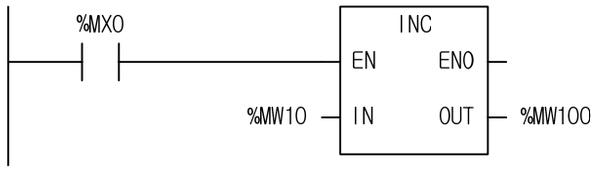
*ANY_BIT: ANY_BIT 타입 중 BOOL 제외

■ 기능

1. EN이 1이면, IN의 비트스트링 데이터를 1만큼 증가시켜서 OUT으로 출력합니다.
2. 오버플로우가 발생해도 에러는 발생하지 않으며, 결과는 16#FFFF인 경우에 16#0000이 됩니다.
3. 입력에는 BYTE, WORD, DWORD, LWORD 타입의 데이터가 접속 가능합니다.

FUNCTION	IN/OUT 변수 타입	동작 설명
INC	BYTE	입출력 데이터에 맞추어 4가지 펑션 중 하나를 사용합니다.
INC	WORD	
INC	DWORD	
INC	LWORD	

■ 프로그램 예



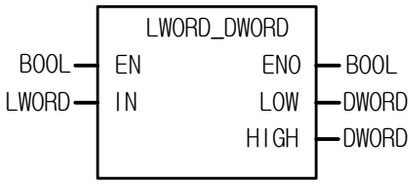
(1) 실행조건(%MX0)이 On 되면 INC 펑션이 실행됩니다.

(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

LWORD를 2개의 DWORD로 나눔

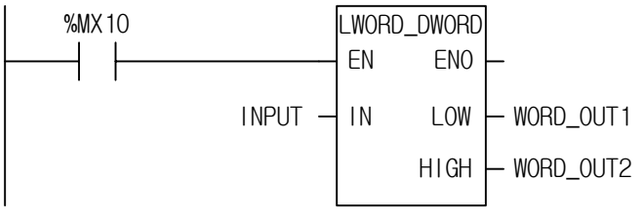
CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 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



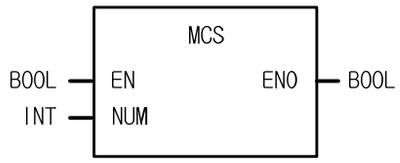
(1) 실행조건(%MX10)이 On 되면, LWORD_DWORD 펑션이 실행됩니다.

(2) 입력변수로 선언된 INPUT=16#AAAA_BBBB_CCCC_DDDD 일 때, 출력 변수로 선언된 WORD_OUT1=16#CCCC_DDDD, WORD_OUT2=16#AAAA_BBBB 가 출력됩니다.

MCS

Master Control

CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 NUM : Nesting (0~15) **출력** ENO : MCS 명령이 실행되면 1을 출력

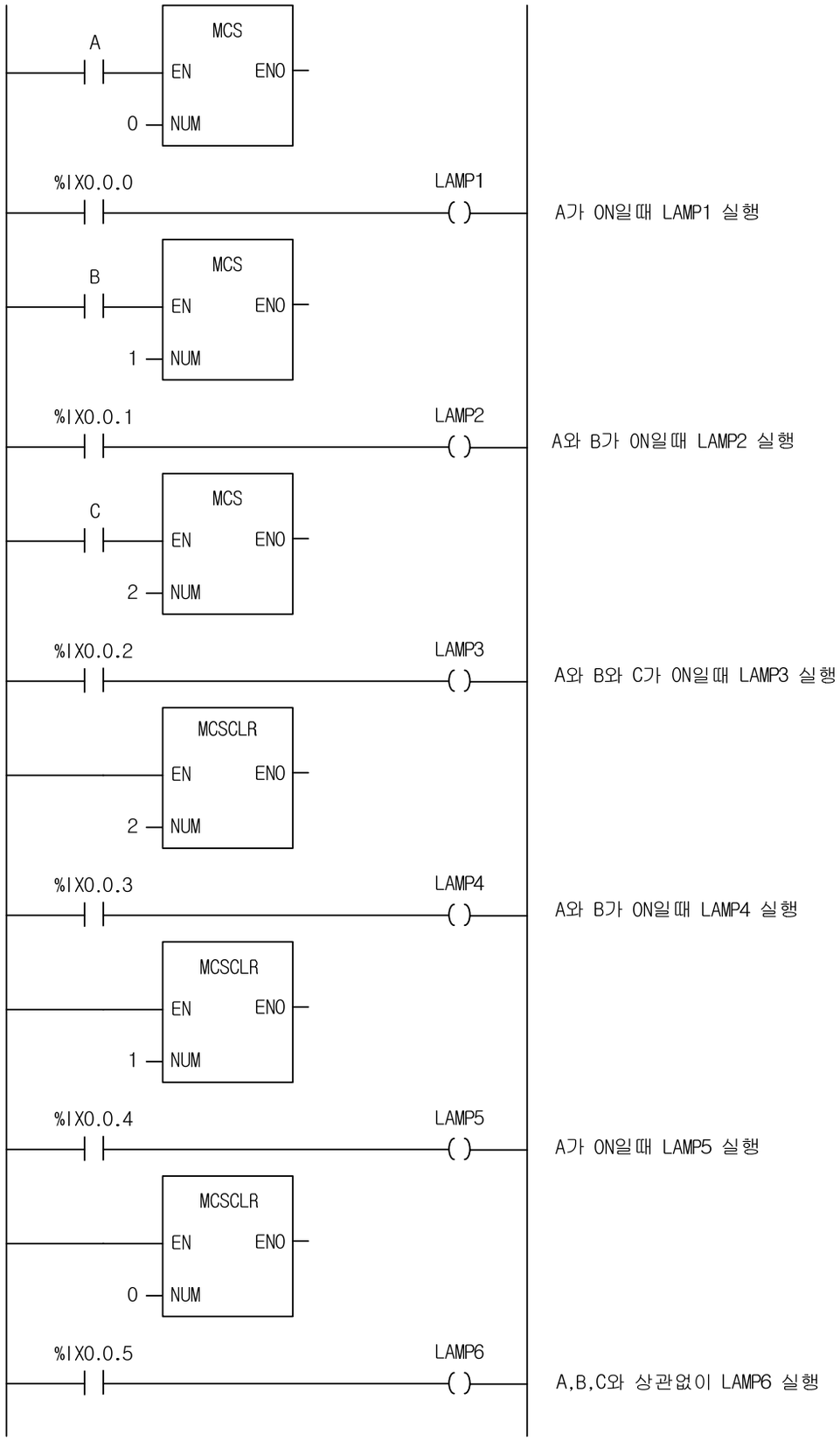
■ 기능

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

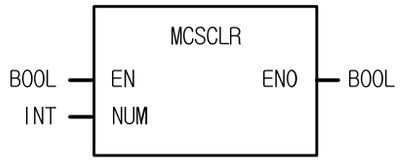

A
MCS
EN
ENO
0
NUM
%IX0.0.0
LAMP1
A가 ON일때 LAMP1 실행
B
MCS
EN
ENO
1
NUM
%IX0.0.1
LAMP2
A와 B가 ON일때 LAMP2 실행
C
MCS
EN
ENO
2
NUM
%IX0.0.2
LAMP3
A와 B와 C가 ON일때 LAMP3 실행
MCSCLR
EN
ENO
2
NUM
%IX0.0.3
LAMP4
A와 B가 ON일때 LAMP4 실행
MCSCLR
EN
ENO
1
NUM
%IX0.0.4
LAMP5
A가 ON일때 LAMP5 실행
MCSCLR
EN
ENO
0
NUM
%IX0.0.5
LAMP6
A,B,C와 상관없이 LAMP6 실행

(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

Master Control 해제 명령

CPU 명	XGI	XGR
적용 가능	●	●

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.

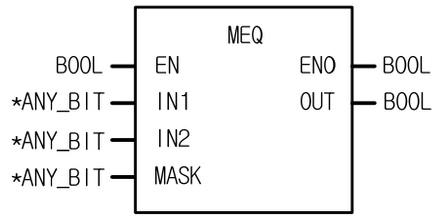
■ 프로그램 예

MCS 펑션의 프로그램 예를 참조 바랍니다.

MEQ

Masked Equal

CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN1 : 입력 1 IN2 : 입력 2 MASK : masking할 입력값 **출력** ENO : EN값을 그대로 출력 OUT : 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		○	○	○	○															
	MASK		○	○	○	○															

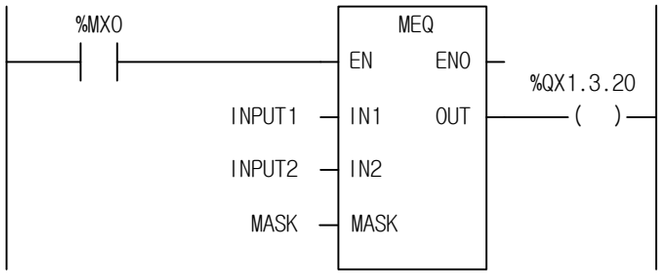
*ANY_BIT: ANY_BIT 타입 중 BOOL 제외

■ 기능

1. 입력된 변수값들을 Masking한 후 두개의 변수값이 서로 동일한지를 비교합니다. 만약 8비트 변수값을 2#1111_1100 으로 Masking하면 하위 2비트는 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) 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

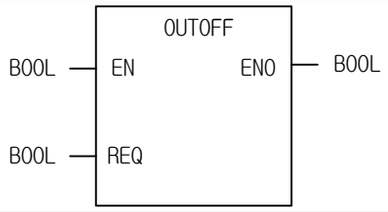
INPUT2(BYTE 타입) = 2#0101_0100

과 같이 되어 서로 동일한 값을 가지므로 출력접점 %QX1.3.20 이 On 됩니다.

OUTOFF

입력 조건이 성립하면 전 출력을 Off

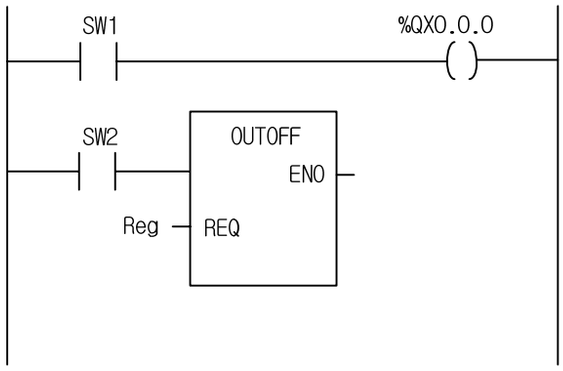
CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 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) 출력 모듈을 장착한 후 위의 예와 같이 프로그램을 구성합니다.

(2) SW1 을 On 시키면 출력(%QX0.0.0)이 SET 됩니다.

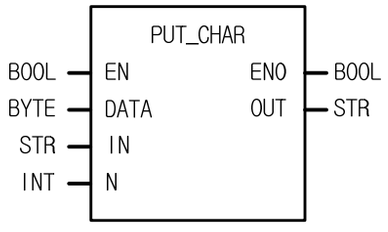
(3) SW2 를 On 시킨 후 Reg = 1 로 동작 시키면 OUTOFF 펑션이 실행이 되면서 모든 출력 모듈이 Off 됩니다.
프로그램 모니터 상으로는 셋(SET)되어있는 것 같이 보이나 실제 출력 모듈은 Off 되어 있습니다.

PUT_CHAR

문자열에 한 문자 써넣기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_ERR, _LER

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 DATA : STRING에 삽입할 BYTE 입력 IN : STRING 입력 N : STRING내의 위치 지정 **출력** ENO : EN값을 그대로 출력 OUT : STRING 출력

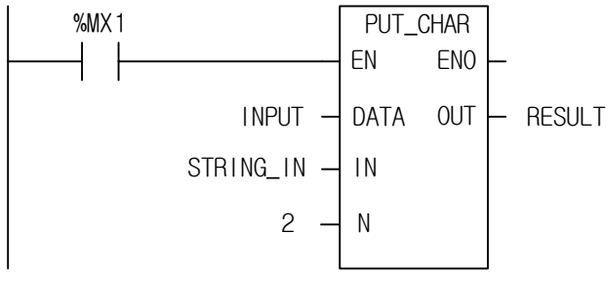
■ 기능

1. 하나의 바이트 입력값을 STRING상의 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



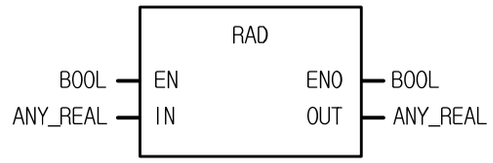
(1) 실행조건(%MX1)이 On되면 PUT_CHAR 펑션이 실행됩니다.

(2) 입력변수인 INPUT = 16#41(“A”)이고 STRING_IN = “TOKEN” 일 때 입력 변수 STRING_IN 의 2 번째 위치에 입력변수 INPUT이 지닌 값을 덮어쓰기 하게 되면 출력 변수인 RESULT는 “TAKEN”이 됩니다.

RAD

각도(DEG)를 Radian 값으로 변환

CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 각도 입력 **출력** ENO : EN 값을 그대로 출력 OUT : 라디안 값 출력

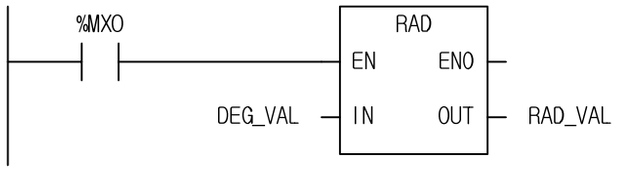
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN														○	○					
	OUT														○	○					

■ 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



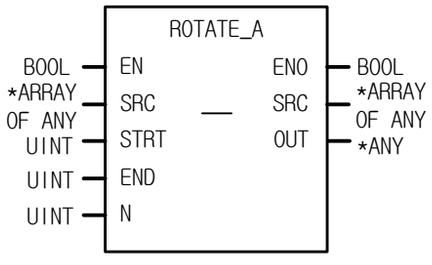
(1) 실행조건(%MX0)이 On 하면, RAD_REAL 펑션이 실행됩니다.
(2) 펑션의 입력 변수로 선언된 DEG_VAL = 127(°)일 경우, 펑션의 출력변수 값은 RAD_VAL = 2.21656823 이 됩니다.

ROTATE_A

지정된 어레이 원소의 ROTATE

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_ERR, _LER

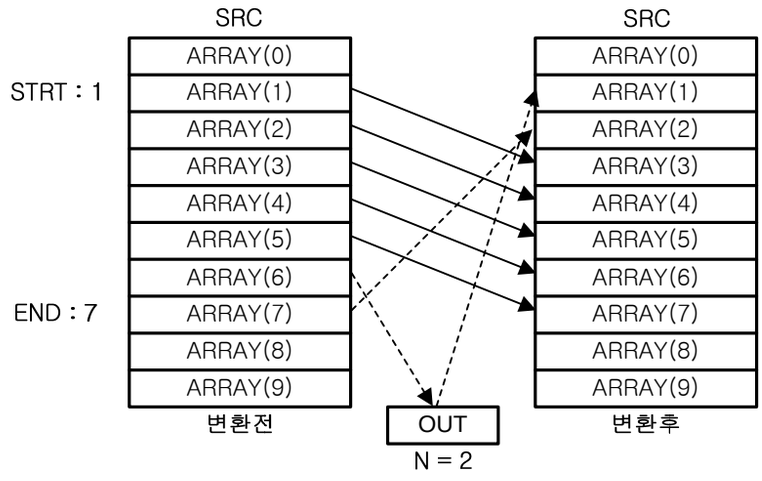
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 N : Rotate 시킬 개수 STRT : 어레이 블록 중 Rotate할 원소의 시작위치 END : 어레이 블록 중 Rotate 할 원소의 종료위치 **출력** ENO : 에러 없이 실행되면 1을 출력 OUT : Rotate하여 밀려나온 데이터를 출력 **입출력** SRC : Rotate 조작이 가해질 어레이 블록

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.



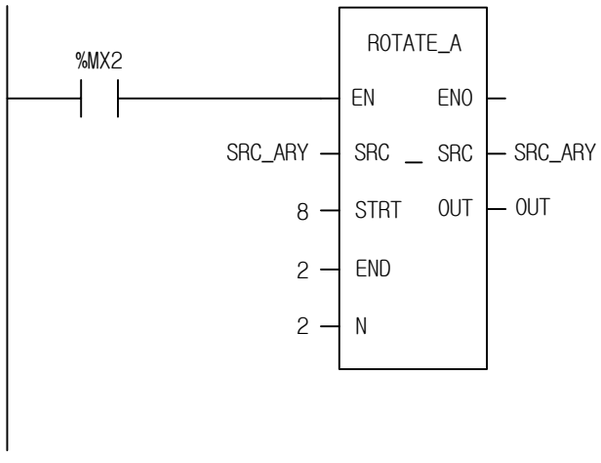
펑션	입출력 어레이 타입	동작 설명
ROTATE_A	BOOL	각 타입 어레이의 지정된 범위의 원소들을 지정된 방향으로 ROTATE 시킵니다.
ROTATE_A	BYTE	
ROTATE_A	WORD	
ROTATE_A	DWORD	
ROTATE_A	LWORD	
ROTATE_A	SINT	
ROTATE_A	INT	
ROTATE_A	DINT	
ROTATE_A	LINT	
ROTATE_A	USINT	
ROTATE_A	UINT	
ROTATE_A	UDINT	
ROTATE_A	ULINT	
ROTATE_A	REAL	
ROTATE_A	LREAL	
ROTATE_A	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.

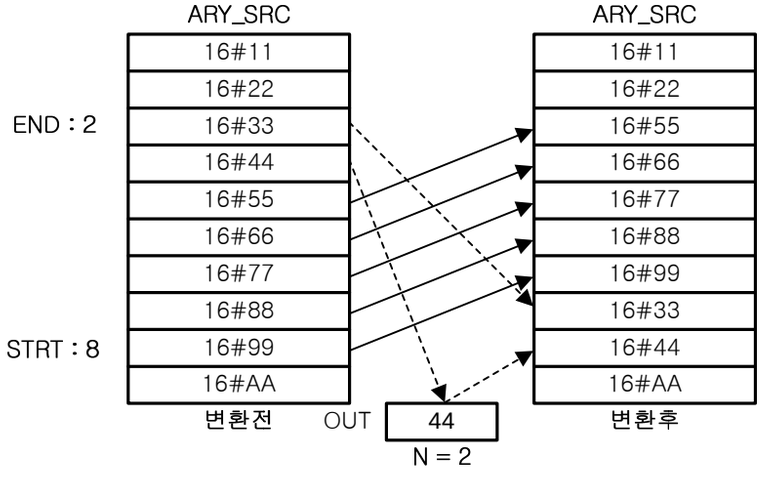
■ 프로그램 예



(1) 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.

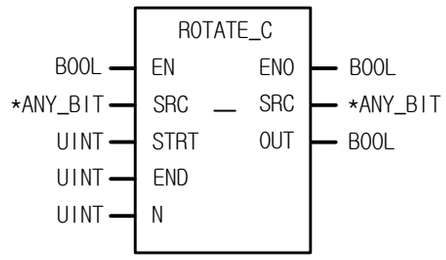


ROTATE_C

Rotate with Carry

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_ERR, _LER

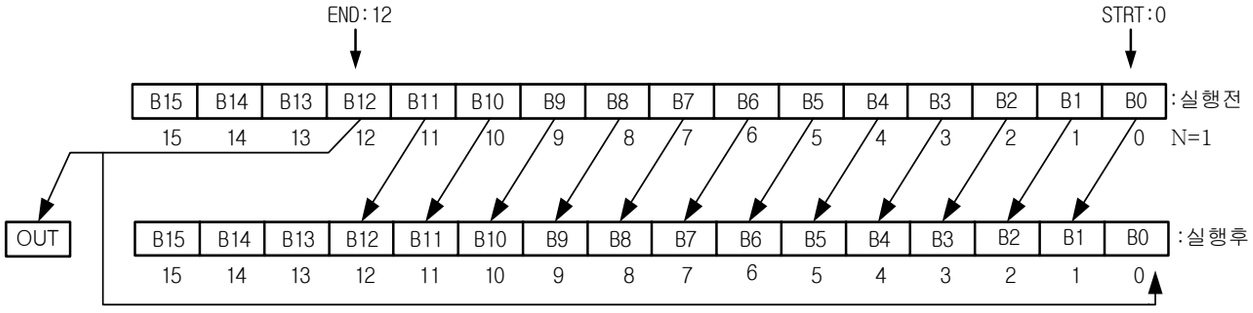
펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 STRT: SRC의 비트열 중 회전할 범위의 시작bit 위치 END : SRC의 비트열 중 회전할 범위의 끝 bit 위치 N : Shift 할 비트수 **출력** ENO : 에러 없이 수행되면 1을 출력 OUT : Carry 출력 **입출력** SRC : 회전할 변수

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.

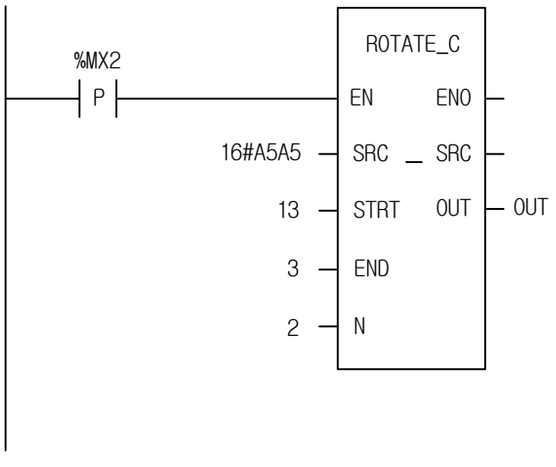


펑션	SRC 변수 타입	동작 설명
ROTATE_C	BYTE	입력값의 지정된 범위에 해당하는 비트들을 지정된 횟수만큼 Rotate 시킵니다.
ROTATE_C	WORD	
ROTATE_C	DWORD	
ROTATE_C	LWORD	

■ 플래그

플래그	설명
_ERR	STRT와 END값이 SRC 변수 타입의 비트 개수를 넘는 경우 SRC 값에는 변화가 없으며 _ERR, _LER 플래그가 셋(SET)됩니다.

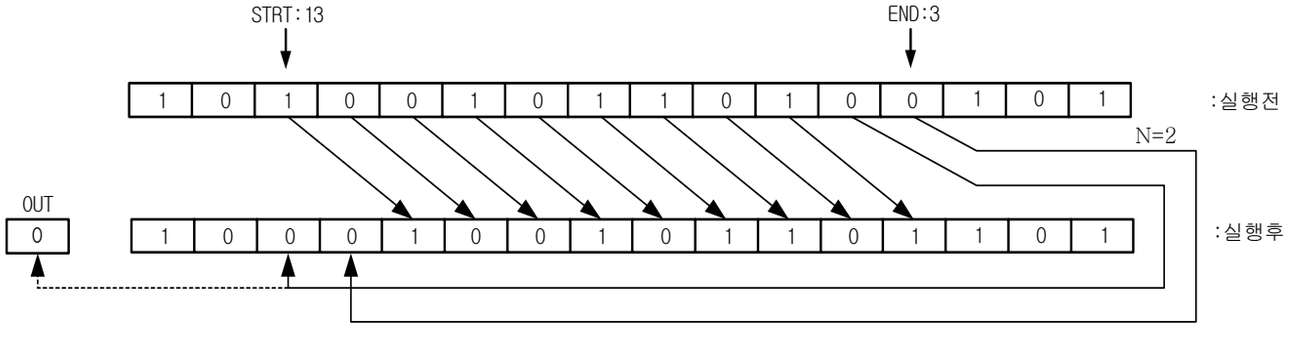
■ 프로그램 예



(1) 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.

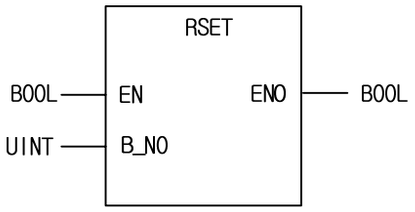
(3) Rotate 된 후의 값이 SRC(16#896D)로 다시 출력되고 회전되면서 END 위치에서 STRT 로 되돌아가는 비트인 0 이 OUT 으로 출력됩니다.



RSET

설정된 블록 번호를 지정된 블록 번호로 전환

CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 B_NO : 전환할 블록 NO(0~1). **출력** ENO : 에러 없이 수행되면 1을 출력

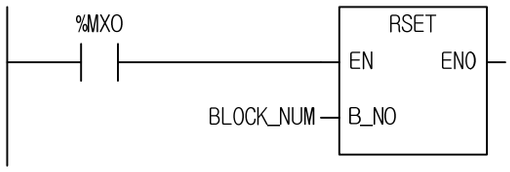
■ 기능

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



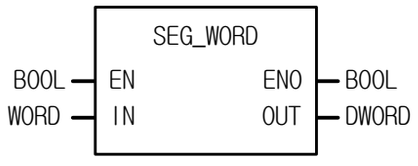
(1) 실행조건(%MX0)이 On 되면 RSET 펑션이 실행됩니다.

(2) BLOCK_NUM(UINT 타입)은 0 과 1 의 둘 중 하나를 선택할 수 있으며 지정된 R 블록으로 전환시킵니다.

SEG_WORD

BCD 또는 HEX값을 7세그먼트 디스플레이 코드로 변환

CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : 7세그먼트 코드로 변환할 입력 데이터 **출력** ENO : EN값이 그대로 출력 OUT : 7세그먼트 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

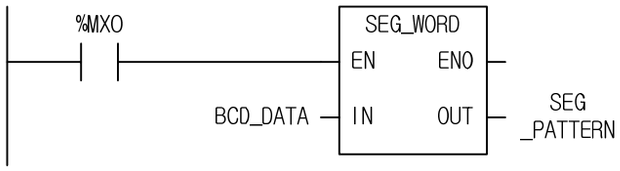
   5) 7 세그먼트의 자릿수가 4 자리를 넘을 때

      가) BCD, HEX 는 4 자리씩 나누어 'SEG' 펑션을 사용

      나) 정수 -> 8 자리 BCD 형 7 세그먼트 코드:

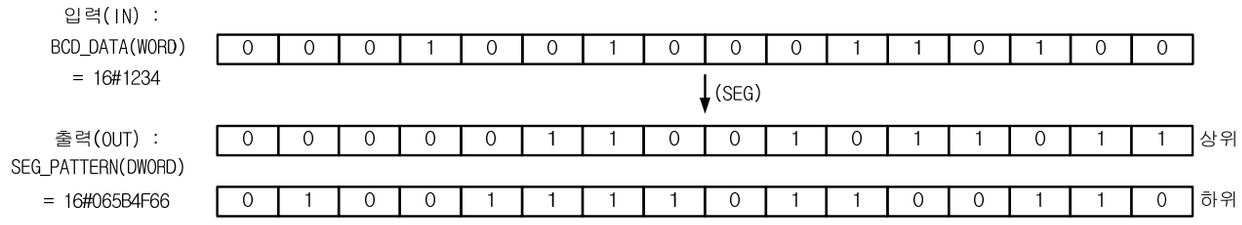
      정수를 10,000 으로 나누어 몫과 나머지를 각각 'INT_TO_BCD' 펑션을 거쳐서 'SEG' 로 변환하여 상위 4 자리와 하위 4 자리의 7 세그먼트 코드를 생성

■ 프로그램 예

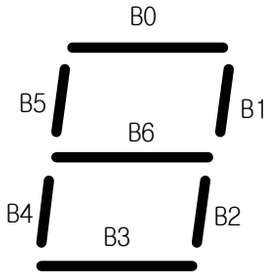


(1) 실행조건(%MX0)이 On 되면 SEG_WORD 펑션이 실행됩니다.

(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

입력 (BCD)	입력 (16 진수)	정수값	출력 B7 B6 B5 B4 B3 B2 B1 B0	표시 데이터
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

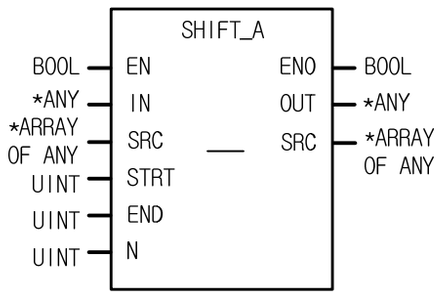
SHIFT_A

지정된 어레이 원소의 SHIFT

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

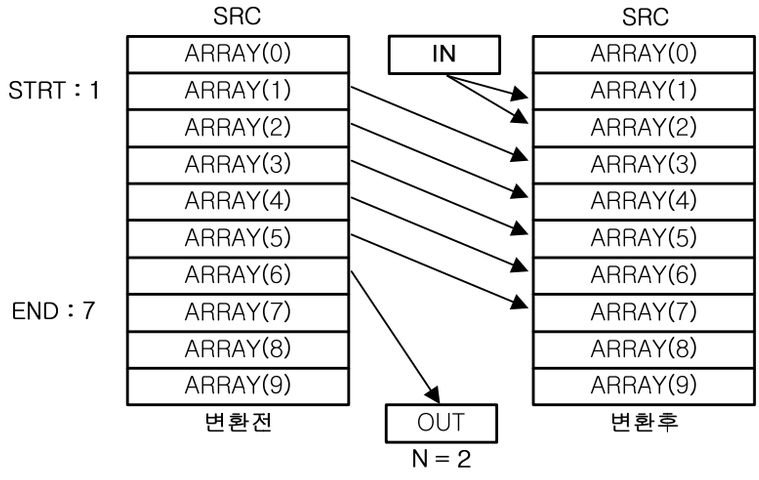
입출력 SRC : Shift 조작이 가해질 어레이 블록

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.



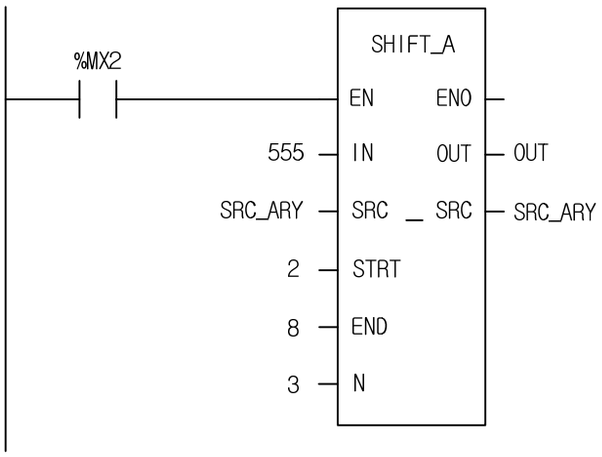
펑션	입출력 어레이 타입	동작 설명
SHIFT_A	BOOL	각 타입 어레이의 지정된 범위의 원소들을 지정된 방향으로 이동 시킵니다.
SHIFT_A	BYTE	
SHIFT_A	WORD	
SHIFT_A	DWORD	
SHIFT_A	LWORD	
SHIFT_A	SINT	
SHIFT_A	INT	
SHIFT_A	DINT	
SHIFT_A	LINT	
SHIFT_A	USINT	
SHIFT_A	UINT	
SHIFT_A	UDINT	
SHIFT_A	ULINT	
SHIFT_A	REAL	
SHIFT_A	LREAL	
SHIFT_A	TIME	
SHIFT_A	DATE	
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



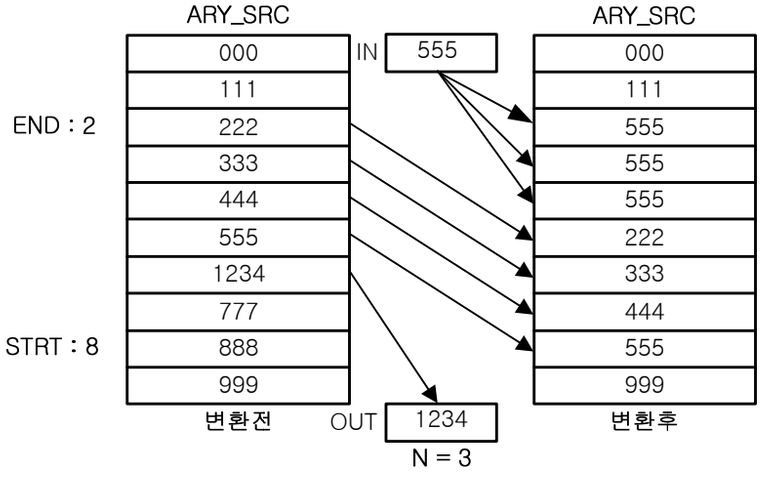
(1) 입력 조건(%MX2)이 On 되면, SHIFT_A 펑션이 실행됩니다.

(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.

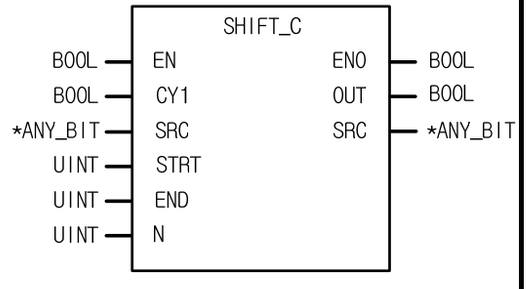


SHIFT_C

Shift with Carry

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

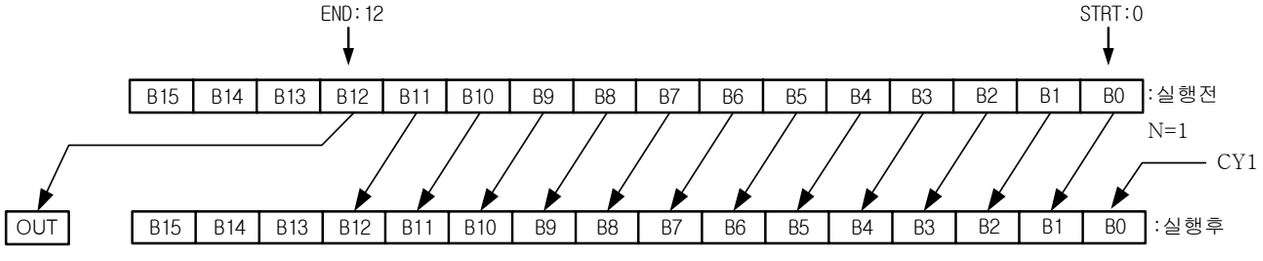
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	OUT		○	○	○	○															

*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.

.

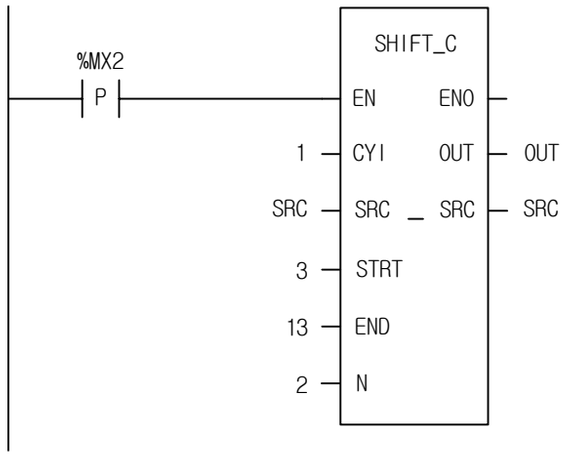


펑션	SRC 변수 타입	동작 설명
SHIFT_C	BYTE	입력값의 지정된 범위에 해당하는 비트들을 지정된 횟수만큼 Shift 시킵니다.
SHIFT_C	WORD	
SHIFT_C	DWORD	
SHIFT_C	LWORD	

■ 플래그

플래그	설명
_ERR	STRT와 END값이 SRC변수 타입의 비트 개수를 넘는 경우 SRS의 변화는 없으며 _ERR, _LER 플래그가 셋(SET)됩니다.

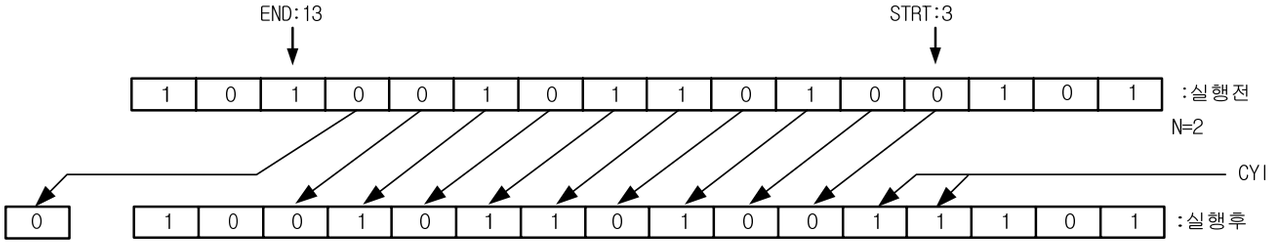
■ 프로그램 예



(1) 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.

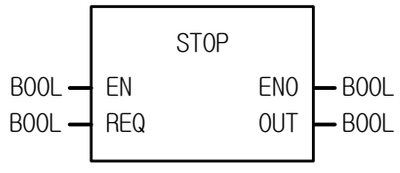
(3) Shift 된 후의 값은 다시 SRC(16#969D)로 출력되고 2 비트 shift 되면서 밀려나온 값 0 이 OUT 으로 출력됩니다.



STOP

프로그램에 의한 운전정지

CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 RE : 프로그램에 의한 운전정지 요구 **출력** ENO : EN값이 그대로 출력 OUT : STOP 동작이 실행되면 1출력

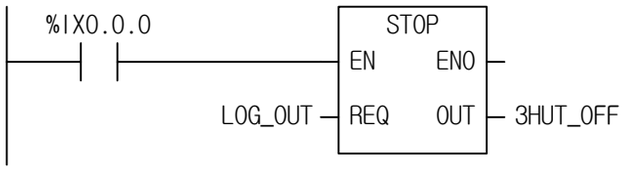
■ 기능

1. EN이 1이고 REQ에 1의 값이 들어오면 운전을 정지하고 STOP 모드로 됩니다.
2. 'STOP'펑션이 수행되면 수행 중인 스캔 프로그램의 수행을 완료한 후 정지합니다.
3. 전원을 재 투입하거나 운전모드를 STOP으로 하였다가 RUN으로 하면 재 기동이 됩니다.

■ 관련 플래그

플래그	설명
_USTOP_ON	STOP 명령어에 의해 STOP된 경우 On 됩니다. 이 상태에서 다시 런 진입시 Off 됩니다.

■ 프로그램 예



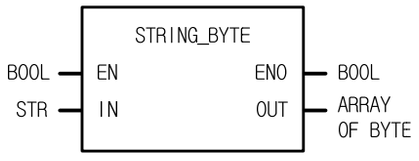
(1) 실행조건(%IX0.0.0)이 On하고 LOG_OUT가 1이 되면 실행중인 스캔 프로그램을 완료한 후 STOP 모드로 됩니다.

(2) 입력변수로 선언한 'STOP'펑션 수행 후 안정된 상태에서 PLC의 전원을 끄는 것이 좋습니다.

STRING_BYTE

문자열을 Byte Array로 변환

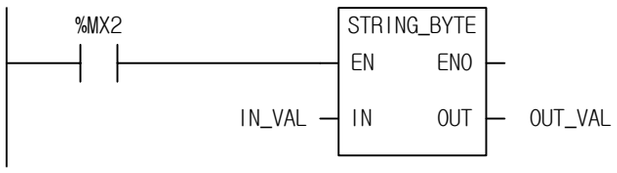
CPU 명	XGI	XGR
적용 가능	●	●

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

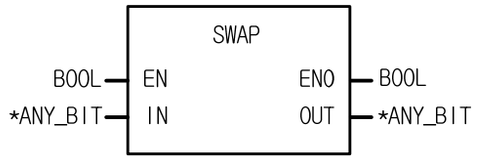


(1) 실행조건(%MX2)이 On 되면 STRING_BYTE 펑션이 실행됩니다.
(2) IN_VAL = 'ABC' 이면 OUT_VAL[0] = 16#41, OUT_VAL[1] = 16#42, OUT_VAL[2] = 16#43 이 됩니다.

SWAP

데이터의 상위 하위 바꾸기

CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 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		○	○	○	○															

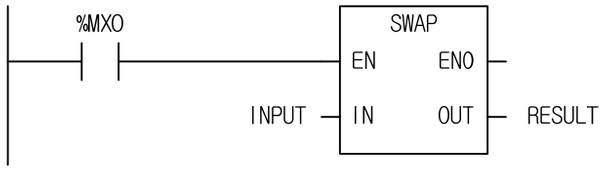
*ANY_BIT: ANY_BIT 타입 중 BOOL 제외

■ 기능

1. 입력된 변수를 2개의 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) 실행조건(%MX0)이 On 되면, SWAP 펑션이 실행됩니다.

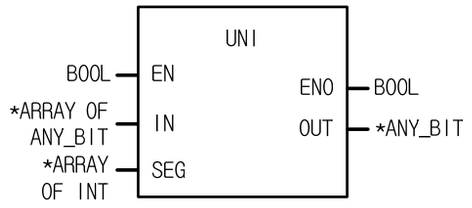
(2) 펑션의 입력 변수 INPUT(BYTE 타입) = 16#5F 일 경우, 펑션의 출력 변수 RESULT(BYTE 타입) = 16#F5 가 됩니다.

UNI

데이터 결합(union)

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

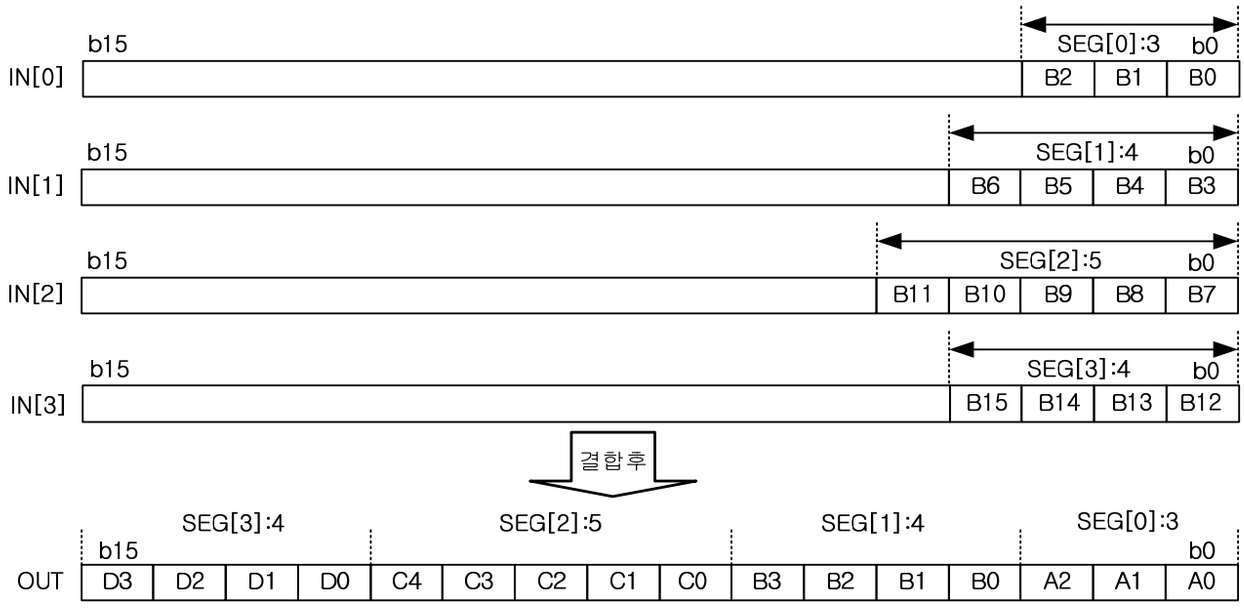
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN		○	○	○	○															
	OUT		○	○	○	○															

*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	

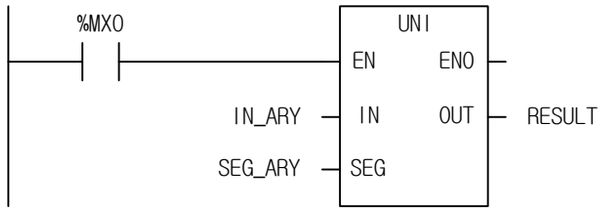


SEG로 지정된 값들의 합이 입력 변수 타입의 비트수를 초과할 경우 _ERR/_LER 플래그가 셋(Set)됩니다.

■ 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) 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

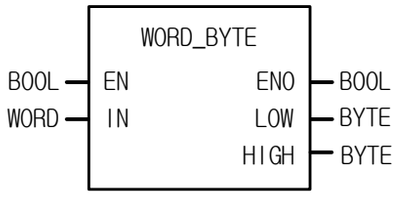
↓

RESULT : 2#00 1010111 0110 101

WORD_BYTE

WORD를 2개의 BYTE로 나눔

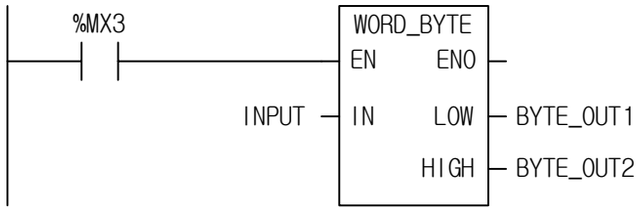
CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 IN : WORD 입력 **출력** ENO : EN 값을 그대로 출력 LOW : 하위 BYTE 출력 HIGH : 상위 BYTE 출력

■ 기능

1. 하나의 워드를 2개의 바이트로 분산합니다.
   LOW: 하위 바이트 출력, HIGH: 상위 바이트 출력

■ 프로그램 예

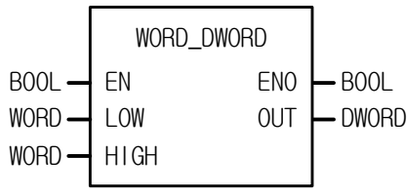


(1) 실행조건(%MX3)이 On 되면 WORD_BYTE 펑션이 실행됩니다.
(2) 입력변수로 선언된 INPUT값이 16#ABCD일 때 출력 변수로 선언된 변수 BYTE_OUT1 = 16#CD, BYTE_OUT2 = 16#AB 값이 저장됩니다.

WORD_DWORD

2개의 WORD를 DWORD로 모음

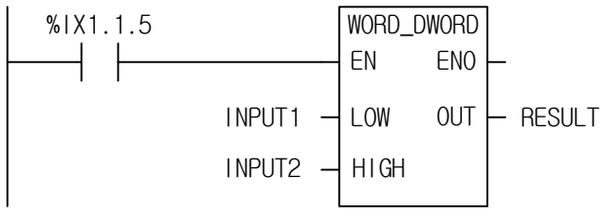
CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 때 펑션 실행 LOW : 하위 WORD 입력 HIGH : 상위 WORD 입력 **출력** ENO : EN값을 그대로 출력 OUT : DWORD 출력

■ 기능

1. 2개의 WORD를 하나의 DWORD로 조합합니다.
   LOW: 하위 워드 입력, HIGH: 상위 워드 입력

■ 프로그램 예

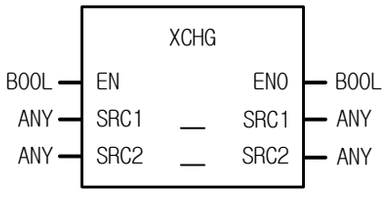


(1) 실행조건(%IX1.1.5)이 On 되면 WORD_DWORD 펑션이 실행됩니다.
(2) 입력변수로 선언된 INPUT1 = 16#1020_3040이고 INPUT2 = 16#A0B0_C0D0일 때 출력변수로 선언된 RESULT = 16#A0B0_C0D0_1020_3040가 됩니다.

XCHG

2개의 입력변수 값을 서로 교환

CPU 명	XGI	XGR
적용 가능	●	●

펑 션	설 명
	**입력** EN : 1일 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	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○

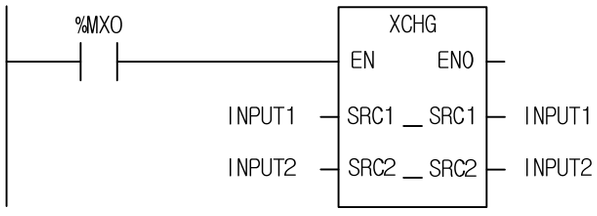
■ 기능

1. 입력 1과 입력 2의 데이터를 교환합니다.

펑 션	입출력타입	동작 설명
XCHG	BOOL	두개의 BOOL 입력 값을 서로 교환합니다.
XCHG	BYTE	두개의 BYTE 입력 값을 서로 교환합니다.
XCHG	WORD	두개의 WORD 입력 값을 서로 교환합니다.
XCHG	DWORD	두개의 DWORD 입력 값을 서로 교환합니다.
XCHG	LWORD	두개의 LWORD 입력 값을 서로 교환합니다.
XCHG	SINT	두개의 SINT 입력 값을 서로 교환합니다.
XCHG	INT	두개의 INT 입력 값을 서로 교환합니다.
XCHG	DINT	두개의 DINT 입력 값을 서로 교환합니다.
XCHG	LINT	두개의 LINT 입력 값을 서로 교환합니다.
XCHG	USINT	두개의 USINT 입력 값을 서로 교환합니다.
XCHG	UINT	두개의 UINT 입력 값을 서로 교환합니다.
XCHG	UDINT	두개의 UDINT 입력 값을 서로 교환합니다.
XCHG	ULINT	두개의 ULINT 입력 값을 서로 교환합니다.
XCHG	REAL	두개의 REAL 입력 값을 서로 교환합니다.
XCHG	LREAL	두개의 LREAL 입력 값을 서로 교환합니다.
XCHG	TIME	두개의 TIME 입력 값을 서로 교환합니다.

펑　　션	입출력타입	동작 설명
XCHG	DATE	두개의 DATE 입력 값을 서로 교환합니다.
XCHG	TOD	두개의 TOD 입력 값을 서로 교환합니다.
XCHG	DT	두개의 DT 입력 값을 서로 교환합니다.
XCHG	STRING	두개의 STRING 입력 값을 서로 교환합니다.

■ 프로그램 예

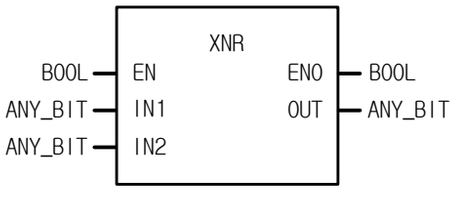


(1) 실행조건(%MX0)이 On 되면 XCHG 펑션이 실행됩니다.
(2) 펑션의 입력변수 INPUT1 = 0 이고, INPUT2 = 1이면 펑션이 실행된 후 2개의 입력값은 서로 바뀌어 출력됩니다.
따라서 펑션이 실행된 후 INPUT1 = 1이고, INPUT2 = 0이 됩니다.

XNR

배타적 논리곱

CPU 명	XGI	XGR
적용 가능	●	●

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.

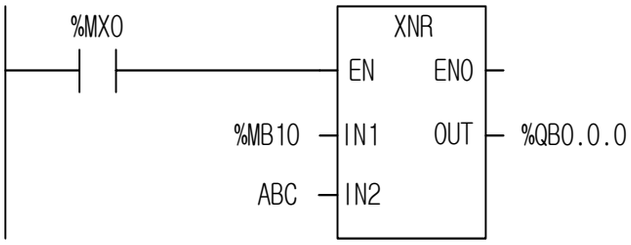
   IN1 1111 0000

   XNR

   IN2 1010 1010

   OUT 1010 0101

■ 프로그램 예



(1) 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.

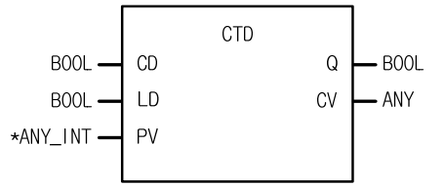
제 9 장 기본 펑션 블록

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

감산 카운터 (펑션 블록)

CPU 명	XGI	XGR
적용 가능	●	●

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							○	○	○		○	○	○							

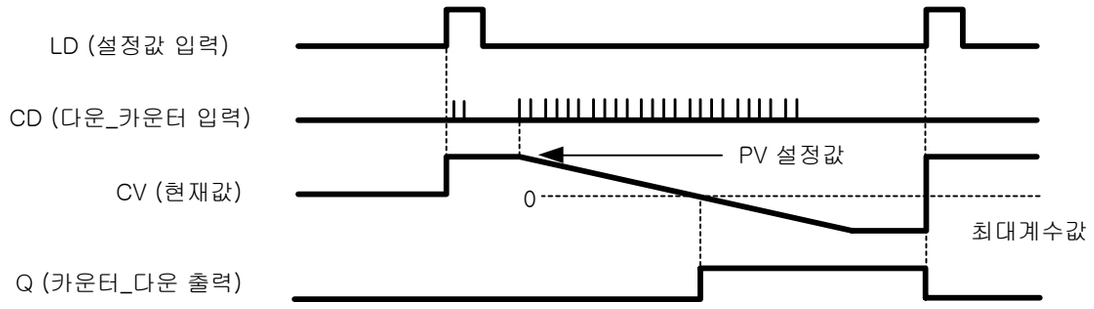
*ANY_INT: ANY_INT 타입 중 SINT, USINT 제외.

■ 기능

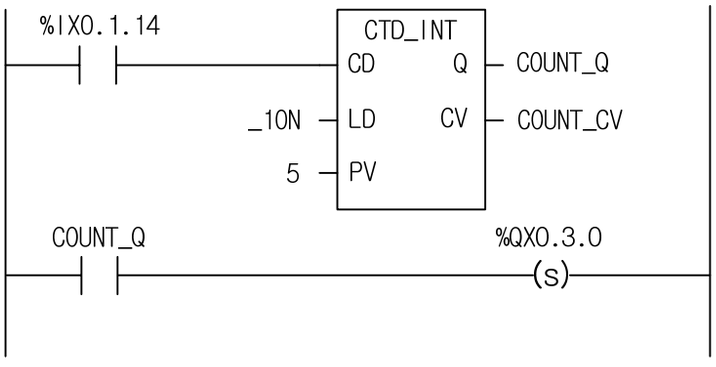
1. 감산 카운터 펑션 블록 CTD는 다운 카운터 펄스입력 CD가 0에서 1이 되면, 현재값 CV가 이전값보다 1만큼 감소하는 카운터 입니다.
2. 단, CV는 PV의 최소보다 클 때만 감소하고, 최소값이 되면 더이상 감소하지 않습니다.
3. 설정값 입력 LD가 1이 되면 현재값 CV에는 설정값 PV값이 로드 됩니다. (CV = PV)
4. 출력 Q는 CV가 0이하일 때만 1이 됩니다.

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



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.

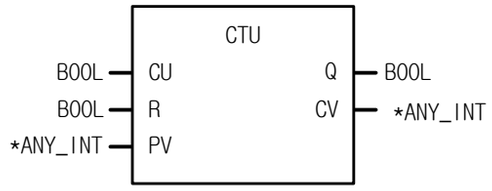
(11) 입력접점에 5 번의 펄스가 들어오면 현재값 CV 는 0 가 되고, Q(COUNT_Q)는 1 이 됩니다.

(12) Q(COUNT_Q)가 1 이 되면 출력접점 %QX0.3.0 이 Set 됩니다.

CTU

가산 카운터 (펑션 블록)

CPU 명	XGI	XGR
적용 가능	●	●

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							○	○	○		○	○	○							

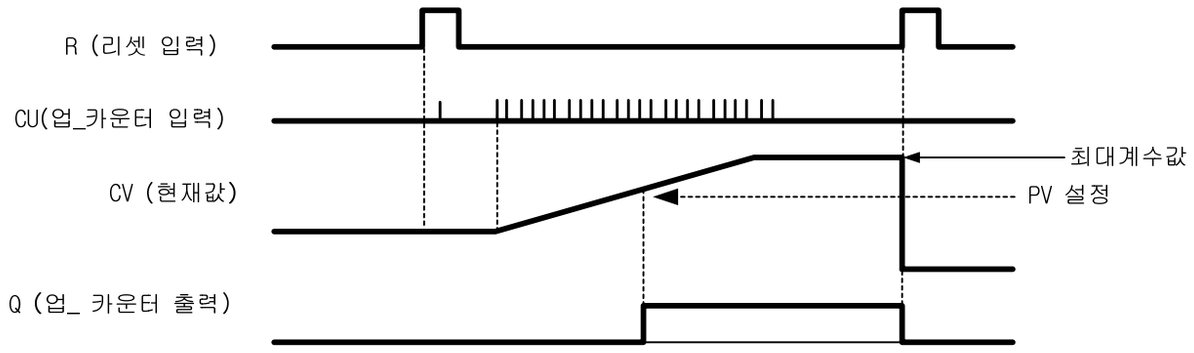
*ANY_INT: ANY_INT 타입 중 SINT, USINT 제외.

■ 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.

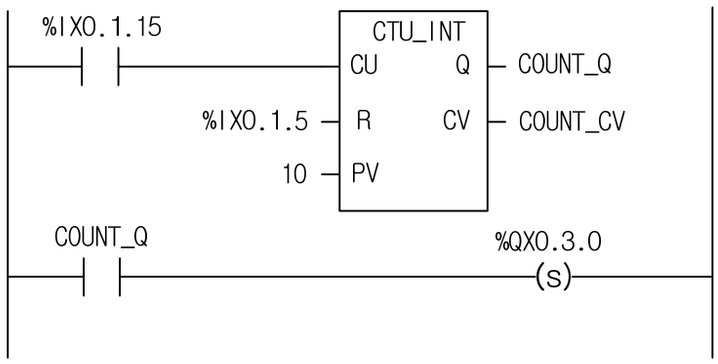
펑션 블록	PV	동작 설명
CTU_INT	INT	INT(설정값)의 최대값(32767)만큼 증가합니다.
CTU_DINT	DINT	DINT(설정값)의 최대값(2147483647)만큼 증가합니다.
CTU_LINT	LINT	LINT(설정값)의 최대값(9223372036854775807)만큼 증가합니다.
CTU_UINT	UINT	UINT(설정값)의 최대값(0)만큼 증가합니다.
CTU_UDINT	UDINT	UDINT(설정값)의 최대값(0)만큼 증가합니다.
CTU_ULINT	ULINT	ULINT(설정값)의 최대값(0)만큼 증가합니다.

■ 타임차트



■ 프로그램 예

1. 입력접점 %IX0.1.15 에 10 번의 펄스가 들어오면, 출력접점 %QX0.3.0 을 Set 시키는 프로그램



(1) 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.

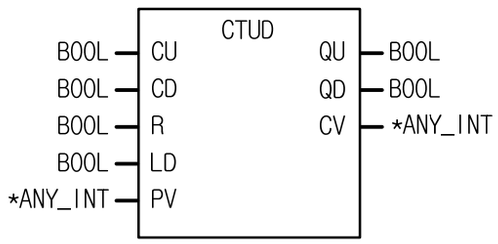
(10) 입력접점에 10 번의 펄스가 들어오면 현재값 CV 는 10 이 되고, 설정값과 같게 되므로 출력 Q(COUNT_Q)가 1 이 됩니다.

(11) Q(COUNT_Q)가 1 이 되면 출력접점 %QX0.3.0 이 셋(Set)됩니다.

CTUD

가감산 카운터 (펑션 블록)

CPU 명	XGI	XGR
적용 가능	●	●

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							○	○	○		○	○	○							

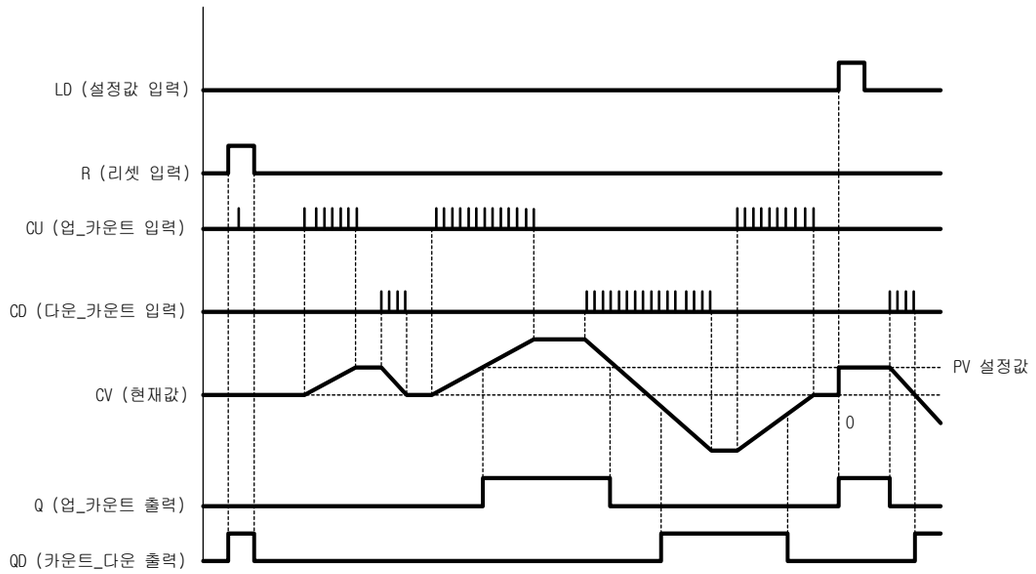
*ANY_INT: ANY_INT 타입 중 SINT, USINT 제외.

■ 기능

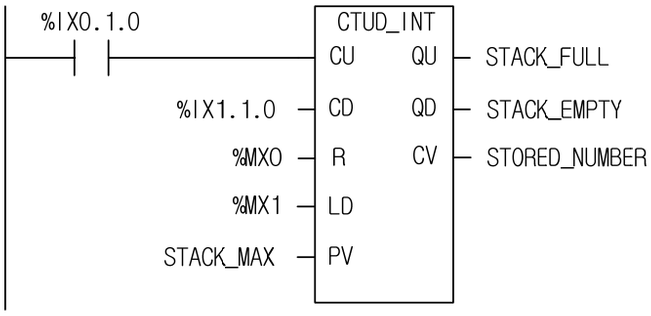
1. 가감산 카운터 펑션 블록 CTUD 는 업카운터 펄스 입력 CU 가 0 에서 1 이 되면 현재값 CV 가 이전값보다 1 만큼 증가하고, 다운 카운터 펄스입력 CD 가 0 에서 1 이 되면 현재값 CV 가 이전값보다 1 만큼 감소하는 카운터 입니다.
2. 단, CV 가 PV 의 최소값과 최대값 사이의 값을 가지며 최대, 최소값에 이르면 각각 더이상 증가, 감소하지 않습니다.
3. 설정값 입력 LD 가 1 이 되면 현재값 CV 에는 설정값 PV 값이 로드됩니다. (CV = PV)
4. 설정값 입력 R 이 1 이 되면 현재값 CV 는 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.

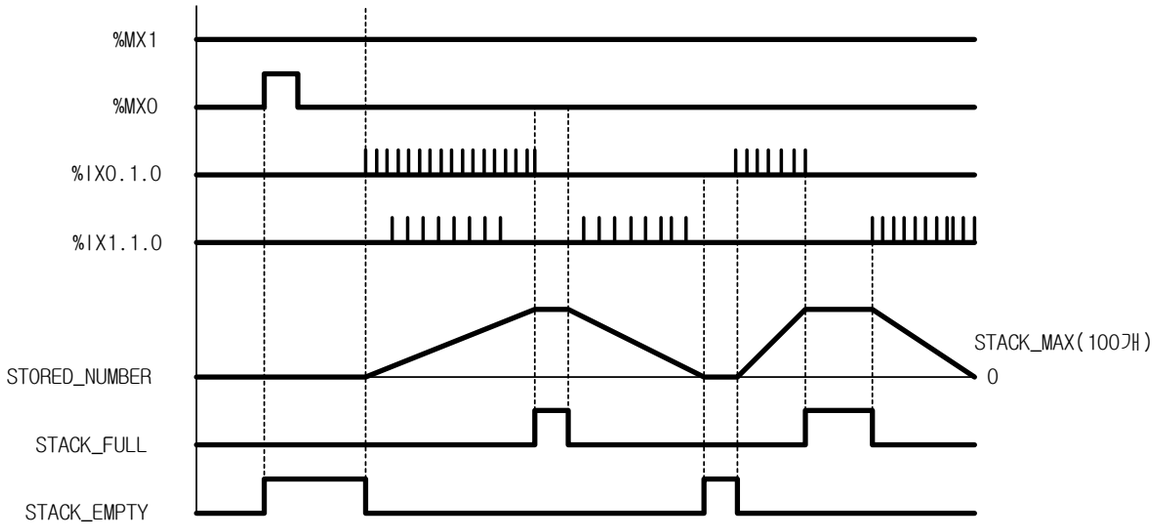
■ 타임차트



■ 프로그램 예



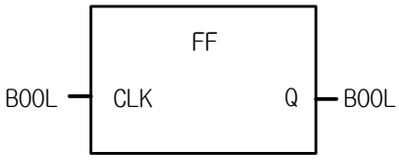
공정라인에서 중간 적재 완충부에 임시 저장될 수 있는 용량 STACK_MAX 의 값이 100 입니다. 적재부에 자재가 투입될 때마다 IN 신호가 1 이 되고, 적재부에서 자재가 빠져 나갈 때마다 OUT 신호가 1 이 되도록 설정되어 있습니다. 운전을 시작하여 선 공정에서 완충부에 투입되는 속도가 다음 공정을 위해 빠져나가는 속도보다 빠를 때, 중간 적재부에 100 개가 쌓이면 카운터의 상한 출력 QU 가 1 이 되어 STACK_FULL 에 1 이 출력됩니다. 반대로 중간 적재부에 남아 있는 것이 없으면 하한 출력 QD 가 1 이 되어 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

출력 비트 반전

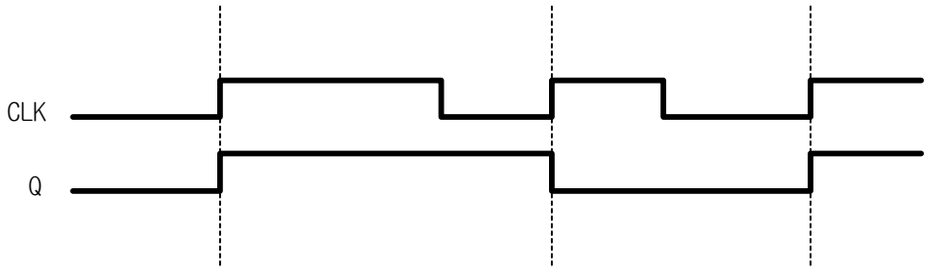
CPU 명	XGI	XGR
적용 가능	●	●

펑션 블록	설 명
	**입력** CLK : 입력신호 **출력** Q : 상승에 지시 출력반전

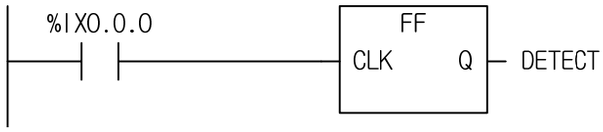
■ 기능

1. FF 는 CLK 에 연결된 입력의 상태가 0 에서 1 로 변할 때 출력 Q 를 반전 시킵니다.

■ 타임차트



■ 프로그램 예

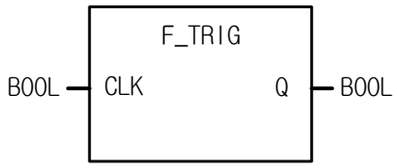


(1) 입력변수 %IX0.0.0 의 상태를 감시하여, 입력변수가 0 에서 1 으로 변할 때마다 출력변수 DETECT 의 출력을 반전시킵니다.

F_TRIG

하강 에지 검출 (펑션 블록)

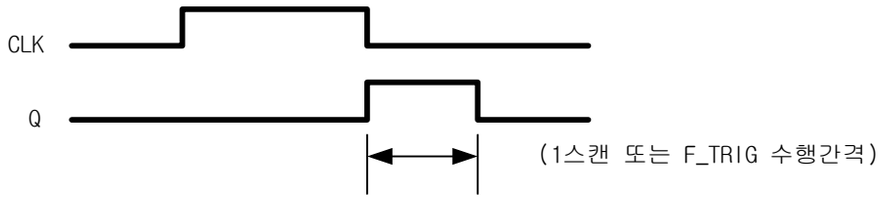
CPU 명	XGI	XGR
적용 가능	●	●

펑션 블록	설 명
	**입력** CLK : 입력신호 **출력** Q : 하강 에지 검출결과

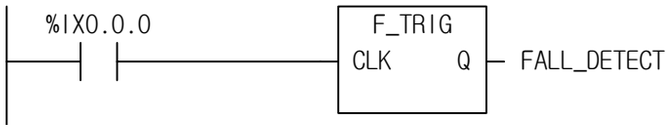
■ 기능

1. F_TRIG는 CLK에 연결된 입력의 상태가 1에서 0으로 변할 때 출력 Q를 1로 만들고 다음 수행에서 0으로 만듭니다. 그 외는, 출력Q가 항상 0이 됩니다.

■ 타임차트



■ 프로그램 예

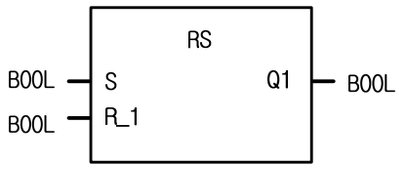


(1) 입력변수 %IX0.0.0 의 상태를 감시하여 1 에서 0 으로 변할 때에 출력변수 FALL_DETECT 에 1 을 출력하고, 다음에 펑션 블록 수행시 FALL_DETECT 에 0을 출력합니다.

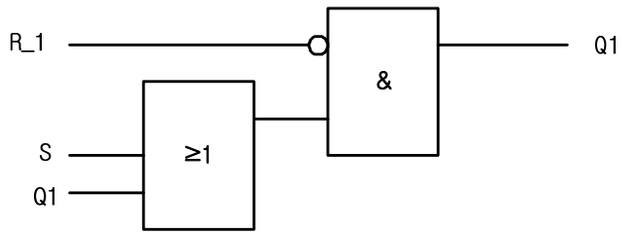
RS

Reset 우선 Bistable(펑션 블록)

CPU 명	XGI	XGR
적용 가능	●	●

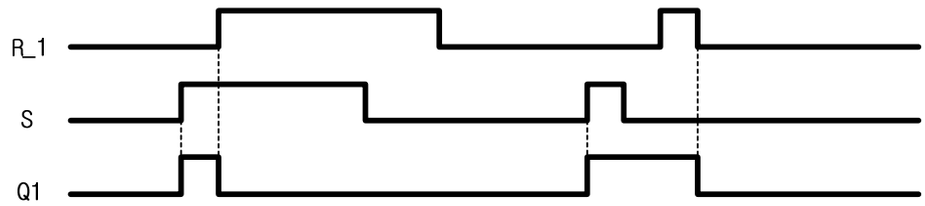
펑션 블록	설 명
	**입력** R_1 : Reset 조건 S : Set 조건 **출력** Q1 : 연산결과

■ 기능

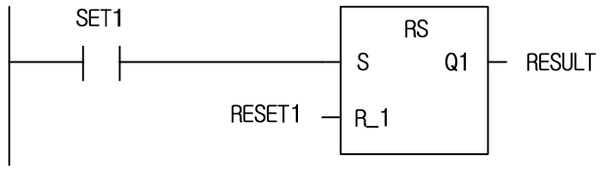


R_1 이 1 이면, S 와 관계없이 출력 Q1 은 항상 0 이 됩니다. 출력 Q1 은 이전 상태를 유지하며, R_1 이 0 이고 S 가 1 일 때 1 이 됩니다. Q1 의 초기상태는 0 입니다.

■ 타임차트



■ 프로그램 예



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.

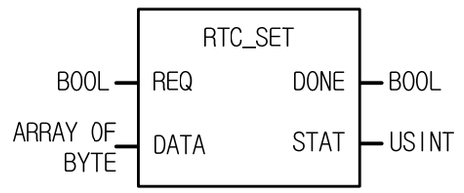
(3) 입력 변수로 선언된 SET1 과 RESET1 이 On 되면 출력 변수 RESULT 는 0 가 됩니다.

RTC_SET

시간 데이터 쓰기

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

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.

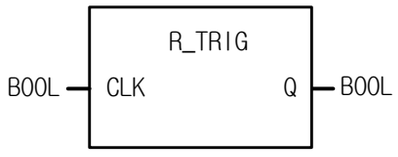
예: 1998. 12. 22. 19:37:46, 화

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

상승에지 검출 (펑션 블록)

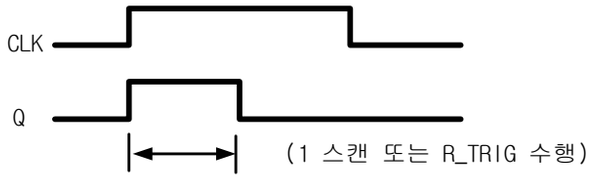
CPU 명	XGI	XGR
적용 가능	●	●

펑션 블록	설 명
	**입력** CLK : 입력신호 **출력** Q : 상승 에지 검출결과

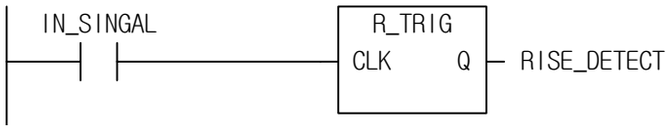
■ 기능

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

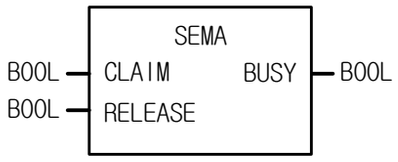


입력변수로 선언된 IN_SIGNAL 의 상태를 감시하여 0 에서 1 로 변할 때, RISE_DETECT 에 1 을 출력하고 R_TRIG 펑션 블록 수행시 RISE_DETECT 에 0 을 출력합니다.

SEMA

시스템 자원 배분 (펑션 블록)

CPU 명	XGI	XGR
적용 가능	●	●

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.

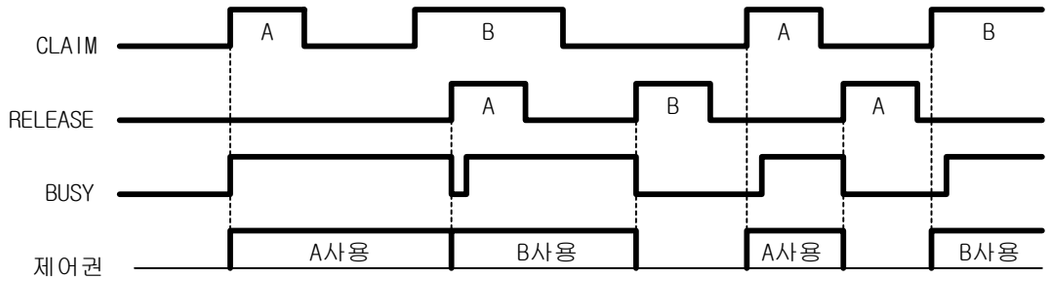
(이때 CLAIM = 0, RELEASE = 1 의 값으로 SEMA 를 수행하는 제어 양도 동작은 반드시 현재 제어권을 가지고 있는 프로그램에서만 수행해야 합니다.)

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
```

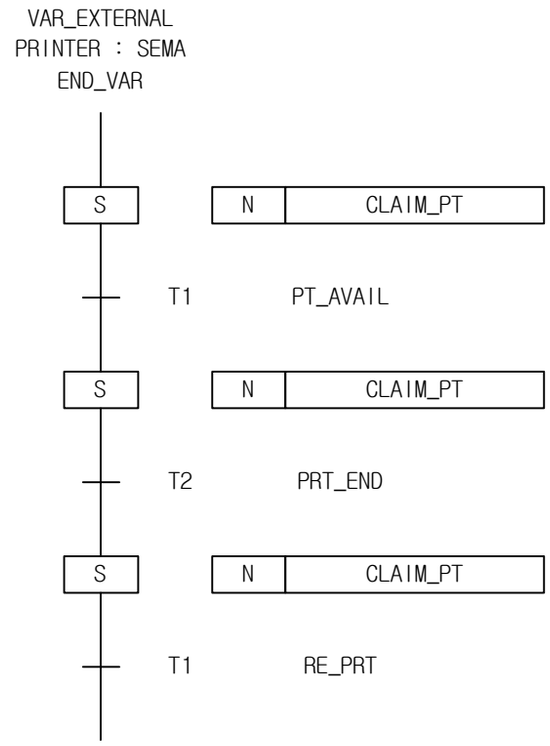
■ 타임차트

프로그램 블록 A 와 프로그램 블록 B 에서 동일한 자원에 대한 액세스권을 주고 받는 경우

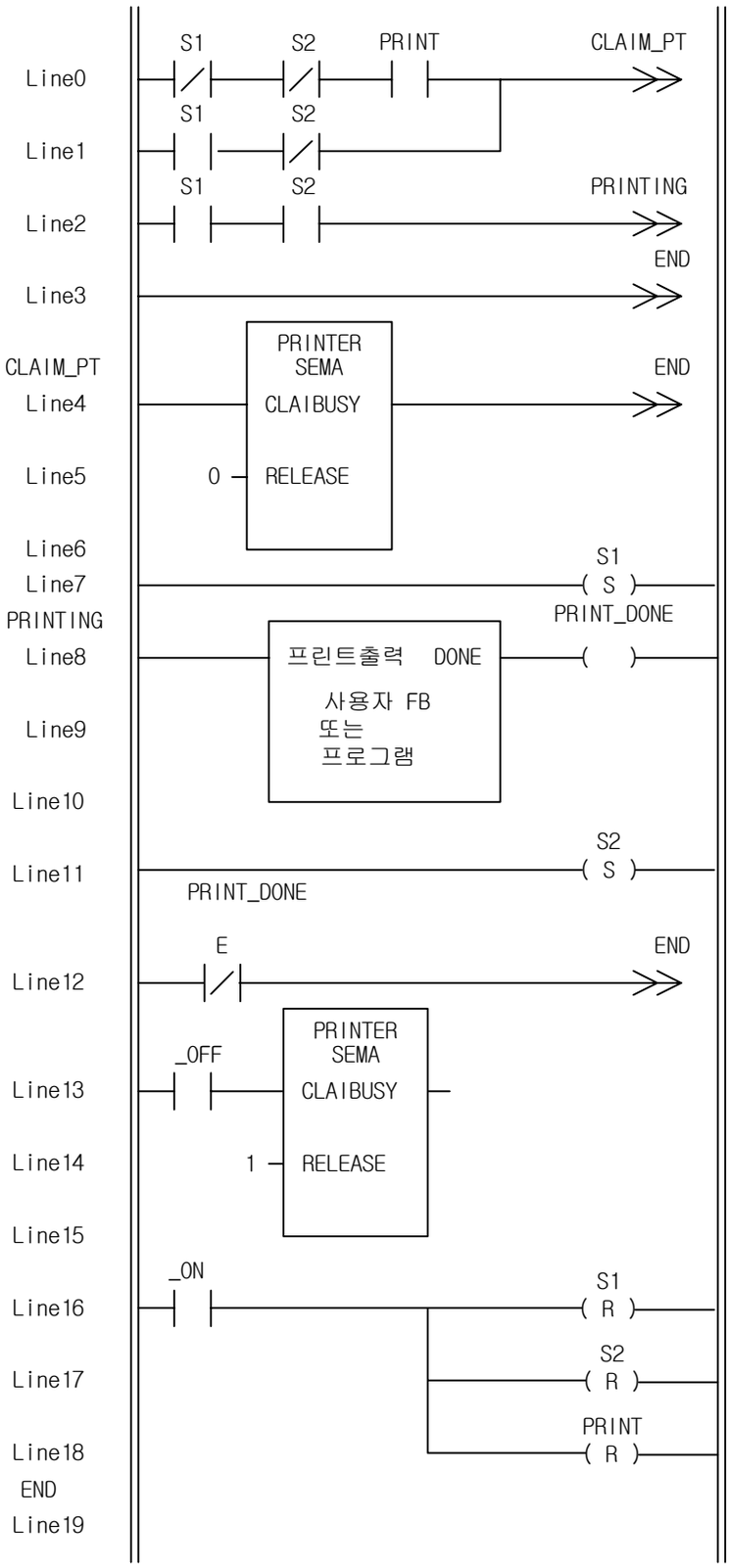


■ 프로그램 예

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.



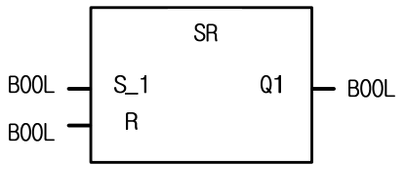
S	CLAIM_PT;프린터 제어권 요구
	CAL SEMA PRINTER CLAIM:= 1 RELEASE:= 0
T	PT_AVAIL;프린터 제어권 획득 체크
	LDN PRINTER.BUSY ST TRANS
S	PRINTING;프린터 출력
	프린터 제어 프로그램 프린터 완료시 PRINT_DONE:=1
T	PRT_END;프린터 완료 체크
	LD PRINTER_DONE ST TRANS
S	REL_PRT;프린터 제어권 양도
	CAL SEMA PRINTER CLAIM:= 0 RELEASE:= 1
T	RE_PRT;프린터 재요구
	LD PRT_REQ 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
PRINT_DONE
Line8
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

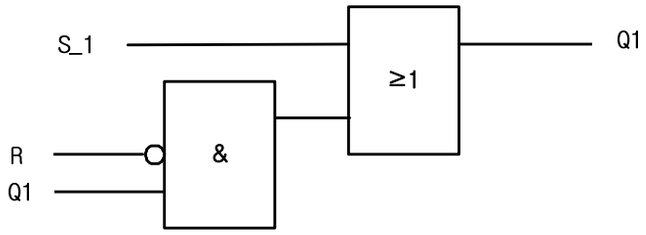
SR

Set우선 Bistable (펑션 블록)

CPU 명	XGI	XGR
적용 가능	●	●

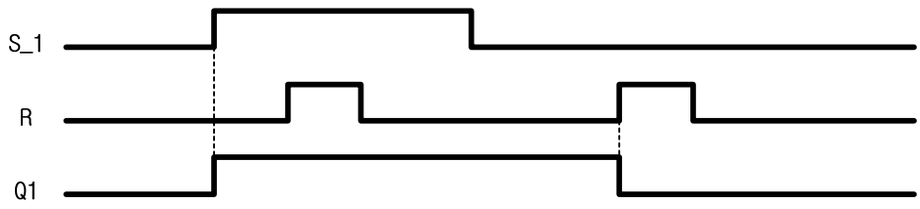
펑션 블록	설 명
	**입력** S_1 : Set 조건 R : Reset 조건 **출력** Q1 : 연산결과

■ 기능

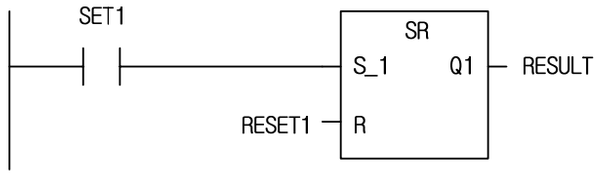


1. S_1이 1이 되면, R과 관계없이 출력 Q1은 항상 1이 됩니다.
2. 출력 Q1은 이전 상태를 유지하며, S_1이 0이고 R이 1일 때 0이 됩니다.
3. Q1의 초기상태는 0입니다.

■ 타임차트



■ 프로그램 예



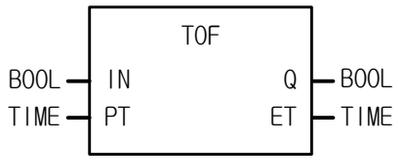
(1) 입력변수로 선언된 SET1 이 On 되면, 출력변수로 선언된 RESULT 같이 1 이 됩니다.

(2) 출력변수로 선언된 RESULT 값이 0 이 되도록 하려면, 입력변수로 선언된 SET1 이 Off 되고, RESET 이 On 되어야 합니다.

TOF

Off 딜레이 타이머 (펑션 블록)

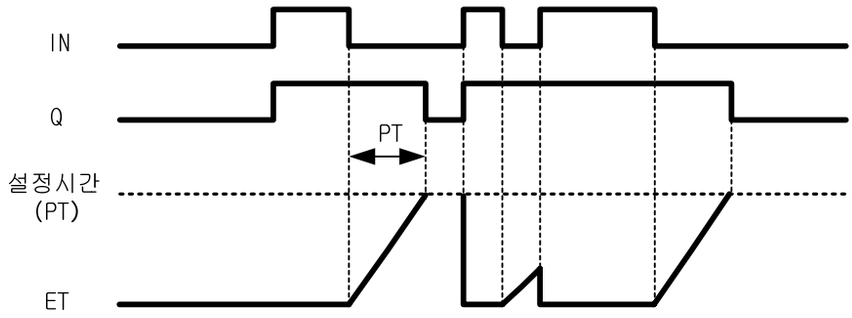
CPU 명	XGI	XGR
적용 가능	●	●

펑션 블록	설 명
	**입력** IN : 타이머 기동 조건 PT : 설정 시간(Preset Time) **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)

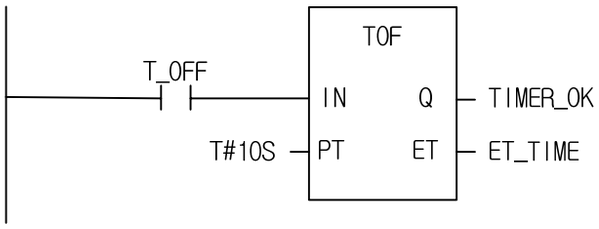
■ 기능

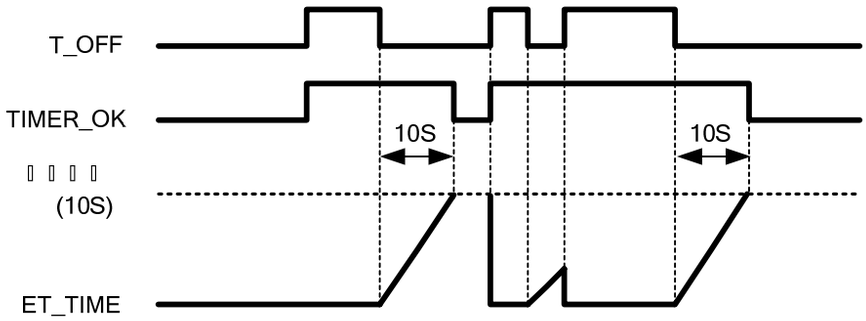
1. IN이 1이 되면, Q가 1이 되고, IN이 0이 된 후부터 PT에 의해서 지정된 설정시간이 경과한 후 Q가 0이 됩니다.
2. IN이 0이 된 후 경과시간이 ET로 출력됩니다.
3. 만일 경과시간 ET가 설정시간에 도달하기 전에 IN이 1이 되면, 경과시간은 다시 0으로 됩니다.

■ 타임차트



■ 프로그램 예



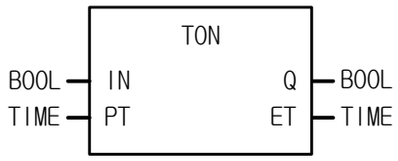


(1) 입력변수로 설정된 T_OFF 가 1 이 되면, 출력변수 TIMER_OK 에 1 이 출력되고 T_OFF 가 0 이 된 후 10 초 후에 TIMER_OK 가 0 이 됩니다.
(2) T_OFF 가 0 이 된 후 10 초 이내에 다시 1 이 되면 타이머는 다시 초기상태가 됩니다.
(3) 타이머의 시간 측정값은 ET_TIME 로 출력됩니다.

TON

On 딜레이 타이머 (펑션 블록)

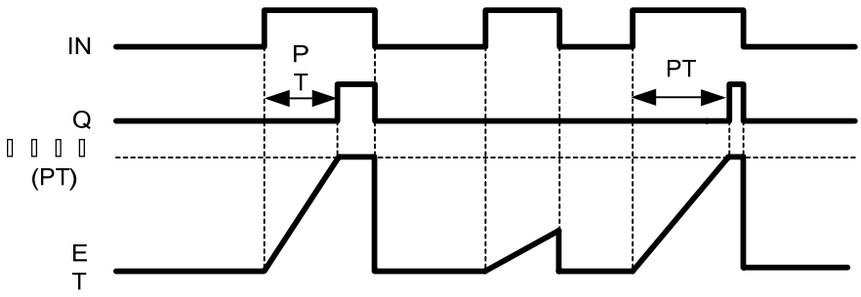
CPU 명	XGI	XGR
적용 가능	●	●

펑션 블록	설 명
	**입력** IN : 타이머 기동 조건 PT : 설정 시간 (Preset Time) **출력** Q : 타이머 출력 ET : 경과 시간(Elapsed Time)

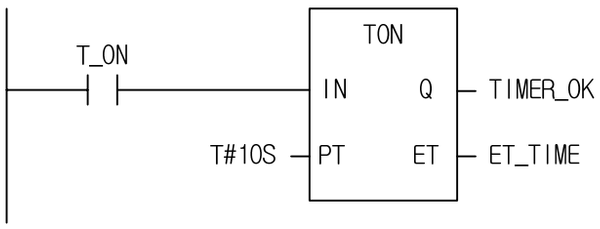
■ 기능

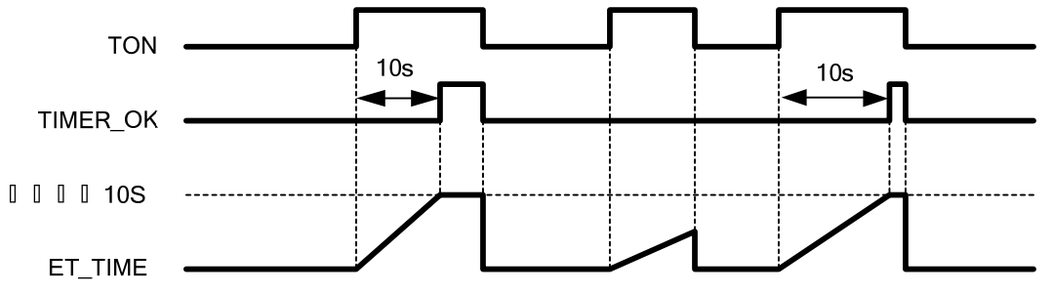
1. IN이 1이 된 후 경과 시간이 ET로 출력됩니다.
2. 만일 경과시간 ET가 설정 시간에 도달하기 전에 IN이 0이 되면, 경과 시간은 0으로 됩니다.
3. Q가 1이 된 후 IN이 0이 되면, Q는 0이 됩니다.

■ 타임차트



■ 프로그램 예





(1) 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.

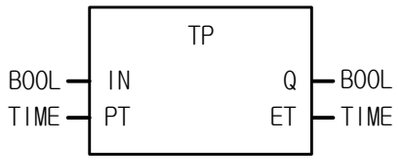
(3) 만일 경과 시간 ET_TIME 이 설정 시간 10 초에 도달하기 전에 T_ON 이 0 이 되면, 경과 시간 ET_TIME 은 0 으로 됩니다.

(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

펄스 타이머 (펑션 블록)

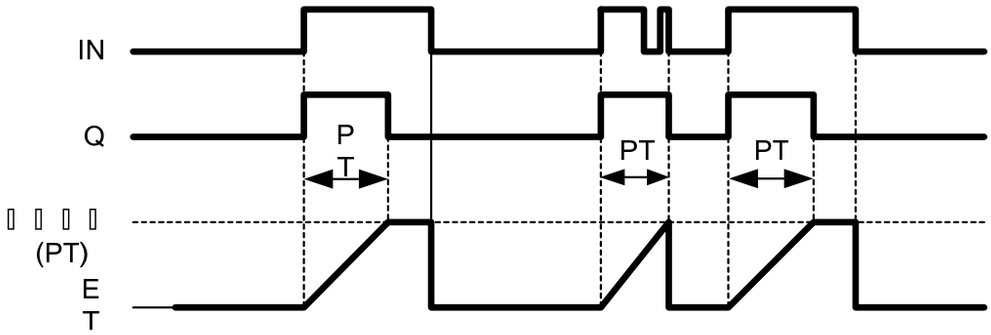
CPU 명	XGI	XGR
적용 가능	●	●

펑션 블록	설 명
	**입력** IN : 타이머 기동조건 PT : 설정 시간 (Preset Time) **출력** Q : 타이머 출력 ET : 경과 시간 (Elapsed Time)

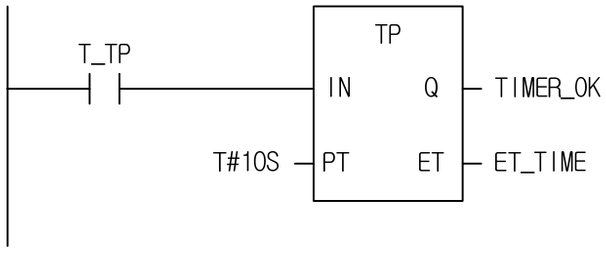
■ 기능

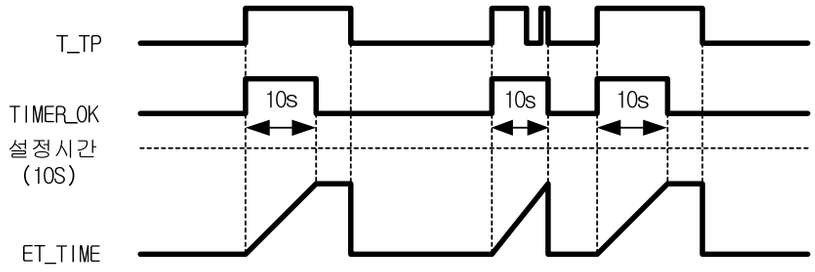
1. IN 이 1 이 되면 PT 에 의해서 지정된 설정 시간 동안만 Q 가 1 이 되고, ET 가 PT 에 도달하면 자동으로 0 이 됩니다.
2. 경과시간 ET 는 IN 이 1 이 되었을 때부터 증가하며 PT 에 이르면 값을 유지하다가 IN 이 0 이 될 때 0 의 값이 됩니다.
3. ET 가 증가할 동안은 IN 이 0 이 되거나 재차 1 이 되어도 영향이 없습니다.

■ 타임차트



■ 프로그램 예





(1) 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.

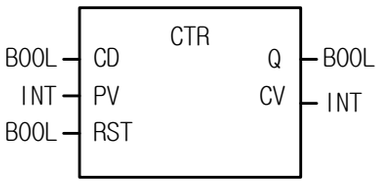
제 10 장 응용 펑션 블록

9 장에서 설명한 기본 펑션 블록과는 다른 응용 펑션 블록에 대하여 설명합니다.

CTR

Ring 카운터

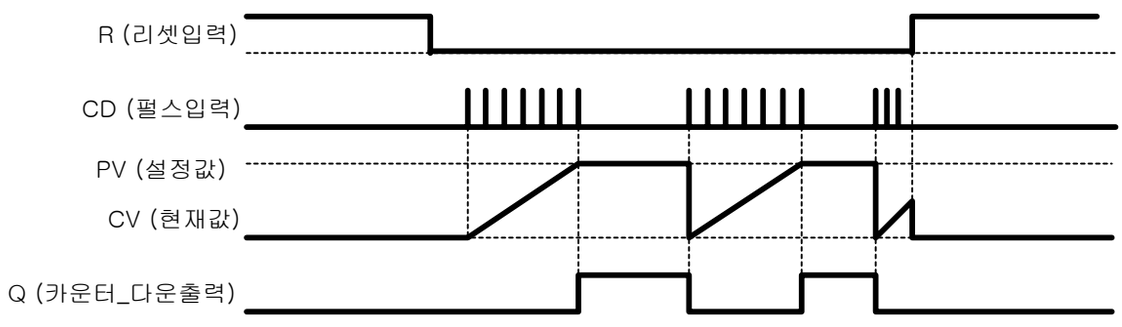
CPU 명	XGI	XGR
적용 가능	●	●

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

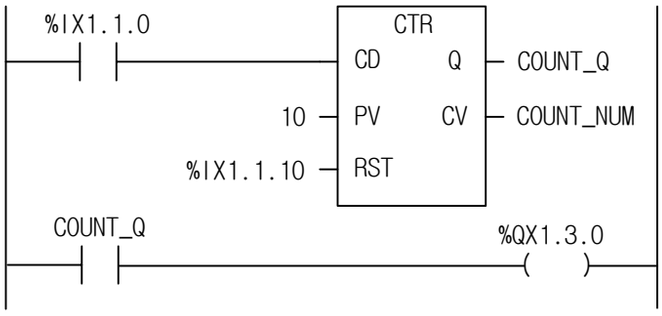
1. 링 카운터 CTR 펑션 블록은 펄스 입력 CD 가 0 에서 1 이 될 때마다 현재값 CV 를 +1 하고 현재값 CV 가 설정값 PV 에 도달한 후 CD 가 다시 0 에서 1 로 되면 현재값은 1 로 됩니다.
2. 현재값이 설정값에 도달하면 출력(Q)은 1 이 됩니다.
3. 현재치가 설정치 미만이거나 Reset 조건이 1 이 되면 출력(Q)은 0 이 됩니다

■ 타임 차트



■ 프로그램 예

입력 접점 %IX1.1.0 에 10 번의 펄스가 들어올 때마다, 출력접점 %QX1.3.1 을 On 시키는 프로그램



(1) 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.

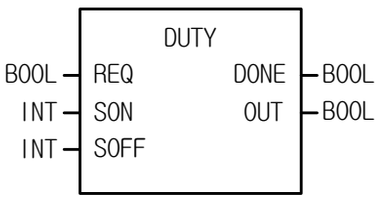
(11) Q(COUNT_Q)가 1 이 되면 출력 접점 %QX1.3.0 은 On 됩니다.

(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

스캔 지정 On/Off

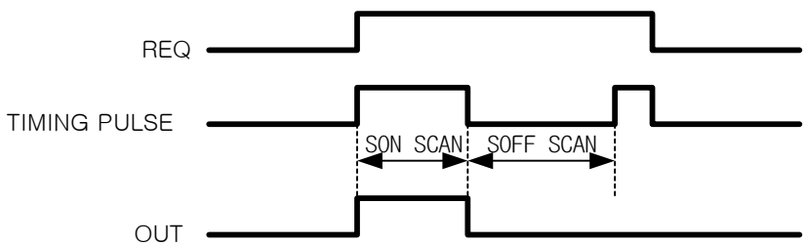
CPU 명	XGI	XGR
적용 가능	●	●

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

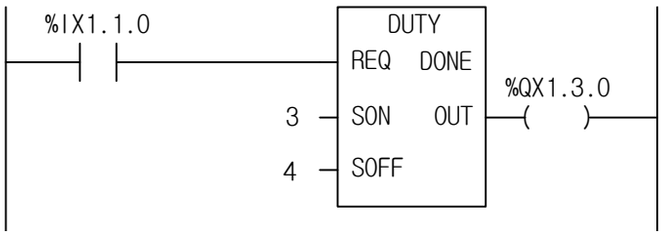
1. DUTY FB 는 REQ 가 On 된 시점부터 SON Scan 동안 On, SOFF Scan 동안 Off 하는 펄스를 발생시킵니다.
2. SON = 0 이면 출력은 항상 Off 가 됩니다.
3. SON > 0, SOFF = 0 이면 출력은 항상 On 이 됩니다.
4. REQ 가 Off 면 출력은 Off 됩니다.
5. SON < 0 이거나 SOFF < 0 이면, DONE 은 Off 되고 OUT = 0 이 됩니다.

■ 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) 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.

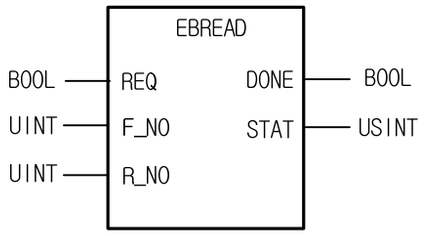
(7) 쓰기를 완료하면 모드전환(Stop → Run) 시킵니다.

(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

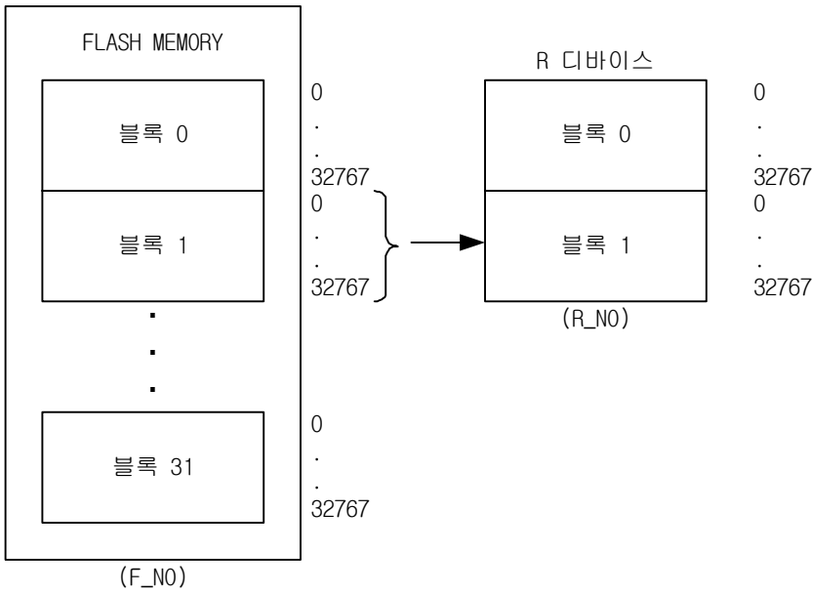
플래시 영역에 R영역 데이터 읽기

CPU 명	XGI	XGR
적용 가능	●	●

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 1을 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.

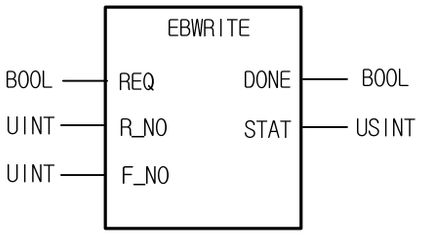


2. R_NO 의 값이 2 이상이면 STAT = 1, F_NO 의 값이 32 이상이면 STAT = 2 가 되며, _ERR, _LER 이 On 됩니다.
   또한 플래시로부터 읽기를 진행하고 있을 경우에는 DONE = 0 이고, STAT = 5 가 됩니다.
   이미 플래쉬 영역에 읽기/쓰기가 진행중인 경우에는 명령어 동작 시 DONE = 0 이고, STAT = 10 이 됩니다.
3. 명령어 수행 중에 _RBLOCK_RD_FLAG 의 F_NO 에 해당하는 비트가 On 됩니다.

EBWRITE

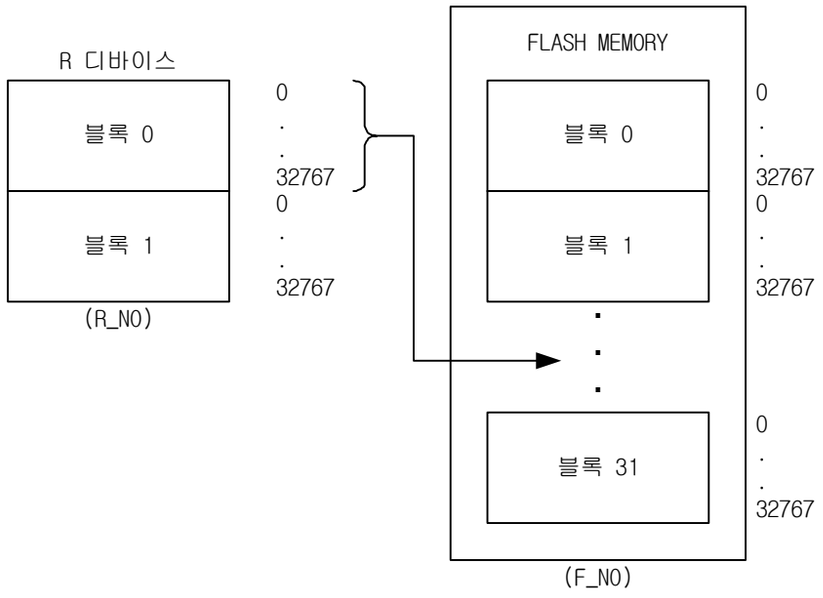
R영역 데이터를 플래시 영역에 쓰기

CPU 명	XGI	XGR
적용 가능	●	●

펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행요구 R_NO : R 디바이스(내부램)의 블록번호 E_NO : 저장할 플래쉬 영역의 블록 번호 **출력** DONE : 정상적으로 동작 후 1을 유지 STAT : ERR 정보

■ 기능

1. 지정된 R 디바이스의 1 개의 블록(64Kbyte) 내용을 저장할 플래쉬 영역의 블록으로 데이터를 전송합니다. 정상적으로 수행 완료시 DONE 이 1 이 됩니다.

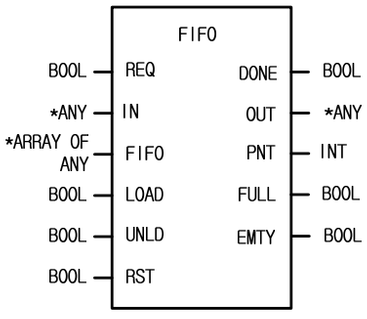


2. R_NO 의 값이 2 이상이면 STAT = 1, F_NO 의 값이 32 이상이면 STAT = 2 가 되며, _ERR, _LER 이 On 됩니다.
   또한 플래시에 쓰기를 진행하고 있을 경우에는 DONE = 0 이고, STAT = 5 가 됩니다.
   이미 플래쉬 영역에 읽기/쓰기가 진행중인 경우에는 명령어 동작 시 DONE = 0 이고, STAT = 10 이 됩니다.
3. 명령어 수행 중에 _RBLOCK_WR_FLAG 의 F_NO 에 해당하는 비트가 On 됩니다.

FIFO

FIFO 스택에 값을 Load / Unload (선입 선출)

CPU 명	XGI	XGR
적용 가능	●	●

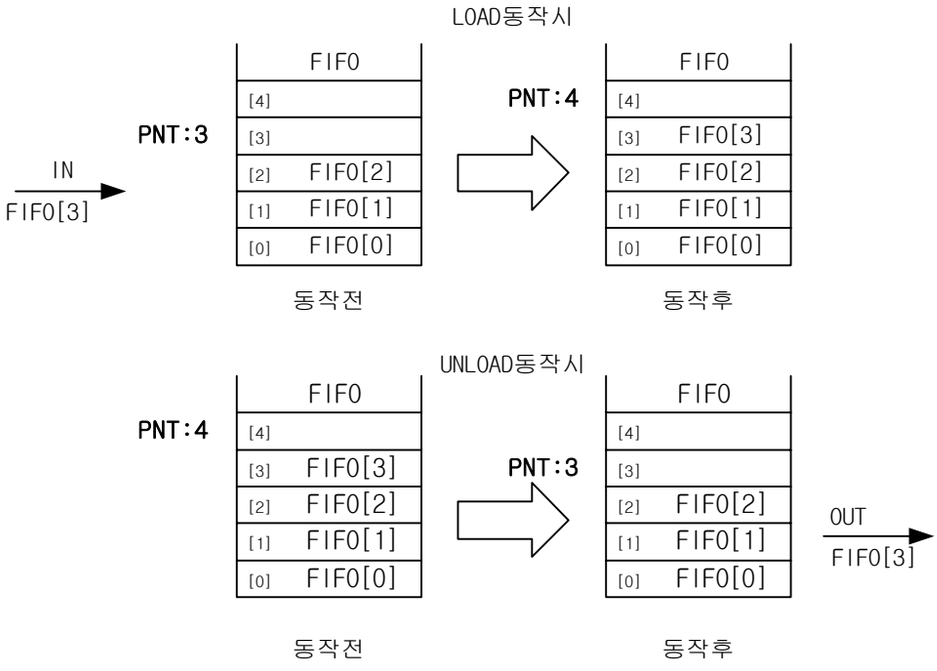
펑션 블록	설 명
	**입력** REQ : 펑션 블록 실행 요구 IN : FIFO 스택에 저장할 변수 또는 상수값 LOAD : On 이면 입력 모드 UNLD : On 이면 출력 모드 RST : POINTER VALUE 리셋 FIFO : FIFO 스택으로 사용되는 어레이 **출력** DONE : 최초 동작 후 1을 유지 OUT : 출력 모드일 경우 FIFO 스택으로부터 나온 값을 출력 PNT : FIFO 스택에 입력된 값에 대한 Pointer FULL : FIFO 스택이 가득차면 1을 출력 EMTY : FIFO 스택에 아무런 값도 저장되어 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	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	FIFO	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ANY: ANY 타입 중 STRING 제외, *ARRAY OF ANY: ARRAY_ANY 타입 중 STRING 제외

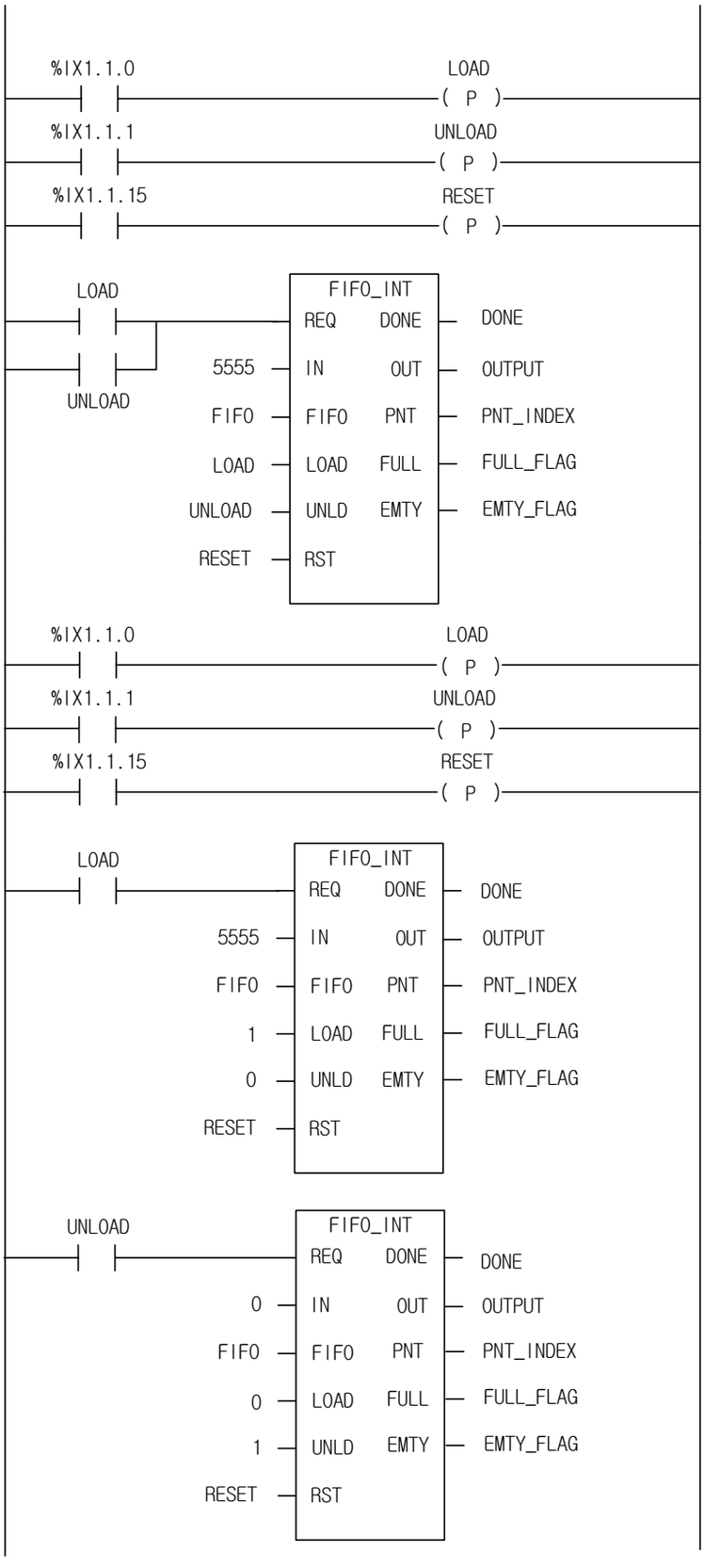
■ 기능

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.



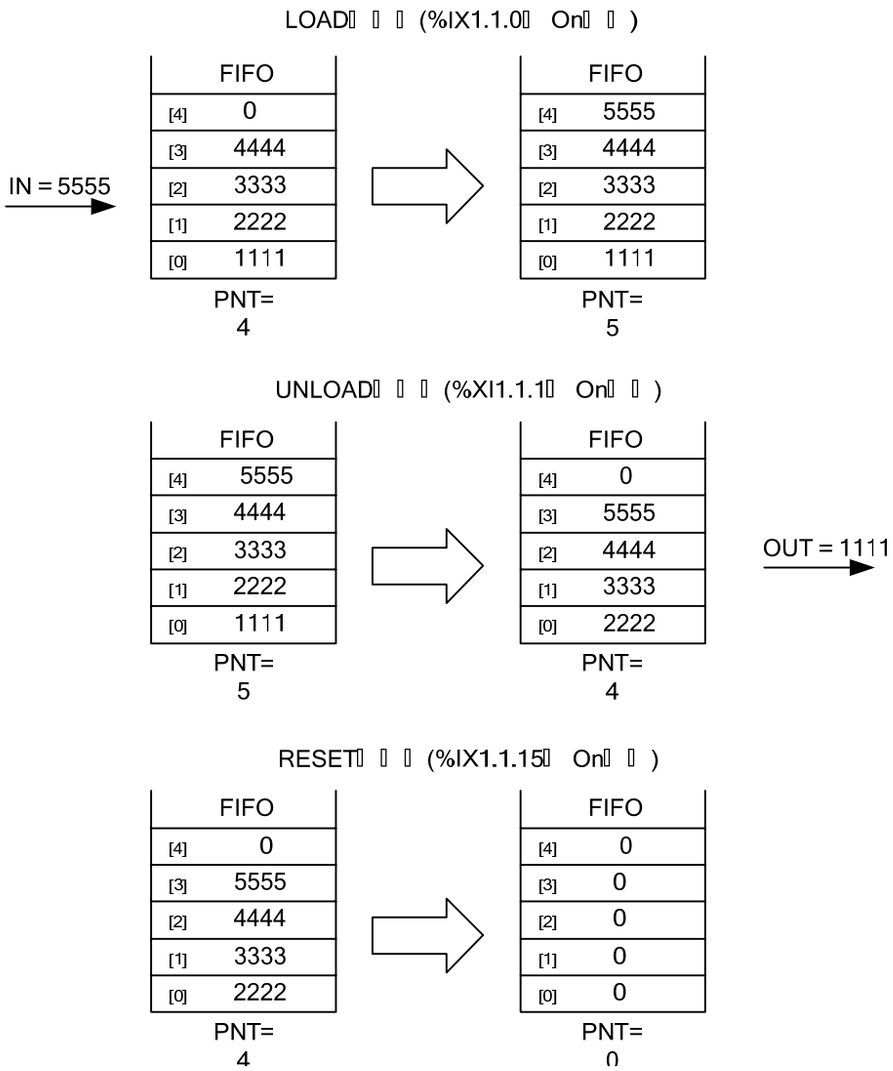
펑션 블록	FIFO 변수 타입	동작 설명
FIFO_BOOL	BOOL	BOOL 타입 DATA에 대한 FIFO 동작을 수행합니다.
FIFO_BYTE	BYTE	BYTE 타입 DATA에 대한 FIFO 동작을 수행합니다.
FIFO_WORD	WORD	WORD 타입 DATA에 대한 FIFO 동작을 수행합니다.
FIFO_DWORD	DWORD	DWORD 타입 DATA에 대한 FIFO 동작을 수행합니다.
FIFO_LWORD	LWORD	LWORD 타입 DATA에 대한 FIFO 동작을 수행합니다.
FIFO_SINT	SINT	SINT 타입 DATA에 대한 FIFO 동작을 수행합니다.
FIFO_INT	INT	INT 타입 DATA에 대한 FIFO 동작을 수행합니다.
FIFO_DINT	DINT	DINT 타입 DATA에 대한 FIFO 동작을 수행합니다.
FIFO_LINT	LINT	LINT 타입 DATA에 대한 FIFO 동작을 수행합니다.
FIFO_USINT	USINT	USINT 타입 DATA에 대한 FIFO 동작을 수행합니다.
FIFO_UINT	UINT	UINT 타입 DATA에 대한 FIFO 동작을 수행합니다.
FIFO_UDINT	UDINT	UDINT 타입 DATA에 대한 FIFO 동작을 수행합니다.
FIFO_ULINT	ULINT	ULINT 타입 DATA에 대한 FIFO 동작을 수행합니다.
FIFO_REAL	REAL	REAL 타입 DATA에 대한 FIFO 동작을 수행합니다.
FIFO_LREAL	LREAL	LREAL 타입 DATA에 대한 FIFO 동작을 수행합니다.
FIFO_TIME	TIME	TIME 타입 DATA에 대한 FIFO 동작을 수행합니다.
FIFO_DATE	DATE	DATE 타입 DATA에 대한 FIFO 동작을 수행합니다.
FIFO_TOD	TOD	TOD 타입 DATA에 대한 FIFO 동작을 수행합니다.
FIFO_DT	DT	DT 타입 DATA에 대한 FIFO 동작을 수행합니다.

■ 프로그램 예


%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.

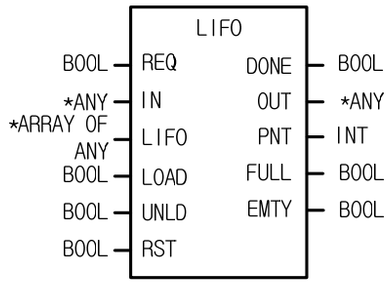
(1) 입력조건(%IX1.1.0, %IX1.1.1, %IX1.1.15)이 성립되면 FIFO_INT 가 실행됩니다.
(2) 입력접점 %IX1.1.0 이 On 되면 Load 동작을 수행합니다. 5555 가 FIFO 스택에 입력되고 PNT_INDEX 가 1 증가합니다.
(3) 입력접점 %IX1.1.1 이 On 되면 Unload 동작을 수행합니다. FIFO 스택으로부터 1111 이 출력되고 PNT_INDEX 가 1 감소합니다.
(4) 입력접점 %IX1.1.15 가 On 되면 Reset 동작을 수행합니다. FIFO 스택의 모든 값이 0 으로 클리어(Clear)되고, PNT_INDEX 가 0 으로 초기화되며, EMTY_FLAG 가 On 됩니다.



LIFO

LIFO 스택에 값을 Load/Unload (후입 선출)

CPU 명	XGI	XGR
적용 가능	●	●

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

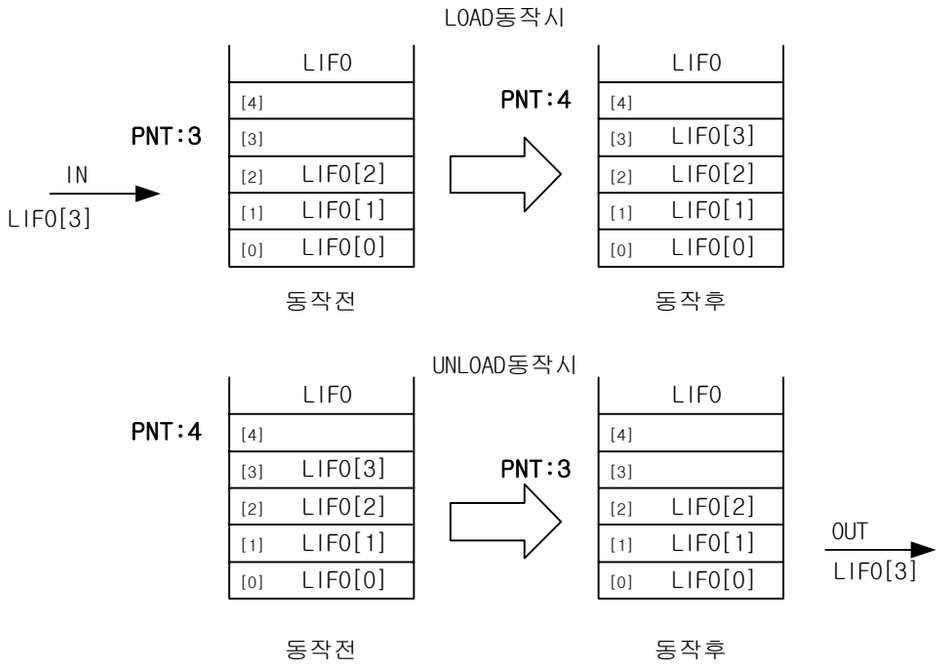
ANY 타입 변수설명	변수명	BOOL	BYTE	WORD	DWORD	LWORD	SINT	INT	DINT	LINT	USINT	UINT	UDINT	ULINT	REAL	LREAL	TIME	DATE	TOD	DT	STRING
	IN	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	LIFO	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	
	OUT	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	

*ANY: ANY 타입 중 STRING 제외, *ARRAY OF ANY: ARRAY OF ANY 타입 중 STRING 제외

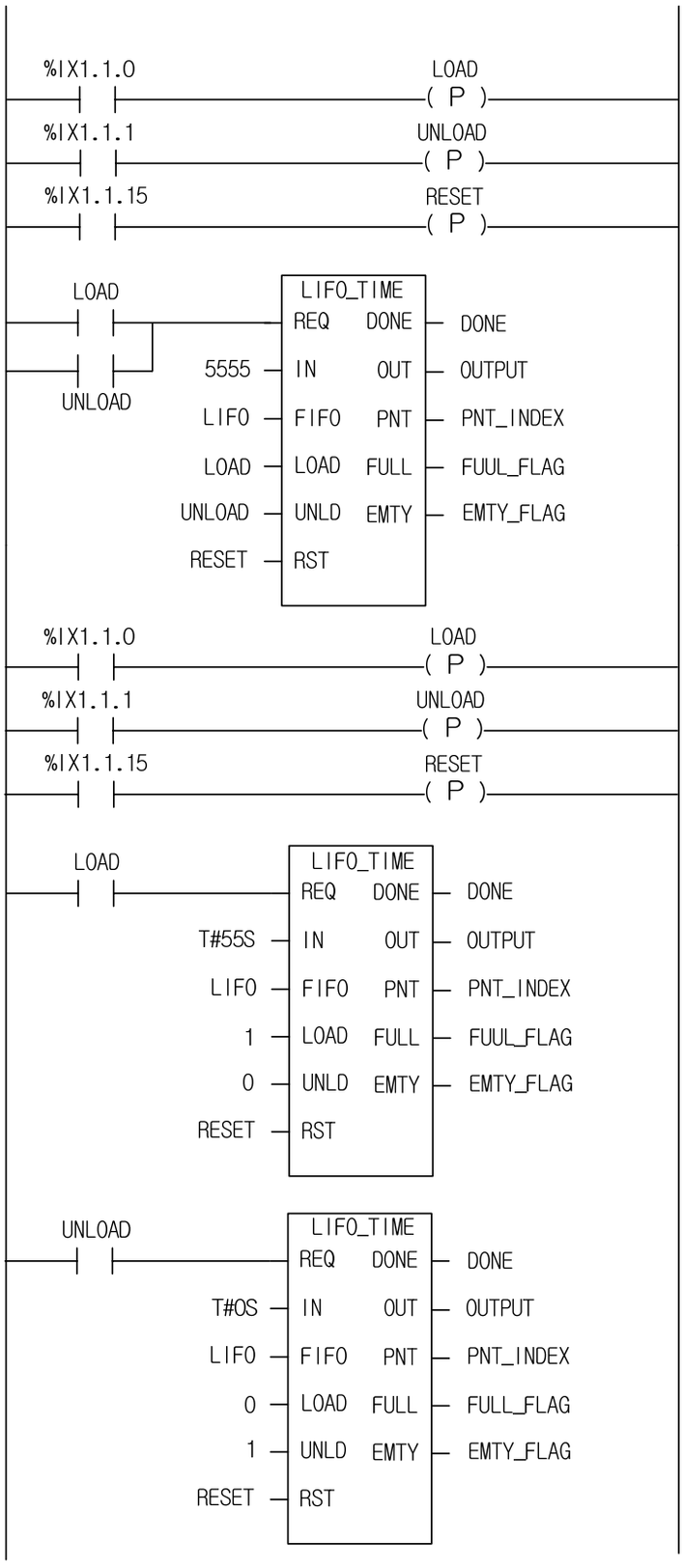
■ 기능

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

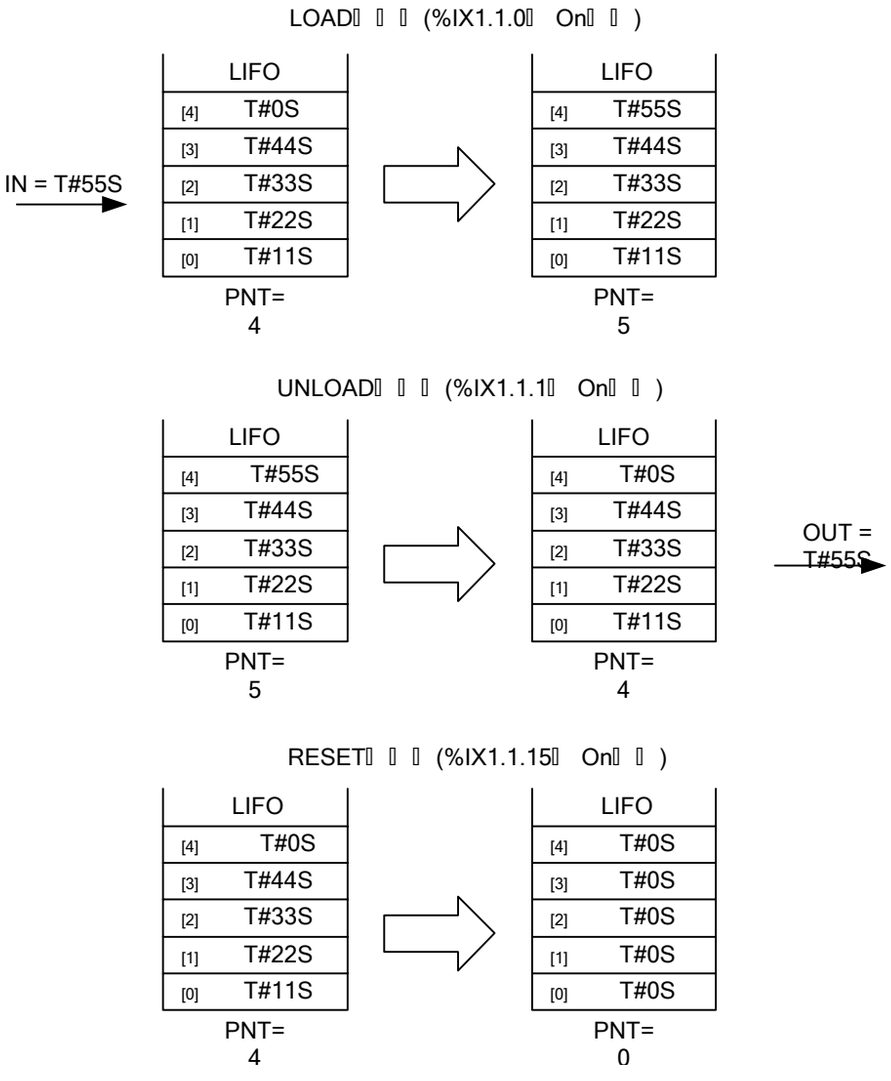
LIFO_TIME 펑션 블록은 위의 그림과 같이 2 가지 방법으로 사용이 가능합니다. 위에서 예를 든 2 가지 방법은 동일한 동작을 수행합니다. 위의 블록은 하나의 펑션 블록을 사용하여 입력과 출력을 동시에 수행할 수 있도록 한 프로그램이고, 아래의 블록은 입력을 위한 펑션 블록과 출력을 위한 펑션 블록을 따로 작성하여 입출력 동작을 다른 위치에서 동작하도록 작성한 프로그램입니다. 단 이때 주의할 점은 인스턴스 이름이 같도록 설정해야 합니다.

(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

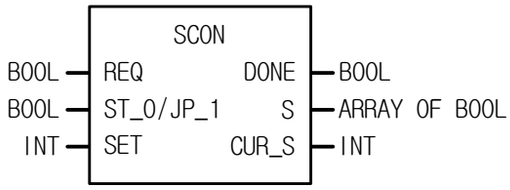


SCON

스텝 컨트롤러(순차스텝 및 스텝점프)

CPU 명	XGI	XGR
적용 가능	●	●

발생플래그	_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

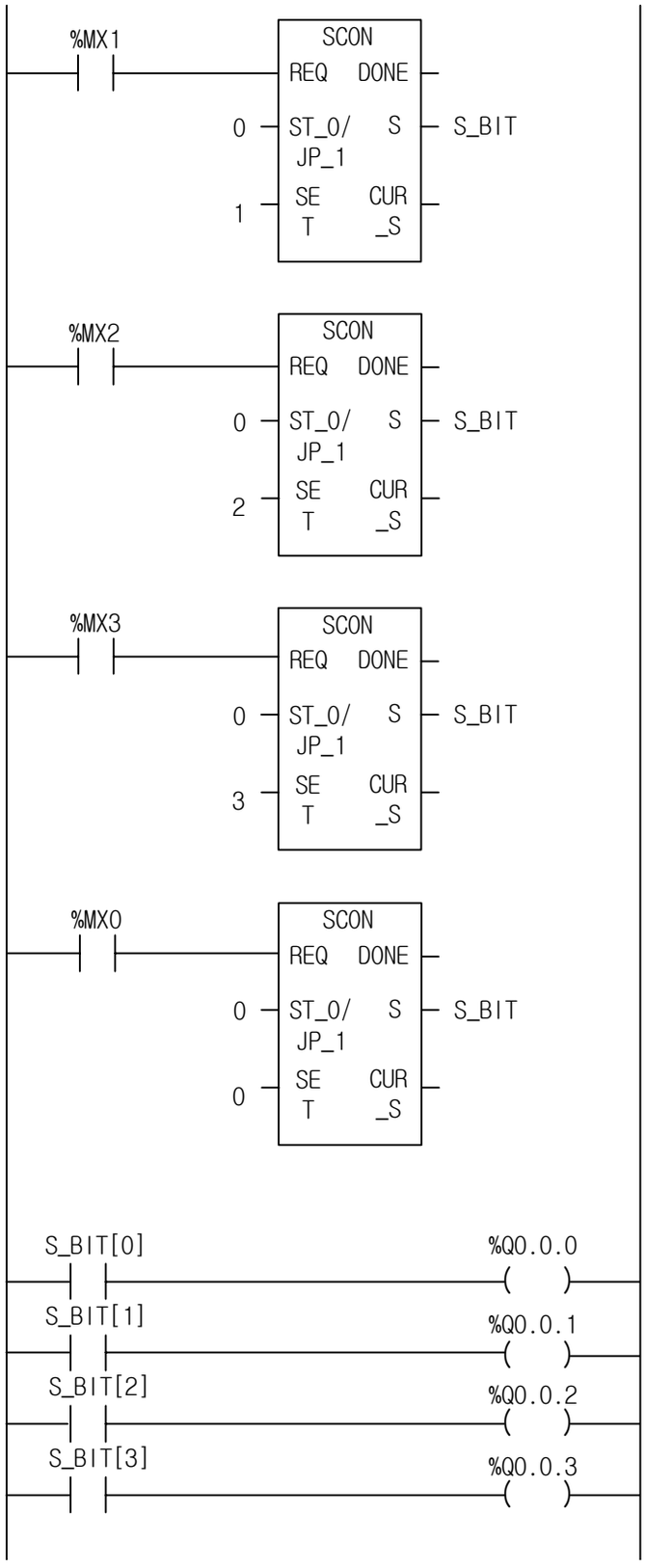
■ 기능

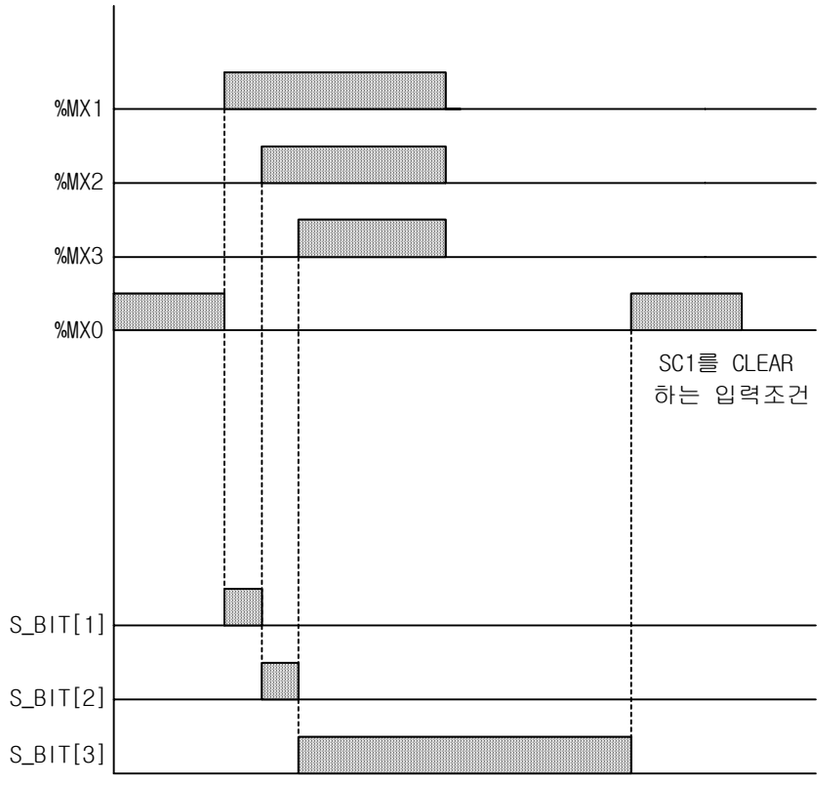
1. 순차작업 조의 설정
   펑션 블록의 인스턴스 이름이 하나의 순차작업 조의 이름이 됩니다.
   (펑션 블록 선언 예: S00, G01, 제조1, 스텝 접점 예: S00.S[1], G01.S[1], 제조1.S[1])
2. SET 동작일 경우(ST_0/JP_1 = 0)
   동일 조 내에서 바로 이전의 스텝 번호가 On 되었을 때 현재 스텝 번호가 On 됩니다.
   현재 스텝 번호가 On 되면 자기 유지되어 입력 접점이 Off 되어도 On 의 상태를 유지합니다.
   입력 조건 접점이 동시에 On 되어도 한 조 내에서는 한 스텝 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.

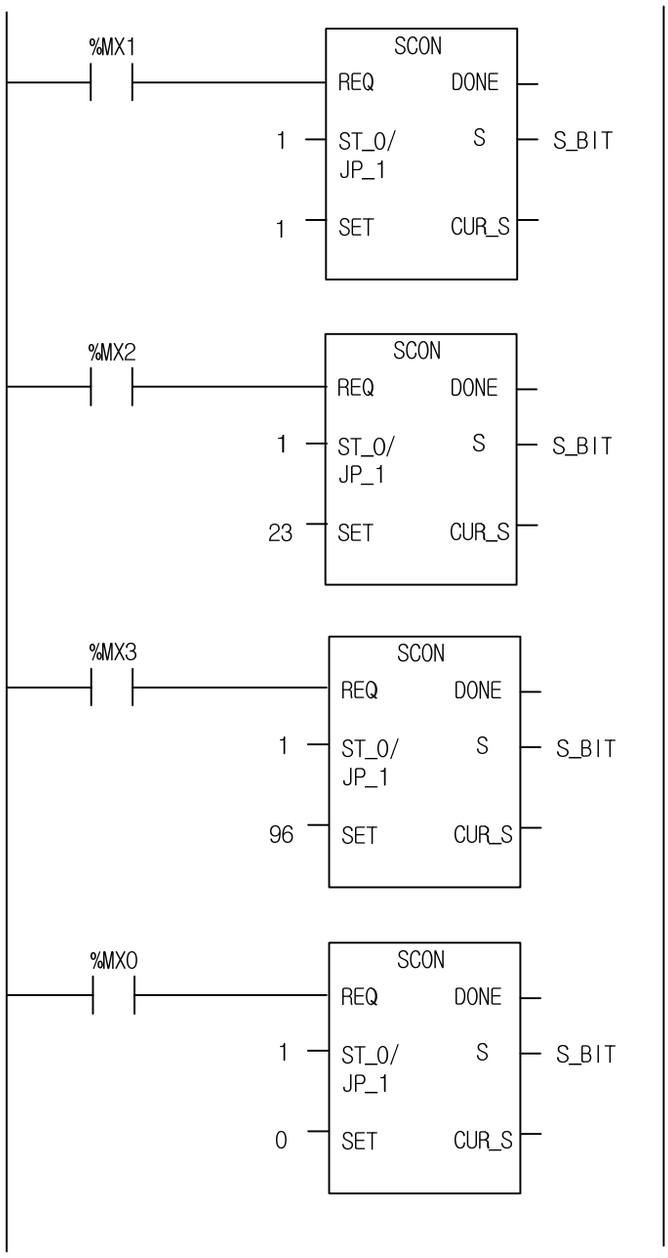
■ 플래그

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




%MX1
%MX2
%MX3
%MX0
SC1를 CLEAR
하는 입력조건
S_BIT[1]
S_BIT[2]
S_BIT[3]

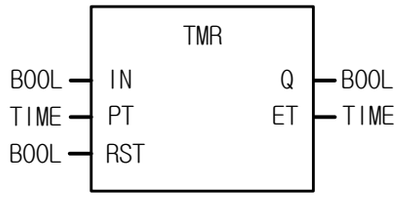


NO	%MX1	%MX2	%MX3	%MX4	S_0 [1]	S_0 [23]	S_0 [98]	S_0 [0]
1	On	Off	Off	Off	○			
2	On	On	Off	Off		○		
3	On	On	On	Off			○	
4	On	On	On	On				○

TMR

적산 타이머

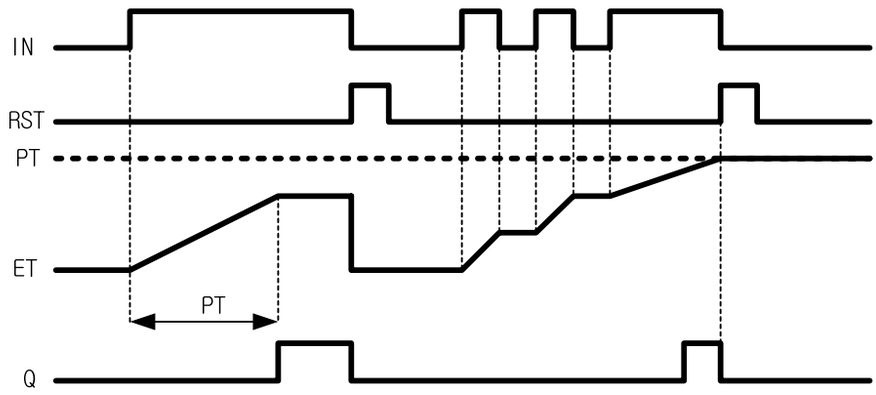
CPU 명	XGI	XGR
적용 가능	●	●

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)

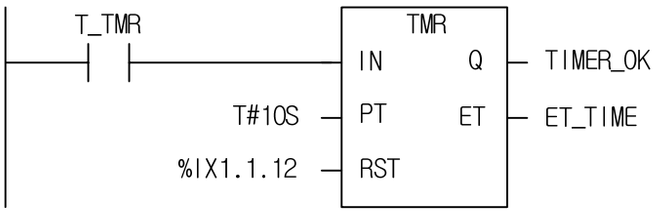
■ 기능

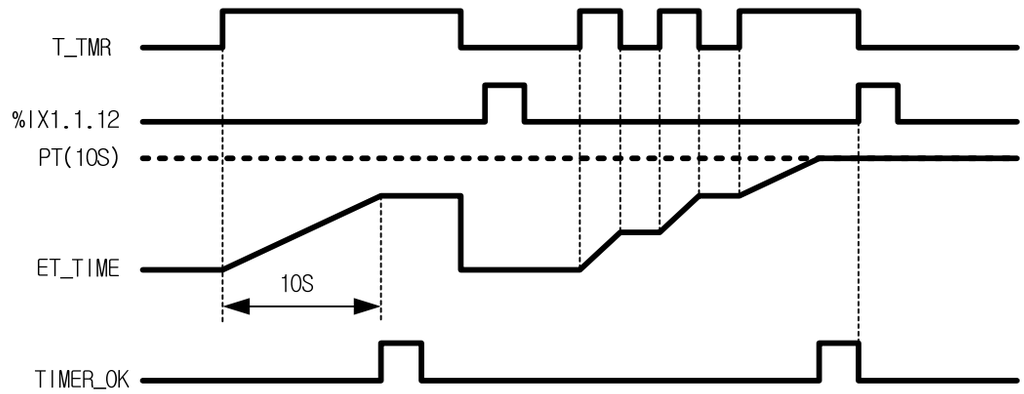
1. TMR 펑션 블록은 IN 이 1 이 된 후 경과 시간이 ET 로 출력됩니다.
2. 경과 시간 ET 가 설정시간에 도달하기 전에 IN 이 0 이 되어도 현재의 경과 시간을 유지하다가 IN 이 다시 1 이 되면 경과 시간을 다시 증가시킵니다.
3. 경과 시간이 설정 시간에 도달하면 Q 가 1 이 됩니다.
4. Reset 입력 조건이 성립되면 Q 는 0 이 되며 경과 시간도 0 이 됩니다.

■ 타임 차트



■ 프로그램 예





(1) 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.

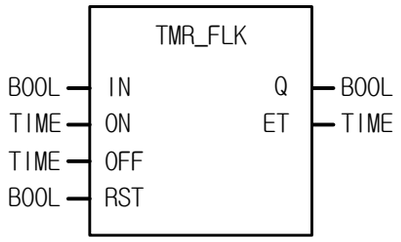
(4) 입력 변수 T_TMR 이 다시 1 이 되면 정지 이전의 경과 시간부터 다시 시작합니다.

(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

점멸기능 타이머

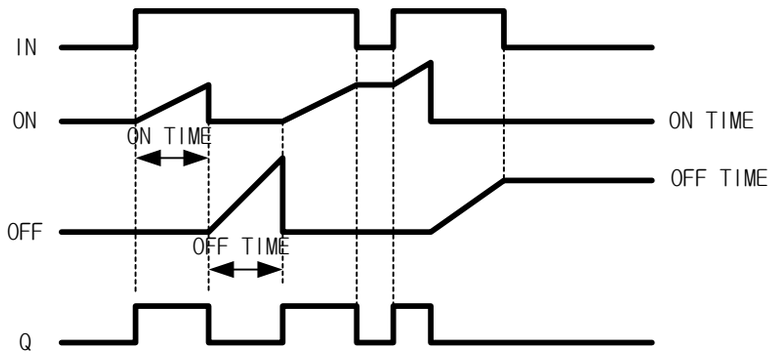
CPU 명	XGI	XGR
적용 가능	●	●

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)

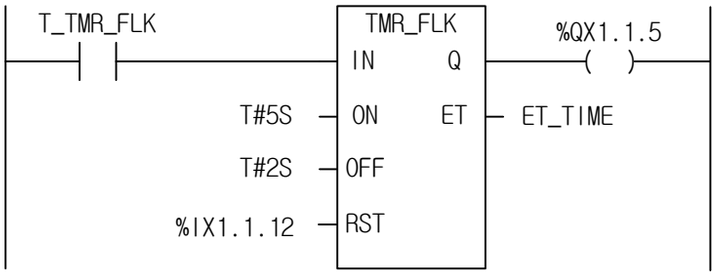
■ 기능

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 Q는 0이 됩니다.
5. ON이 0이면 출력 Q는 항상 0이 됩니다.

■ 타임 차트



■ 프로그램 예



(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.

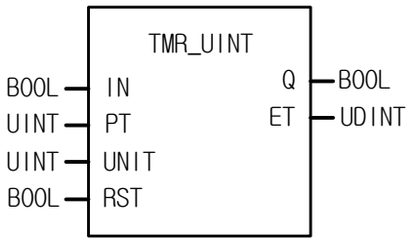
(5) 입력변수 T_TMR_FLK 가 0 이 되면 동작 중인 시간을 유지하고 출력접점 %QX1.1.5 는 0 이 되며, T_TMR_FLK 가 다시 1 이 되면 정지되었던 시간부터 다시 동작합니다.

(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

정수 설정 적산 타이머

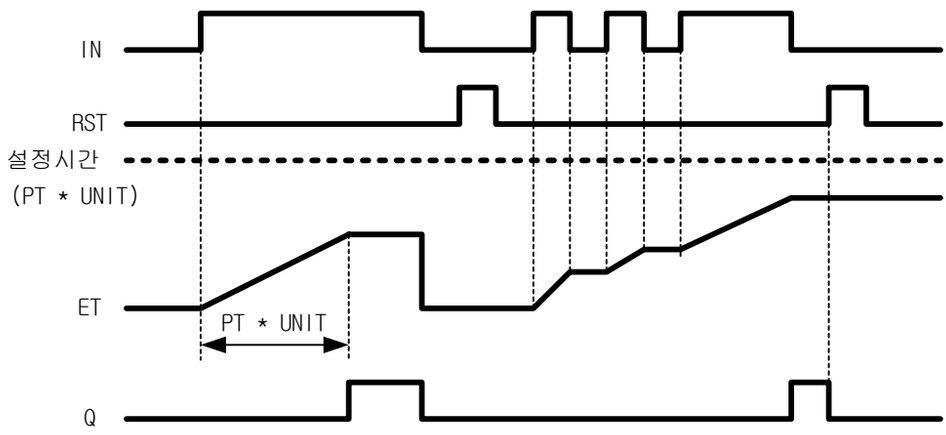
CPU 명	XGI	XGR
적용 가능	●	●

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)

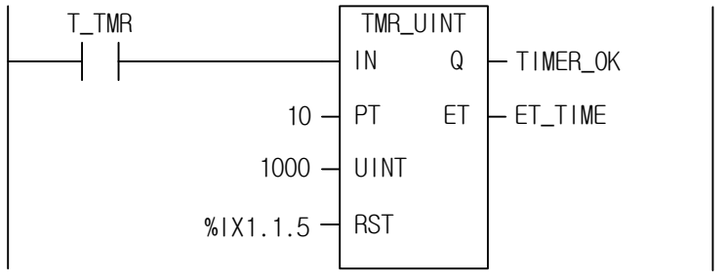
■ 기능

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) 설정 시간은 PT * UNIT[ms] = 10 * 1000[ms] = 10[s]가 됩니다.

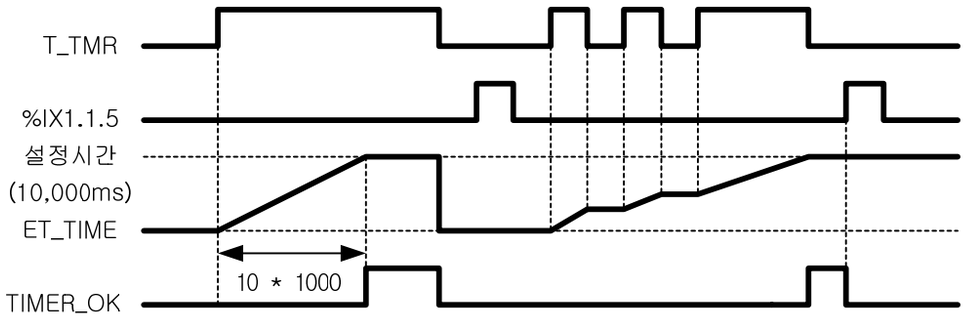
(2) 입력변수 T_TMR이 1이 된 후 10초가 경과하면 출력 변수 TIMER_OK가 1이 됩니다.

(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.

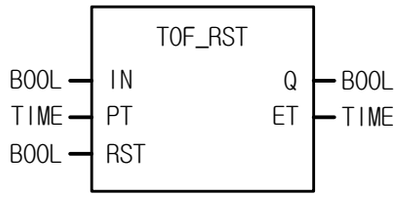
(6) 입력 접점 %IX1.1.5가 1이 되면 경과 시간 ET_TIME 및 출력 변수 TIMER_OK 모두 0으로 클리어(Clear)됩니다.



TOF_RST

동작 중 출력 Off가 가능한 딜레이 타이머

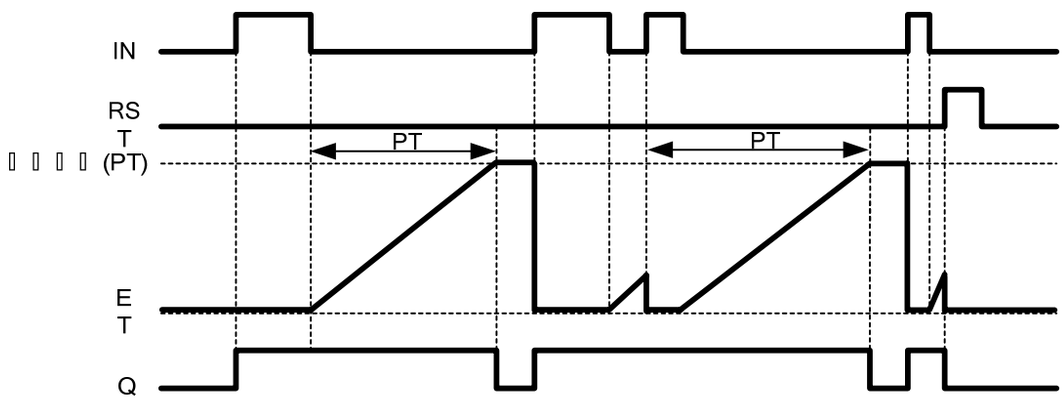
CPU 명	XGI	XGR
적용 가능	●	●

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)

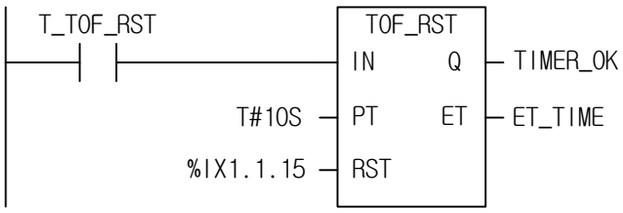
■ 기능

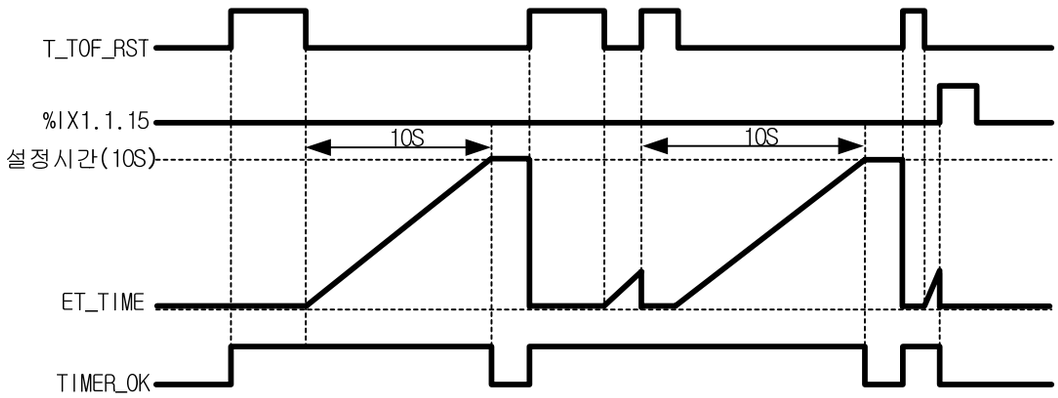
1. TOF_RST 펑션 블록은 기동 조건 IN이 1이 되는 순간 Q는 1이 되고, IN이 0이 된 후부터 PT에 의하여 지정된 설정 시간이 경과한 후 Q가 0이 됩니다.
2. IN이 0이 된 후 경과 시간이 ET로 출력됩니다.
3. 만일 경과시간 ET가 설정 시간에 도달하기 전에 IN이 1이 되면, 경과 시간은 다시 0으로 됩니다.
4. Reset 입력 조건이 성립하면 타이머 출력 Q는 0이 되고 경과 시간도 0이 됩니다.

■ 타임 차트



■ 프로그램 예





(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.

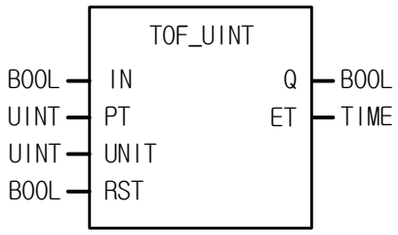
배열인덱스를 이용한 변수를 사용했을 경우 배열인덱스 에러는 접점이 On 이 되었을 때만 발생합니다.

때문에 TOF_RST 펑션 블록에서는 펑션 블록이 동작하더라도 접점이 Off 되어 있다면 배열인덱스 에러가 발생하지 않습니다.

TOF_UINT

정수 설정 Off 타이머

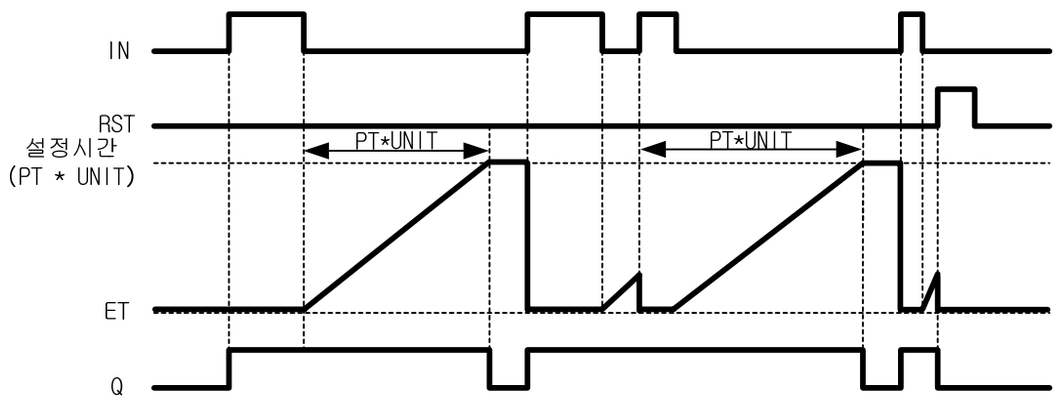
CPU 명	XGI	XGR
적용 가능	●	●

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)

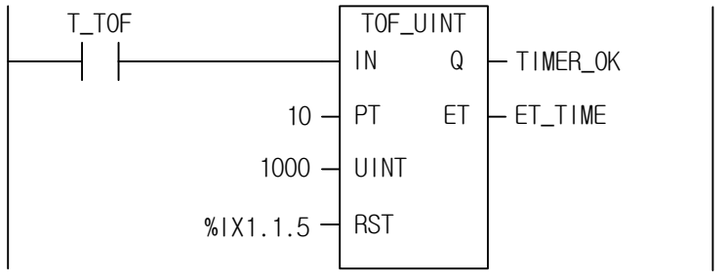
■ 기능

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



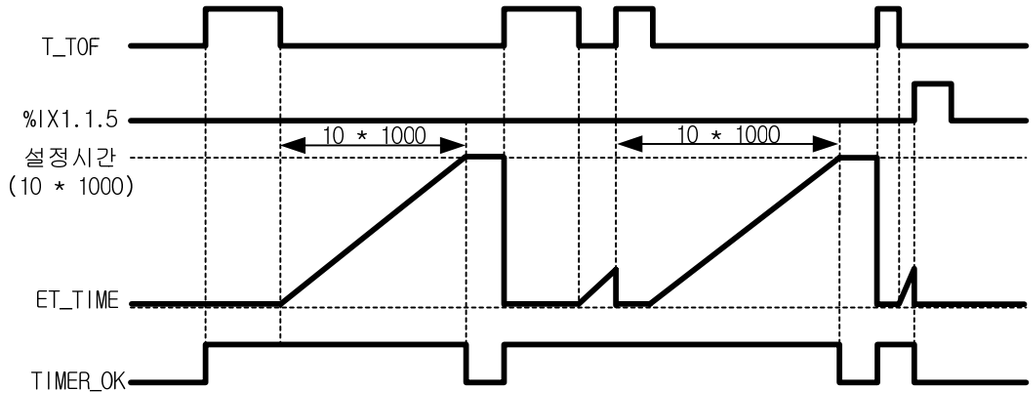
(1) 설정 시간은 PT * UNIT[ms] = 10 * 1000[ms] = 10[s]가 됩니다.

(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.

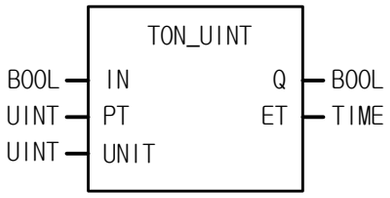
배열인덱스를 이용한 변수를 사용했을 경우 배열인덱스 에러는 접점이 On 이 되었을 때만 발생합니다.

때문에 TOF_UINT 펑션 블록에서는 펑션 블록이 동작하더라도 접점이 Off 되어 있다면 배열인덱스 에러가 발생하지 않습니다.

TON_UINT

정수 설정 On 타이머

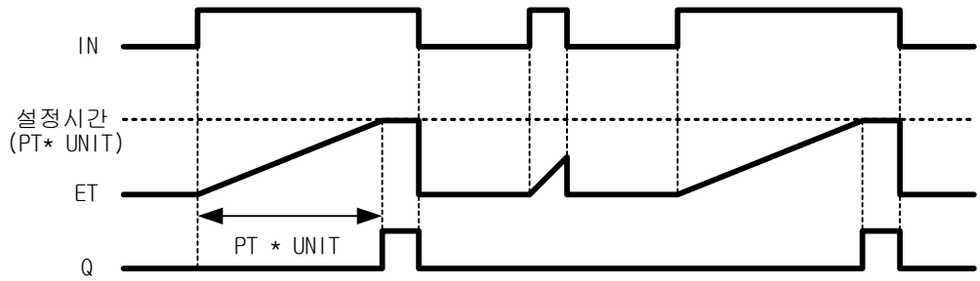
CPU 명	XGI	XGR
적용 가능	●	●

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)

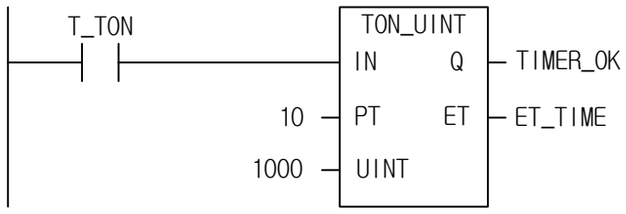
■ 기능

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



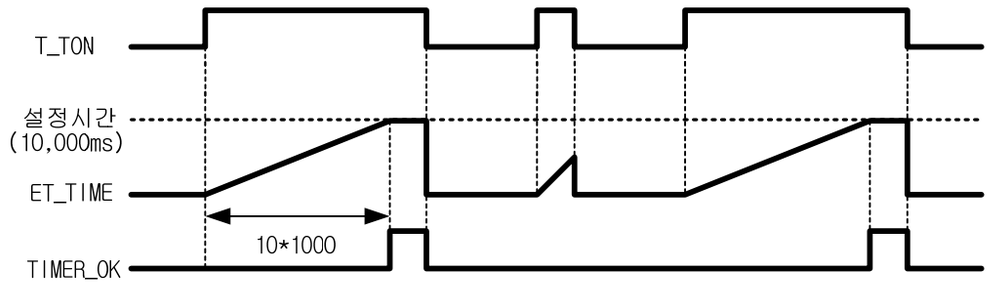
(1) 설정 시간은 PT * UNIT[ms] = 10 * 1000[ms] = 10[s]가 됩니다.

(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.

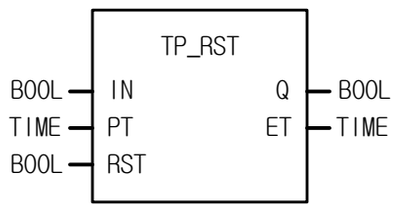
(5) TIMER_OK 가 1 이 된 후 T_TON 이 0 이 되면, TIMER_OK 는 0 이 되고 경과 시간 ET_TIME 도 0 이 됩니다.



TP_RST

접점 출력 Off가 가능한 펄스 타이머

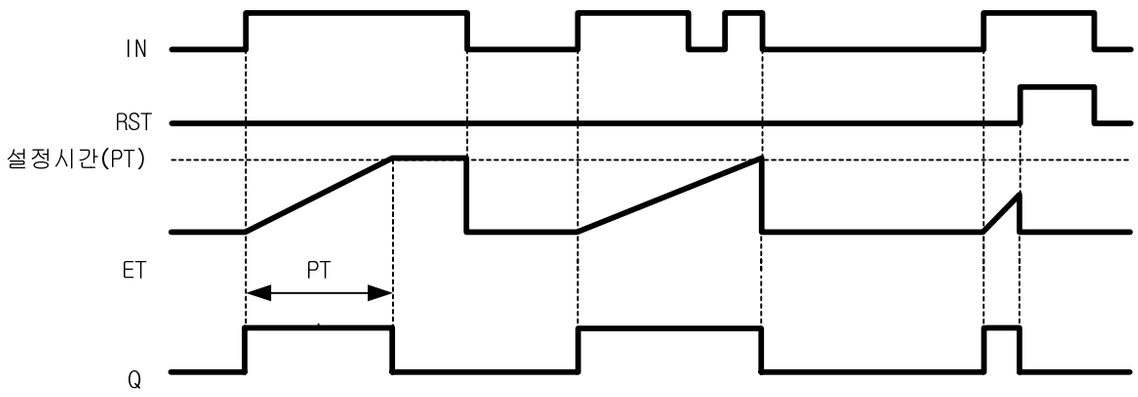
CPU 명	XGI	XGR
적용 가능	●	●

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)

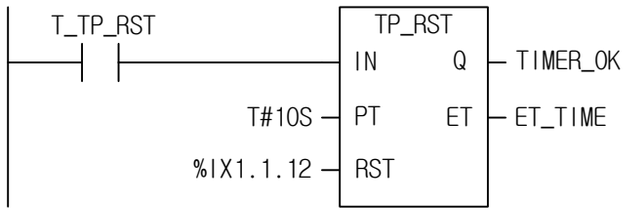
■ 기능

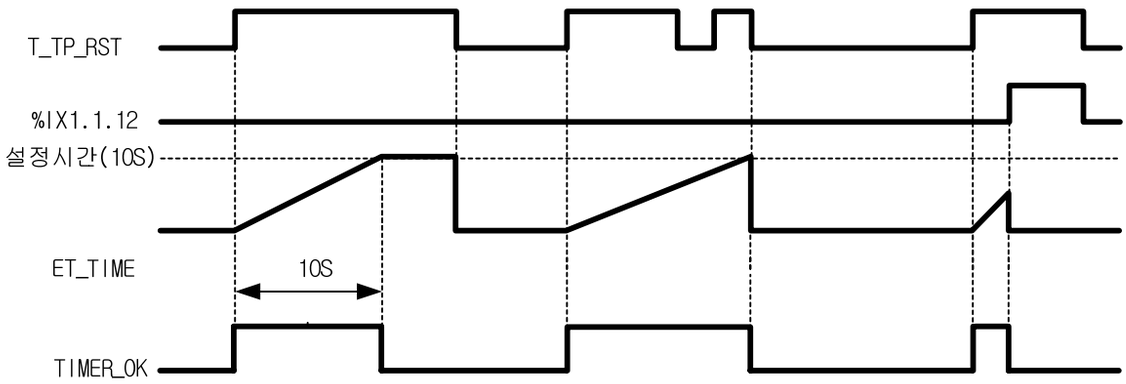
1. TP_RST 펑션 블록은 IN이 1이 되는 순간 Q는 1이 되고, 경과 시간이 설정 시간에 도달하면 타이머 출력 Q는 0이 됩니다.
2. 경과 시간 ET는 IN이 1이 되었을 때부터 증가하며 PT에 이르면 값을 유지하다가 IN이 0이 될 때 0으로 클리어 (clear) 됩니다.
3. 타이머 출력 Q가 1인 동안(펄스 출력 중)에는 타이머 기동조건 IN이 1, 0 변화를 하여도 무시합니다.
4. Reset 입력 조건이 성립하면 펄스 출력 중에도 타이머 출력 Q는 0이 되고 경과 시간도 0이 됩니다.

■ 타임 차트



■ 프로그램 예





(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 TIMER_OK 와 ET_TIME 모두 0 으로 클리어(Clear)됩니다.

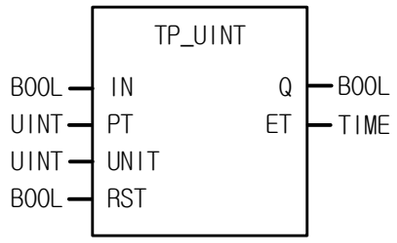
☆ 참고

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

정수 설정 펄스 타이머

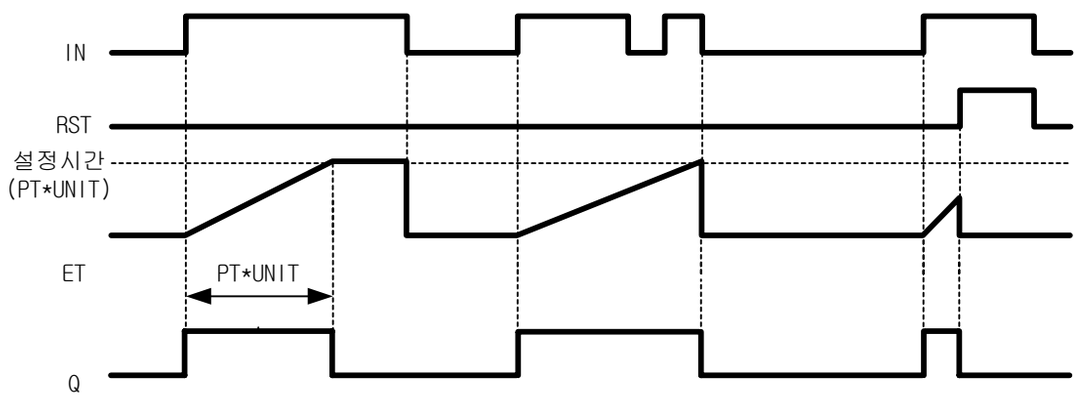
CPU 명	XGI	XGR
적용 가능	●	●

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)

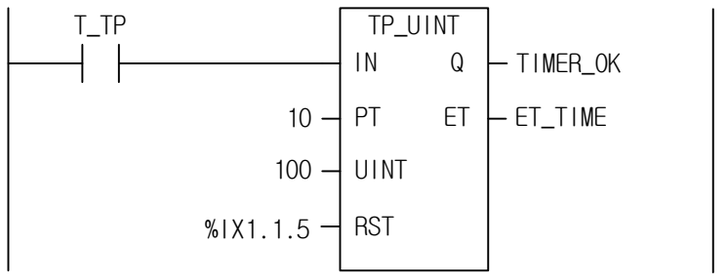
■ 기능

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

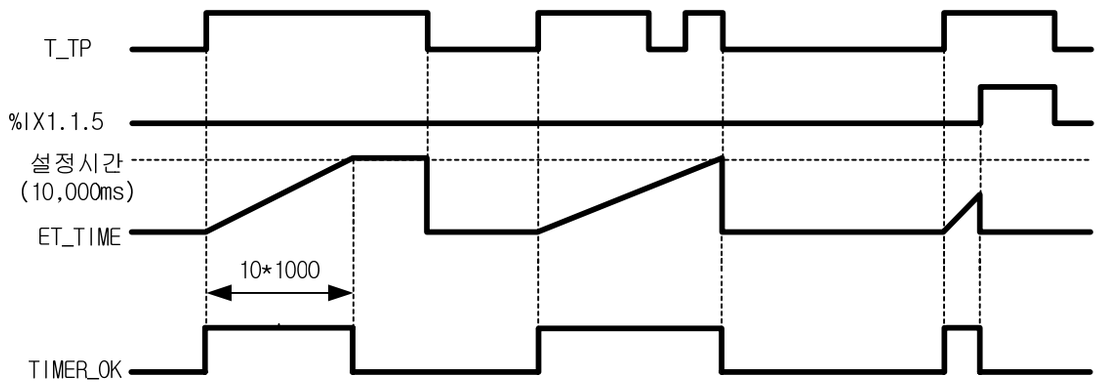


(1) 설정 시간은 PT * UNIT[ms] = 10 * 100[ms] = 1[s]가 됩니다.

(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.



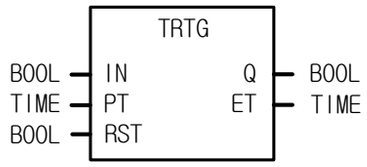
☆ 참고

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

리트리거블 타이머

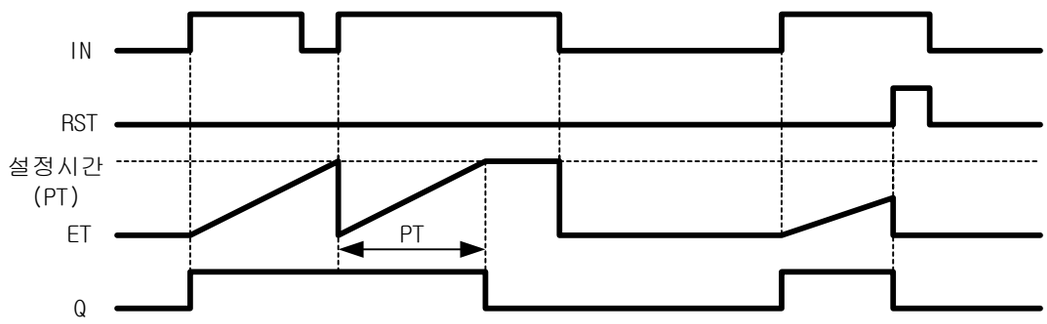
CPU 명	XGI	XGR
적용 가능	●	●

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)

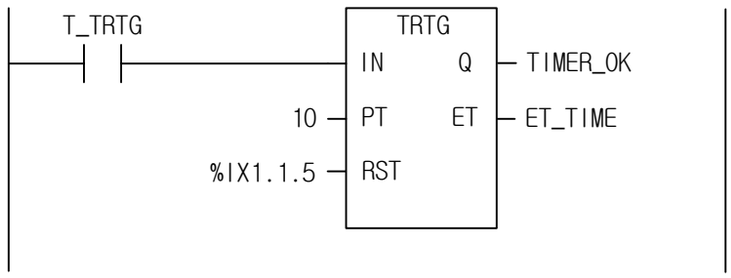
■ 기능

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

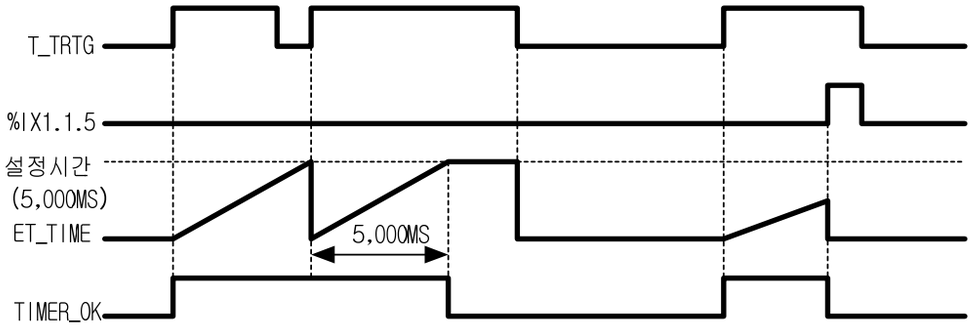


(1) 입력변수 T_TRTG 가 0 에서 1 이 된 후 10 초 동안 TIMER_OK 는 1 이 됩니다. 타이머가 가동된 후 T_TRTG 가 다시 0 에서 1 이 되면 ET_TIME 은 0 부터 다시 시작됩니다.

(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.



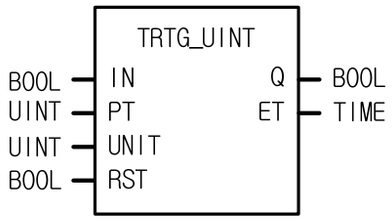
☆ 참고

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

정수 설정 리트리거블 타이머

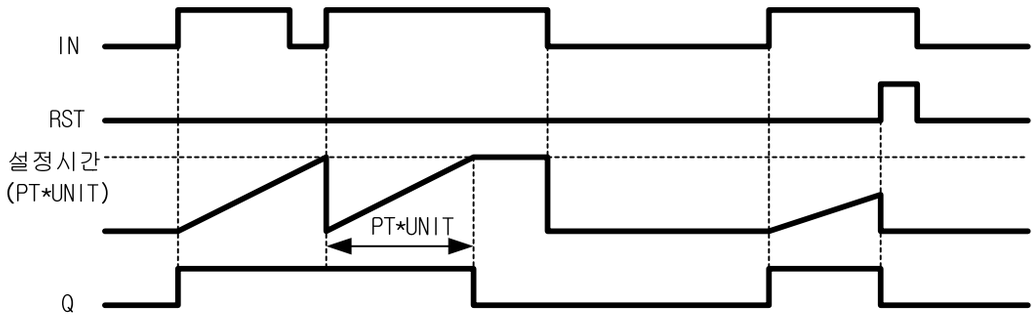
CPU 명	XGI	XGR
적용 가능	●	●

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)

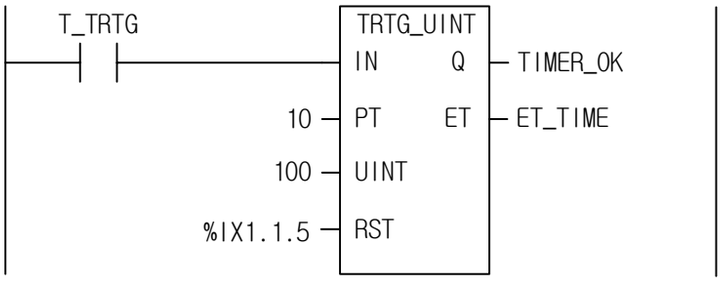
■ 기능

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



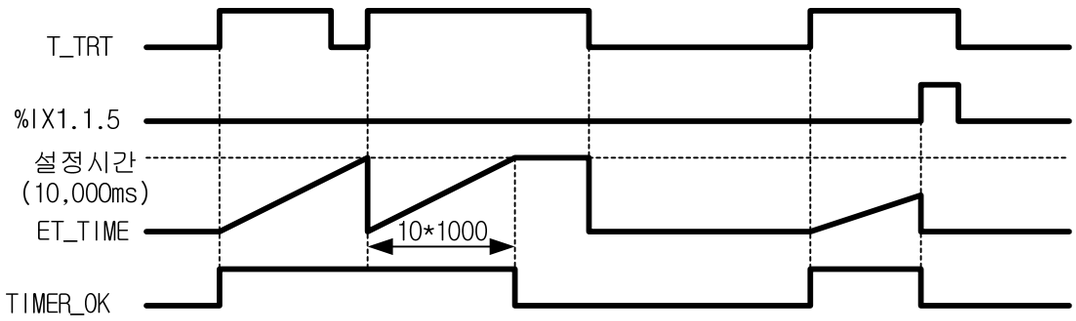
(1) 설정 시간은 PT * UNIT[ms] = 10 * 1000[ms] = 10[s]가 됩니다.

(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.

제 11 장 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.

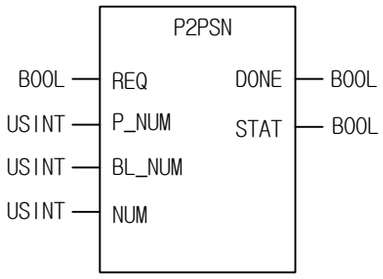
11.1 통신 펑션 블록

1. 각각의 통신 펑션 블록에 대한 설명입니다.

P2PSN

국번 설정

CPU 명	XGI	XGR
적용 가능	●	●

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

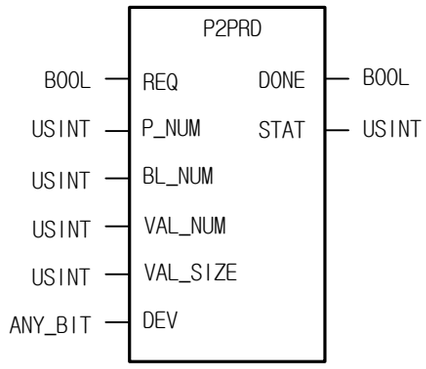
1. 에러 발생 시 STAT에 해당 에러 번호를 표시합니다.

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

P2PRD

읽기 영역 지정

CPU 명	XGI	XGR
적용 가능	●	●

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.)
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

PD 에서 설정된 P2P 파라미터의 허용 범위를 벗어난 설정을 하는 경우 해당 에러 번호가 발생합니다.

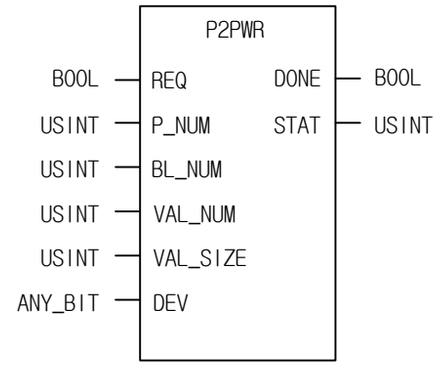
STAT	내용	세부 설명
1	P2P번호 설정 에러	P_NUM(1~8)이외의 값 설정 시 발생
2	블록 번호 설정 에러	BL_NUM(0~63)이외의 값 설정 시 발생 <단, Cnet 인 경우 0~31>
3	변수 번호 설정 에러	PD에서 설정된 P2P 파라미터에 허용되지 않는 변수 번호 입력 시 발생
4	슬롯 존재 안함	-
5	모듈 불일치	통신 모듈 아님
6	모듈 불일치	해당 명령어에 사용할 수 없는 통신모듈

STAT	내용	세부 설명
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

P2PWR

쓰기 영역 지정

CPU 명	XGI	XGR
적용 가능	●	●

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

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	-
5	모듈 불일치	통신 모듈 아님
6	모듈 불일치	해당 명령어에 사용할 수 없는 통신모듈

STAT	내용	세부 설명
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

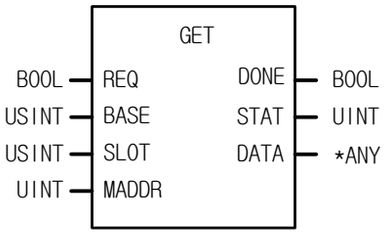
11.2 특수 펑션 블록

1. 각각의 특수 펑션 블록에 대한 설명입니다.

GET

특수 모듈 데이터 읽기

CPU 명	XGI	XGR
적용 가능	●	●

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

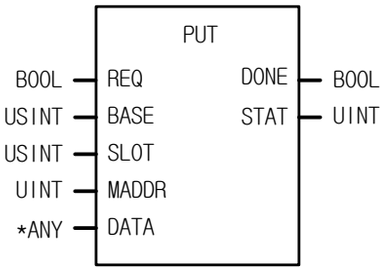
1. 지정한 특수 모듈로부터 데이터를 읽어 옵니다.

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.

PUT

특수 모듈에 데이터 쓰기

CPU 명	XGI	XGR
적용 가능	●	●

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

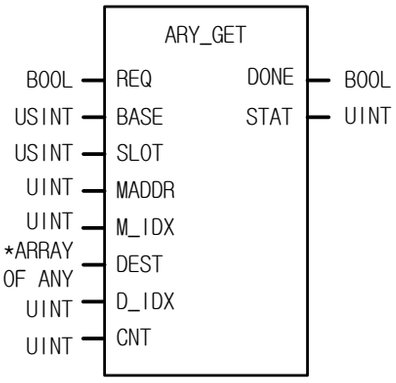
1. 지정한 특수 모듈로부터 데이터를 읽어 옵니다.

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.

ARY_GET

특수 모듈 데이터 읽기(어레이)

CPU 명	XGI	XGR
적용 가능	●	●

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

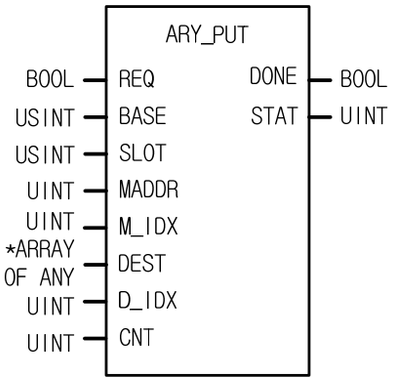
1. 지정한 특수 모듈로부터 데이터를 읽어 옵니다.

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.

ARY_PUT

특수 모듈 데이터 쓰기(어레이)

CPU 명	XGI	XGR
적용 가능	●	●

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.

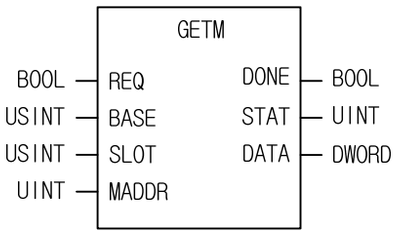
11.3 모션 제어 펑션 블록

1. 모션 제어 펑션 블록에 대한 설명입니다.

GETM

모션 제어 모듈 데이터 읽기

CPU 명	XGI	XGR
적용 가능	●	●

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

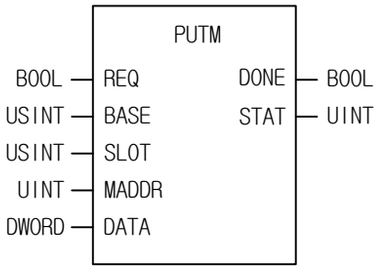
1. 지정한 모션 제어 모듈의 읽기 공유 메모리 주소 MADDR 에서 데이터를 읽어옵니다.

펑션 블록	출력(DATA) 타입	동작 설명
GETM_DWORD	DWORD	지정한 모듈 어드레스(MADDR)부터 DWORD 만큼 데이터를 읽어옵니다.

PUTM

특수 모듈(모션모듈)에 데이터 쓰기

CPU 명	XGI	XGR
적용 가능	●	●

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

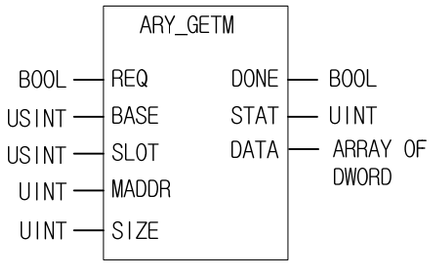
1. 지정한 모션 제어 모듈의 쓰기 공유 메모리 주소 MADDR에 데이터를 저장합니다.

펑션 블록	DATA 타입	동작 설명
PUT_DWORD	DWORD	지정한 모듈 어드레스(MADDR)에 DWORD 데이터를 저장합니다.

ARY_GETM

모션 제어 모듈 데이터 읽기(어레이)

CPU 명	XGI	XGR
적용 가능	●	●

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)

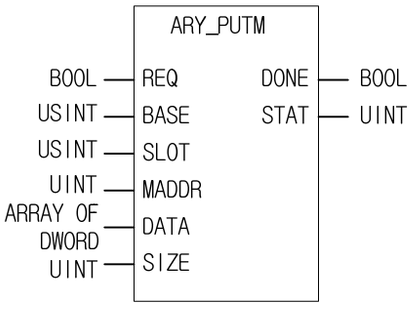
■ 기능

1. 지정한 모션 제어 모듈의 읽기 공유 메모리 MADDR 에서 SIZE 크기만큼 데이터를 읽어옵니다.

ARY_PUTM

모션 제어 모듈 데이터 쓰기(어레이)

CPU 명	XGI	XGR
적용 가능	●	●

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.

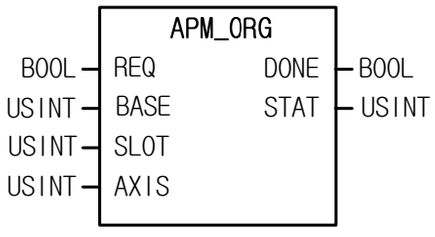
11.4 위치결정 펑션 블록

	이름	상세 설명	동작 조건	수행 시간 (ms)
1	ORG	원점 복귀 기동	Edge	5
2	FLT	부동 원점 설정	Edge	5
3	DST	직접 기동	Edge	5
4	IST	간접 기동	Edge	5
5	LIN	직선 보간 기동	Edge	5
6	CIN	원호 보간 기동	Edge	5
7	SST	동시 기동	Edge	5
8	VTP	속도/위치 제어 전환	Edge	5
9	PTV	위치/속도 제어 전환	Edge	5
10	STP	감속 정지	Edge	5
11	SKP	스킵 운전	Edge	5
12	SSP	위치 동기	Edge	5
13	SSS	속도 동기	Edge	5
14	POR	위치 오버라이드	Edge	5
15	SOR	속도 오버라이드	Edge	5
16	PSO	위치 지정 속도 오버라이드	Edge	5
17	NMV	연속 운전	Edge	5
18	INC	인칭 기동	Edge	5
19	RTP	수동 운전 이전 위치로 복귀	Edge	5
20	SNS	기동 스텝 번호 변경	Edge	5
21	SRS	반복 스텝 번호 변경	Edge	5
22	MOF	M 코드 해제	Edge	5
23	PRS	현재 위치 프리셋	Edge	5
24	ZONE	Zone 출력 허용/금지	Edge	5
25	EPRE	엔코더값 프리셋	Edge	5
26	TEA	단독 티칭	Edge	5
27	ATEA	복수 티칭	Edge	5
28	SBP	기본 파라미터 티칭	Edge	5
29	SEP	확장 파라미터 티칭	Edge	5
30	SHP	원점 복귀 파라미터 티칭	Edge	5
31	SMP	수동 운전 파라미터 티칭	Edge	5
32	SIP	외부 신호 파라미터 티칭	Edge	5
33	SCP	공통 파라미터 티칭	Edge	5
34	SMD	운전 데이터 티칭	Edge	5
35	EMG	비상 정지	Edge	5
36	RST	에러 리셋/출력 금지 해제	Edge	5
37	PST	포인트 운전	Edge	5
38	WRT	파라미터/운전 데이터 저장	Edge	1,000
39	CRD	운전 정보 읽기	Level	0.02
40	SRD	운전 상태 읽기	Level	0.02
41	ENCRD	엔코더값 읽기	Level	0.02
42	JOG	조그 운전	Level	5
43	MPG	수동 펄스 발생기(MPG) 운전	Edge	5

APM_ORG

원점 복귀 기동

CPU 명	XGI	XGR
적용 가능	●	●

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

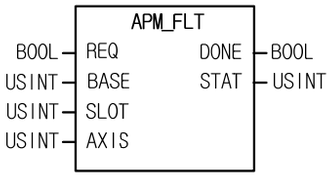
1. 이 명령은 위치결정 모듈에 원점 복귀 기동을 실행하는 명령입니다.
2. 각 축의 원점 복귀 파라미터에 설정된 방향, 보정량, 속도(고속, 저속), 어드레스 및 드웰 시간 등으로 원점을 찾는 운전 지령입니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 원점 복귀 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러6” 이 발생합니다.

   0: X 축, 1: Y 축, 2: Z 축

APM_FLT

부동 원점 설정

CPU 명	XGI	XGR
적용 가능	●	●

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

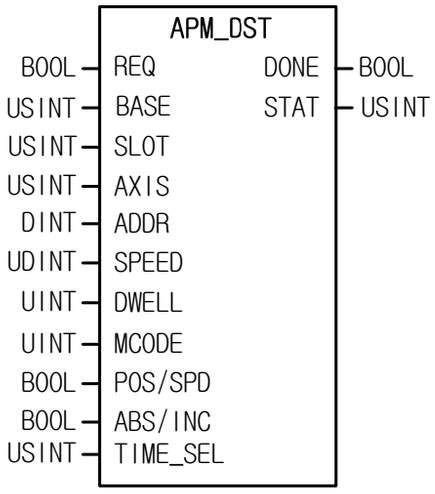
■ 기능

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

APM_DST

직접 기동

CPU 명	XGI	XGR
적용 가능	●	●

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

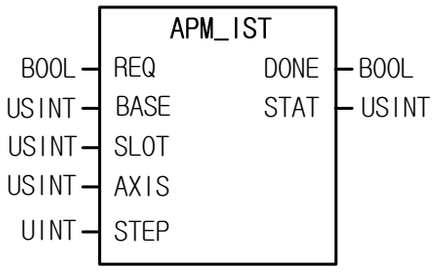
■ 기능

1. 이 명령은 위치결정 모듈에 직접 기동 실행하는 명령입니다.
2. 운전 데이터로 설정된 축의 운전 스탭 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 직접기동 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
5. SPEED, DWELL, TIME_SEL에 설정된 값이 설정범위를 넘을 경우 STAT에 "에러11"이 발생합니다.

APM_IST

간접 기동

CPU 명	XGI	XGR
적용 가능	●	●

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

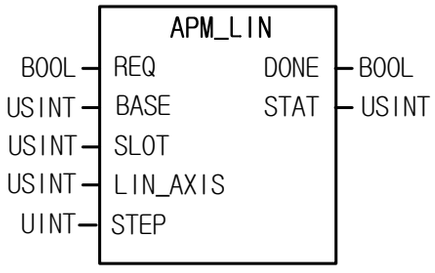
■ 기능

1. 이 명령은 위치결정 모듈에 직접 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
5. STEP에 설정된 값이 설정 범위(0 ~ 400)를 넘을 경우 STAT에 "에러11"이 발생합니다.
6. STEP에 0을 설정하면 현재 스텝을 운전합니다.

APM_LIN

직선 보간 기동

CPU 명	XGI	XGR
적용 가능	●	●

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.

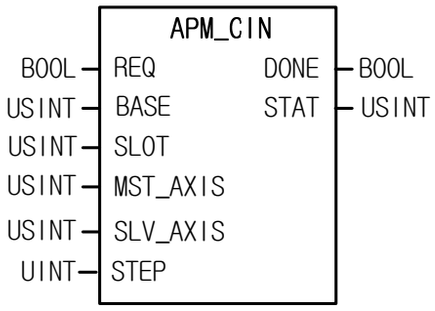
15 ~ 4	2	1	0
-	Z축	Y축	X축

5. STEP에 설정된 값이 설정 범위(0 ~ 400)를 넘을 경우 STAT에 “에러11”이 발생합니다.
6. STEP에 0을 설정하면 현재 스텝을 운전합니다.

APM_CIN

원호 보간 기동

CPU 명	XGI	XGR
적용 가능	●	●

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

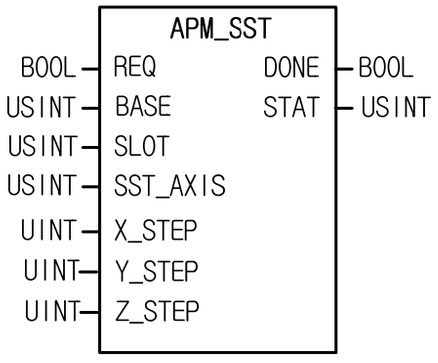
■ 기능

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.

APM_SST

동시기동

CPU 명	XGI	XGR
적용 가능	●	●

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 1을 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.

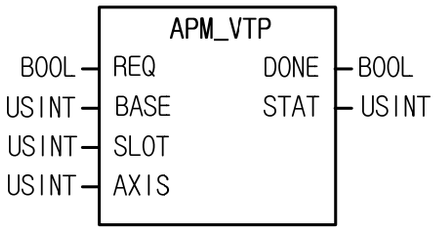
15 ~ 4	2	1	0
-	Z축	Y축	X축

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)를 넘을 경우 STAT에 "에러11"이 발생합니다.
7. X_STEP, Y_STEP, Z_STEP에 0을 설정하면 각 축의 현재 스텝을 운전합니다.

APM_VTP

속도/위치 제어 변경

CPU 명	XGI	XGR
적용 가능	●	●

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

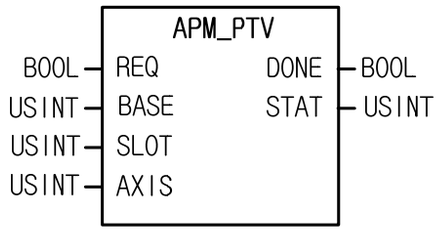
■ 기능

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

APM_PTV

위치/속도 제어 전환

CPU 명	XGI	XGR
적용 가능	●	●

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

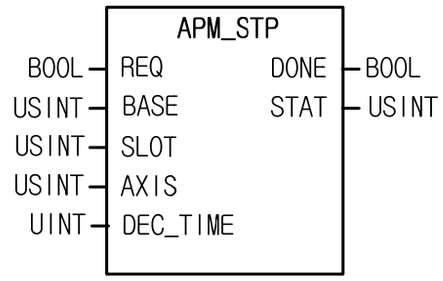
■ 기능

1. 이 명령은 위치결정 모듈에 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 "에러6"이 발생합니다.
   0: X축, 1: Y축, 2: Z축

APM_STP

감속 정지

CPU 명	XGI	XGR
적용 가능	●	●

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 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

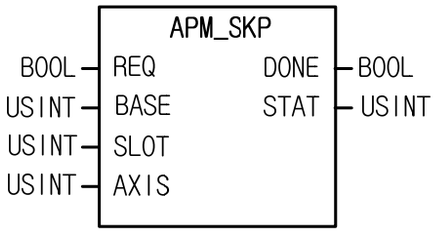
■ 기능

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

APM_SKP

스킵 운전

CPU 명	XGI	XGR
적용 가능	●	●

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

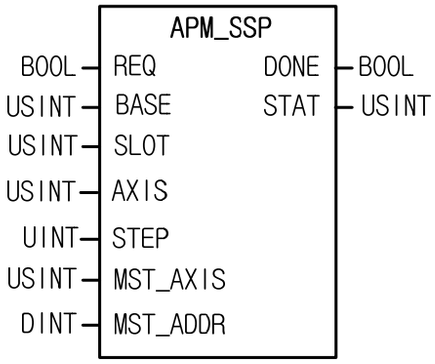
■ 기능

1. 이 명령은 위치결정 모듈에 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

APM_SSP

위치 동기

CPU 명	XGI	XGR
적용 가능	●	●

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

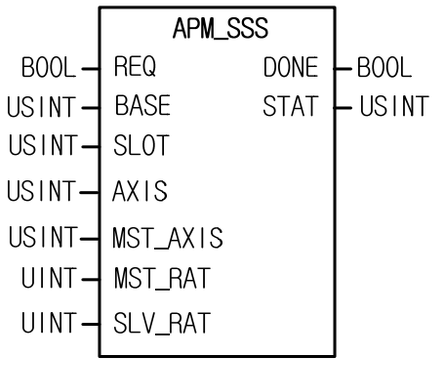
■ 기능

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 운전합니다.
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
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

APM_SSS

속도 동기

CPU 명	XGI	XGR
적용 가능	●	●

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

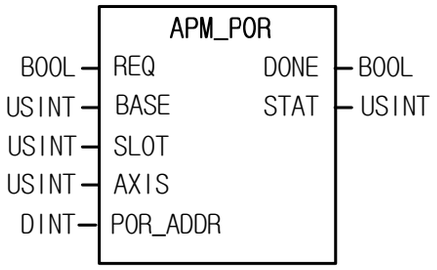
■ 기능

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

APM_POR

위치 오버라이드

CPU 명	XGI	XGR
적용 가능	●	●

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

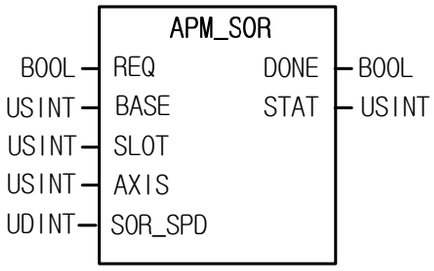
■ 기능

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
5. POR_ADDR 에는 변경할 목표 위치를 설정합니다.

APM_SOR

속도 오버라이드

CPU 명	XGI	XGR
적용 가능	●	●

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 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

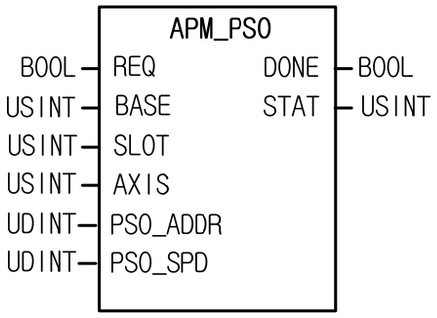
■ 기능

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
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]
   Line Driver: 0 ~ 1,000,000[pps]

APM_PSO

위치 지정 속도 오버라이드

CPU 명	XGI	XGR
적용 가능	●	●

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 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

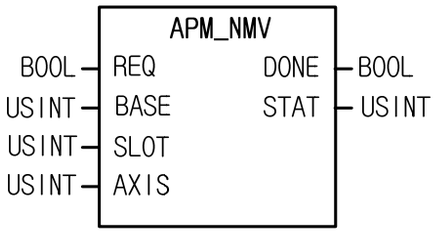
■ 기능

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
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]
   Line Driver: 0 ~ 1,000,000[pps]

APM_NMV

연속 운전

CPU 명	XGI	XGR
적용 가능	●	●

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

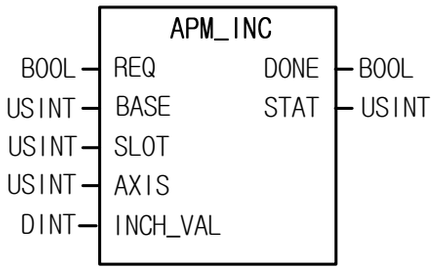
■ 기능

1. 이 명령은 위치결정 모듈에 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

APM_INC

인칭 운전

CPU 명	XGI	XGR
적용 가능	●	●

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 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

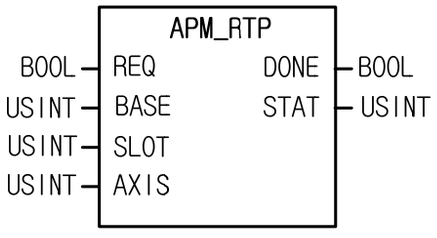
■ 기능

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 “에러6” 이 발생합니다.
   0: X 축, 1: Y 축, 2: Z 축

APM_RTP

수동 운전 이전 위치로 복귀

CPU 명	XGI	XGR
적용 가능	●	●

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

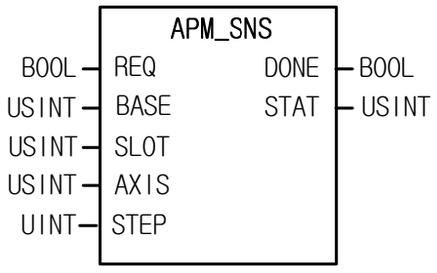
■ 기능

1. 이 명령은 위치결정 모듈에 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

APM_SNS

기동 스텝 번호 변경

CPU 명	XGI	XGR
적용 가능	●	●

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

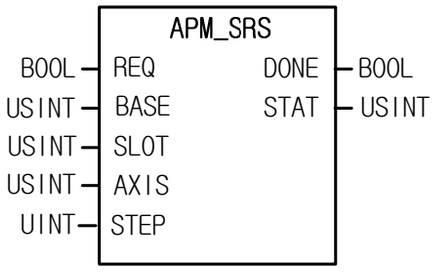
■ 기능

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
5. STEP 에는 운전하고자 하는 스텝 번호를 설정합니다. 설정값의 범위는 1 ~ 400 이며 설정값 이외의 값을 설정했을 경우 “에러 11” 이 발생합니다.

APM_SRS

반복 스텝 번호 변경

CPU 명	XGI	XGR
적용 가능	●	●

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

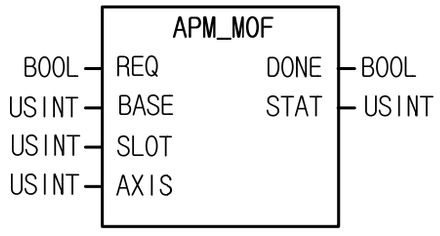
■ 기능

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 스텝번호를 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.

APM_MOF

M 코드 해제

CPU 명	XGI	XGR
적용 가능	●	●

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

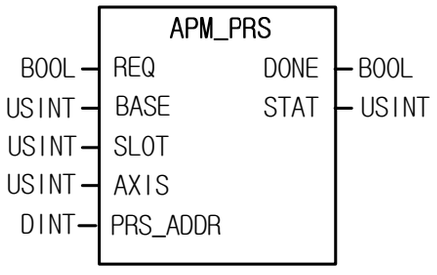
■ 기능

1. 이 명령은 위치결정 모듈에 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

APM_PRS

현재 위치 프리셋

CPU 명	XGI	XGR
적용 가능	●	●

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 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

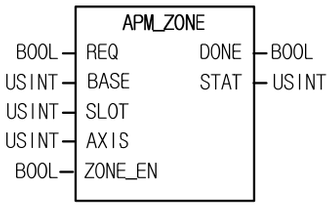
■ 기능

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

APM_ZONE

Zone 출력 허용/금지

CPU 명	XGI	XGR
적용 가능	●	●

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

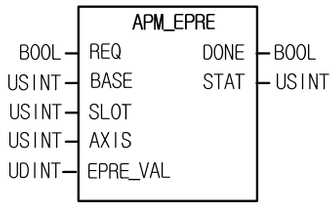
1. 이 명령은 위치결정 모듈에 Zone 출력 허용/금지 지령을 실행하는 명령입니다.
2. 공통 파라미터에서 설정한 Zone 에 대해 지령축의 위치 데이터와 Zone1, Zone2, Zone3 에서 설정한 위치 데이터 값을 이용하여 Zone 출력을 허가 또는 금지하는 지령입니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 Zone 출력 허용/금지 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.

0: X축, 1: Y축, 2: Z축

APM_EPRE

엔코더 프리셋

CPU 명	XGI	XGR
적용 가능	●	●

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

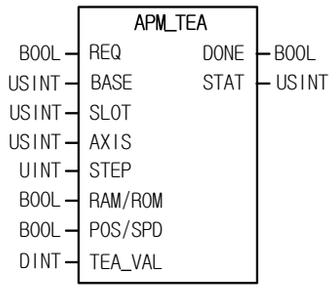
1. 이 명령은 위치결정 모듈에 엔코더 프리셋을 실행하는 명령입니다.
2. EPRE_VAL 에 설정된 값으로 엔코더의 현재값을 프리셋하는 지령입니다.
3. BASE(위치결정 모듈의 베이스 번호)와 SLOT(위치결정 모듈의 슬롯 번호)으로 지정된 위치결정 모듈의 AXIS 로 지정된 축에 엔코더 프리셋 지령을 내립니다.
4. AXIS 에는 지령을 내릴 축을 설정하며 다음과 같은 값을 설정할 수 있습니다. 설정값 이외의 값을 설정했을 경우 “에러 6” 이 발생합니다.

   0: X축, 1: Y축, 2: Z축

APM_TEA

단독 티칭

CPU 명	XGI	XGR
적용 가능	●	●

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

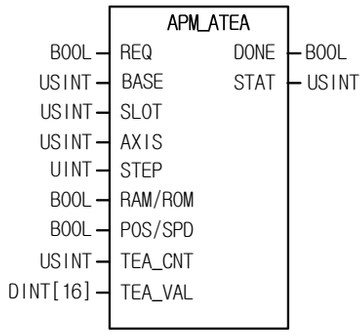
■ 기능

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]

APM_ATEA

복수 티칭

CPU 명	XGI	XGR
적용 가능	●	●

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

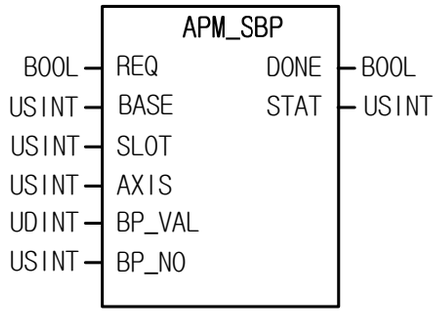
■ 기능

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]

APM_SBP

기본 파라미터 설정

CPU 명	XGI	XGR
적용 가능	●	●

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

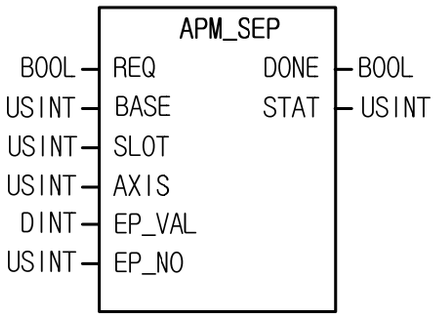
■ 기능

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

APM_SEP

확장 파라미터 티칭

CPU 명	XGI	XGR
적용 가능	●	●

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 1 을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

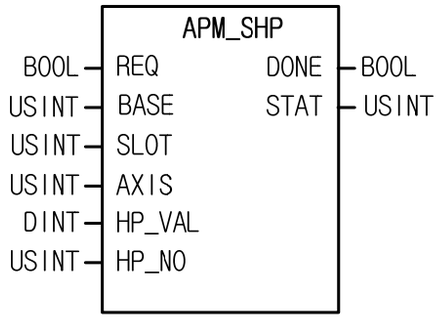
■ 기능

1. 이 명령은 위치결정 모듈에 확장 파라미터 티칭을 실행하는 명령입니다.
2. 확장 파라미터 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

APM_SHP

원점 복귀 파라미터 설정

CPU 명	XGI	XGR
적용 가능	●	●

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

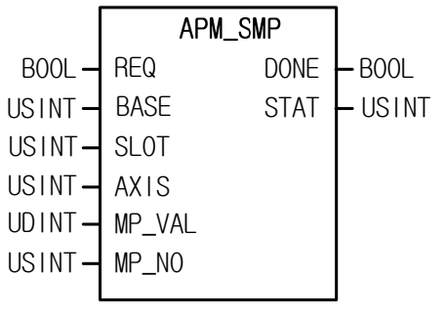
■ 기능

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/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.
   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

APM_SMP

수동 운전 파라미터 티칭

CPU 명	XGI	XGR
적용 가능	●	●

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 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

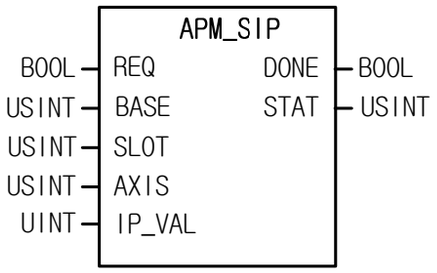
■ 기능

1. 이 명령은 위치결정 모듈에 수동 운전 파라미터 티칭을 실행하는 명령입니다.
2. 수동 운전 파라미터 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/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.
   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

APM_SIP

입력 신호 파라미터 티칭

CPU 명	XGI	XGR
적용 가능	●	●

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/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.

   0: X 축, 1: Y 축, 2: Z 축

각 입력 신호 설정 영역의 설정값은 아래와 같은 의미를 가진다.

0: A접점, 1: B접점

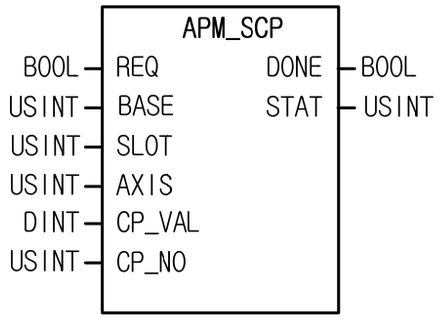
변경할 입력 신호 파라미터 값의 각 Bit에 할당된 신호는 아래와 같습니다.

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	-

APM_SCP

공통 파라미터 티칭

CPU 명	XGI	XGR
적용 가능	●	●

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

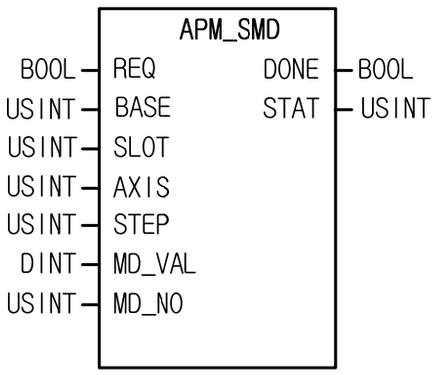
공통 파라미터 항목 번호에 설정할 값은 아래와 같습니다.

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

APM_SMD

운전 데이터 티칭

CPU 명	XGI	XGR
적용 가능	●	●

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 1을 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

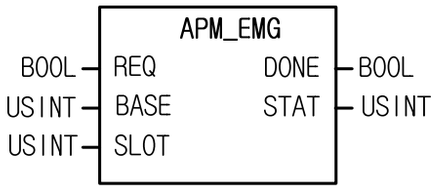
운전 데이터 항목번호에 설정할 값은 아래와 같습니다.

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

APM_EMG

비상 정지

CPU 명	XGI	XGR
적용 가능	●	●

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

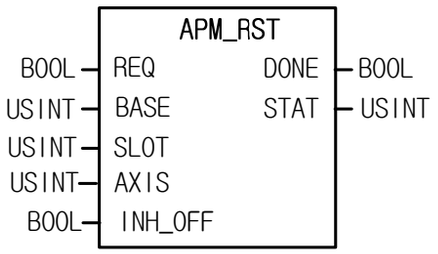
■ 기능

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.

APM_RST

에러 리셋/출력 금지 해제

CPU 명	XGI	XGR
적용 가능	●	●

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 1을 유지 STAT : 펑션 블록 실행 중 발생한 에러번호 출력

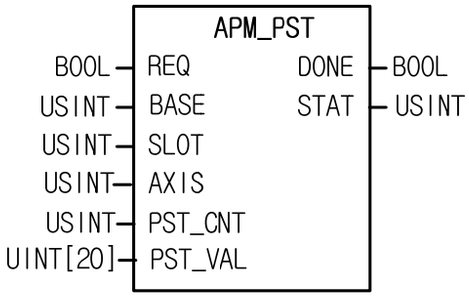
■ 기능

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
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.

APM_PST

포인트 운전

CPU 명	XGI	XGR
적용 가능	●	●

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

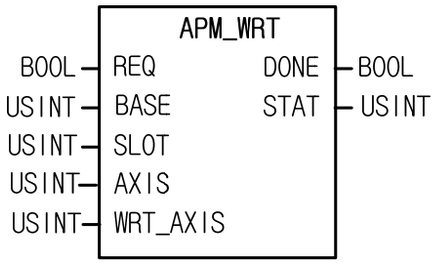
■ 기능

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.

APM_WRT

파라미터/운전 데이터 저장

CPU 명	XGI	XGR
적용 가능	●	●

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

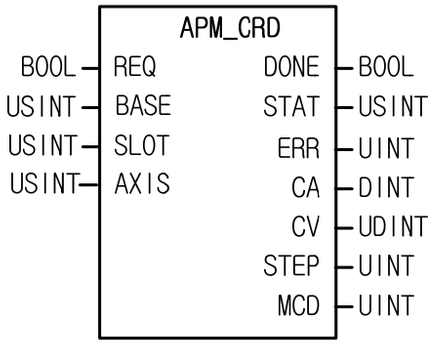
■ 기능

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
4. 지령축에 WRT_AXIS 에 설정된 축의 현재 운전 중인 파라미터와 운전 데이터를 Flash ROM 에 저장하는 지령을 내립니다.

APM_CRD

운전 정보 읽기

CPU 명	XGI	XGR
적용 가능	●	●

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 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

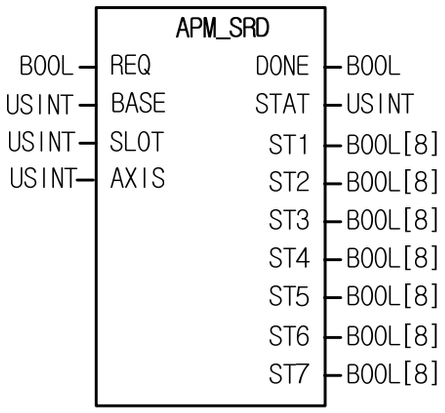
■ 기능

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 “에러6” 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.

APM_SRD

운전 상태 읽기

CPU 명	XGI	XGR
적용 가능	●	●

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 “에러6” 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
	[1]	하한 검출	[5]	정속 중
	[2]	비상 정지 상태	[6]	감속 중

	Bit	설명	Bit	설명
	[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]	-

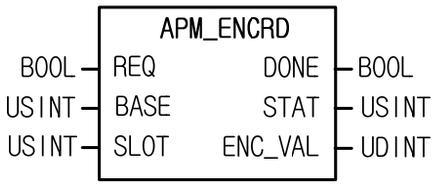
6. ST5 ~ ST7 의 각 Bit 가 의미하는 것은 아래와 같습니다.

	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]	-

APM_ENCRD

엔코더값 읽기

CPU 명	XGI	XGR
적용 가능	●	●

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

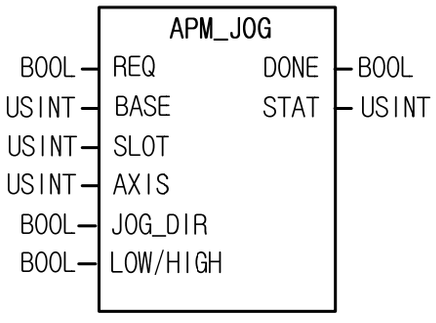
■ 기능

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 엔코더값 읽기 지령을 내립니다.

APM_JOG

조그 운전

CPU 명	XGI	XGR
적용 가능	●	●

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

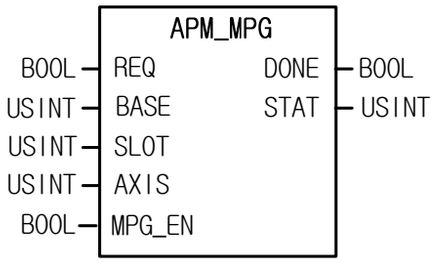
■ 기능

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

APM_MPG

수동 펄스 발생기(MPG) 운전

CPU 명	XGI	XGR
적용 가능	●	●

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

부록 1 수치체계 및 데이터 구조

부 1.1 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.

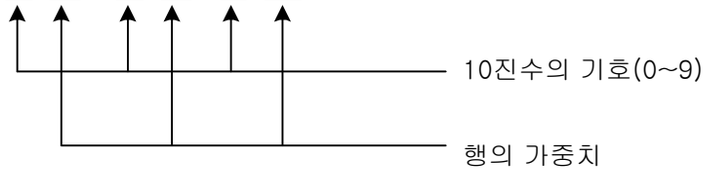
그리고 0, 1, 2, 3, 4, …, 9 다음에 '10' 으로 자리올림하고 계속 진행됩니다.

예를 들면, 10 진수 153 을 행과 "행의 가중치" 란 측면에서 보면 아래와 같습니다.

$$135=100+50+3$$

$$=1*100+5*10+3*1$$

$$=1*10^{2}+5*10^{1}+3*10^{0}$$



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.

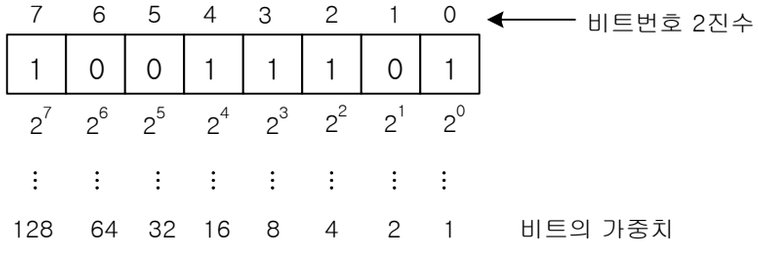
즉, 0,1 의 한 자리 수를 비트라고 합니다.

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.

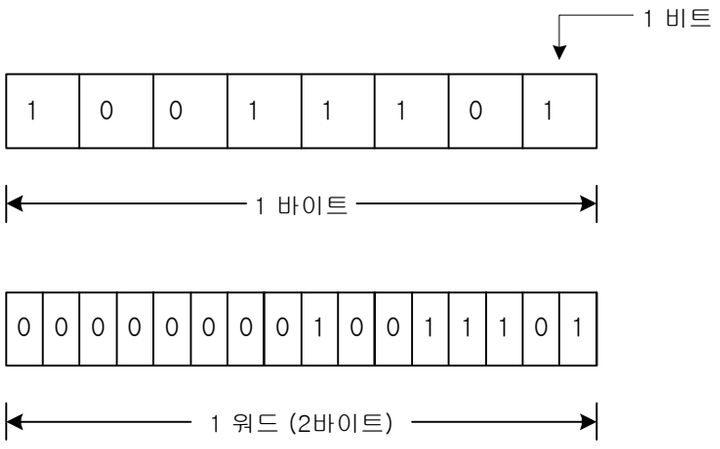
= 1 × 128 + 0 × 64 + 0 × 32 + 1 × 16 + 1 × 8 + 1 × 4 + 0 × 2 + 1 × 1

= 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.

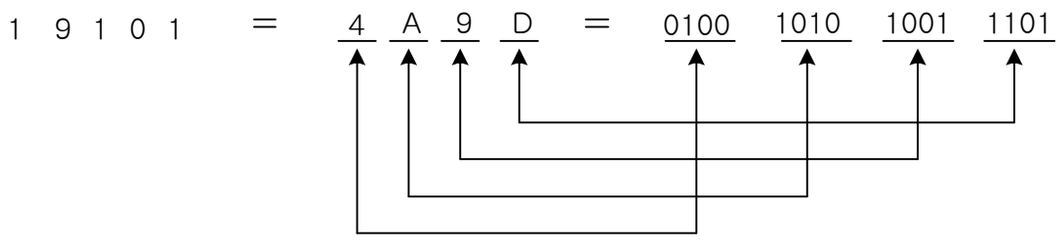


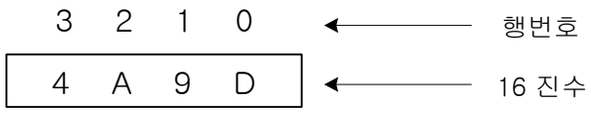
3) 16 진수 (Hexadecimal ····· HEX)

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

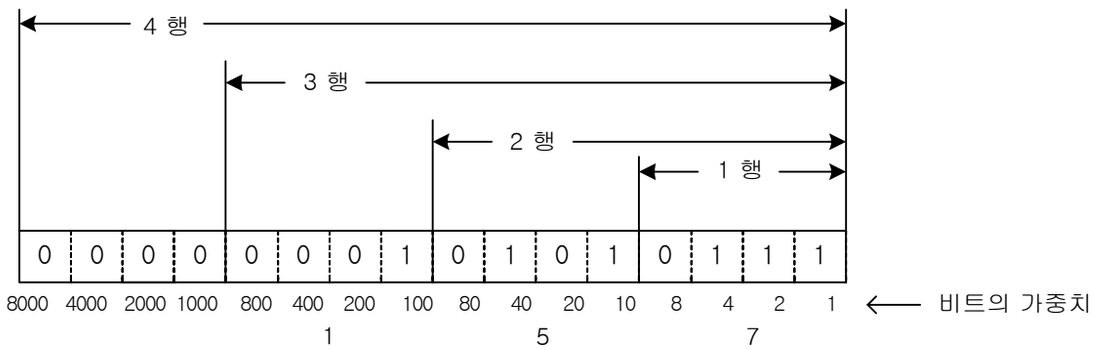
16 진수의 한자리는 2 진수의 4 비트로 대응됩니다.

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

부 1.2 정수표현

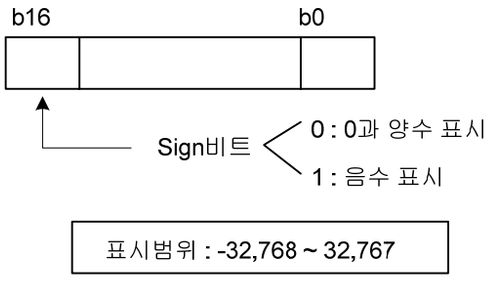
XGI 명령어에서는 음수체계연산(Signed)을 기본으로 합니다.

이때 정수표시는 최상위 비트(MSB)가 0 이 되면 양수를 나타내고 1 이면 음수로 나타나게 됩니다.

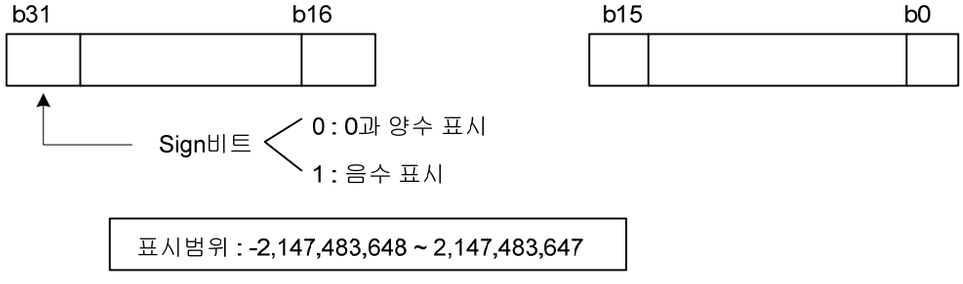
음수, 양수를 표시하는 최상위 비트를 Sign 비트라고 합니다.

16 비트, 32 비트에서는 MSB 의 위치가 다르기 때문에 Sign 비트 위치에 주의해야 합니다.

✸ 16 비트 일 경우



✸ 32 비트 일 경우



부 1.3 음수의 표현

예) - 0001 을 표기하는 방법

(1) 음수번호를 땐 0001 을 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 +1 을 한다.

b15							b0
1	1				~	1	1

-0001 = 16#FFFF

부록2 플래그 일람

부2.1 모드와 상태

예 약 변 수	데이터타입	내 용
_SYS_STATE	DWORD	PLC 의 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.
_MODBUS_CON	BOOL	모드버스 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 PADT 에 의해 운전모드가 변경 되었습니다.
_CMOD_RPADT	BOOL	리모트 PADT 에 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 SKIP 이 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 코드 1 이 선택되었습니다.
_PB2	BOOL	프로그램 코드 2 가 선택되었습니다.
_USER_WRITE_F	WORD	프로그램에서 사용 가능한 접점
_RTC_WR	BOOL	RTC 에 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
_USER_STAUS_F	WORD	유저접점

예 약 변 수	데이터타입	내 용
_INIT_DONE	BOOL	초기화 태스크 수행 완료를 표시
_KEY	DWORD	로컬 키의 현재 상타를 나타냅니다.
_KEY_PREV	DWORD	로컬 키의 이전 상태를 나타냅니다.
_RBLOCK_STATE	WORD	플래쉬 블록 상태

부2.2 시스템 에러

예 약 변 수	데이터타입	내 용
_CNF_ER	WORD	시스템의 중고장 상태를 보고합니다.
_CPU_ER	BOOL	CPU 구성에 에러가 있습니다.
_IO_TYER	BOOL	모듈 타입이 일치하지 않습니다.
_IO_DEER	BOOL	모듈이 착탈되었습니다.
_FUSE_ER	BOOL	퓨즈가 끊어졌습니다.
_IO_RWER	BOOL	모듈 입출력에 문제가 발생했습니다.
_IP_IFER	BOOL	특수 / 통신 모듈 인터페이스에 문제가 발생했습니다.
_IO_TYER_N	WORD	모듈 타입 불일치 슬롯 넘버
_IO_DEER_N	WORD	모듈 착탈 슬롯 넘버
_FUSE_ER_N	WORD	퓨즈 단선 슬롯 넘버
_IO_RWER_N	WORD	입출력 모듈 읽기/쓰기 에러 슬롯 넘버
_IP_IFER_N	WORD	특수/통신 모듈 인터페이스 에러 슬롯 넘버
_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.
_CDOVER_ER	BOOL	실행코드 영역 초과 에러
_CODE_ER	BOOL	프로그램 코드에 에러가 있습니다.
_TMRIDX_ER	BOOL	타이머 인덱스 사용 에러
_COMPILE_ER	BOOL	컴파일 에러
_INST_ER	BOOL	연산에러
_SWDT_ER	BOOL	시스템 워치독 타이머가 작동했습니다.
_BASE_POWER_ER	BOOL	베이스 전원에 이상이 있습니다.
_WDT_ER	BOOL	스캔 워치독 타이머가 작동했습니다.
_IO_RWER0	WORD	메인 베이스 모듈 읽기/쓰기 에러
_IO_RWER1	WORD	증설 베이스 1단 모듈 읽기/쓰기 에러

예 약 변 수	데이터타입	내 용
_IO_RWER2	WORD	증설 베이스 2 단 모듈 읽기/쓰기 에러
_IO_RWER3	WORD	증설 베이스 3 단 모듈 읽기/쓰기 에러
_IO_RWER4	WORD	증설 베이스 4 단 모듈 읽기/쓰기 에러
_IO_RWER5	WORD	증설 베이스 5 단 모듈 읽기/쓰기 에러
_IO_RWER6	WORD	증설 베이스 6 단 모듈 읽기/쓰기 에러
_IO_RWER7	WORD	증설 베이스 7 단 모듈 읽기/쓰기 에러
_FUSE_ER0	WORD	메인 베이스 퓨즈 단선 에러
_FUSE_ER1	WORD	증설 베이스 1 단 퓨즈 단선 에러
_FUSE_ER2	WORD	증설 베이스 2 단 퓨즈 단선 에러
_FUSE_ER3	WORD	증설 베이스 3 단 퓨즈 단선 에러
_FUSE_ER4	WORD	증설 베이스 4 단 퓨즈 단선 에러
_FUSE_ER5	WORD	증설 베이스 5 단 퓨즈 단선 에러
_FUSE_ER6	WORD	증설 베이스 6 단 퓨즈 단선 에러
_FUSE_ER7	WORD	증설 베이스 7 단 퓨즈 단선 에러
_IO_TYER0	WORD	메인 베이스 모듈 타입 에러
_IO_TYER1	WORD	증설 베이스 1 단 모듈 타입 에러
_IO_TYER2	WORD	증설 베이스 2 단 모듈 타입 에러
_IO_TYER3	WORD	증설 베이스 3 단 모듈 타입 에러
_IO_TYER4	WORD	증설 베이스 4 단 모듈 타입 에러
_IO_TYER5	WORD	증설 베이스 5 단 모듈 타입 에러
_IO_TYER6	WORD	증설 베이스 6 단 모듈 타입 에러
_IO_TYER7	WORD	증설 베이스 7 단 모듈 타입 에러
_IO_DEER0	WORD	메인 베이스 모듈 착탈 에러
_IO_DEER1	WORD	증설 베이스 1 단 모듈 착탈 에러
_IO_DEER2	WORD	증설 베이스 2 단 모듈 착탈 에러
_IO_DEER3	WORD	증설 베이스 3 단 모듈 착탈 에러
_IO_DEER4	WORD	증설 베이스 4 단 모듈 착탈 에러
_IO_DEER5	WORD	증설 베이스 5 단 모듈 착탈 에러
_IO_DEER6	WORD	증설 베이스 6 단 모듈 착탈 에러
_IO_DEER7	WORD	증설 베이스 7 단 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.
_DBCK_ER	BOOL	데이터 벡업에 문제가 발생했습니다.
_HBCK_ER	BOOL	핫 리스타트가 불가능합니다.
_ABSD_ER	BOOL	비정상 운전으로 인하여 정지합니다.
_TASK_ER	BOOL	태스크가 충돌하고 있습니다.
_BAT_ER	BOOL	배터리 상태에 이상이 있습니다.
_ANNUM_WAR	BOOL	외부 기기의 경고장이 검출 되었습니다.
_LOG_FULL	BOOL	로그 메모리가 꽉 찼습니다.
_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
_USER_F	WORD	사용자가 사용할 수 있는 타이머입니다.
_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.
_10N	BOOL	첫 스캔만 On 상태인 비트입니다.
_10FF	BOOL	첫 스캔만 Off 상태인 비트입니다.
_STOG	BOOL	매 스캔 반전됩니다.
_USER_CLK	WORD	사용자가 설정 가능한 CLOCK 입니다.

부2.5 연산 결과 플래그

예 약 변 수	데이터타입	내 용
_LOGIC_RESULT	WORD	로직 결과를 표시합니다.
_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
_PUT_CNT	DWORD	PUT 수행 시 증가합니다.
_GET_CNT	DWORD	GET 수행 시 증가합니다.
_FPU_FLAG_E	BOOL	비정규화값 입력 에러 플래그
_FPU_FLAG_I	BOOL	부정확 에러 플래그
_FPU_FLAG_O	BOOL	오버플로우 에러 플래그
_FPU_FLAG_U	BOOL	언더플로우 에러 플래그
_FPU_FLAG_V	BOOL	무효연산 에러 플래그
_FPU_FLAG_Z	BOOL	영나누기 에러 플래그
_FPU_LFLAG_I	BOOL	부정확 에러 래치 플래그

예 약 변 수	데이터타입	내 용
_FPU_LFLAG_O	BOOL	오버플로우 에러 래치 플래그
_FPU_LFLAG_U	BOOL	언더플로우 에러 래치 플래그
_FPU_LFLAG_V	BOOL	무효연산 에러 래치 플래그
_FPU_LFLAG_Z	BOOL	영나누기 에러 래치 플래그
_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.
_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.
_MON_YEAR	WORD	PLC 의 월, 년 데이터입니다.
_TIME_DAY	WORD	PLC 의 시, 일 데이터입니다.
_SEC_MIN	WORD	PLC 의 초, 분 데이터입니다.
_HUND_WK	WORD	PLC 의 백년, 요일 데이터입니다.
_MON_YEAR_DT	WORD	시계 정보 데이터 (월 / 년)
_TIME_DAY_DT	WORD	시계 정보 데이터 (시 / 일)
_SEC_MIN_DT	WORD	시계 정보 데이터 (초 / 분)
_HUND_WK_DT	WORD	시계 정보 데이터 (백년 / 요일)
_RTC_DATE	WORD	RTC 의 현재 날짜
_RTC_WEEK	WORD	RTC 의 현재 요일
_RTC_TOD	DWORD	RTC 의 현재 시간 (ms 단위)
_AC_FAIL_CNT	DWORD	전원이 차단 된 횟수를 저장합니다.
_ERR_HIS_CNT	DWORD	에러가 발생한 횟수를 저장합니다.
_MOD_HIS_CNT	DWORD	모드가 전환된 횟수를 저장합니다.
_SYS_HIS_CNT	DWORD	시스템 이력 발생 횟수를 저장합니다.
_LOG_ROTATE	DWORD	로그 로테이트 정보를 저장합니다.
_BASE_INFO0	WORD	메인 베이스 슬롯 정보

예 약 변 수	데이터타입	내 용
_BASE_INF01	WORD	증설 베이스 1단 슬롯 정보
_BASE_INF02	WORD	증설 베이스 2단 슬롯 정보
_BASE_INF03	WORD	증설 베이스 3단 슬롯 정보
_BASE_INF04	WORD	증설 베이스 4단 슬롯 정보
_BASE_INF05	WORD	증설 베이스 5단 슬롯 정보
_BASE_INF06	WORD	증설 베이스 6단 슬롯 정보
_BASE_INF07	WORD	증설 베이스 7단 슬롯 정보
_RBANK_NUM	WORD	현재 사용중인 블록 번호
_RBLOCK_RD_FLAG	DWORD	플래시 N블록의 데이터 읽을 때 On
_RBLOCK_WR_FLAG	DWORD	플래시 N블록의 데이터 쓸 때 On
_RBLOCK_ER_FLAG	DWORD	플래시 N블록 서비스중 에러 발생
_REF_COUNT	DWORD	모듈 리프레시 수행시 증가
_REF_OK_CNT	DWORD	모듈 리프레시가 정상일 때 증가
_REF_NG_CNT	DWORD	모듈 리프레시가 비정상일 때 증가
_REF_LIM_CNT	DWORD	모듈 리프레시가 비정상일 때 증가(TIME OUT)
_REF_ERR_CNT	DWORD	모듈 리프레시가 비정상일 때 증가
_MOD_RD_ERR_CNT	DWORD	모듈 1워드를 비정상적으로 읽으면 증가합니다.
_MOD_WR_ERR_CNT	DWORD	모듈 1워드를 비정상적으로 쓰면 증가합니다.
_CA_CNT	DWORD	모듈의 블록데이터 서비스 시 증가
_CA_LIM_CNT	DWORD	블록데이터 서비스 비정상 시 증가
_CA_ERR_CNT	DWORD	블록데이터 서비스 비정상 시 증가
_BUF_FULL_CNT	DWORD	CPU 내부버퍼 FULL일 경우 증가
_AC_F_CNT	WORD	순시 정전 발생 횟수를 알려줍니다.
_FALS_NUM	WORD	FALS의 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*
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**

예 약 변 수	데이터타입	내 용
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**
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**

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.