목 차

PLC 기초

I

PLC(Programmable Logic Controller)의 기초

1장. PLC의 개요

1-1. 제어 요소

㉮ 입력장치 : 물리적 신호를 전기 신호로 변환하는 감지기와 제어장치로 신호를 전송하는 변환 장치로 구성된다.

㉯ 출력장치 : 장치에서 만들어지는 제어 신호를 실제 구동장치를 유발시키는 조작신호로 변환하는 장치이다.

㉰ 제어장치 : 입력 조건에 따른 제어정책 및 제어 알고리즘을 수행하여 제어신호를 출력에 내보내는 장치로서, 제어 알고리즘을 소프트웨어로 메모리 장치에 저장하여 유연하게 변경할 수 있는 프로그래머블 제어기와 한 번 결정하여 설치하면 제어 알고리즘이 고정적이어서 변경하기 어려운 하드웨어 시스템으로 크게 나눌 수 있다.

[그림 1-1] 제어 시스템의 요소

1-2. PLC의 정의

PLC(Programmable Logic Controller)란 종래에 사용하던 제어반에 사용하는 릴레이, 타이머, 카운터 등의 기능을 IC, 트랜지스터 등의 반도체 소자로 대체시켜, 기본적인 시퀀스 제어 기능에 수치 연산기능을 추가하여 프로그램 제어가 가능하도록 한 자율성이 높은 제어장치이며 미국전기

공업협회(NEMA : National Electrical Manufacturers Association)에서는 “디지털 또는 아날로그 입·출력 모듈을 통하여 로직, 시퀀싱, 타이밍, 카운팅, 연산과 같은 특수한 기능을 수행하기 위하여 프로그램 가능한 메모리를 사용하고 여러 종류의 기계나 프로세서를 제어하는 디지털 동작의 전자 장치”로 정의하고 있다.

1-3. PLC의 발달과정

산업사회의 발전에 따라 각 공정들이 대규모, 고도화, 복잡화 되고 다양한 형태의 제어 시스템이 요구되고 있다. 이러한 제어시스템을 유기적으로 연결 또는 변경하기 위해서는 많은 시간과 비용이 소요된다. 자동화를 위한 지금까지의 제어 시스템은 회로도에 따라 여러 관련소자(Relay, Contractor, Timer, Counter 등)들을 연결하여 사용함으로써 실제 배선작업이 어렵고, 시퀀스 제어를 위하여 많은 공간이 필요하였으며, 처리속도에 한계가 있다는 문제점이 발생하였다, 이러한 여러 원인에 의하여 1968년 미국의 자동차 메이커인 제너럴모터스(General Motors)사가 [표 1-1] PLC의 10가지 조건을 제시하였으며, 이러한 조건이 PLC 개발의 계기가 되었고, [표 1-2]는 PLC의 역사를 연도별로 정리한 것이다.

[표 1-1] GM의 10가지 조건

(1) 프로그램 및 시퀀스 시스템의 작성과 변경이 용이할 것.
(2) 점검 및 보수가 용이하고 Plug-In 방식일 것.
(3) 계전기 제어반 보다 신뢰성이 높을 것.
(4) 출력은 상위 컴퓨터와 결합기능을 가질 것.
(5) 계전기 제어반보다 소형일 것.
(6) 계전기 제어반보다 가격 면에서 유리할 것.
(7) 입력을 AC115[V]를 공급받을 것.
(8) 출력은 AC115[V], 2[A]를 공급받을 것.
(9) 전체 시스템 변경을 최소화하면서 기본 시스템은 확장이 가능할 것.
(10) 최저 4K Word까지 확장 가능한 프로그래머블 메모리를 가질 것.

[표 1-2] PLC의 역사

구 분	발 달 과 정
1968	PLC의 개념 태동
1970	논리지시 및 1K 기억용량과 128 I/O 점수 제어기 등장
1974	타이머, 카운터, 산술연산 기능과 12K 기억용량과 1024 I/O 점수처리
1976	원격 입·출력 시스템 소개(최초로 규격 제정 - 미국)
1977	마이크로프로세서 PLC 등장
1980	고성능 I/O 모듈, 고성능 통신장비, 고기능 소프트웨어 등장 및 프로그래밍 도구로 마이크로컴퓨터 사용
1983	저렴한 소형 PLC 등장
1985	표준화, 컴퓨터와의 네트워크 기능으로 분산 및 계층 제어 가능
1991	퍼지이론을 도입한 퍼지 모듈 및 전용 퍼지 제어기 등장

1-4. PLC의 표준화 및 특징

1) IEC 표준 언어

그 동안 PLC를 접하는 엔지니어는 메이커(maker)마다 사용 언어와 통신 네트워크가 서로 달라 많은 불편함을 겪어 왔으나 이러한 불편함을 해소하고, PLC 고객에게 편리성을 도모하고자 IEC(International Electrotechnical Commission 국제 전기 표준회의)에서는 [표 1-3]과 같이 PLC 국제 표준화 규격(IEC 1131)을 5 Part로 구성하여 제정하였다.

[표 1-3] PLC의 변천과정

구 분	설 명
Part 1	PLC의 기본 기능 및 용어 정의
Part 2	설비의 요구 기능 및 시험조건
Part 3	프로그램 언어
Part 4	사용자 지침
Part 5	통신 및 네트워크

IEC에서 새로 도입한 가장 중요한 특징들은 다음과 같다.

① 다양한 데이터 타입(type)을 지원한다.

② 펑션, 펑션블록, 프로그램 같은 구성 요소가 도입되어 상향식, 또는 하향식 설계가 가능하며, PLC 프로그램을 구조적으로 작성할 수 있다.

③ 사용자가 작성한 프로그램을 라이브러리화하여 다른 환경에서 소프트웨어를 재사용 할 수 있다.

④ 다양한 언어를 지원하므로 사용자는 최적의 언어를 선택하여 사용할 수 있다.

⑤ IEC에서 표준화한 PLC용 언어는 두 개의 도형 기반 언어와 두 개의 문자 기반언어, 그리고 SFC로 이루어져 있다.

(1) 도형식(Graphic)언어

① LD(Ladder Diagram) : 미국의 사다리형(Ladder)에서 기원한 언어이며 입력과 출력을 조합하여 프로그램을 작성하며, 종래의 릴레이 로직 표현 방식의 언어이다.

② FBD(Function Block Diagram) : 블록화한 기능을 서로 연결하여 프로그램을 표현하는 언어

(2) 문자식(Text) 언어

① IL(Instruction List) : 유럽에서 많이 사용하며, 어셈블리 형태의 명령어 방식의 언어이다.

② ST(Structured Text) : 리얼타임 어플리케이션용으로 개발되었으며 파스칼이나 C를 기원으로 한 고급언어이다.

③ SFC(Sequential Function Chart) : 공정의 흐름이나 조건 등의 동작을 순차적으로 표현한 형태의 언어이며 시간과 이벤트 등의 제어 시퀀스를 블록으로 정의하도록 되어있다.

2) PLC의 특징

PLC와 다른 제어장치들의 특징을 아래의 [표 1-4]에서 나열하였으며, 나열된 제어장치들의 기능과 장·단점을 살펴 볼 수 있으며, 다음과 같은 특징을 얻을 수 있다.

[표 1-4] 제어장치의 비교

구 분	릴레이 제어반	디지털 로직	컴퓨터	PLC
가격	매우 저가	저가	고가	저가
크기	대형	매우 소형	적당	매우 소형
처리속도	느림	매우 빠름	매우 빠름	빠름
노이즈	우수	양호	아주 우수	양호
제어	설계와 설치시 많은 시간 소요	설계시 많은 시간 소요	Program시 아주 많은 시간 소요	간단
복합기능	없다.	있다.	있다.	있다.
기능의 변화	매우 어렵다.	어렵다.	아주 쉽다.	아주 쉽다.
유지보수	매우 어렵다.	어렵다.	아주 쉽다.	아주 쉽다.

◈ PLC의 특징

- 기능의 다양화
- 프로그램의 고기능성 (제어회로 설계용이)
- 조작의 간편성
- 유지보수의 편리성
- 고 신뢰성
- 설치의 간편성

1-5. PLC의 선정 및 적용분야

1) PLC의 선정

PLC를 선정하려면 제어대상에 대한 시방과 제어내용을 정확히 이해하고 제어대상의 기능, 가격, 확장성, 사후관리, 기종의 계속성, 사용자의 수준 등을 고려하여야 한다.

(1) 입력 점수의 파악

조작반의 누름 스위치, 리밋 스위치 등의 명령을 내리는 입력신호 수가 근접센서, 포토센서, 리드 스위치 등의 신호수를 더하여 입력점수로 산정하고, 개수를 적절히 선정한다. 또한 입력으로 사용되는 센서 등의 사용전압을 고려하여 입력 모듈(AC 및 DC 전압)의 사양을 선정한다.

(2) 출력점수의 파악

전원 표시등, 운전표시등, 과부하 표시등, 부져 등의 표시 또는 솔레노이드 밸브, 릴레이, 전자 접촉기의 수를 더하여 출력점수로 산정하고 8점, 16점, 32점 등의 모듈을 적절히 혼합하여 출력 모듈 수를 선정한다. 또한 출력방식은 릴레이 접점방식, TR출력방식, SSR출력 방식 등이 있으므로 출력부의 사용전압을 고려하여 선정하여야 한다.

※ 일반적으로 출력전압에 구애를 받지 않는 릴레이 접점 출력방식의 모듈이 많이 사용된다.

(3) CPU 및 특수모듈의 지원

일반적으로 Digital I/O외에 Analog I/O와 특수카드(HSC, POP, PID등) 지원기능 등과 CPU의 특성을 고려하여 선정하여야 한다.

2) PLC의 적용분야

설비의 자동화와 고 능률화의 요구에 따라 PLC의 적용범위는 확대되고 있다. 특히 공장 자동화와 FMS에 따른 PLC의 요구는 과거 중규모 이상의 릴레이 제어반 대체 효과에서 현재 고기능화, 고속화의 추세로 소규모 공작기계에서 대규모 시스템설비 등에 적용되고 있다. [표 1-5]는 PLC 제어대상에 따른 적용분야를 나타낸 것이다.

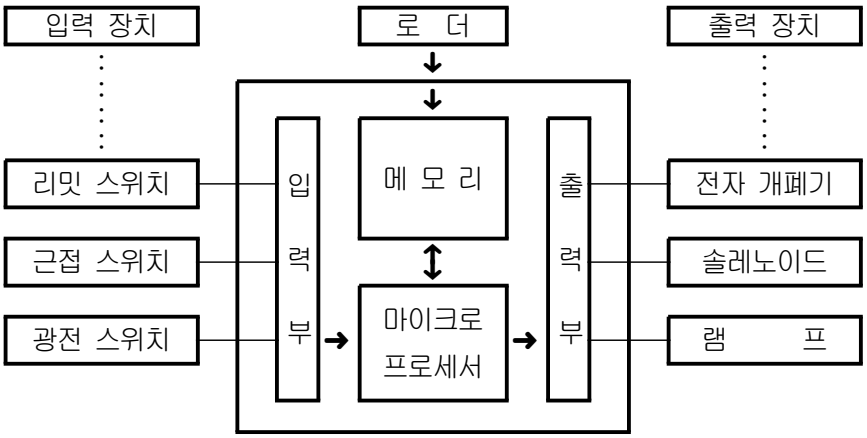
[표 1-5] PLC의 적용분야

분 야	제 어 대 상
식료산업	컨베이어 총괄 제어, 생산라인 자동제어
제철, 제강산업	작업장 하역제어, 원료 수송제어, 압연라인제어, 하역운반제어
섬유, 화학공업	원조 수입출하 제어, 컨베이어 제어, 직조 염색 라인 제어
자동차 공업	전송라인제어, 자동조립 라인제어
기계 산업	산업용 로봇제어, 공작기계제어, 송·배수 펌프제어
상하수도	정수장 제어, 하수처리제어, 송·배수 펌프 제어
물류산업	자동창고 제어, 하역 설비제어, 반송라인제어
공장설비	압축기 제어
공해방지산업	쓰레기 소각로 자동제어, 공해 방지기 제어

2장. PLC의 구조

2-1. 하드웨어의 구조

PLC는 마이크로프로세서(Microprocessor) 및 메모리를 중심으로 구성되어 인간의 두뇌 역할을 하는 중앙처리장치(CPU), 외부 기기와의 신호를 연결시켜 주는 입·출력부, 각 부에 전원을 공급하는 전원부, PLC내의 메모리에 프로그램을 기록하는 주변 장치로 구성되어 있다. [그림 2-1]은 PLC의 전체 구성도를 나타낸 것이다.



[그림 2-1] PLC의 전체 구성도

1) PLC의 CPU

PLC의 두뇌에 해당하는 부분으로서, 메모리에 저장되어 있는 프로그램을 하나씩 꺼내서 해독하여 처리 내용을 실행한다. 이 절차는 매우 빠른 속도로 반복되며, 모든 정보는 2진수로 처리된다.

2) PLC CPU의 메모리

(1) 메모리의 종류

IC 메모리 종류에는 ROM과 RAM이 있다. ROM은 읽기 전용으로, 메모리 내용을 변경할 수 없다. 따라서 고정된 정보를 써넣는다. 이 영역의 정보는 전원이 끊어져도 기억 내용이 보존되는 불휘발성 메모리이다. RAM은 메모리에 정보를 수시로 읽고 쓰기가 가능하여 정보를 일시 저장하는 용도로 사용되나, 전원이 끊어지면 기억시킨 정보 내용을 상실하는 휘발성 메모리이다. 그러나 필요에 따라 RAM 영역 일부를 배터리 백업에 의하여 불휘발성 영역으로 사용할 수 있다.

(2) 메모리 내용

PLC의 메모리는 사용자 프로그램 메모리, 데이터 메모리, 시스템 메모리의 3가지로 구분된다. 사용자 프로그램 메모리는 제어하고자 하는 시스템 사양에 따라 사용자가 작성한 프로그램이 저장되는 영역으로, 제어 내용이 프로그램 완성 전이나, 완성 후에도 바뀔 수 있으므로 RAM이 사용된다. 프로그램이 완성되어 고정이 되면 ROM에 써넣어 ROM 운전을 할 수 있다. 데이터 메모리는 입·출력 릴레이, 보조 릴레이, 타이머와 카운터의 접점상태 및 설정값, 현재 값 등의 정보가 저장되는 영역으로 정보가 수시로 바뀌므로 RAM 영역이 사용된다.

시스템 메모리는 PLC 제작 회사에서 작성한 시스템 프로그램이 저장되는 영역이다. 이 시스템 프로그램은 PLC의 기능이나 성능을 결정하는 중요한 프로그램으로, PLC 메이커에서 직접 ROM에 써넣는다.

2-2. PLC의 입·출력 구조

PLC의 입·출력부는 현장의 외부 기기에 직접 접속하여 사용한다. PLC내부는 DC+5(V)의 전원(TTL 레벨)을 사용하지만 입·출력부는 다른 전압 레벨을 사용하므로 PLC 내부와 입·출력의 접속(interface)은 시스템 안정에 결정적인 요소가 되며, PLC의 입·출력부는 다음의 사항이 요구된다.

① 외부 기기와 전기적 규격이 일치해야 한다.

② 외부 기기로부터의 노이즈가 CPU쪽에 전달되지 않도록 해야 한다.
(포토커플러(photocoupler)를 사용)
③ 외부 기기와의 접속이 용이해야 한다.
④ 입·출력의 각 접점 상태를 감시할 수 있어야 한다. (LED부착)

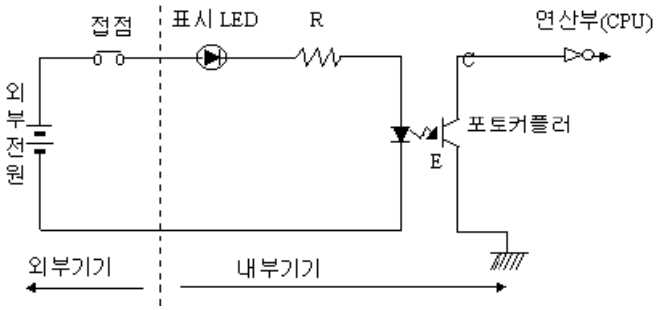
[표 2-2]는 입·출력에 접속되는 외부 기기의 예는 와 같다.

[표 2-2] 입·출력기기

I/O	구분	부착장소	외부 기기의 명칭
입력부	조작입력	제어반과 조작반	푸시버튼 스위치 선택 스위치 토클 스위치
	검출입력 (센서)	기계장치	리밋 스위치 광전 스위치 근접 스위치 레벨 스위치
출력부	표시경보 출력	제어반 및 조작반	파일럿램프 부져
	구동출력 (액츄에이터)	기계장치	전자밸브 전자 클러치 전자 브레이크 전자 개폐기

1) 입력부

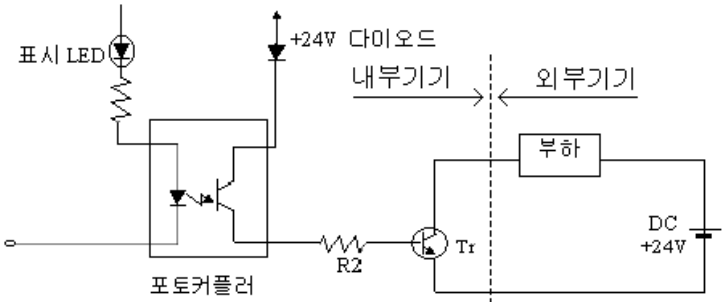
외부 기기로부터의 신호를 CPU의 연산부로 전달해 주는 역할을 한다. 입력의 종류는 DC 24[V], AC 110/220[V]등이 있고, 그 밖의 특수 입력모듈로는 아날로그 입력(A/D)모듈, 고속카운터(high speed counter)모듈 등이 있다. 입력부 회로의 예는 [그림 2-3]과 같다.



[그림 2-3] 입력부 회로

2) 출력부

내부 연산의 결과를 외부에 접속된 전자 접촉기나 솔레노이드에 전달하여 구동시키는 부분이다. 출력의 종류에는 릴레이 출력, 트랜지스터 출력, SSR(Solid State Relay)출력 등이 있고 그 밖의 출력 모듈로는 아날로그 출력(D/A)모듈, 위치결정(POP) 모듈 등이 있다. 트랜지스터 출력부 회로의 예는 [그림 2-4]와 같다.



[그림 2-4] 트랜지스터 출력회로

[표 2-3] PLC 출력 모듈의 종류

출력 신호 종류	개 폐 소 자	
	유접점	무접점(반도체)
직류(DC)	릴레이 출력	트랜지스터 출력
교류(AC)	릴레이 출력	SSR 출력

[표 1-3]에서와 같이 릴레이 출력은 직류나 교류를 모두 사용할 수 있으나 기계적 수명의 한계 때문에 접점의 개폐가 빈번할 경우는 교류 전용인 무접점 SSR 출력이나 직류 전원 전용인 트랜지스터 출력을 사용하여야 한다.

※ 특수모듈

㉮ 위치제어 모듈

지정된 주파수 영역과 전압레벨로 고속 접점 출력을 처리한다.

㉯ PID 제어 모듈

아날로그 입력모듈에서 받은 현장 데이터를 정해진 설정치에 도달시키도록 최적의 조건에 의해 연산하여 그 결과를 아날로그 출력 모듈로 출력시킨다.

㉰ 설정치 제어 모듈

정해진 시간 동안 정해진 설정치를 상승, 유지, 하강을 반복하며 제어한다.

㉱ 기타

통신모듈, 네트워크 모듈, 특정기기의 제어 모듈 등이 있다.

2-3. 소프트웨어의 구조

1) 프로그래밍의 개요

PLC를 사용할 경우의 제어 Sequence와 종래의 Relay, Timer등을 사용한 경우의 Sequence와는 근본적으로 차이가 없다. Sequence를 이해하고 작성하기 위해서는 다음의 3가지 사항을 이해하여야 한다.

① 제어해야 할 대상의 특성을 이해하여야 한다. 즉, 제어 목적, 운전 방법, 동작 등의 각종 전기적인 조건을 알고 있어야 한다.

② 제어 장치에 대한 지식이 있어야 한다. 즉, Relay와 PLC의 특성 및 사용방법을 알고 있어야 한다.

③ Sequence를 작성하기 위한 약속을 알고 있어야 한다. 즉, 도 기호, 기구 번호, 상태 등에 대한 약속(규칙)을 알고 있어야 한다.

PLC를 사용 하던가, Relay/Timer를 사용 하던가 상기 두 항에는 차이가 없다. 따라서 PLC를 사용한 경우의 제어 Sequence를 이해하기 위해서는 PLC의 특성과 그 사용 방법을 아는 것이 중요하다. 예를 들면 입·출력 카드의 정격전압, 접점 수 등과 같은 물리적 특성(Hardware)과 논리연산, Timer, Counter와 같은 기능적인 특성(Software)이 있다. 이것에 대해서는 통상 Catalog의 사양 항목에 기재되어 있다. 따라서 PLC가 갖고 있는 많은 기능 중에서 제어장치의 어떤 부분의 기능을 실행시키는가 하는 것이 PLC를 사용하는 방법으로 가장 중요한 점이다.

2) 하드 와이어드와 소프트 와이어드

종래의 릴레이 제어 방식은 일의 순서를 회로도에 전개하여 그곳에 필요한 제어기기를 결합하여 리드선으로 배선 작업을 하여 요구하는 동작을 실현하였다. 이 같은 방식을 하드 와이어드 로직(hardwired logic)이라고 한다.

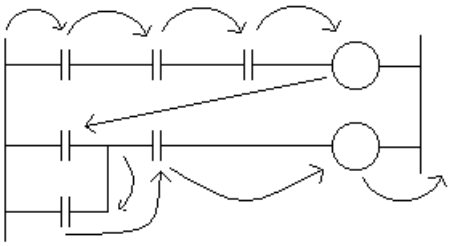
하드 와이어드 로직 방식에서는 하드(기기)와 소프트가 한 쌍이 되어 있어 사양이 변경되면 하드와 소프트를 모두 변경해야 하므로, 따라서 하드와 소프트를 분리하는 연구 끝에 컴퓨터 방식이 개발되었다. 컴퓨터는 하드웨어(hardware)만으로는 동작할 수 없다. 하드웨어 속에 있는 기억 장치에 일의 순서를 넣어야만 비로소 기대되는 일을 할 수가 있다. 이 일의 순서를 프로그램이라 하며, 또 기억 장치를 메모리라 하고, 이 메모리에 일의 순서를 넣는 작업을 프로그래밍이라 한다. 이는 마치 배선 작업과 같다고 생각하면 된다. 이 방식을 소프트 와이어드 로직(softwired logic)이라 한다. PLC는 이 소프트 와이어드 로직 방식을 취하고 있다.

3) 릴레이 시퀀스와 PLC 프로그램의 차이점

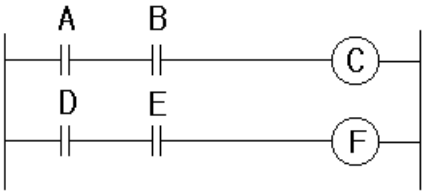
PLC는 LSI 등 전자 부품의 집합으로 릴레이 시퀀스와 같은 접점이나 코일은 존재하지 않으며, 접점이나 코일을 연결하는 동작은 소프트웨어로 처리되므로 실제로 눈에 보이는 것이 아니다. 또, 동작도 코일이 여자 되면 접점이 닫혀 회로가 활성화되는 릴레이 시퀀스와는 달리, 메모리에 프로그램을 기억시켜 놓고 순차적으로 내용을 읽어서 그 내용에 따라 동작하는 방식으로 사용자가 제어 로직에 맞게 프로그램의 내용을 마음대로 작성할 수 있는 차이점이 있다.

(1) 직렬 처리와 병렬 처리

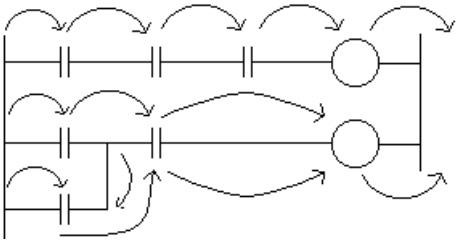
PLC제어와 릴레이 시퀀스의 가장 근본적인 차이점은 [그림 2-5]에 나타낸 것과 같이 "직렬처리와 병렬 처리"라는 동작상의 차이에 있다. PLC는 메모리에 있는 프로그램을 순차적으로 연산하는 직렬 처리 방식이고, 릴레이 시퀀스는 여러 회로가 전기적인 신호에 의해 동시에 동작하는 병렬 처리방식이다. 따라서 PLC는 어느 한 순간을 포착해 보면 한 가지 일밖에 하지 않는다.



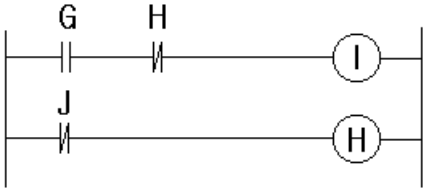
(a) 직렬 처리 방식



(a)



(b) 병렬처리 방식



(b)

[그림 2-5] 연산처리 방식

[그림 2-6] 시퀀스도

먼저 그림 2-5(a)의 시퀀스도로 PLC와 릴레이의 동작상의 차이점을 설명한다.
릴레이 시퀀스에서는 전원이 투입되어 접점 "A"와 "B", 그리고 접점 "D"와 "E"가 동시에 닫히면, 출력 "C"와 "F"는 동작하고 어느 한쪽이 빠를수록 먼저 동작한다. 이에 비하면 PLC는 연산 순서에 따라 "C"가 먼저 출력되고 다음에 "F"가 출력된다.
PLC와 릴레이의 동작상의 차이점을 그림 2-6(b)의 경우에서 살펴보면, 먼저 릴레이 시퀀스에서는 전원이 투입되면 접점 "J"가 닫힘과 동시에 "H"가 동작되어 출력 "I"는 동작될 수 없다. PLC는 직렬 연산 처리되므로 최초의 연산 때 "G"가 닫히면 "I"가 출력되고 "J"가 닫히

면 “H”가 출력된다. 둘째 번 연산 때 비로소 최초의 연산 때 출력된 “H”에 의해서 “I”의 출력이 해제된다.

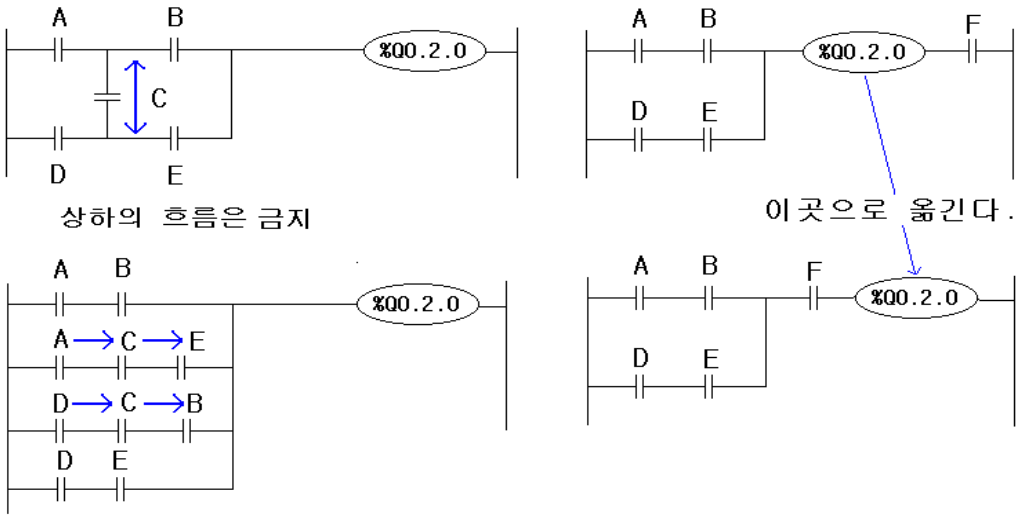
(2) 사용 접점 수의 제한

릴레이는 일반적으로 개당 가질 수 있는 접점 수에 한계가 있다. 따라서 릴레이 시퀀스를 작성할 때에는 가능한 접점 수를 절약해야 한다. 이에 비하여 PLC는 동일 접점에 대하여 사용 횟수에 제한을 받지 않으며 이는 접점에 대한 정보(ON/OFF)를 정해진 메모리에 저장해 놓고 연산할 때 메모리에 있는 정보를 읽어서 처리하기 때문이다.

(3) 접점이나 코일 위치의 제한

PLC는 릴레이 시퀀스에 없는 약속 사항이 있다. 그 하나는 코일 이후 접점을 금지하는 사항이며 출력코일을 반드시 오른쪽 모선에 붙여서 작성해야 한다. 그 밖에, PLC는 항상 신호가 왼쪽에서 오른쪽으로 전달되도록 구성되어 있어 릴레이 시퀀스와는 다르게 오른쪽에서 왼쪽으로 그리고 상하로 흐르는 회로 구성을 금지하고 있다.

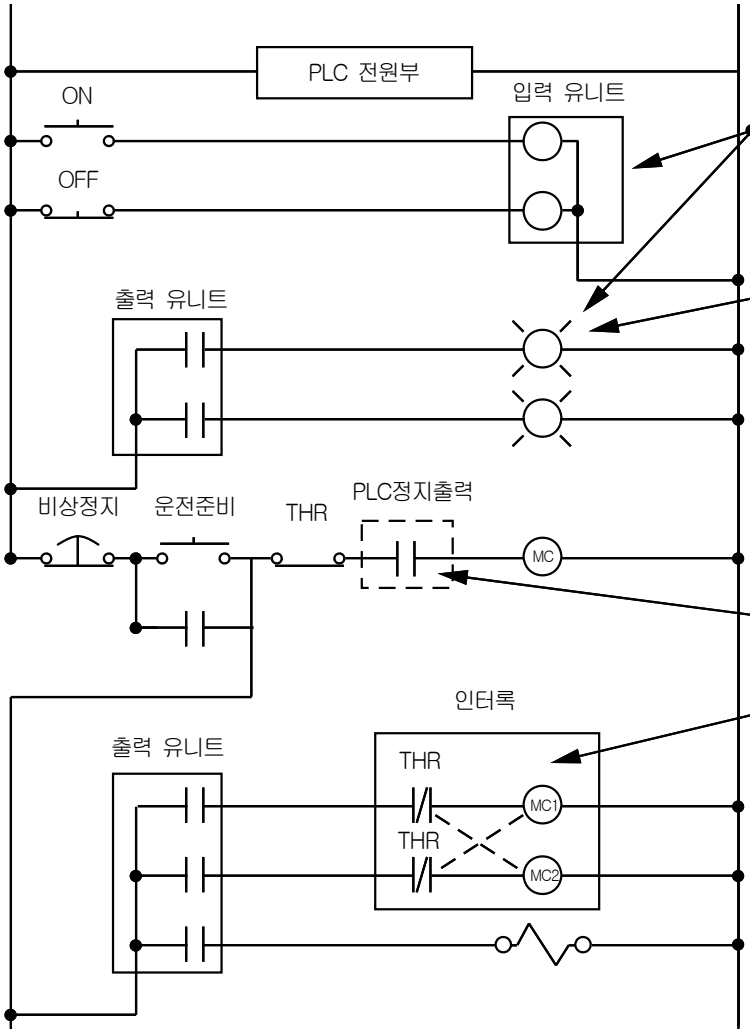
[그림 2-7]은 PLC 시퀀스의 약속사항을 나타내고 있다.



[그림 2-7] PLC 시퀀스의 약속 사항

(4) PLC 구성시 주의 사항

릴레이 회로의 경우 고장이 발생해도 그 이상 동작은 한정되지만 PLC의 경우는 시스템 전체의 이상 동작을 일으킨다. 이러한 점에서 제어 전체를 PLC에 맡기는 것은 좋은 방법이 아니며, 기계의 고장이나 전압 기기의 조작 회로 등은 PLC의 외부에서 구성 시켜야 한다. 또, PLC에 전원을 투입하는 순간에 출력 유니트의 출력이 ON으로 될 경우 사고를 일으키기 쉬우므로 아래와 같이 외부 출력기기 동작을 방지할 필요가 있다.



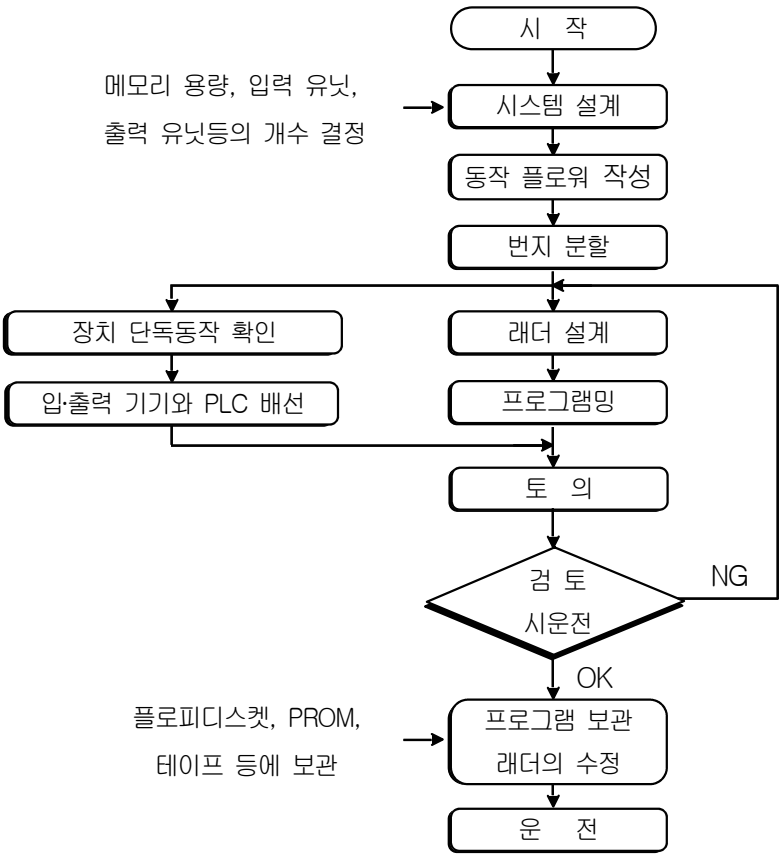
1. 입력유니트, 램프점등용 출력유니트는 비상정지회로 전에 접속시키는 것이 운전의 정지상태 등의 파악이 가능하다.

2. 출력유니트는 정지 직전에 대하여 ON/OFF 상태를 유지한다. 이 경우 시스템 메모리의 정전 유지 번지를 사용할 경우 점등 유지되고, 일반 번지를 사용하여 점등유지를 시킬 수도 있다.

3. 정지출력

4. 인터록 회로 : 정 운전, 역운전등의 상반되는 동작이나 기계의 파손, 인사사고로 이어지는 부분은 외부에서 인터록 회로를 구성하여야 한다.

(5) PLC 프로그래밍의 설계 순서



3장. PLC Program (GMWIN) Setup

3-1. PC의 사양 및 환경

GMWIN을 사용하려면 다음과 같은 하드웨어와 소프트웨어가 필요하다.

1) 컴퓨터와 메모리

586 이상의 CPU에 확장 메모리를 포함하여 적어도 16MB 이상의 메모리를 지닌 퍼스널 컴퓨터가 필요하다.

2) 시리얼 포트

GMWIN의 기능을 최대한 활용하고 PLC 본체와 통신을 하려면 적어도 두 개 이상의 시리얼 포트가 필요하다.

3) 하드 디스크

GMWIN과 관련된 파일을 모두 설치하고 GMWIN을 원활히 사용하려면 하드디스크에 사용 가능한 용량이 20MB이상 있어야 한다.

4) 플로피 디스크 드라이브

GMWIN을 설치하거나 작성한 데이터를 플로피 디스크에 저장하려면 적어도 하나 이상의 플로피 디스크 드라이브가 필요하다.(CD 드라이브 추천)

5) 마우스

GMWIN의 기능을 최대한 활용하려면 컴퓨터 본체에 연결할 수 있고 한글/영문 윈도우 95 이상의 버전에 적합한 마우스가 필요하다.

6) 프린터

GMWIN을 인쇄하려면 윈도우 95/98 이상에서 사용할 수 있는 프린터가 필요하다.

7) MS-DOS/한글 MS-DOS

사용하는 컴퓨터 본체에 적합하고 한글/영문 윈도우 3.1/95에서 실행되는 버전 MS-DOS 또는 한글 MS-DOS가 필요하다.

8) 한글/영문 윈도우 95/98

한글/영문 윈도우 95/98이상의 버전이 필요하다.

3-2. GMWIN의 설치

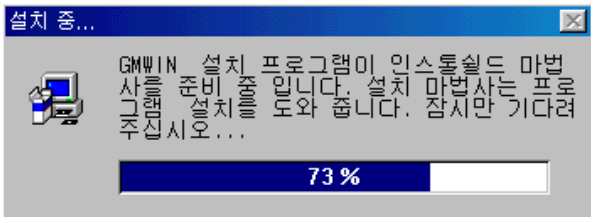
GMWIN을 설치하는데 필요한 프로그램은 CD에 저장되어 있으므로 CD드라이브에 CD를 삽입하고 SETUP.EXE를 실행시킨다.

1) 설치 방법

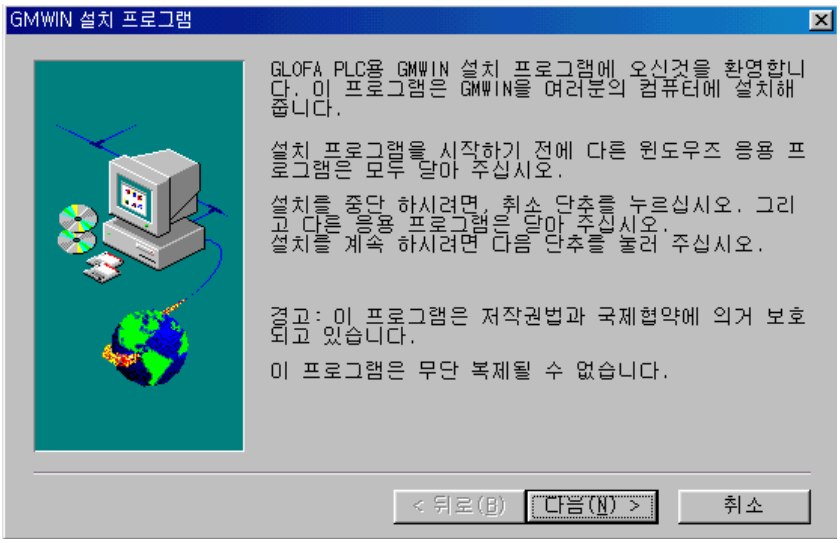
① CD드라이브에 CD를 삽입한다.

② 실행 명령어인 SETUP.EXE를 실행한다.

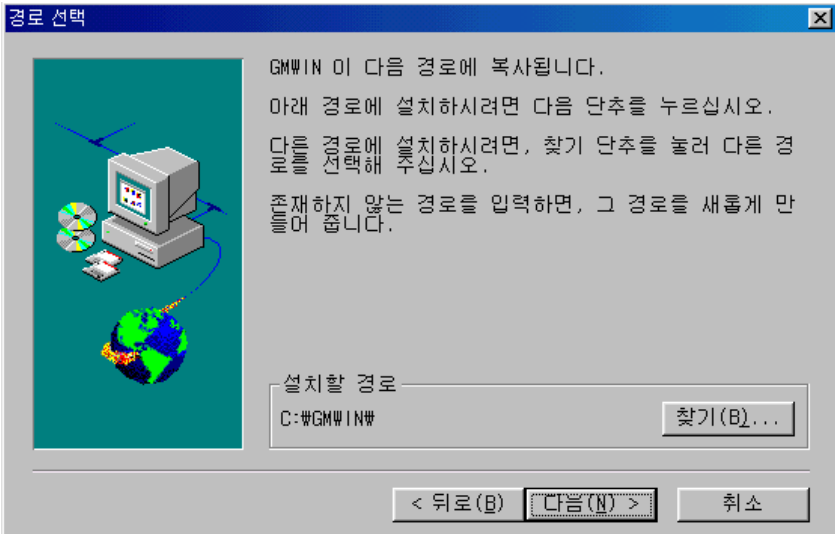
③ 설치로고 화면이 나타나면서, 설치 마법사가 설치를 준비한다.



④ 환영 메시지를 보여주는 대화상자가 나타난다. GMWIN 설치 중에는 다른 윈도우즈 응용 프로그램은 종료하는 것이 좋다.



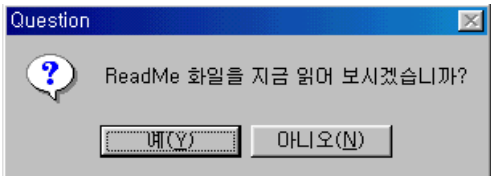
⑤ 다음(N) > 단추를 눌러 다음 화면으로 이동한다.



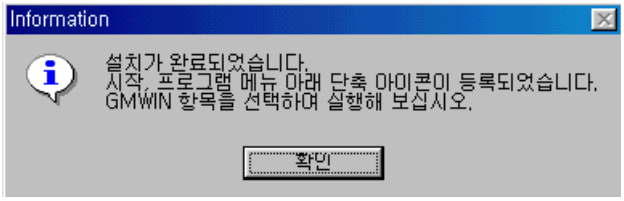
⑥ 설치할 경로를 보여주는 대화상자가 나타난다. 다른 경로로 설치하려면, 찾기(B)... 단추를 눌러 다른 경로를 선택한다. 설치를 중단하려면, 설치 중 어느 곳에서든 취소 단추를 누르면 된다. 물론 설치가 완료되지 않은 상태이므로 GMWIN을 실행해 볼 수 는 없다.

⑦ 디스크에 있는 파일들을 사용자의 컴퓨터에 복사하기 시작한다.

⑧ 복사가 완료되면, ReadMe 파일을 읽을지 묻는다. ReadMe 파일에는 GMWIN 버전이 대략적으로 소개되어 있다. ReaadMe 파일은 설치가 완료된 후, 나중에 읽어도 무방하다.



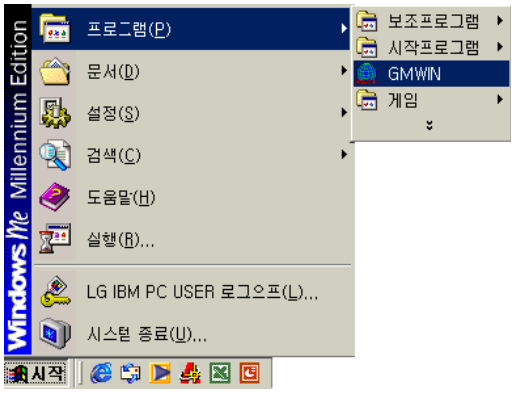
⑨ 시작의 프로그램 메뉴 아래 단축아이콘이 등록 되어있다. GMWIN 항목을 실행시킨다.



4장. Programming Tool (GMWIN)

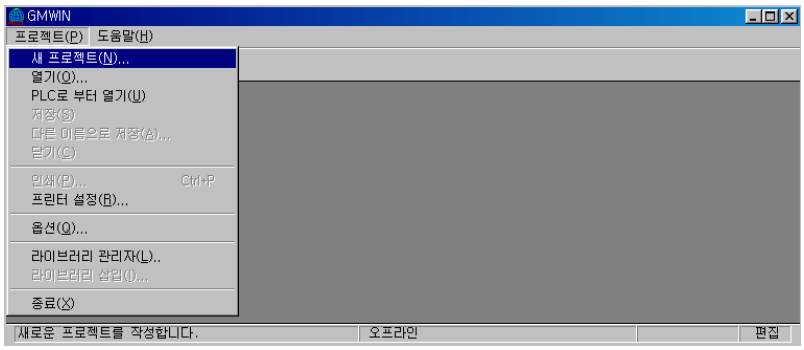
4-1. GMWIN의 기동

1) 윈도우의 시작 메뉴에서 - 프로그램 - GMWIN을 더블 클릭 하면 [그림 4-1]과 같은 창이 나타난다.



[그림 4-1] GMWIN의 프로그램 창

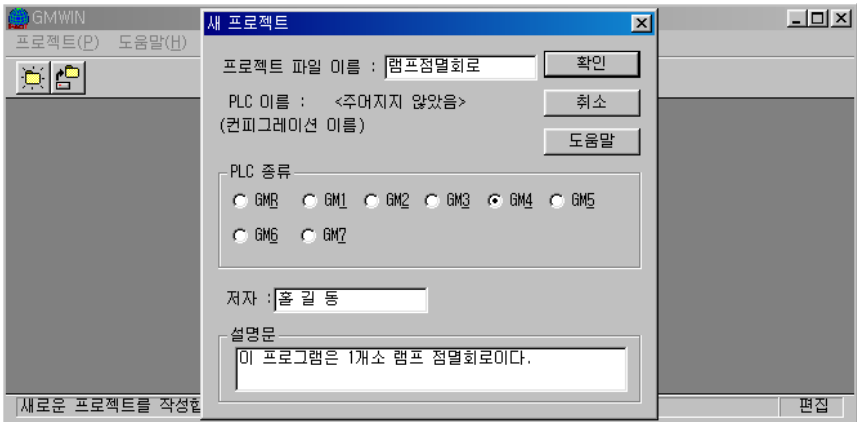
2) 1)항을 실행하면 아래와 같이 GMWIM 초기화면이 나오는데 새로운 프로그램을 작성하기 위하여 [그림 4-2]의 프로젝트에서 새 프로젝트를 클릭 한다.



[그림 4-2] GMWIN의 프로젝트 창

3) 프로젝트

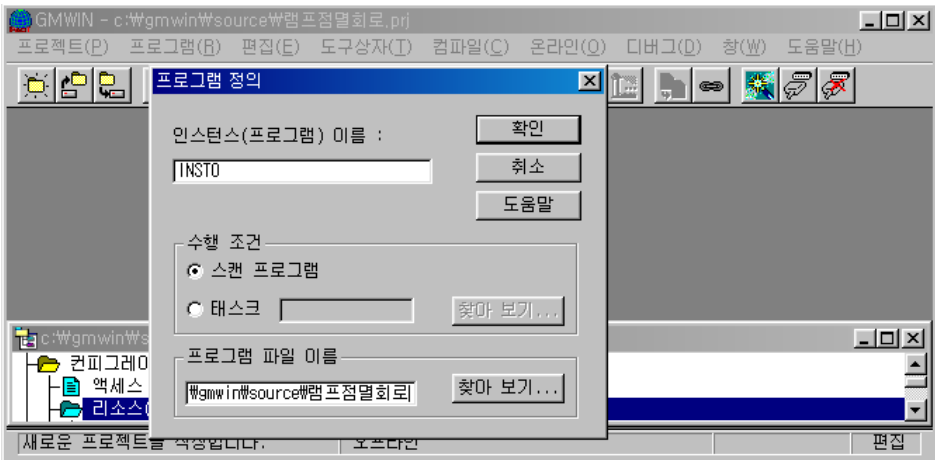
새 프로젝트 창 중 프로젝트 파일 이름은 사용자가 작성한 프로젝트 파일명이며 작성을 하지 않아도 디폴트 네임이 자동적으로 생성 된다. PLC의 종류는 사용하고자 하는 PLC의 종류를 선택하면 된다. 또한 저자, 설명문은 프로젝트 파일 이름에 대한 참고자료이며 기록을 하지 않아도 무방하다.[그림 4-3] 참조



[그림 4-3] GMWIN의 새 프로젝트 창

4) 프로그램 정의

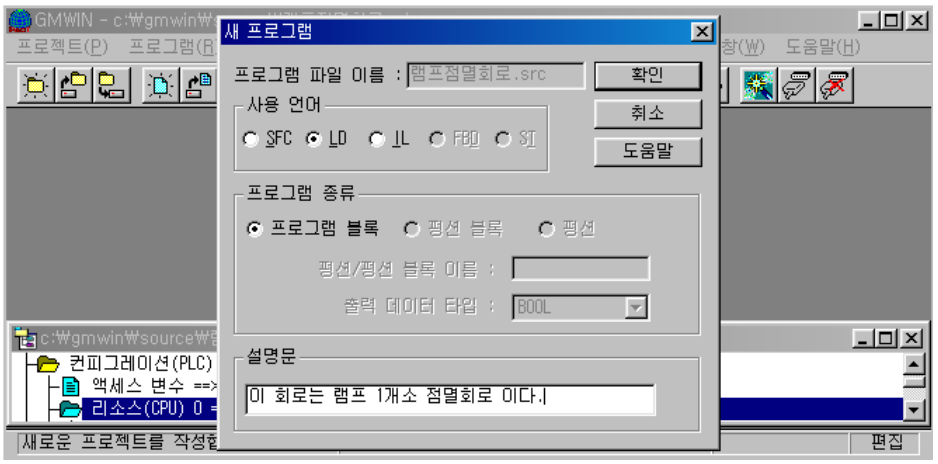
[그림 4-4]는 프로그램 정의 창이며 인스턴스는 사용자가 새로운 프로그램 명을 주어도 되지만, 때에 따라 인스턴스 이름을 기억해야만 파일을 열 수 있으므로 디폴트 값으로 사용하기를 추천한다. 수행조건에서 태스크는 프로그램을 실행시킬 조건을 정의할 때 사용하므로 조건에 관계없이 실행시키는 스캔 프로그램을 선택한다. 프로그램 파일 이름은 프로젝트 파일명과 동일한 파일명을 주면 사용이 편리하며, 이미 작성되어진 프로그램 파일은 찾아보기를 누르고 사용할 파일을 선택하면 된다.(컴파일 시 몇 개의 다른 확장자가 생기는데 프로젝트명과 프로그램명이 다르면 구분하기 곤란하다)



[그림 4-4] GMWIN의 프로그램 정의 창

5) 프로그램

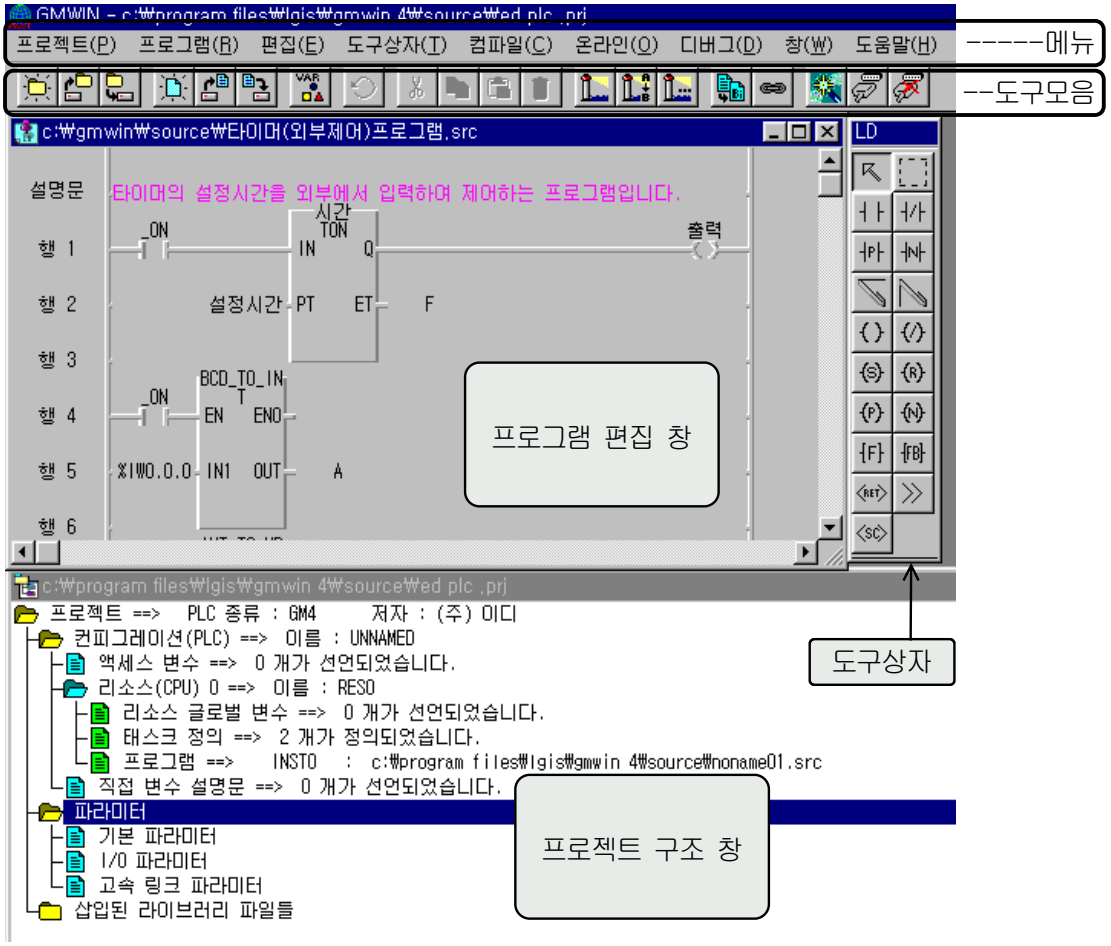
[그림 4-5]는 프로그램의 사용 언어를 지정하는 창으로 사용언어는 SFC, LD, IL 중 사용하기 편리한 기능을 선택하는데 본 장에서는 LD(Ladder Diagram)를 설명하고자 한다.



[그림 4-5] GMWIN의 새 프로그램 창

4-2. 화면구성

GMWIN 화면은 [그림 4-6]과 같이 프로그램 창과 도구상자 창 그리고 프로젝트 구조 창으로 구분 되어있으며 프로그램 창과 도구상자 창에 대해서는 다음 부분에서 자세히 설명하기로 한다.



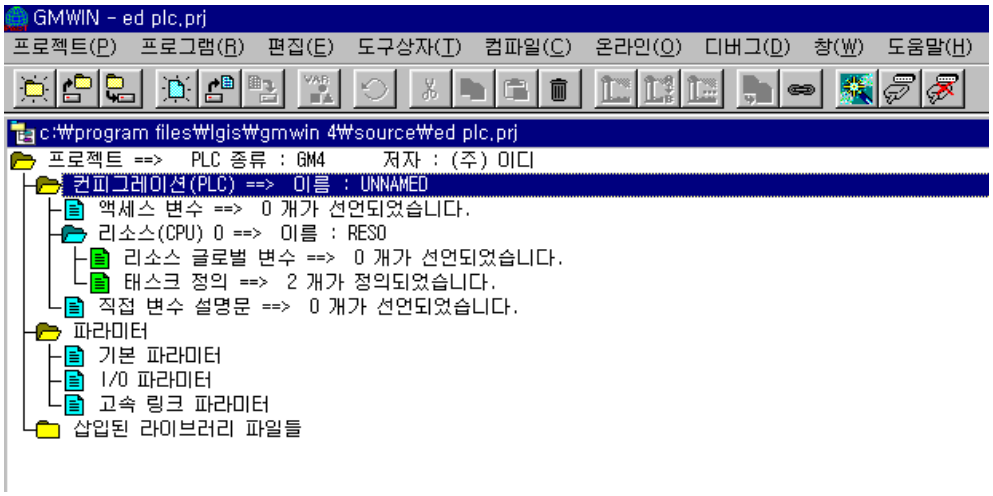
[그림 4-6] GMWIN의 화면구성

4-3. 프로젝트의 구조

프로젝트는 GLOFA PLC의 프로그램을 구성하는 가장 기본적인 요소로서 PLC 시스템당 하나의 프로젝트를 작성하는 것을 기본으로 한다.

프로젝트는 크게 컨피그레이션(Configuration)부, 파라미터(Parameter)부, 삽입된 라이브러리 파일들로 나눌 수 있으며, 컨피그레이션부는 글로벌(Global) 변수, 액세스(Access) 변수, 리소스(Resource) 내용 등 소프트웨어(Software)적인 것들을 작성하는 부분이고, 파라미터부는 기본 파라미터, I/O 파라미터, 링크 파라미터 등 하드웨어(Hardware)적인 것들을 작성하는 부분이다.

그리고, 삽입된 라이브러리 파일들에서 라이브러리 파일을 추가, 삭제할 수 있으며 프로젝트는 [그림 4-7]과 같은 구조이며 이들의 기능은 [표 4-8]과 같다.



[그림 4-7] 프로젝트의 구성화면

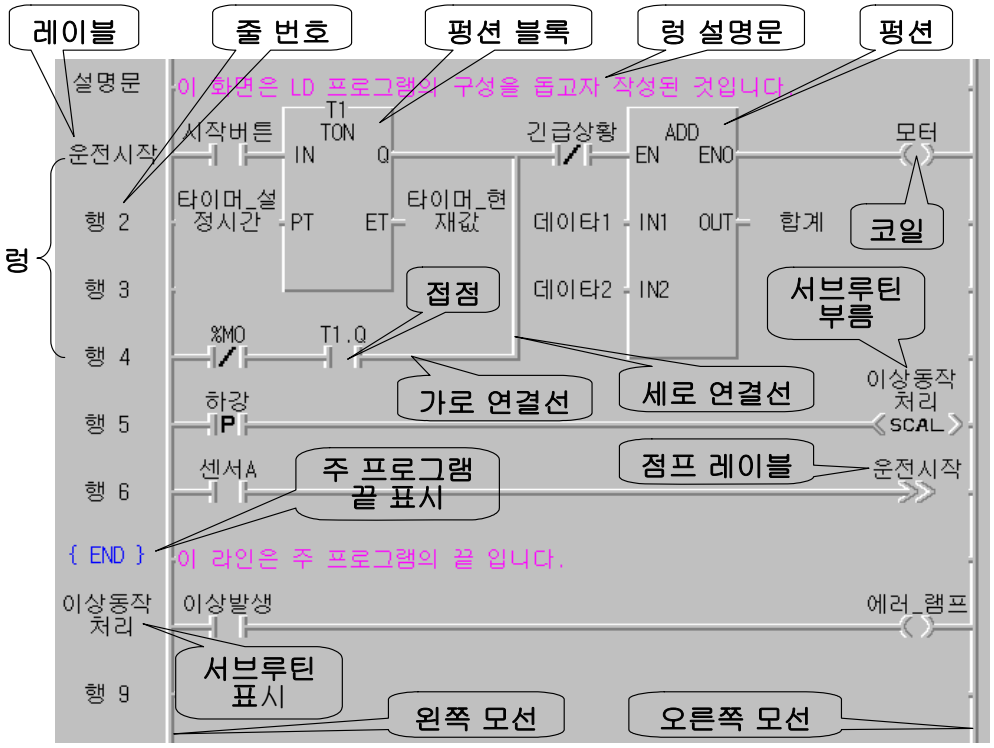
[표 4-8] 프로젝트 계층 구조

계층 항목	설 명
프로젝트	PLC시스템 전체를 정의
컨피그레이션	PLC프로그램에 관한 여러 정의 사항들을 설정
컨피그레이션 글로벌 변수	컨피그레이션 전체에서 사용되는 변수 리스트
액세스 변수	다른 컨피그레이션이 접근 가능한 변수 리스트
리소스	CPU 모듈에 해당
리소스 글로벌 변수	한 리소스 전체에서 사용되는 변수 리스트
태스크 정의	프로그램의 실행 조건 정의
프로그램 정의	각 프로그램과 그 실행 조건 기술
직접 변수 설명문	직접 변수에 사용한 설명문 리스트
파라미터	PLC시스템의 하드웨어에 관한 내용 정의
기본 파라미터	기본적인 하드웨어 파라미터 정의
I/O 파라미터	입·출력 모듈에 관한 내용 기술
고속 링크 파라미터	고속 링크 파라미터에 관한 내용 기술
삽입된 라이브러리 파일들	현재 삽입되어 있는 라이브러리 파일들의 리스트

4-4. LD 편집

LD 프로그램은 릴레이 논리 다이어그램에서 사용되는 코일이나 접점 등의 그래픽 기호를 통하여 PLC 프로그램을 표현하는 것이다.

아래 그림에서 '렁 설명문'이란 해당 렁에 관한 설명문이다. 렁이란 세로로 연결되어 있는 연속된 줄의 모양이다. 즉, 아래 그림에서는 행1에서 행4까지가 하나의 렁이 되며, 행5도 하나의 렁이 된다.



위의 그림에서 행 7의 **{END}**는 주 프로그램의 끝을 표시하는 것이다. 다음 행에 이어지는 **이상동작 처리**는 서브프로그램의 한 형태이며 행5에서 이 서브루틴을 부르고 있다.

도구상자의 임의의 요소를 선택하면 마우스의 모양이 그 요소 같은 모양으로 나타나며, 원하는 위치로 마우스를 옮긴 후에 마우스 왼쪽 단추를 누르면 접점이 생성 된다.

4-5. 업-로드(UP load)

압축된 프로젝트 파일과 소스 파일을 PLC의 RAM(프로그램 영역) 또는 플레시 메모리에 저장한 후 PLC로부터 프로그램을 업-로드 한다.

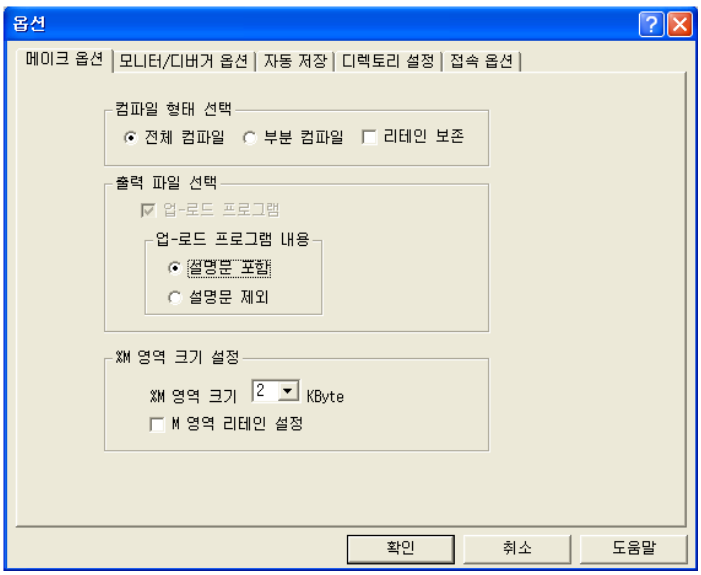
1) 업-로드 파일 만들기

옵션에서 메이크할 때 업-로드 파일을 만들도록 하고, 메이크 메뉴를 선택하면 업-로드 파일을 생성한다.

업-로드 파일은 프로젝트, 프로그램, 프로그램에서 사용한 사용자 펑션, 펑션블록이 포함되어 있다.

◆ 메뉴 **프로젝트 - 옵션**을 선택하여 옵션 대화상자를 부른다.

◆ 메이크 옵션 대화상자에서 업-로드 프로그램을 선택하고 확인 단추를 누른다. 변수 테이블은 선택할 필요가 없다.(추후 타 기기로 변수를 모니터하기 위해 마련)



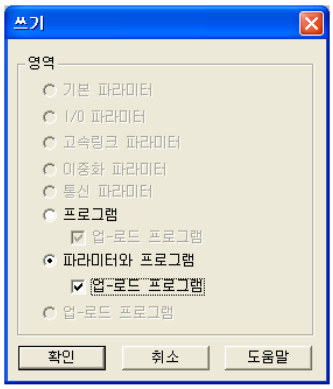
이때 PLC 프로그램 실행 파일과 업-로드 프로그램을 생성한다.

2) PLC에 쓰기

PLC에 프로그램을 쓸 때 업-로드 프로그램을 선택한다.

◆ 메뉴 **온라인 - 쓰기**를 선택한다.

◆ 쓰기 대화상자에서 파라미터와 프로그램, 업-로드 프로그램을 선택한다.



프로그램 크기에 따라 다음과 같이 실행한다.

프로그램 크기	업-로드 프로그램이 저장되는 위치
(실행프로그램 + 업-로드 프로그램) 크기 < 프로그램 램 크기	CPU의 램(REM)에 저장된다.
프로그램 램 크기 > (실행프로그램 + 업-로드 프로그램) 크기	플래시 메모리가 장착된 경우 사용자에게 확인한 후 플래시 메모리에 저장된다.

3) PLC로부터 읽기 (업-로드)

◆ 메뉴 프로젝트 - PLC로부터 열기를 선택합니다.

PLC로부터 업-로드 파일을 읽어서 프로젝트, 프로그램 파일, 사용자 라이브러리를 생성한 후 프로젝트를 연다.

GMWIN에 이미 같은 이름의 프로젝트나 프로그램이 있을 경우 Overwrite 시키거나, 사용자가 지정한 다른 디렉토리 또는 다른 이름으로 저장 한다.

4-6. 메뉴

1) 프로젝트

명 령	설 명
새 프로젝트	프로젝트를 처음 생성한다.
열기	기존의 프로젝트를 연다.
PLC로부터 열기	PLC에 있는 프로젝트 및 프로그램을 업-로드 한다.
저장	프로젝트를 저장합니다. 프로그램은 저장되지 않는다.
다른 이름으로 저장	프로젝트를 다른 이름으로 저장한다.
닫기	프로젝트를 닫는다.
프로젝트 항목 추가	프로젝트에 새로운 항목을 추가한다.
프로젝트 항목 수정	프로젝트에 속해 있는 항목을 편집한다.
프로젝트 항목 삭제	프로젝트에 속해 있는 항목(프로그램 정의, 리소스)을 삭제한다.
위로(프로그램)	프로젝트 창에서 위에 있는 프로그램 항목과 순서를 바꾼다.
아래로(프로그램)	프로젝트 창에서 아래에 있는 프로그램 항목과 순서를 바꾼다.
인쇄	활성화되어 있는 창의 내용을 인쇄한다.
프린터 설정	프린터 옵션을 설정한다.
옵션	GMWIN에 해당되는 옵션을 설정한다.
라이브러리 관리자	라이브러리를 편집한다.
라이브러리 삽입	새로운 라이브러리를 삽입한다.
시뮬레이터 시작	현재 열려 있는 프로젝트를 시뮬레이터 한다.
종료	GMWIN을 끝마친다.

2) 프로그램

명 령	설 명
새 프로그램	프로그램을 처음 생성한다.
열기	기존의 프로그램을 연다.
저장	프로그램을 저장한다.
다른 이름으로 저장	프로그램을 다른 이름으로 저장한다.
닫기	프로그램을 닫는다.
프로그램 속성	프로그램의 속성을 바꾼다.
지역 변수	변수를 편집한다.
입·출력 변수	펑션, 펑션 블록인 경우 입·출력 변수를 편집 한다

3) 편집

명 령	설 명
편집 취소	프로그램 편집 창에서 편집을 취소하고 바로 이전 상태로 되돌린다.
잘라 내기	블록을 잡아 삭제하면서 클립보드에 복사한다.
복사	블록을 잡아 클립보드에 복사한다.
붙여 넣기	클립보드로부터 편집 창에 복사한다.
삭제	Del 블록을 잡아 삭제한다.
찾기	원하는 문자를 찾는다.
바꾸기	원하는 문자를 찾아 새로운 문자로 바꾼다.
다시 찾기	이전에 실행한 찾기(Find) 또는 바꾸기(Replace)를 반복 실행한다.
찾아 가기	원하는 위치로 커서를 이동한다.
화면확대/축소	화면 크기를 조절한다.
변수 설명문	LD 창에서 변수 설명문을 본다.
라인 삭제	한 줄을 지운다.
셀 삭제	한 셀을 지운다.
라인 삽입	한 줄을 삽입한다.
셀 삽입	한 셀을 삽입한다.

4) 도구상자

명 령	설 명
도구상자 형태선택	도구상자를 없애거나 없어진 도구상자를 다시 나오게 한다. 또는 도구상자의 위치를 정해준다.

5) 컴파일

명 령	설 명
컴파일	프로그램을 컴파일 한다.
메이크	프로젝트에 속해 있는 프로그램 중 컴파일이 안 된 프로그램들을 컴파일 한 후 PLC 실행파일을 만든다.
모두 컴파일	프로젝트에 속해 있는 모든 프로그램을 컴파일 한 후 PLC 실행파일을 만든다.
메시지 보기	컴파일 후 에러 메시지를 본다.
메모리 참조	사용된 글로벌 변수 및 직접 변수를 볼 수 있다.

6) 온라인

명 령		설 명
접속 + 쓰기 + 모드전환(런) + 모니터시작(G)		GMWIN과 옵션에서 지정한 PLC를 접속시켜 사용자가 작성한 프로그램을 PLC에 쓴 후 모드를 절환 하여 모니터링 한다.
접속		GMWIN과 옵션에서 지정한 PLC를 접속시킨다.
접속 끊기		GMWIN과 PLC 접속을 해제한다.
읽기		PLC의 데이터를 읽어 온다.
쓰기		GMWIN의 프로그램을 PLC에 쓴다.
모니터	모니터 시작/끝	프로그램을 모니터링 한다. / 모니터링을 끝낸다.
	변수 모니터	변수만 모니터링 한다.
	I/O 모니터	I/O를 모니터링 한다.
	타임차트	BOOL 변수에 대하여 타임차트 형식으로 모니터링 한다.
	링크 파라미터	고속 링크 파라미터를 모니터링 한다.
모드 전환		PLC 모드를 전환한다.
데이터 클리어		PLC 데이터를 "0"으로 지운다.
CPU 전환		GM1에서 통신할 CPU를 전환한다.(GM1기능)
리셋		PLC의 CPU를 리셋 한다.
플래시메모리		CPU에 장착된 플래시메모리의 타입정보를 읽거나 플래시메모리에 데이터 쓰기를 한다.
링크 허용설정		고속 링크 허용을 금지한다.
PLC 정보		PLC 정보를 보여준다.
I/O 정보		PLC I/O 구성 상태를 보여준다.
강제 I/O 설정		I/O 강제 입·출력 값/실행 허용을 설정한다.
링크 정보		링크 모듈의 타입, 장착, 슬롯, 국번 등을 보여준다.
Mnet 파라미터		Mnet 파라미터를 입력한다.
Mnet 정보		Mnet 정보를 본다.
I/O Skip		스킵할 I/O를 지정한다.

7) 디버그

명 령	설 명
디버그 시작/끝	디버그 모드로 전환하여 디버그를 시작합니다. / 디버그를 끝낸다.
런	브레이크 포인트까지 런 시킨다.
스텝 오버	한 스텝씩 런 시킨다.
스텝 인	평션, 평션 블록을 디버깅한다.
스텝 아웃	평션, 평션 블록을 디버그 시 현재 블록을 빠져 나간다.
일시 정지	런을 중지시킨다.
커서 위치까지 런	커서 위치까지 런 시킨다.
브레이크 포인트 설정/해제	브레이크, 포인트를 설정 또는 해제한다.
브레이크 포인트 목록/조건	설정된 브레이크 포인트의 목록을 보여주고 브레이크 조건을 설정한다.
태스크 수행 설정	디버깅 중 태스크 전환을 허용한다.

8) 창

명 령	설 명
계단식 배열	GMWIN에 속해 있는 여러개의 창들을 계단식으로 배열한다.
수평 배열	GMWIN에 속해 있는 여러개의 창들을 수평으로 배열한다.
수직 배열	GMWIN에 속해 있는 여러개의 창들을 수직으로 배열한다.
아이콘 정렬	GMWIN에 속해 있는 아이콘들을 정열 한다.
모두 닫기	GMWIN에 속해 있는 여러개의 창들을 모두 닫는다.

4-7. 도구모음

GMWIN에서 사용되는 도구들이다.

GMWIN에서 자주 사용되는 메뉴들을 단축 형태인 도구로 제공하고 있다. 원하는 도구를 마우스로 선택하면 실행되며 아래 표에서는 아이콘의 모양을 설명하고 있다.

아이콘	명 령	설 명
	새 프로젝트	- 새로운 프로젝트를 생성한다. - 메뉴 - 프로젝트 - 새 프로젝트...( )를 선택한다.
	프로젝트열기	- 이미 작성된 프로젝트를 연다. - 메뉴 - 프로젝트 - 열기 - ...( )를 선택한다.
	프로젝트저장	- 작성한 프로젝트 파일을 저장한다. - 메뉴 - 프로젝트 - 저장 - ...( )을 저장한다.
	새 프로그램	- 한 프로젝트 안에 1개 이상의 프로그램을 열 때 사용한다. - 메뉴 - 프로그램 - 새 프로그램
	프로그램열기	- 메뉴에서 선택한 프로그램을 연다. - 메뉴 - 프로그램 - 열기...( )를 선택한다. [알아두기] - 수정 모드나 디버그 모드에서는 같은 프로그램을 두 개 이상 열 수 없다. - 모니터 모드인 경우에는 긴 프로그램을 한 화면에 볼 수 없는 경우에 대비하여 같은 프로그램을 두개 이상 열고 각각 다른 부분을 모니터할 수 있다.
	프로그램저장	- 작성한 프로그램을 파일로 저장한다.. - 메뉴 - 프로그램 - 저장...( )을 선택한다.
	변수목록	- 현재 활성화된 프로그램 윈도우에 해당하는 지역 변수(Local Variable)를 편집 할 수 있다. - 프로그램 - 지역 변수...( )를 선택한다. - 각 변수를 추가, 삭제, 수정한 후 닫기 단추를 클릭 한다.
	편집취소	- 프로그램 작성 중 앞서 편집된 동작을 취소 하고자 하는 기능이다. - 메뉴 - 편집 취소(Ctrl+Z)를 선택한다.
	잘라내기	- 잘라내기를 하고자 하는 부분을 블록 지정한 상태에서 - 메뉴 - 편집 - 잘라내기(Ctrl+X)를 선택한다.
	복사	- 복사 하고자 하는 부분을 블록 지정한 상태에서 - 메뉴 - 편집 - 복사(Ctrl+C)를 선택한다.

아이콘	명 령	설 명
	붙여넣기	- 잘라내기나 복사를 한 다음 - 붙여넣기를 실행할 위치로 커서를 이동한다.. - 메뉴 - 편집 - 붙여넣기(Ctrl+V)를 선택한다.
	삭제	- 블록지정 아이콘으로 삭제하고자 하는 부분을 지정한 다음 삭제 아이콘을 사용하여 삭제한다.
	찾기	- 현재 편집하고 있는 프로그램 중에서 인스트럭션이나 피연산자를 찾는다. - 메뉴 - 편집 - 찾기...( )를 선택한다. - 찾기 대화 상자의 찾고자 하는 문자열 입력란에 찾을 문자열을 입력한다. - 다음의 옵션을 선택하고 확인 단추를 누른다. 1) 종류 : 찾을 문자열의 종류를 선택한다. 2) 범위 : 찾기를 실행할 범위를 선택한다. - 커서부터 : 현재 커서 위치부터 찾는다. - 전체 : 프로그램 전체를 찾는다. 3) 방향 : 찾기를 실행할 방향을 선택한다. - 아래로 : 아래 방향으로 찾는다. - 위 로 : 위 방향으로 찾는다. 4) 단어 : 찾을 단어의 일치정도를 선택한다. - 전체일치 : 전부 일치하는 단어를 찾는다. - 부분일치 : 부분적으로 일치해도 찾는다.
	바꾸기	- 메뉴 - 편집 - 바꾸기...( )를 선택하여 바꾸기 대화 상자를 부른다. - 이름 입력란에서 바꿀 문자를 입력한다. - 접점/코일 목록상자에서 바꿀 LD 프로그램 요소를 선택한다. - 새 이름 입력란에 새로 대치하고자 하는 문자를 입력한다. - 새 접점/코일 목록상자에서 새로 대치하고자 하는 LD 프로그램 요소를 선택한다. - 나머지 옵션은 찾기에서와 같이 선택하고, 확인 단추를 누른다.
	다시 찾기	- 이전에 찾기/바꾸기를 실행했을 경우, 이전에 설정되어 있는 찾기/바꾸기 조건대로 다시 찾기/바꾸기를 실행한다. - 메뉴 편집 - 다시 찾기(Ctrl+F3)를 선택한다.
	컴파일	- 현재 활성화된 프로그램 윈도우에 대해서만 컴파일을 실행하여 오브젝트 파일을 생성한다. - 메뉴 컴파일 - 컴파일...( )을 선택한다. [알아두기] 프로그램 컴파일만 실행하면 실행파일은 생성되지 않는다.
	메이크 실행파일 만들기	- 프로젝트에 속해 있는 프로그램 중 컴파일 할 필요가 있는 프로그램만 컴파일한 후 실행 파일을 생성한다. - 메뉴 - 컴파일 - 메이크...( )를 선택한다.

아이콘	명 령	설 명
	접속+쓰기+모드전환+모니터시작	- 작성된 프로그램을 한번의 메뉴 조작으로 실행하는 매크로 명령이다. - 이 명령이 실행되면 접속, 프로그램 쓰기, 모드전환(런), 모니터 시작을 한번 조작으로 실행 가능하다.
	접속	- GMWIN과 PLC와의 접속을 한다. - 메뉴 - 온라인 - 접속을 선택한다.
	런	- RUN 모드 : 프로그램을 정상적으로 수행하는 모드 - 메뉴 - 온라인 - 모드전환 - 런
	스톱	- STOP모드 : 프로그램을 연산하지 않고 정지 상태인 모드 - 메뉴 - 온라인 - 모드전환 - 스톱
	일시정지	- PAUSE 모드 : 프로그램의 연산이 일시 정지된 모드 - 메뉴 - 온라인 - 모드전환 - 일시정지
	디버그 시작	- DEBUG 모드 : 프로그램의 오류를 찾거나, 연산과정을 추적하기 위한 모드 - 메뉴 - 온라인 - 모드전환 - 디버그
	디버그 런	브레이크 포인트까지 런 시킨다.
	오버 스텝	한 스텝씩 런 시킨다.
	스텝 인	펑션, 펑션 블록을 디버깅한다.
	스텝 아웃	펑션, 펑션 블록 디버그 시 현재 블록을 빠져 나간다.
	일시정지	런을 일시 중지시킨다.
	커서위치까지 런	커서 위치까지 런 시킨다.
	브레이크 포인트 설정/해제	브레이크 포인트를 설정 또는 해제한다.
	접속 끊기	- GMWIN과 PLC의 접속을 끊는다. - 메뉴 - 온라인 - 접속 끊기를 선택한다.

아이콘	명 령	설 명
	쓰기	- GMWIN의 파라미터 및 프로그램을 PLC로 쓰는 기능이다. - 메뉴 - 온라인 - 쓰기를 선택하여 쓰기 대화 상자를 부른다. (접속 실행 후 실시) - 현재 PLC의 상태가 Run인 경우에는 아래 화면이 나타난다. - 기본 파라미터 : 기본 파라미터만 PLC로부터 읽는다. - I/O 파라미터 : I/O 파라미터만 PLC로부터 읽는다. - 고속 링크 파라미터 : 고속 링크 파라미터만 PLC로부터 읽는다. - 이중화 파라미터 : 이중화 파라미터만 PLC로부터 읽는다. (이중화 선택시만 활성화) - 프로그램 : 프로그램만 PLC로부터 읽는다. - 파라미터와 프로그램 : 파라미터들과 프로그램을 함께 PLC로부터 읽는다. - 업-로드 : 업-로드 프로그램을 PLC로부터 읽는다.
	모니터 시작/끝	- GMWIN에서는 운전 중인 PLC 연산 상태를 모니터링 할 수 있다. - 온라인 - 모니터 - 모니터시작 1) 프로그램 모니터링 2) I/O 모니터링 3) 변수 모니터링 4) 타임 차트 모니터링 5) 링크 파라미터 모니터링

4-8. GMWIN에서 생성되는 파일

사용자가 프로젝트를 생성하고 프로그램을 편집하여 PLC실행 파일을 만들면 다음과 같은 확장자를 가진 파일이 만들어진다.

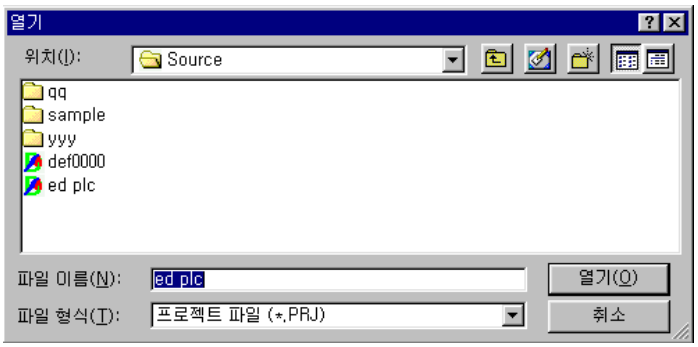
① * . PRJ : 사용자가 작성한 프로젝트 파일
② * . BNO : PLC 실행 파일
③ * . MON : 모니터링을 위한 정보 파일
④ * . CRO : PLC 실행 파일을 만들 때 생성.
⑤ * . SRC : 사용자가 작성한 프로그램 파일
⑥ * . ASV : 사용자가 작성한 프로그램을 이 이름으로 주기적으로 저장한다.
⑦ * . OP? : 프로그램을 컴파일하면 생성된다.(프로그램 블록인 경우)
⑧ * . OB? : 프로그램을 컴파일하면 생성된다.(펑션 블록인 경우)
⑨ * . OP? : 프로그램을 컴파일하면 생성된다.(펑션인 경우)

4-9. 파일 열기

사용자가 기존의 프로젝트를 열고 프로그램을 작성하려면 파일을 열어야 한다.

1) 프로젝트를 열 경우

◈ 메뉴 프로젝트 - 열기를 선택한다.



2) 프로그램을 열 경우

① 메뉴 - 프로그램 - 열기를 선택한다.

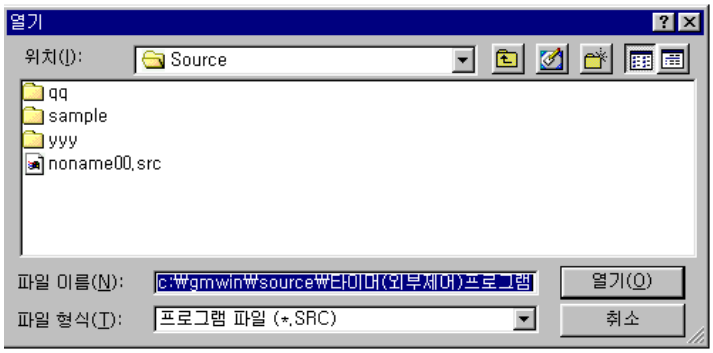
② 파일이 있는 위치를 지정하기 위하여 위치 목록 상자에서 드라이브 및 디렉토리를 선택한다.

③ 파일 이름 입력란에 파일 이름을 직접 입력하거나 목록상자에서 선택한다.

파일형식 목록상자에서 선택된 확장자를 가진 파일만 나타난다.

◈ 프로젝트 파일 : *.PRJ　　프로그램 파일: *.SRC

④ 열기 단추를 누른다.



4-10. 파일 저장

1) 새로운 파일 저장

◈ 한 번도 저장하지 않은 새 파일을 저장한다.

2) 프로젝트를 저장할 경우

◈ 메뉴 - 프로젝트 - 저장을 선택한다.

3) 프로그램을 저장할 경우

(1) 메뉴 - 프로그램 - 저장을 선택한다.

(2) 파일이 저장될 위치를 지정하기 위하여 위치 목록상자에서 드라이브 및 디렉토리를 선택한다.

(3) 파일이름 입력란에 파일 이름을 입력한다.
프로젝트 파일에는 PRJ, 프로그램 파일에는 SRC의 확장자를 입력한다.

(4) 저장 단추를 누른다.

4) 작업 중 파일 저장

(1) 프로젝트를 저장할 경우

◈ 메뉴 - 프로젝트 - 저장을 선택한다.

(2) 프로그램을 저장할 경우

◈ 메뉴 - 프로그램 - 저장을 선택한다.

5) 다른 이름으로 저장

프로젝트 이름 또는 프로그램 이름을 변경한다.

(1) 프로젝트를 저장할 경우

◈ 메뉴 - 프로젝트 - 다른 이름으로 저장을 선택한다.

(2) 프로그램을 저장할 경우

① 메뉴 - 프로그램 - 다른 이름으로 저장을 선택한다.

② 파일이 저장될 위치를 찾기 위하여 위치 목록상자에서 드라이브 및 디렉토리를 선택한다.

③ 파일이름 입력란에 파일 이름을 입력한다.

프로젝트 파일에는 PRJ, 프로그램 파일에는 SRC 확장자를 입력한다.

④ 저장 단추를 누른다.

6) 파일 닫기

(1) 방법 1 : 해당 창의 왼쪽 위 모서리의 조절메뉴 상자를 두 번 누른다.

(2) 방법 2

① 프로젝트를 닫을 경우

◈ 메뉴 - 프로젝트 - 닫기를 선택한다.

② 프로그램을 닫을 경우

◈ 메뉴 - 프로그램 - 닫기를 선택한다.

5장. 데이터의 구성

5-1. 변수 표현 방식

프로그램 안에서 사용되는 데이터는 값을 가지고 있는데, 프로그램이 실행되는 동안에 값이 바뀌지 않는 상수와 그 값이 변하는 변수가 있다. 프로그램 블록, 펑션, 펑션 블록 등의 프로그램 구성 요소에서 변수를 사용하기 위해서는 우선 변수의 표현 방식을 결정해야 한다. 변수 표현 방식은 직접변수와 네임드 변수의 두 가지로 구분된다.

① **직접변수** : 변수 선언 불필요(종래의 PLC 방식)
② **네임드 변수** : 변수 선언 필요

첫 번째 변수 표현 방식은 사용자가 이름을 부여하지 않고 이미 Maker에 의해 지정된 메모리 영역의 식별자를 사용하는 직접변수 방식이고, 두 번째 변수 표현 방식은 사용자가 이름으로 부여하고 사용하는 네임드(Named) 변수방식이다.

1) 직접변수

직접변수에는 **%I**, **%Q**의 입·출력 변수와 **%M**의 내부 메모리 변수가 있다.

직접변수는 반드시 퍼센트 문자(%)로 시작하고 다음에 위치 접두어와 크기 접두어를 붙이며, 그리고 마침표로 분리되는 하나 이상의 부호 없는 정수의 순으로 나타낸다.

(1) 직접변수 지정의 예

입력 변수 지정 : %I0.0.0 , %I0.0.1 , %I0.0.2 등
출력 변수 지정 : %Q0.2.0 , %Q0.2.1 , %Q0.2.2 등
내부메모리 변수 지정 : %M0 , %M1 , %M2 등

(2) 위치 접두어

번 호	접두어	의 미
1	I	입력위치(Input Location)
2	Q	출력위치(Output Location)
3	M	내부 메모리 위치(Memory Location)

(3) 크기 접두어

번 호	접두어	의 미
1	X	1 비트의 크기(X 문자에 한하여 생략 가능)
2	B	1 바이트(8 비트)의 크기
3	W	1 워드(16 비트)의 크기
4	D	1 더블워드(32 비트)의 크기
5	L	1 롱 워드(64 비트)의 크기

※ 직접변수의 입·출력 표시형식

% [위치 접두어] [크기 접두어] [베이스 번호]. [슬롯 번호]. [접점 번호]

입 력 % I X 0. 0. 0
출 력 % Q X 0. 3. 0

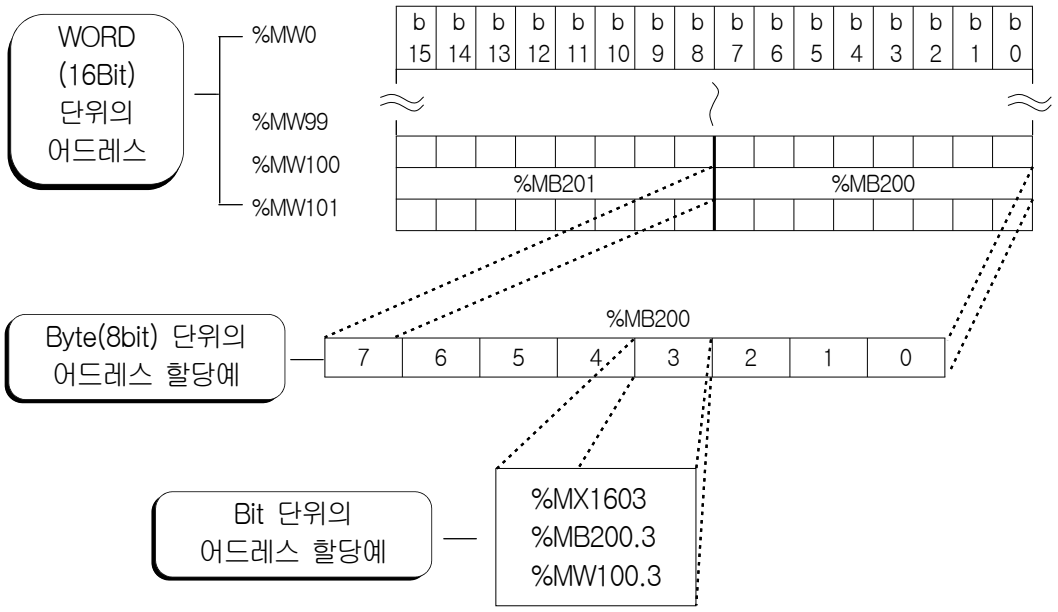
- 입·출력 모듈의 접점번호를 나타내며, 최대 0~63의 값을 갖는다.
- 입·출력 모듈이 장착된 슬롯번호를 나타내며, 0~7의 값을 갖는다.
- 베이스 번호 : 0~3의 값을 갖는다.
- 크기접두어 : X는 1비트의 크기를 나타낸다.
- 위치접두어 : I(입력), Q(출력), M(내부메모리)으로 나타낸다.
- 직접변수임을 나타내는 예약어

EX) %IX0.0.3 : 직접변수, 입력, 비트단위, 0번 베이스, 0번 슬롯, 3번 접점
%QX0.2.7 : 직접변수, 출력, 비트단위, 0번 베이스, 2번 슬롯, 7번 접점

(4) 내부 메모리

번호	내부메모리	의 미
1	%MX0	0의 위치에 있는 비트 단위의 접점 번호를 나타낸다.
2	%MB1	1의 위치에 있는 바이트 단위의 메모리를 나타낸다.
3	%MD48	48의 위치에 있는 더블 워드 단위의 메모리를 나타낸다.
4	%MW20.3	20의 위치에 있는 워드 단위의 메모리중 3번 비트를 나타낸다.

(5) 내부 메모리 M



2) 네임드(Named) 변수

네임드 변수는 사용자가 변수 이름과 형 등을 선언하고 사용한다.

① 네임드 변수의 이름은 한글은 8자, 영문은 16자 까지 선언 가능하며

② 한글 영문, 숫자 및 밑줄(_)문자를 조합하여 사용 할 수 있다.

③ 영문의 경우 대소문자를 구별하지 않고 모두 대문자로 인식하며 빈칸을 포함하지 않아야 한다.

(1) 네임드 변수의 예

종 류	사 용 예
한글, 숫자 및 밑줄 문자	모터10, 디지털_스위치1, 누름_검출, 수동_배출_스위치밸브1, 설비_자동_운전중, 사이클_정지_완료
한글, 영문, 숫자 및 밑줄 문자	AGV_주행_완료, 모터2_ON, BCD값, VAL2, 자동_SOL배출

(2) 네임드 변수의 종류

번 호	변수 종류	의 미
1	VAR	읽고 쓸 수 있는 일반적인 변수
2	VAR_RETAIN	정전 시 값이 유지되는 변수
3	VAR_CONSTANT	읽기만 할 수 있는 변수
4	VAR_EXTERNAL	외부 변수(VAR_GLOBAL)임을 지정하는 변수

(3) Named(네임드) 변수의 데이터 형(Type)(데이터의 고유 성질을 나타냄)

① 데이터형은 크게 수치(ANY_NUM)와 비트 상태(ANY_BIT)로 구분할 수 있다.

② 수치의 대표적인 경우는 정수(INT: Interger)인데 셀 수 있고 산술 연산을 할 수 있다.

③ 정수의 예는 카운터의 현재값, A/D(아날로그 입력) 변환값 등이 있다.

④ 비트 상태는 BOOL(Boolean: 1 비트), BYTE(8 개의 비트 열), WORD(16 개의 비트열) 등이 있는데 비트 열의 ON/OFF 상태를 나타내며 논리 연산을 할 수 있다.

⑤ 비트 상태의 예는 입력 스위치의 ON/OFF 상태, 출력 램프의 소등/점등 상태 등이 있다.

⑥ BCD는 10 진수를 4비트의 2진 코드로 나타낸 것이므로 비트 상태(ANY_BIT)이다.

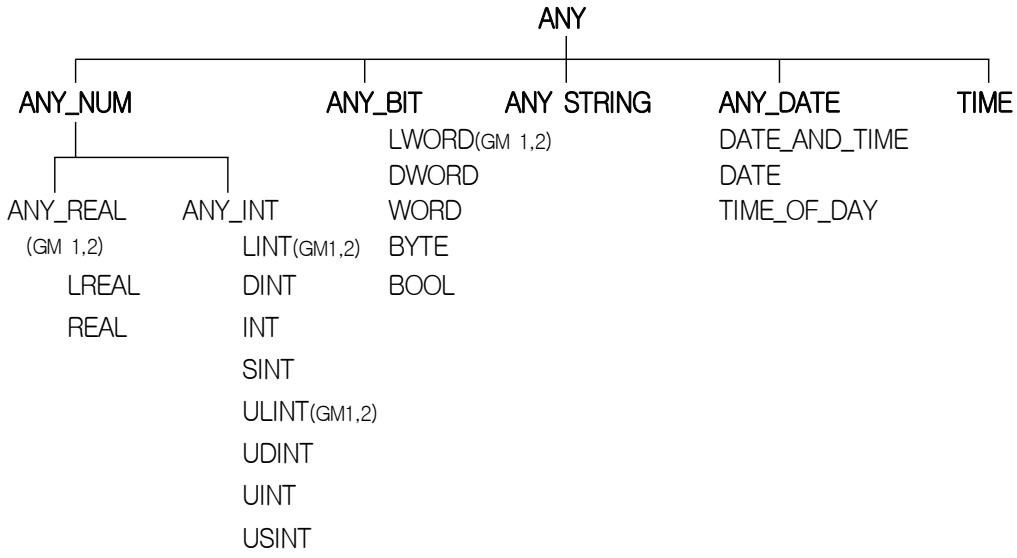
⑦ 비트 상태는 산술연산이 불가능하지만 형(Type) 변환 펑션을 사용하여 수치로 변환하면 산술 연산이 가능하다.

(4) 기본 데이터 형

구 분	예약어	데이터 형	크기 (비트)	범 위
수치 (ANY_ NUM)	SINT	Short Integer	8	-128 ~ 127
	INT	Integer	16	-32768 ~ 32767
	DINT	Double Integer	32	-2147483648 ~ 2147483647
	LINT	Long Integer	64	-263 ~ -263 ~ -1
	USINT	Unsigned Short Integer	8	0 ~ 255
	UINT	Unsigned Integer	16	0 ~ 65535
	UDINT	Unsigned Double Integer	32	0 ~ 4294967295
	ULINT	Unsigned Long Integer	64	0 ~ 264 ~ -1
	REAL	Real Numbers	32	-3.402823E38 ~ -1.401298E-451. 401298E-45 ~ 3.402823E38
	LREAL	Long Reals	64	-1.7976931E308 ~ -4.9406564E-324 4.9406564E-324 ~ 1.7976931E308
시간	TIME	Duration	32	T#OS ~ T#49D17H2M47S295MS
날짜	DATE	Date	16	D#1984-01-01 ~ D#2163-6-6
	TIME_OF_DAY	Time Of Day	32	TOD#00:00:00 ~ TOD#23:59:59.999
	DATE_AND _TIME	Date And Time Of Day	64	DT#1984-01-01-00:00:00 ~ DT#2163-12-31-23:59:59.999
문자열	STRING	Character String	30*8	-
비트상태 (ANY_ BIT)	BOOL	Boolean	1	0,1
	BYTE	Bit String Of Length 8	8	16#0 ~ 16#FF
	WORD	Bit String Of Length 26	16	16#0 ~ 16FFFF
	DWORD	Bit String Of Length 32	32	16#0 ~ 16FFFFFFFF
	LWORD	Bit String Of Length 64	64	16#0 ~ 16FFFFFFFFFFFFFFFF

(5) 데이터 형(Type) 계층도

GLOFA PLC에서 사용되는 데이터 타입은 다음과 같이 분류하여 표현한다.



① ANY_REAL(LREAL, REAL) 및 LINT, ULINT, LWORD는 GM1, GM2만 적용된다.

② 앞으로 데이터 타입을 표현할 때 ANY_NUM으로 나타내면 위의 계층도와 같이 LREAL, REAL, LINT, DINT, INT, SINT, ULINT, UDINT, UINT, USINT를 모두 포함한다.

③ 예를 들어 GM3에서 타입이 ANY_BIT로 표현되면, DWORD, WORD, BYTE, BOOL 중 하나를 사용할 수 있다.

(6) 초기값(데이터의 초기값을 지정하지 않으면 자동적으로 아래와 같이 지정된다.)

데이터 타입	초 기 값
SINT, INT, DINT, LINT, 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)"

① VAR_EXTERNAL의 선언은 외부에서 선언한 변수를 간접 지정한 것이므로 초기 값을 줄 수 없다.

② 변수 선언 시 %I와 %Q로 할당한 변수는 입·출력에 해당하므로 초기 값을 줄 수 없다.

(7) Named(네임드) 변수의 메모리 할당

네임드 변수의 메모리 할당에는 자동 할당과 사용자 정의가 있다.

(8) 자동 할당

컴파일러가 내부 메모리 영역에 변수 위치를 자동으로 지정한다.

예를 들어 "밸브 1"이란 변수를 자동 메모리 할당으로 지정할 경우 변수의 내부 위치는 프로그램이 작성된 후 컴파일(Compile) 과정에서 정해지므로 사용자는 변수 위치에 신경을 쓸 필요가 없다. 선언된 변수는 외부 입·출력과 관계없이 내부 연산 도중 신호의 중계, 신호 상태(내부 정보)의 일시 저장, 타이머나 카운터의 접점 이름(펑션 블록의 인스턴스)지정 등에 사용된다.

(9) 사용자 정의

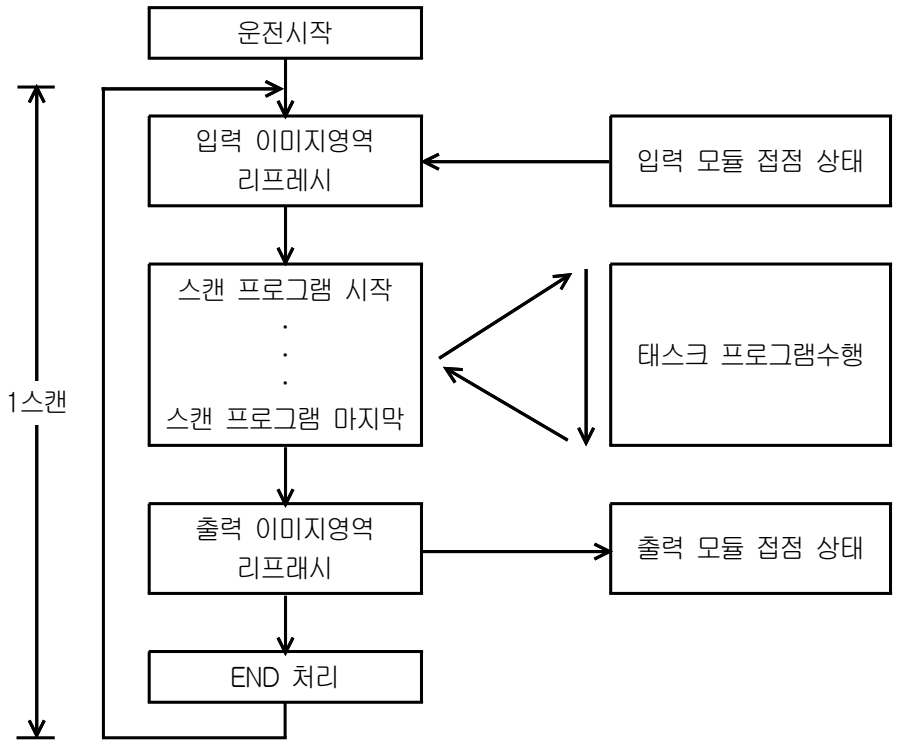
사용자가 직접 변수(%I, %Q, %M)를 사용하여 강제로 위치를 지정한다. 선언된 변수는 입·출력용(%I, %Q) 변수와 통신 파라미터에서 사용할 통신용 변수에 사용한다.

※ **데이터의 유형**은 상당히 중요함으로 예제를 다룰 때 자세히 언급하기로 하고, 이러한 데이터의 유형들은 기본적인 시퀀스 제어 이외에 산술연산 또는 비교, 변환 등의 명령어 사용 시 주의를 해야 한다.

6장. 연산처리

6-1. 스캔 타임 (Scan time)

PLC의 연산 처리 방법은 입력 리프레시(refresh)된 상태에서 이들 조건으로 프로그램을 처음부터 마지막까지 순차적으로 연산을 실행하고 출력 리프레시(refresh)를 한다. 이러한 동작은 고속으로 반복되는데, 이러한 방식을 '반복 연산 방식'이라 하며 한 번 실행하는데 걸리는 시간을 '1 스캔 타임'(1 연산 주기)이라고 한다.



6-2. 입·출력 리프레시

프로그램의 연산을 시작하기 전에 입력 모듈로부터 접점의 상태를 읽어 들여 입력 이미지(input image) 영역에 저장하고, 프로그램의 연산이 끝나면 프로그램 실행에 의해 변화된 출력 이미지(output image) 결과 값을 출력 모듈에 쓰는데 이를 '입·출력 리프레시'(I/O refresh)라고 한다.

6-3. 입·출력 이미지 영역

GLOFA PLC는 반복 연산 방식이므로 프로그램 연산 도중에는 입·출력의 상태를 직접 바꾸지 않고, 스캔 단위로 입·출력 리프레시를 수행한다. 따라서 프로그램 연산 도중 변화된 각 입·출력 접점의 상태를 PLC 내의 메모리 영역에 저장해 놓는데, 이 영역을 입·출력 이미지 영역이라고 한다.

6-4. 운전 모드

1) RUN 모드

프로그램 연산을 정상적으로 수행하는 모드이다.

(1) 모드 변경 시 처리

처음 스캔 시작 시에 데이터 영역의 초기화가 수행된다.

(2) 연산 처리 내용

입·출력 리프레시와 프로그램의 연산을 수행한다.

2) PAUSE 모드

프로그램의 연산이 일시 정지되는 모드이다. 다시 RUN 모드로 돌아갈 경우에는 정지되기 이전의 상태부터 연속하여 운전된다.

(1) 모드 변경 시 처리

데이터 영역의 초기화와 입·출력 이미지 영역의 클리어(clear)를 수행하지 않고 모드 변경 직전의 운전 상태를 유지한다.

(2) 연산 처리 내용

입·출력 리프레시를 수행한다.

3) STOP 모드

프로그램 연산을 하지 않는 정지상태인 모드이다. 리모트 STOP 모드에서만 GMWIN을 통한 프로그램의 전송이 가능하다.

(1) 모드 변경 시 처리

출력 이미지 영역을 소거하고 출력 리프레시를 수행한다.

(2) 연산 처리 내용

입·출력 리프레시를 수행하여 장착된 모듈의 정상동작, 착·탈 여부 검사를 판단하며, 통신 서비스 및 기타 내부 처리를 한다.

4) DEBUG 모드

프로그램의 오류를 찾거나 연산 과정을 추적하기 위한 모드로 이 모드로의 전환은 STOP에서만 가능하다. 프로그램의 수행상태와 각 데이터의 내용을 확인해 보며 프로그램을 검증할 수 있다.

(1) 모드 변경시의 처리

모드 변경 초기에 파라미터에 설정된 리스타트 모드에 따라 데이터 영역이 초기화되며, 출력 이미지 영역을 소거하고 입력 리프레시를 수행한다.

(2) 연산처리 내용

① 입력 리프레시를 수행한다.

② 설정 상태에 따른 디버그 운전을 한다.

③ 프로그램의 마지막까지 디버그 운전을 한 후, 출력 리프레시를 수행한다.

④ 장착된 모듈의 정상 동작, 착·탈 여부를 검사한다.

⑤ 통신등 기타 서비스를 수행한다.

6-5. 모드 변경의 변경 방법

1) 모드 변경의 방법

(1) CPU 모듈의 모드 Key에 의한 변경

(2) CPU 모듈의 통신 포트에 GMWIN을 접속하여 변경

(3) GMWIN을 F-net 상에 연결된 다른 CPU 모듈에 접속하여 변경

(4) FAM, 컴퓨터 링크 모듈 등을 이용하여 사용자 명령으로 변경

(5) 프로그램 수행 중 'STOP 펑션'에 의한 변경

2) 모드 키에 의한 운전 모드 변경

모드 키 위치	운전 모드
RUN	로컬 RUN
STOP	로컬 STOP
STOP → PAU/REM	리모트 STOP
PAU/REM → RUN *	로컬 RUN
RUN → PAU/REM	로컬 PAUSE
PAU/REM → STOP	로컬 STOP

리모트 RUN 모드에서 모드 키에 의해 로컬 RUN 모드로 변경되는 경우 PLC는 중단 없이 연속 운전한다.

3) 리모트 운전 모드 변경

리모트 모드 변경은 모드 키의 위치가 리모트 STOP으로 설정된 경우만 가능하다.
(모드 키가 STOP → PAU/REM의 위치에 있을 경우)

모드 키 위치	모드 변경	GMWIN에 의한 모드 변경	FAM, 컴퓨터 통신 등을 이용한 변경
PAU/REM	리모트 STOP → 리모트 RUN	○	○
	리모트 STOP → 리모트 PAUSE	×	○
	리모트 STOP → DEBUG	○	×
	리모트 RUN → 리모트 PAUSE	○	○
	리모트 RUN → 리모트 STOP	○	○
	리모트 RUN → DEBUG	×	×
	리모트 PAUSE → 리모트 RUN	○	○
	리모트 PAUSE → 리모트 STOP	○	○
	리모트 PAUSE → 리모트 DEBUG	×	×
	DEBUG → 리모트 STOP	○	○
	DEBUG → 리모트 RUN	×	×
	DEBUG → 리모트 PAUSE	×	×

4) 리모트 운전 모드 변경 허가

시스템의 보호를 위하여 운전 모드 변경 소스(Source)중 일부가 모드 변경을 할 수 없도록 할 수 있으며, 리모트 운전 모드 변경 금지 시에는 모드 키와 GMWIN에 의해서 운전 모드 변경이 가능하다. 설정은 기본 파라미터의 '통신에 의한 PLC 제어 허용'항에서 한다.

6-6. 리스타트 모드

리스타트 모드는 전원을 재투입하거나 또는 모드 전환에 의해서 RUN 모드로 운전을 시작할 때, 변수 및 시스템을 어떻게 초기화한 후 RUN 모드 운전을 할 것인가를 설정하는 것으로, 콜드, 웜, 핫의 3종류가 있으며, 각 리스타트 모드의 수행조건은 다음과 같다.

1) 콜드 리스타트 (Cold Restart)

(1) 파라미터의 리스타트 모드를 콜드 리스타트로 설정하는 경우 수행된다.
(2) 모든 데이터를 '0'으로 소거하고, 초기값이 설정된 변수에 대해서만 그 값으로 설정한다.
(3) 파라미터를 웜 리스타트 모드로 설정해도 수행할 프로그램이 변경된 후 최초 수행시는 콜드 리스타트 모드로 수행된다.
(4) 운전 중 수동 리셋 스위치를 누르면(GMWIN에서 리셋 명령을 한 경우와 동일) 파라미터에 설정된 리스타트 모드에 관계없이 콜드 리스타트 모드로 수행된다.

2) 웜 리스타트 (Warm Restart)

(1) 파라미터의 리스타트 모드를 웜 리스타트로 설정하는 경우 수행된다.
(2) 이전 값 유지를 설정한 데이터는 이전 값을 그대로 유지하고, 초기값만 설정된 데이터는 초기값으로 설정한다. 이외의 데이터는 '0'으로 소거한다.
(3) 파라미터를 웜 리스타트 모드로 설정해도, 프로그램 다운로드 및 에러에 의한 운전 정지 후 최초 수행시는 콜드 리스타트 모드로 수행된다.
(4) 파라미터를 웜 리스타트 모드로 설정해도, 데이터 내용이 비정상일 경우(데이터의 정전 유지가 되지 못함)에는 콜드 리스타트 모드로 수행된다.
→ 변수의 종류를 정전 시 값이 유지되는 VAR_RETAIN 으로 설정하였을 경우 다음의 규칙에 따른다.
① 파라미터를 웜 리스타트 모드로 설정해야 정전 시 그 값이 유지된다.
② 파라미터를 콜드 리스타트 모드로 설정하면 사용자가 정의한 초기값이나 기본 초기값으로 초기화된다.

→ VAR_RETAIN 으로 선언하지 않은 변수는 콜드 리스타트나 웜 리스타트 어느 경우에도 사용자가 정의한 초기값이나 기본 초기값으로 초기화 된다

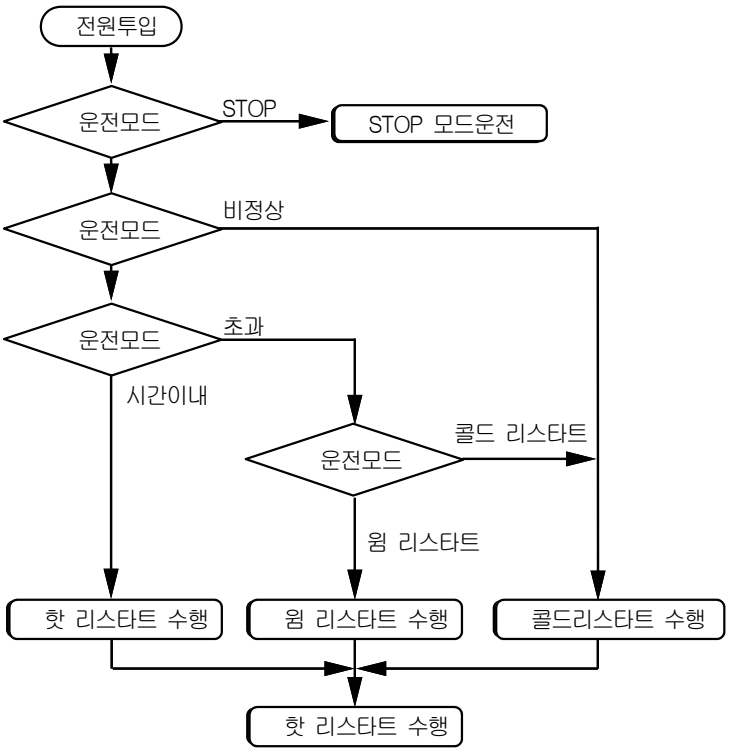
3) 핫 리스타트(Hot Restart)

(1) 정상 운전 중 전원이 꺼진 후 전원이 재투입 될 때 RUN 모드이고, 전원이 꺼진 후 재투입되기까지의 시간이 핫 리스타트 허용 시간이내면, 핫 리스타트 모드를 수행한다.
(2) 모든 데이터와 프로그램 수행 요소들을 전원이 꺼지기 이전의 상태로 복원하여 수행한다.
(3) 전원이 꺼지기 직전의 상태에서 다시 프로그램을 수행하므로, 순간적인 정전 등에도 프로그램의 연속성을 유지할 수 있다.
(4) 핫 리스타트 허용 시간 초과 시는 파라미터에 설정된 리스타트 모드(콜드/웜)로 수행된다.
(5) 데이터 내용이 비정상일 경우(데이터의 정전 유지가 되지 못함)에는 콜드 리스타트 모드로 수행된다.

4) 리스타트 모드에 따른 데이터의 초기화

리스타트 모드 수행 시 각 변수에 대한 초기화 방법은 다음과 같다.

변수지정 / 모드	콜드(COLD)	웜(WARM)	핫(HOT)
디폴트	“0” 으로 초기화	“0”으로 초기화	이전 값 유지
리테인	“0” 으로 초기화	이전 값 유지	이전 값 유지
초기화	사용자 지정 값으로 초기화	사용자 지정 값으로 초기화	이전 값 유지
리테인 & 초기화	사용자 지정 값으로 초기화	이전 값 유지	이전 값 유지



[운전 중 전원 재투입 시 리스타트 모드 수행도]

7장. Programming의 기초

7-1. 도구상자의 활용

프로그램 편집, 변수 모니터, 타임 차트, I/O 모니터링 등을 할 때 자주 사용하는 명령을 도구상자를 통해서 실행할 수 있다.

또한 메뉴 - 도구상자 - 도구상자형태를 선택하면 도구상자의 위치와 화면상에 출현을 조정할 수 있으며 [그림 5-1]과 같다.

[그림 5-1] 도구상자

1) 화살표 ( )

접점, 코일, 펑션, 펑션 블록 등을 선택하여 이동, 편집, 삭제할 경우 사용한다.

2) 블록지정 ( )

한 행 이상을 삭제, 복사 등의 편집기능을 할 경우 사용된다.

※ 블록지정 아이콘을 선택 후 설정하고자 하는 부분으로 이동하여 마우스 좌측을 누르고 드래그 하여 설정하고자 하는 부분을 설정하여 편집한다.

3) 평상시 열린 접점(Normally Open Contact) ( )

평상시 열린 접점으로 시퀀스 제어의 “A”접점이라고 생각하면 된다.(단축키 F2)

※ 아이콘을 이용하여 편집하고자 하는 부분에 커서를 이동시키고 좌측 마우스를 누른다.

4) 평상시 닫힌 접점(Normally Closed Contact) ()

평상시 닫힌 접점으로 시퀀스 제어의 "B"접점이라고 생각하면 된다.(단축키 F3)

※ 아이콘을 이용하여 편집하고자 하는 부분에 커서를 위치시키고 좌측 마우스를 누른다.

[주의]  ,  은 맨 오른쪽 끝에는 올 수 없다.(약속사항)

5) 양 변환 검출접점 ()

양 변환 검출 접점이란 접점이 동작되는 순간 즉 "0"에서 "1"로 변환하는 순간을 검출하여 오른쪽 연결선이 해당 스캔에서 1스캔 동안 연결된다.(단축키 Shift + F1)

※ 양 변환 검출 접점 아이콘을 이용하여 편집하고자 하는 부분에 커서를 이동시키고 좌측 마우스를 누른다.

6) 음 변환 검출접점 ()

음 변환 검출 접점이란 접점이 동작되는 순간 즉 "1"에서 "0"으로 변환하는 순간을 검출하여 오른쪽 연결선이 해당 스캔에서 1스캔 동안 연결된다.(단축키 Shift + F2)

※ 음 변환 검출 접점 아이콘을 이용하여 편집하고자 하는 부분에 커서를 이동시키고 좌측 마우스를 누른다.

7) 가로선과 세로선 ( , )

가로선과 세로선은 접점과 접점, 접점과 코일 또는 분기 회로 시 가로 및 세로로 회로를 연결하는 연결선이다.(단축키 F4, F5)

8) 코일과 역 코일 ( , )

코일은 출력을 나타내며 왼쪽의 입력이 ON상태이면 여자 되고 OFF 상태이면 소자 된다. 역코일 또한 출력을 나타내며 왼쪽의 입력이 ON상태이면 소자 되고 OFF 상태이면 여자 된다.

구 분	왼쪽 회로 입력이 ON인 경우	왼쪽 회로 입력이 OFF인 경우
코 일	여 자	소 자
역 코일	소 자	여 자

※ 사용법 및 기능 설명

아래 회로는 코일인 경우와 역 코일인 경우를 비교하여 설명하고 있다.

◈ 코일인 경우 : %I0.0.00이 "1"이면 코일 %Q0.2.0은 출력 "1"이 나온다.

%I0.0.00이 "0"이면 코일 %Q0.2.0은 출력이 "0"이 나온다.

◈ 역 코일인 경우는 코일과 반대 동작을 한다.

%I0.0.00이 "1"이면 코일 %Q0.3.0은 출력 "0"이 나온다.

%I0.0.00이 "0"이면 코일 %Q0.3.0은 출력이 "1"이 나온다.

9) 셋팅 코일과 리셋코일 ({S} , {R})

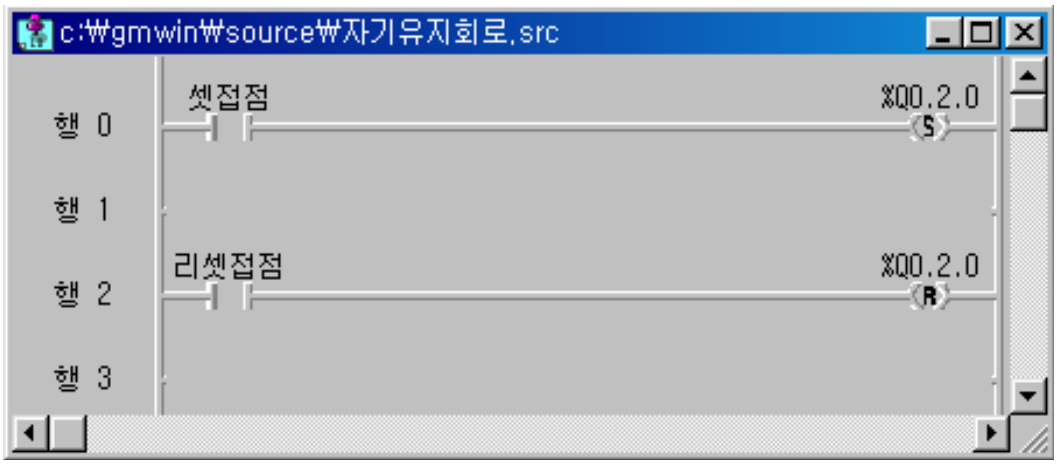
셋팅 코일은 왼편의 연결선 상태가 ON 이 되었을 때 관련된 부울 변수는 ON이 되고, RESET 코일에 의해 reset 되기 전까지는 set 되어 있는 상태로 유지가 된다.

리셋 코일은 셋 코일에 의하여 ON되어 있는 코일상태를 reset 시키고자 할 때 사용한다.(단축키 Ctrl+F3, Ctrl+F4)

※ 사용법 및 기능 설명

◈ 아래 회로와 같이 셋 접점 신호가 ON인 경우 셋팅 코일 %Q0.2.00이 ON되며 셋 접점이 OFF시에도 계속 자기유지 상태로 유지된다.

◈ 리셋 접점을 ON 시켰을 경우 %Q0.2.0 출력 리셋코일이 ON되며 셋팅 코일의 자기유지가 해제 된다.



10) 양 변환 검출코일과 음 변환 검출코일 ({P}, {N})

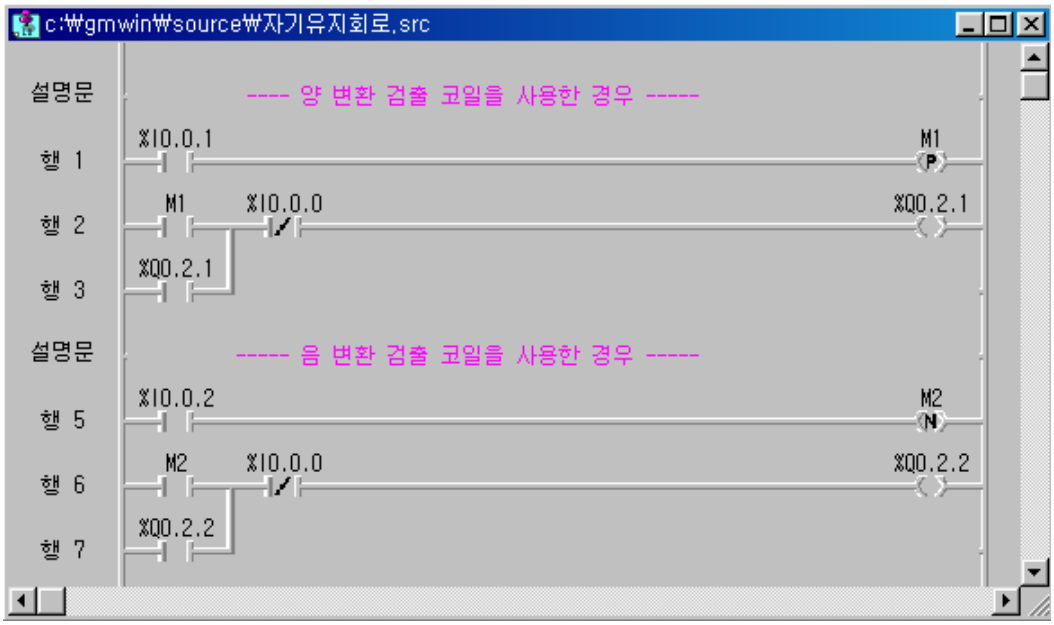
양 변환 검출코일은 왼편의 연결선 상태가 바로 전 스캔에서 OFF였던 것이 현재 스캔에서 ON이 되는 상승에지 상태에서 1스캔 동안만 ON이 된다.

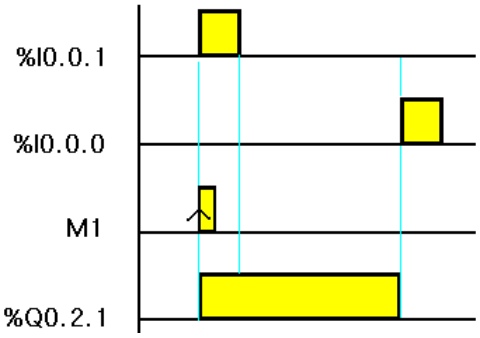
음 변환 검출코일은 왼편의 연결선 상태가 바로 전 스캔에서 ON이었던 것이 현재 스캔에서 OFF되는 하강에지 상태에서 1스캔 동안만 ON이 된다.

※ 사용법 및 기능 설명

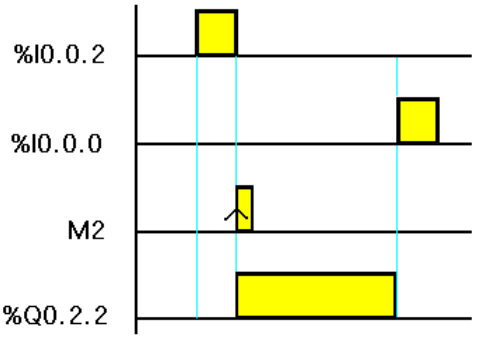
◈ 양 변환 검출 코일은 아래 타임차트에서와 같이 %I0.0.1의 입력 스위치가 ON이 되는 순간 M1이 1스캔 동안 ON이 되며, 그 후로는 동작하지 않으므로 자기유지 접점을 추가하였다. 출력 %Q0.2.1의 여자는 %I0.0.0의 입력 스위치에 의하여 소자 된다.

◈ 음 변환 검출 코일은 입력 스위치 %I0.0.2를 눌렸다 떼는 순간 M2가 동작 하고 %Q0.2.2의 출력코일에 의하여 자기유지가 된다.





[양 변환 검출 코일]



[음 변환 검출 코일]

11) 펑션 ({F})

기본 펑션에는 전송 펑션, 형 변환 펑션, 비교 펑션, 산술연산 펑션, 논리연산 펑션, 비트 시프트 펑션 등이 있다.

(1) 펑션은 입력에 대한 연산 결과를 1스캔에 즉시 출력하며, 출력이 하나이다.

(2) 전송 펑션(MOVE)의 IN, OUT 변수는 모든 데이터형이 지정될 수 있으나, 같은 데이터형이어야 한다.

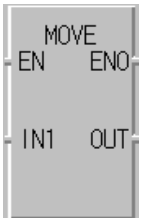
(3) 산술 연산 펑션(ADD, MUL 등)의 IN1, (IN2), OUT 변수는 수치(ANY_NUM) 데이터형만이 지정될 수 있으며, 또한 모두 같은 데이터 형이어야 한다.

(4) 논리 연산 펑션(AND, OR 등)의 IN1, (IN2), OUT 변수는 비트 상태 (ANY_BIT) 데이터형만이 지정될 수 있으며, 또한 모두 같은 데이터 형이어야 한다.

(5) 형 변환 펑션(INT_TO_BCD, BCD_TO_INT등)의 IN, OUT의 변수는 지정된 데이터형에 의해 고정되어 있으며, 펑션은 라이브러리에 넣어서 사용한다.

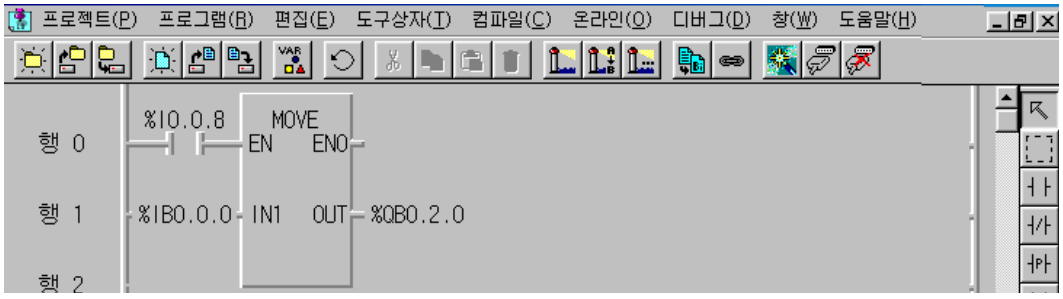
(6) 전송 펑션의 종류와 사용법

MOVE(데이터 전송)

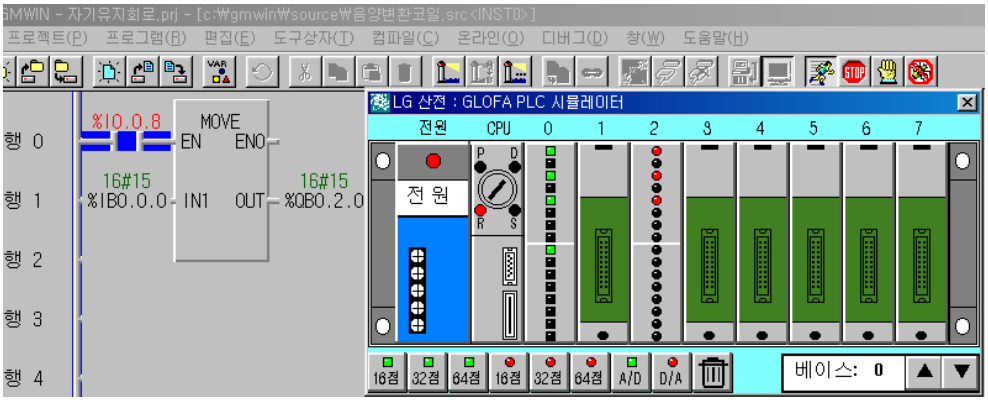
펑 션	설 명
	입력 EN : 1일 때 펑션 실행 I N : 전송할 값 출력 ENO : EN 값이 그대로 출력 OUT : 전송된 값 IN, OUT은 모두 같은 데이터 형이어야 함 예) IN %IB0.0.0 이면 OUT %QB0.2.0이여야 한다.

가. 프로그램 적용 예1

실행조건(%I0.0.8)이 ON되면 MOVE 펑션이 실행되어 1Byte(8bit)인 0.0.0~0.0.7까지의 ON/OFF 정보가 복사되어 0.2.0~0.2.7까지 각 해당 비트에 데이터를 전송한다. (Byte 단위로 동작)

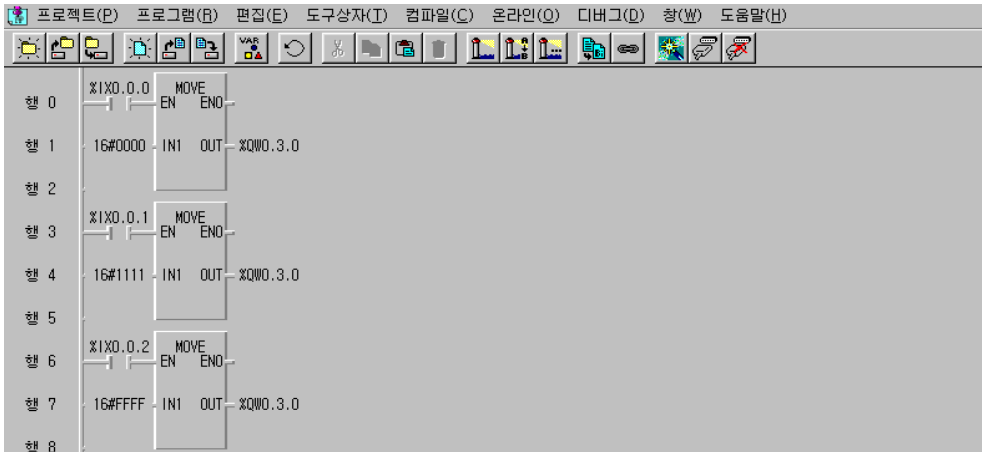


◈ 시뮬레이터에 의한 LD 확인



나. 프로그램 적용 예2

스위치 0, 1, 2 중 하나를 ON 하면 MOVE 펑션이 실행되어 코드값을 %QW0.3.0으로 전송한다.



◈ 시뮬레이터에 의한 LD 확인

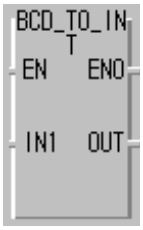
① 첫 번째 IN1의 16#0000이란 %IX0.0.0(비트)의 값이 ON 되었을 때 출력 %QW0.3.0(16진수)의 자리에 16진수로 0000을 표현하라는 지시이다.

② 두 번째 IN1의 16#1111이란 %IX0.0.1(비트)의 값이 ON 되었을 때 출력 %QW0.3.0(16진수)의 자리에 16진수로 1111을 표현하라는 지시이다.

③ 세 번째 IN1의 16#FFFF란 %IX0.0.2(비트)의 값이 ON 되었을 때 출력 %QW0.3.0(16진수)의 자리에 16진수로 FFFF를 표현하라는 지시이다.

(7) 형 변환 펑션의 종류와 사용법

가. **BCD_TO_*** (BCD형을 정수로 변환)

펑 션	설 명
	입력 EN : 1일 때 펑션 실행 I N : BCD 형태의 데이터를 갖고 있는 ANY_BIT 입력 값 출력 ENO : EN 값이 그대로 출력 OUT : 타입 변환된 데이터

가) 기능

INPUT의 타입을 변환해서 OUT으로 출력한다.

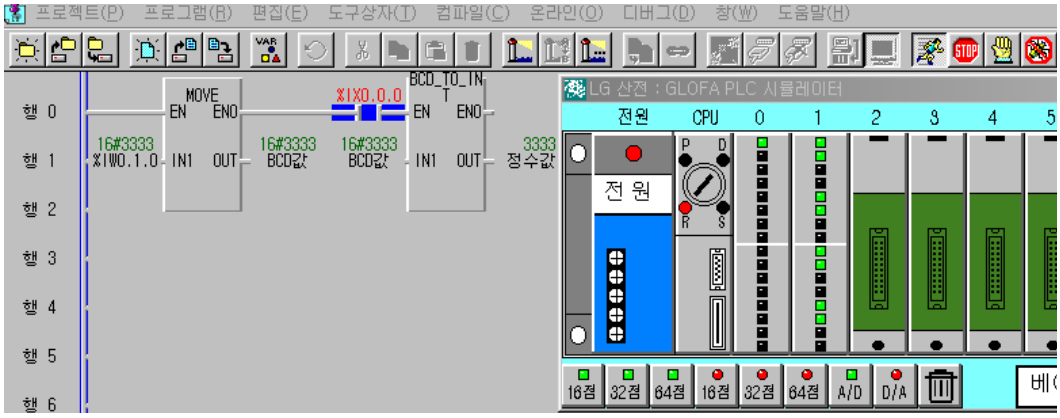
FUNCTION	입력타입	출력타입	내 용
BCD_TO_SINT	BYTE	SINT	입력이 BCD 값일 경우에만 정상 변환된다. (입력 데이터 타입이 WORD 일 경우 0~16#9999값만 정상 변환된다.)
BCD_TO_INT	WORD	INT	
BCD_TO_DINT	DWORD	DINT	
BCD_TO_USINT	BYTE	USINT	
BCD_TO_UINT	WORD	UINT	
BCD_TO_UDINT	DWORD	UDINT	

IN이 BCD 형태의 데이터가 아닌 경우 출력은 “0”이 되고 _ERR(연산 에러 플래그), _LER(연산 에러 래치 플래그)은 ON 된다.

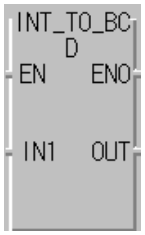
나) 프로그램 적용

%IW0.1.0을 사용하여 BCD 값 3333을 입력하고 %IX0.0.0을 ON 하면 BCD값이 정수로 변환되어 정수 값에 출력이 된다.

◈ 시뮬레이터에 의한 LD 확인



나. **INT_TO_*** ** (INT 타입 변환)

펑 션	설 명
	입력 EN : 1일 때 펑션 실행 I N : 정수형의 입력값 출력 ENO : 에러 없이 실행되면 1을 출력 OUT : 타입 변환된 데이터

가) 기능

IN을 타입 변환해서 OUT으로 출력한다.

FUNCTION	데이터 타입	내 용
INT_TO_SINT	SINT	일력이 -128~127일 경우 정상 변화되나, 그 외 값은 에러가 발생한다.
INT_TO_DINT	DINT	DINT 타입으로 정상 변환 한다.
INT_TO_USINT	USINT	입력이 0~255일 경우 정상 변화되나, 그 외 값은 에러가 발생한다.
INT_TO_UINT	UINT	입력이 0~32767일 경우 정상 변화되나, 그 외 값은 에러가 발생한다.
INT_TO_UDINT	UDINT	입력이 0~32767일 경우 정상 변화되나, 그 외 값은 에러가 발생한다.
INT_TO_BOOL	BOOL	하위 1비트를 취해 BOOL 타입으로 변환한다.
INT_TO_BYTE	BYTE	하위 8비트를 취해 BYTE 타입으로 변환한다.
INT_TO_WORD	WORD	내부비트 배열의 변화 없이 WORD 타입으로 변환한다.
INT_TO_DWORD	DWORD	상위 비트 등을 0으로 채운 DWORD 타입으로 변환 한다.
INT_TO_LWORD	LWORD	상위 비트 등을 0으로 채운 LWORD 타입으로 변환한다.
INT_TO_BCD	WORD	입력이 0~9,999 일 경우 정상 변환되나, 그 외 값은 에러가 발생한다.

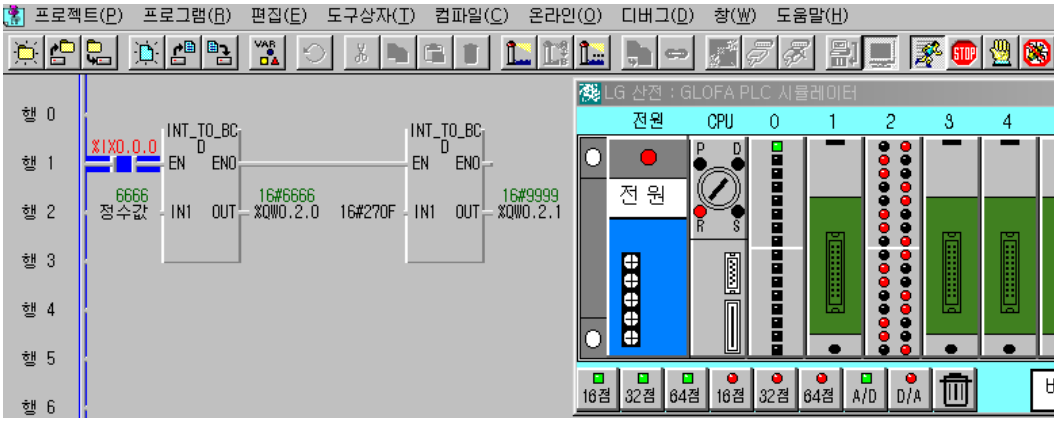
나) 에러

변환 에러 발생시 _ERR(연산 에러 플러그), _LEE(연산 에러 래치 플래그) 이 ON 된다.

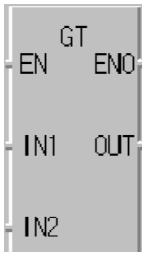
다) 프로그램 적용

%I0.0.0의 입력 스위치가 ON일 때 정수 값 란에 강제변수 6666을 입력하면 BCD 값이 %QW0.2.0에 6666값이 출력되며, 펑션 INT_TO_BCD 입력에 최대 정수 값 9999 또는 16#270F를 입력하면 %QW0.2.1에 BCD 값 16#9999가 표시된다.

◈ 시뮬레이터에 의한 LD 확인



다. GT ('크다' 비교)

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 : EN 값이 그대로 출력 OUT : 비교 결과 값

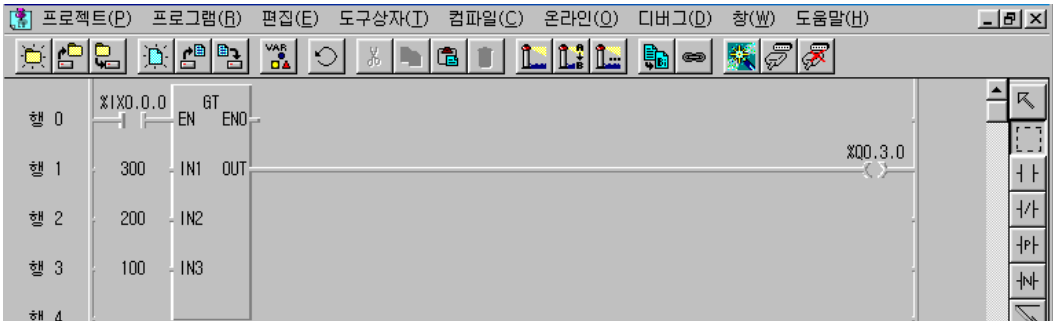
가) 기능

입력 값의 비교 결과 IN1 〉 IN2 〉 IN3... 〉 INn이 참이면 OUT으로 "1"을 출력한다. 거짓의 경우에는 OUT으로 "0""을 출력한다.

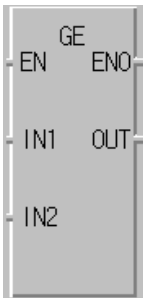
나) 프로그램 적용

(가) 실행 조건 %IX0.0.0을 ON하면 GT 펑션이 실행된다.

(나) 입력 변수로 선언된 값 IN1=300, IN2=200, IN3=100을 입력하고 비교 결과 IN1 〉 IN2 〉 IN3이 되므로, 출력 결과 값 %QX0.3.0은 "1"이 된다.



라. GE ('크거나 같다' 비교)

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 : EN 값이 그대로 출력 OUT : 비교 결과 값

가) 기능

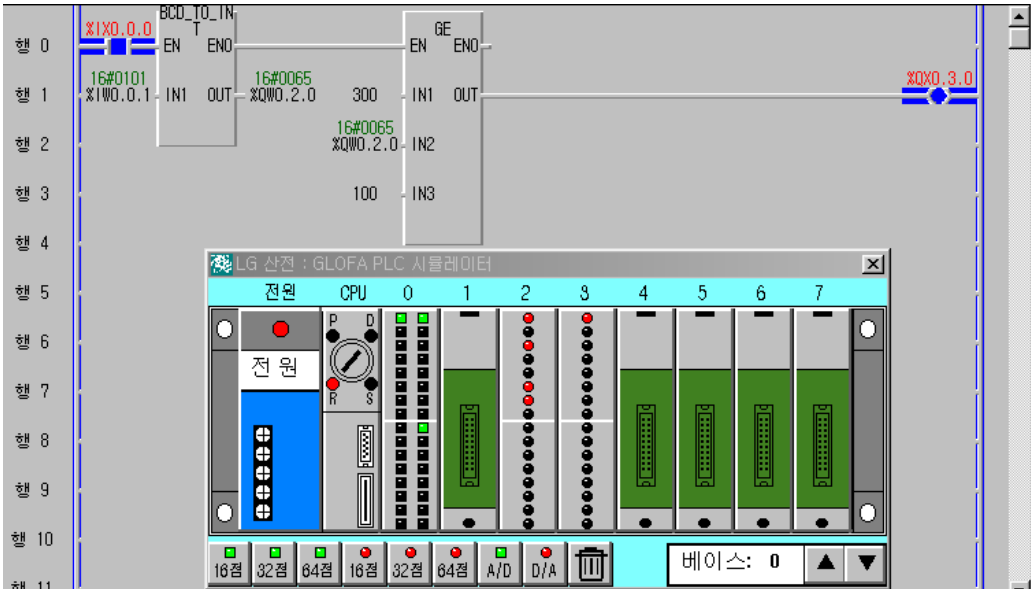
입력값의 비교 결과 IN1 ≥IN2 ≥IN3... ≥INn(n은 입력개수)이 참이면 OUT으로 "1"을 출력한다. 거짓의 경우에는 OUT으로 "0"을 출력한다.

나) 프로그램 적용

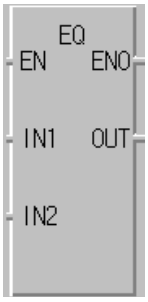
(가) 실행 조건 %IX0.0.0을 ON 하면 GE 펑션이 실행된다.

(나) 입력 변수로 선언된 값 IN1=300, IN2=%IW0.0.1, IN3=100이면, 비교 결과 IN1 ≥ IN2 ≥ IN30이므로, 출력 결과 값 %QX0.3.0은 "1"이 된다. 즉 IN2의 값이 100보다 같거나 크면 또는 300보다 같거나 작으면 %QX0.3.0은 "1"이 된다.

다) 시뮬레이터에 의한 LD 확인



마. **EQ** ('같다' 비교)

평 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 : EN 값이 그대로 출력 OUT : 비교 결과 값

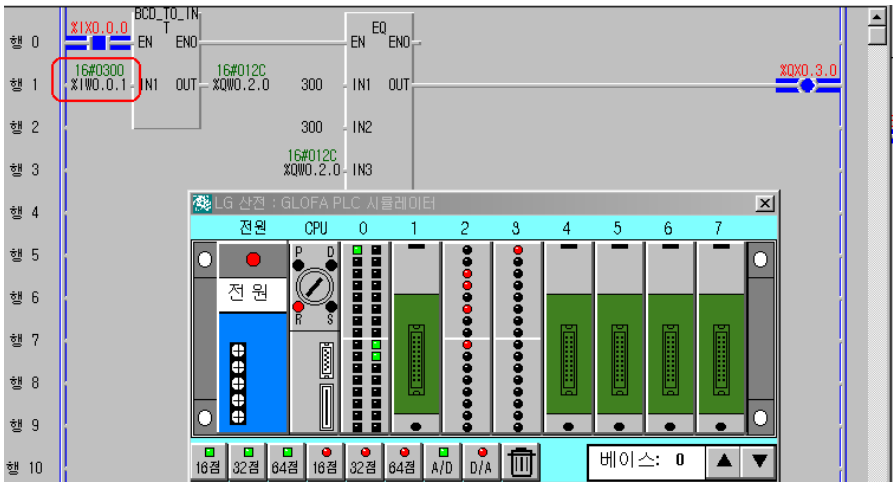
가) 기능

입력 값의 비교 결과 IN1 = IN2 = IN3... = INn(n은 입력개수)이 참이면 OUT으로 "1"을 출력한다. 거짓의 경우에는 OUT 으로 "0"을 출력한다.

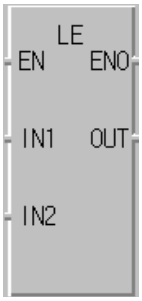
나) 프로그램 적용

%IX0.0.0의 값이 ON일 때 BCD_TO_INT펑션이 실행되며, EQ 값 중 IN3 값이 300 (즉 %IW0.0.1)일 때만 %QX0.3.00이 "1"이 된다.

◈ 시뮬레이터에 의한 LD 확인



바. LE (‘작거나 같다’ 비교)

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

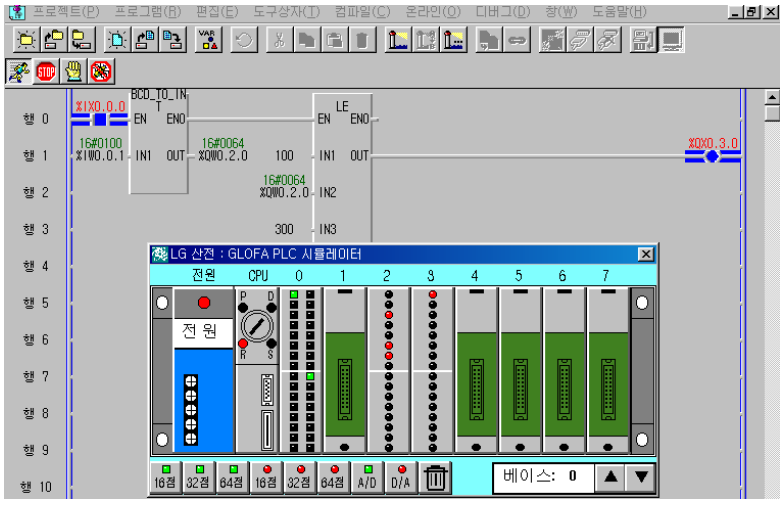
가) 기능

입력 값의 비교 결과 IN1 ≤ IN2 ≤ IN3... ≤ INn(n은 입력개수)이 참이면 OUT으로 “1”을 출력한다. 거짓의 경우에는 OUT으로 “0”을 출력한다.

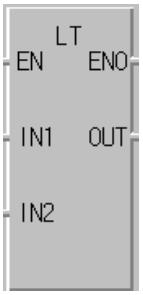
나) 프로그램 예

%IX0.0.0의 값이 ON일 때 BCD_TO_INT펑션이 실행되며, LE 값 중 IN2 값이 100~300(즉 %IW0.0.1)일 때만 %QX0.3.0은 “1”이 된다.

◈ 시뮬레이터에 의한 LD 확인



사. LT (‘작다’ 비교)

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

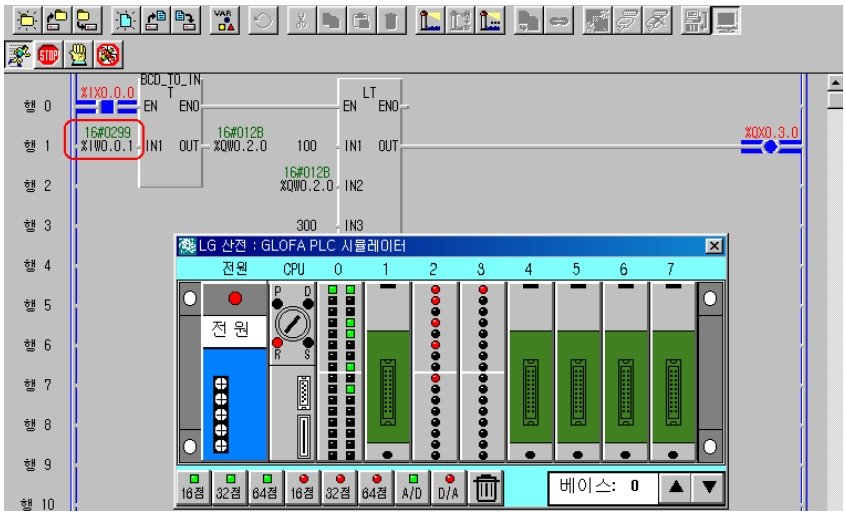
가) 기능

입력 값의 비교 결과 IN1 〈 IN2 〈 IN3... 〈 INn(n은 입력개수)이 참이면 OUT으로 “1”을 출력한다. 거짓의 경우에는 OUT으로 “0”을 출력한다.

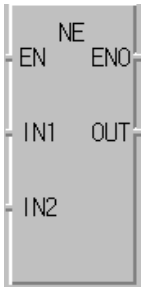
나) 프로그램 적용

%IX0.0.0 의 값이 ON일 때 BCD_TO_INT평션이 실행되며, LT 값 중 IN2 값이 101~299(즉 %IW0.0.1 값이)일 때만 %QX0.3.00이 “1”이 된다.

다) 시뮬레이터에 의한 LD 확인



아. **NE** ('같지 않다' 비교)

펑 션	설 명
	입력 EN : 1일 때 펑션 실행 IN1 : 비교할 값 IN2 : 비교할 값 ※ IN1, IN2,... 는 모두 같은 타입이어야 함. 출력 ENO : EN 값이 그대로 출력 OUT : 비교 결과 값

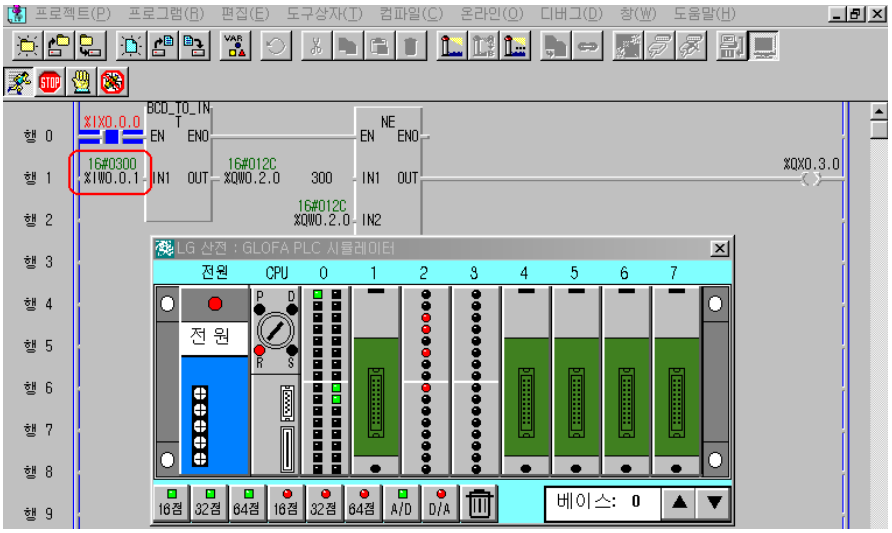
가) 기능

비교 결과 IN1 과 IN2 가 같지 않으면 OUT으로 "1"을 출력한다. 같으면 OUT으로 "0"을 출력한다.

나) 프로그램 적용

%IX0.0.0의 값이 ON일 때 BCD_TO_INT펑션이 실행되며, NE 값 중 IN2 값이 300 (즉 %IW0.0.1 값이)일 때만 %QX0.3.00이 "0"이 되고 그 외에는 "1"이 된다.

다) 시뮬레이터에 의한 LD 확인



12) 펑션 블록 (FB)

펑션 블록은 여러 스캔에 걸쳐 누계된 연산 결과를 출력하므로, 연산 중 누계되는 데이터를 잠시 보관하기 위한 내부 메모리가 필요하다. 따라서 펑션 블록은 사용하기 전에 변수를 선언하는 것처럼 인스턴스 변수를 선언해야 한다. 인스턴스 변수는 펑션 블록에서 사용하는 변수들의 집합이며, 기본 펑션 블록 중 가장 일반적인 것은 타이머와 카운터이다.

※ 사용자가 펑션 블록을 생성할 수 있으며 생성된 펑션 블록은 사용자 펑션에 추가된다.

(1) 기본 펑션 블록 일람표

① 카운터

NO	펑션 블록이름	기 능
1	C T U	가산 카운터 (Up Counter)
2	C T D	감산 카운터 (Down Counter)
3	C T U D	가감산 카운터 (Up Down Counter)

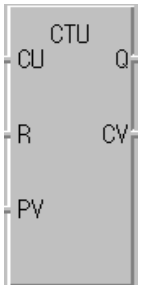
② 타이머

NO	펑션 블록이름	기 능
1	T O N	ON 딜레이 타이머 (On Delay Timer)
2	T O F	OFF 딜레이 타이머 (Off Delay Timer)
3	T P	펄스 타이머 (Pulse Timer)

(2) 기본 펑션 블록 설명 및 프로그램 사용법

가. CTU

가산 카운터 (Up Counter)

펑 션	설 명
	입력 CU : 업 카운터(Up Counter) 펄스입력 R : 리셋 입력(Reset) PV : 설정값(Preset Value) 출력 Q : 업 카운터(Up Counter) 출력 CV : 현재 값(Current Value)

가) 기능

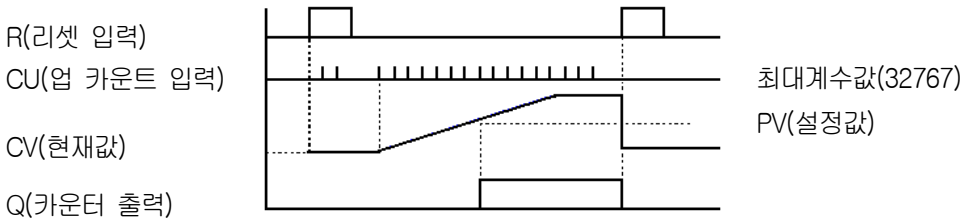
(가) 펄스 입력 CU가 "0"에서 "1"(Rising Edge)이 되면 현재값 CV가 이전 값 보다 1만큼 증가 한다.

(나) 단, CV는 정수(INT)의 최대값 32767을 넘지 않는다.

(다) 리셋 입력 R이 "1"이 되면 현재값 CV는 "0"으로 소거(Clear)된다.

(라) 출력 Q는 현재값(CV)이 설정값(PV) 이상이면 "1"이 된다.

나) 타임 차트



다) 프로그램 예

(가) CTU는 펑션 블록이므로 연산 중 누계되는 데이터를 잠시 보관하기 위한 인스턴스 변수를 반드시 선언해야 한다.

(나) GMWIN에서 프로그램 편집시 CTU의 인스턴스 변수를 선언하면 카운터 출력은 인스턴스 이름.Q, 현재값은 인스턴스 이름.CV로 변수가 자동 생성됩니다.

(다) 아이콘  을 선택하여 펑션 블록 목록 창에서 CTU를 선택한다.

(라) 변수 이름 선언은 제한이 없으나 편의상 C1으로 정의하고 나머지 창들은 디폴트값 으로 정의한다.

(마) 토글스위치"0"(%IX0.0.0)으로 CU에 입상(Rising Edge)펄스를 입력하면 현재 값이 증가한다.

(바) 현재값을 우측 %QW0.2.0에 출력할 것이다.

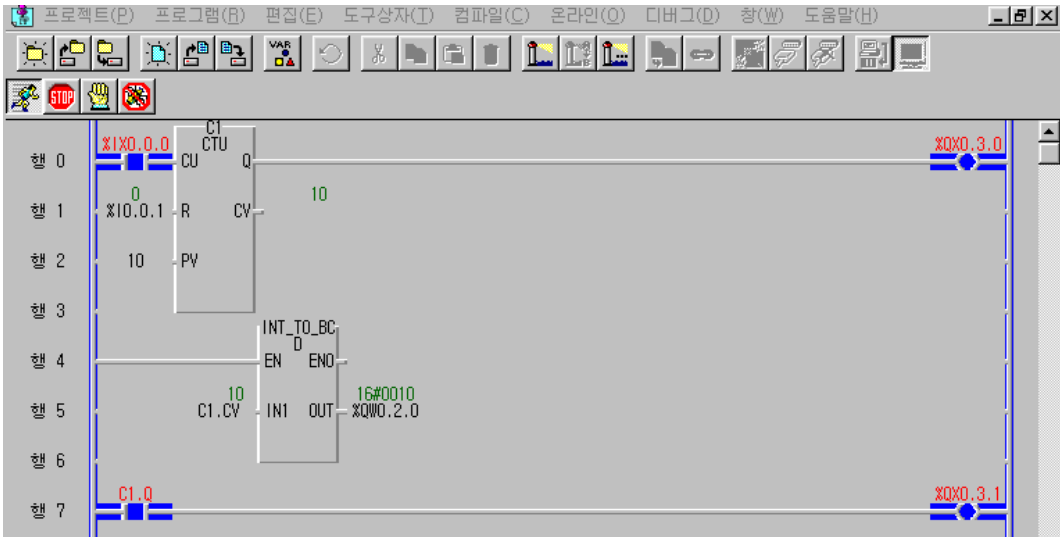
(사) 현재값이 10 이상이면 카운터 출력(C1.Q)이 "1"이 되어 램프(%QX0.3.0)가 점등할 것이다.

(아) 토글스위치 "1"(%IX0.0.1)을 ON하면 현재값 및 카운터 출력이 리셋되어 0이 된다.

(자) 현재값(C1.CV)이 0~9999 사이를 벗어나면 펑션 INT_TO_BCD에 의해 _ERR, _LER 플래그가 ON된다.

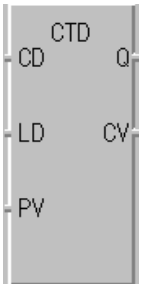
(차) 아래 프로그램의 예 중 출력 %QX0.3.0, %QX0.3.1은 똑 같은 동작을 실행한다. 그런데 %QX0.3.1은 카운터 C1.Q의 값에 따라 동작하는 예를 나타내고 있다.

◈ 시뮬레이터에 의한 LD 확인



나. CTD

감산 카운터 (Down Counter)

펑 션	설 명
	입력 CD : 다운 카운터(Down Counter) 펄스입력 LD : 설정값 입력(Load) PV : 설정값(Preset Value) 출력 Q : 다운 카운터(Down Counter) 출력 CV : 현재값(Current Value)

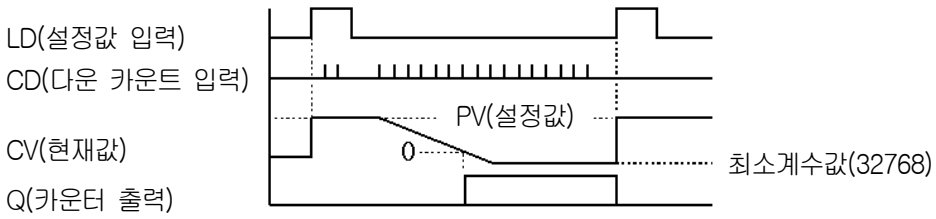
가) 기능

(가) 펄스 입력 CD가 "0"에서 "1"(Rising Edge)이 되면 현재 값 CV가 이전 값 보다 "1" 만큼 감소한다.(단, CV는 정수(INT)의 최소값 -32768)

(나) 설정값 입력 접점 LD가 "1"이 되면, 설정값 PV값이 현재값 CV에 로드 된다.

(다) 출력 Q는 현재값(CV)이 "0"이하이면 "1"이 된다.

나) 타임 차트



다) 프로그램 예

(가) CTD는 펑션 블록이므로 연산 중 누계 되는 데이터를 잠시 보관하기 위한 인스턴스 변수를 반드시 선언해야 한다.

(나) GMWIN에서 프로그램 편집 시 CTD의 인스턴스 변수를 선언하면 카운터 출력은 인스턴스 이름.Q, 현재값은 인스턴스 이름.CV로 변수가 자동 생성된다.

(다) CTD의 인스턴스 변수 C2를 선언한다.

(라) 설정값을 10으로 셋팅 한다.

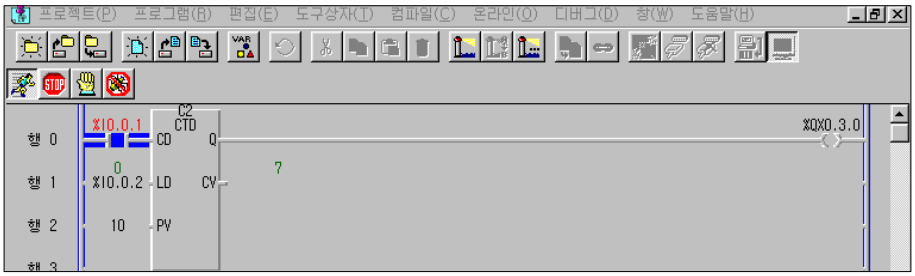
(마) 초기에 %IX0.0.2를 누르면 LD가 “1”이 되어 설정값이 현재 값에 로드 된다.

(바) 스위치 %IX0.0.1을 신호에 의하여 CD에 입상(Rising Edge) 펄스를 입력하면 현재 값이 감소한다.

(사) 현재값이 “0”이하이면 카운터출력(C2.Q)이 “1”이 되어 (%QX0.3.0)이 점등된다.

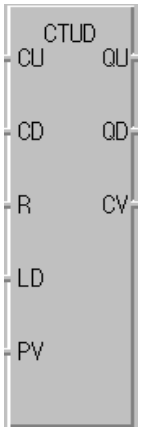
(아) 스위치(%IX0.0.2)를 ON하면 LD가 “1”이 되어 설정값이 현재값에 로드 된다.

◈ 시뮬레이터에 의한 LD 확인



다. CTUD

가감산 카운터 (Up Down Counter)

펑 션	설 명
	입력 CU : 업 카운터(Up Counter) 펄스입력 CD : 다운 카운터(Down Counter) 펄스입력 R : 리셋 입력(RESET) LD : 설정값 입력(Load) PV : 설정값(Preset Value) 출력 QU : 업 카운터(Up Counter) 출력 QD : 다운 카운터(Down Counter) 출력 CV : 현재값(Current Value)

가) 기능 1

(가) CTUD는 CU가 “0”에서 “1”이 되면 현재값 CV가 이전 값보다 1만큼 증가하고, CD가 “0”에서 “1”이 되면 현재값 CV가 이전값 보다 “1”만큼 감소하는 카운터이다. 단, 현재값 CV는 정수(INT)의 최소값 -32768~최대값 32767 사이의 값을 갖는다.

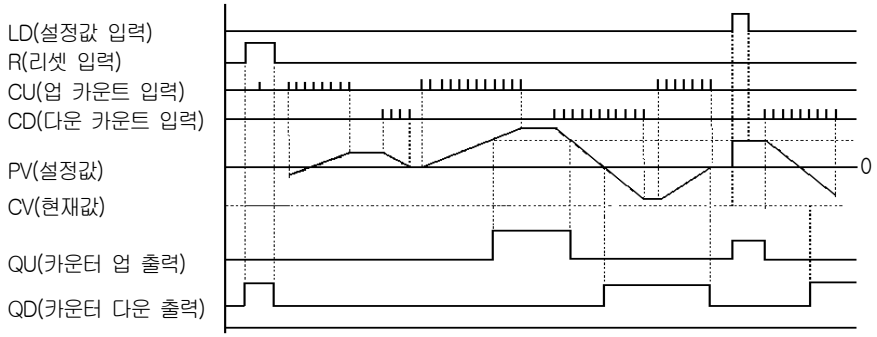
(나) 설정값 입력 접점 LD가 “1”이 되면 현재값 CV에 설정값 PV값이 로드 된다. (CV=PV)

(다) 설정값 입력 R이 “1”이 되면 현재값 CV는 “0”으로 클리어(Clear)된다.(CV=0)

(라) 출력 QU는 CV가 PV이상이면 “1”이 되고, QD는 CV가 “0”이하 일 때 “1”이 된다.

(마) 각 입력 신호에 대해서 R>LD>CU>CD순으로 동작을 수행하며, 신호의 중복 발생시 우선순위가 높은 동작 하나만 수행한다.

나) 타임 차트



다) 프로그램 예

(가) CTUD는 펑션 블록이므로 연산 중 누계되는 데이터를 잠시 보관하기 위한 인스턴스 변수를 반드시 선언해야 한다.

(나) GMWIN에서 프로그램 편집 시 CTUD의 인스턴스 변수를 선언하면 업 카운트 출력은 **인스턴스 이름.QU**, 다운 카운트 출력은 **인스턴스 이름.QD** 그리고 현재값은 **인스턴스 이름.CV**로 변수가 자동 생성된다.

(다) CTUD의 인스턴스 변수 C3를 선언한다.

(라) 설정값을 10으로 셋팅 한다.

(마) 스위치 %IX0.0.0으로 CU에 입상(Rising Edge) 펄스를 입력하면 현재값이 증가한다.

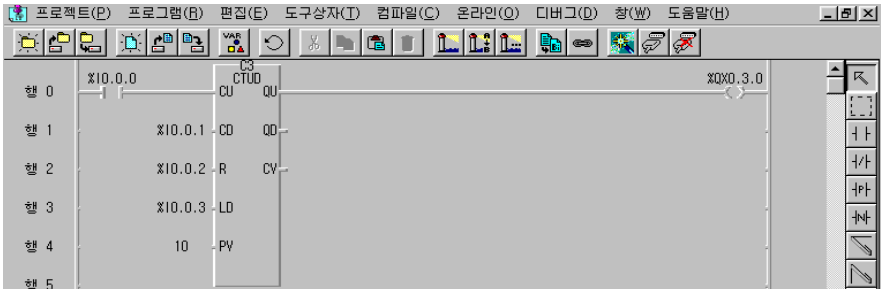
(바) 입력 %IX0.0.1로 CD에 입상(Rising Edge) 펄스를 입력하면 현재값이 감소한다.

(사) 현재값이 설정값 이상이면 C3.QU가 "1"이 되어 %QX0.3.0이 "1"이 된다.

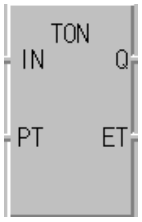
(아) 현재값이 "0"이하이면 C3.QD가 "1"이 된다.

(자) %IX0.0.2를 ON하면 리셋되어 현재값은 "0"으로 클리어(Clear) 된다.

(차) %IX0.0.3을 ON하면 LD가 "1"이 되어 설정값이 현재값에 로드 된다.



라. **TON** (ON Delay Timer)

펑 션	설 명
	입력 IN : 타이머 기동 조건 PT : 설정시간(Preset Time) 출력 Q : 타이머 출력 ET : 경과시간(Elapsed Time)

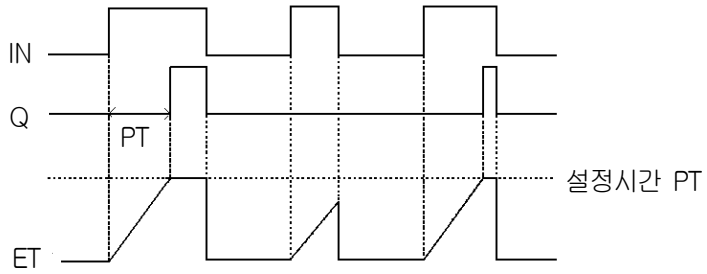
가) 기능

(가) IN이 "1"이 된 후 경과 시간이 ET로 출력된다.

(나) 만일, 경과 시간 ET가 설정 시간에 도달하기 전에 IN이 "0"이면, 경과 시간은 "0"으로 된다.

(다) Q가 "1"이 된 후 IN이 "0"이 되면, Q는 "0"이 된다.

나) 타임 차트



다) 프로그램 예

(가) TON은 펑션 블록이므로 연산 중 누계 되는 데이터를 잠시 보관하기 위한 인스턴스 변수를 반드시 선언해야 한다.

(나) GMWIN에서 프로그램 편집 시 TON의 인스턴스 변수를 선언하면 타이머 출력은 **인스턴스 이름.Q**, 경과 시간은 **인스턴스 이름.ET**로 변수가 자동 생성된다.

(다) TON의 인스턴스 변수 T1을 선언한다.

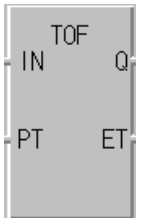
(라) 타이머 T1의 설정 시간을 PT에는 5초(T#5S)로 설정한다.

(마) 기동 스위치 %IX0.0.0을 ON하면 경과 시간(T1.ET)이 디지털 표시기에 출력된다.

(바) 경과 시간 T1.ET가 설정 시간 5초에 도달하면 타이머 출력 T1.Q가 ON되어 출력 %Q0.2.00이 1이 되며 %Q0.2.1은 "0"이 된다.

(사) T1.Q가 ON된 후 기동 스위치 "0"(%IX0.0.0)를 OFF하면 T1.Q는 OFF된다.

마. **TOF** (OFF Delay Timer)

펑 션	설 명
	입력 IN : 타이머 기동 조건 PT : 설정시간(Preset Time) 출력 Q : 타이머 출력 ET : 경과시간(Elapsed Time)

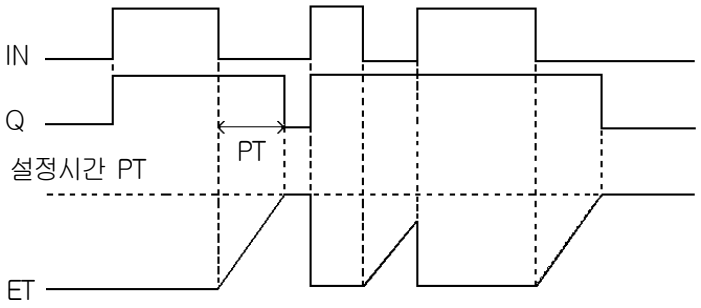
가) 기능

(가) IN이 "1"이 되면, Q가 "1"이 되고, IN이 "0"이 된 후부터 PT에 의해서 지정된 설정 시간이 경과한 후 Q가 "0"이 된다.

(나) IN이 "0"이 된 후 경과 시간이 ET로 출력된다.

(다) 만일 경과 시간 ET가 설정 시간에 도달하기 전에 IN이 "1"이 되면, 경과 시간은 다시 "0"으로 된다.

나) 타임 차트



다) 프로그램 예

(가) TOF는 펑션 블록이므로 연산 중 누계 되는 데이터를 잠시 보관하기 위한 인스턴스 변수를 반드시 선언해야 한다.

(나) GMWIN에서 프로그램 편집 시 TOF의 인스턴스 변수를 선언하면 타이머 출력은 **인스턴스 이름.Q**, 경과 시간은 **인스턴스 이름.ET**로 변수가 자동 생성된다.

(다) TOF의 인스턴스 변수 T2를 선언한다.

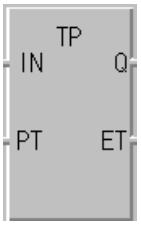
(라) 타이머 T2의 설정 신간을 3초(T#3S)로 설정한다.

(마) 기동 스위치 %IX0.0.0을 ON하면 타이머 출력 T2.Q가 ON이 된다.

(바) 기동 스위치 %IX0.0.0을 OFF하면 경과 시간(T2.ET)이 디지털 표시기에 출력 된다.

(사) 경과 시간 T2.ET가 설정 시간 3초에 도달하면 타이머 출력 T2.Q가 OFF된다.

바. **TP** (Pulse Timer)

펑 션	설 명
	입력 IN : 타이머 기동 조건 PT : 설정시간(Preset Time) 출력 Q : 타이머 출력 ET : 경과시간(Elapsed Time)

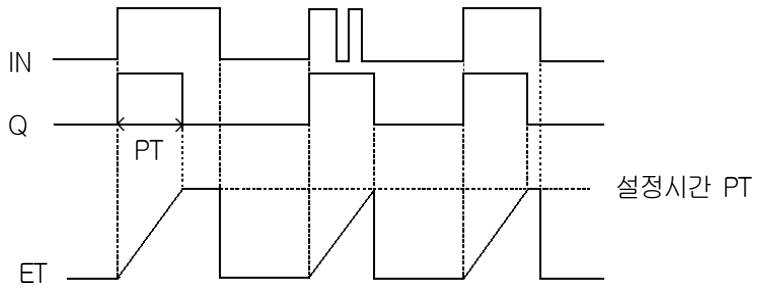
가) 기능

(가) IN이 "1"이 되면 PT에 의해서 지정된 설정 시간 동안만 Q가 "1"이 되고, ET가 PT에 도달하면 자동으로 "0"이 된다.

(나) 경과 시간 ET는 IN이 "1"이 되었을 때부터 증가하며 PT에 이르면 값을 유지하다가 IN이 "0"이 될 때 "0"의 값이 된다.

(다) ET가 증가할 동안은 IN이 "0"이 되거나 재차 "1"이 되어도 영향이 없다.

나) 타임 차트



다) 프로그램 예

(가) TP는 펑션 블록이므로 연산 중 누계 되는 데이터를 잠시 보관하기 위한 인스턴스 변수를 반드시 선언해야 한다.

(나) GMWIN에서 프로그램 편집 시 TP의 인스턴스 변수를 선언하면 타이머 출력은 **인스턴스 이름.Q**, 경과 시간은 **인스턴스 이름.ET**로 변수가 자동 생성된다.

(다) TP의 인스턴스 변수 T3를 선언한다.

(라) 타이머 T3의 설정 시간을 3초(T#3S)로 설정한다.

(마) 기동 스위치 %IX0.0.0를 OFF→ON하면 타이머 출력 T3.Q가 3초 동안 ON했다 OFF된다.

(바) T3.ET가 증가할 동안 기동 스위치가 OFF되거나 다시 ON되어도 영향이 없다.

(사) T3.ET가 증가할 동안 경과 시간(T3.ET)이 디지털 표시기에 출력된다.

라) 프로그램 예

13) 리턴(Return) (<RET>)

프로그램 중 [Return]을 만나면 프로그램을 종료하는 명령으로 [Return] 이후의 프로그램은 실행하지 않는다. (단축키 Shift + F7)

(1) 기능 설명

① 도구 상자에서 <RET> 를 선택한다.

② LD 프로그램 윈도우에서 원하는 위치로 마우스를 옮긴 후에 왼쪽 단추를 누른다.

③ LD 프로그램 윈도우에서 리턴을 입력할 위치로 커서를 이동시킨다.

④ 메뉴 도구 상자 내의 명령 중 리턴(Shift-F7)을 선택한다.

(2) 프로그램의 적용

아래 회로에서와 같이 2행 [Return]의 표시가 있는 부분까지만 실행한다.

프로그램 중 [Return]을 만나면 프로그램을 종료함으로 4행의 프로그램은 실행하지 않는다.

14) 분기(JUMP) (»)

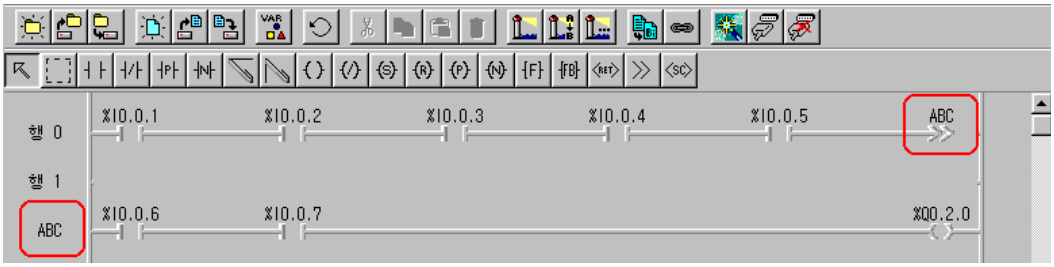
점프는 LD 프로그램 중에 분기하고자 하는 곳(레이블 명)으로 직접 갈 수 있는 방법이다. 목적지는 레이블로 나타낸다. 서브루틴을 포함하는 LD 프로그램에서 주 프로그램에 포함된 점프는 주 프로그램에 포함된 레이블을 입력해야 하고, 서브루틴 영역에 있는 점프는 서브루틴 안에 있는 레이블로만 분기할 수 있다.

※ 프로그램의 예

① 도구 상자에서 » 를 선택한다.

② LD 프로그램 윈도우에서 원하는 위치로 마우스를 옮긴 후에 왼쪽 단추를 누른다.

③ 아이콘 » 을 더블 클릭하여 레이블명을 ABC라고 입력한다.



15) 서브루틴(CALL) (<SC>)

LD 프로그램에서 END 명령어를 만나면 프로그램을 종료한다. 그러나 불가피하게 END 명령어 이후에 프로그램이 존재할 경우 그 프로그램을 불러와야 할 때가 있는데 이러한 경우, 서브루틴 명령어를 사용하면 편리하다.

※ 프로그램의 예

(1) 서브루틴

① 도구 상자에서 <SC>을 선택한다.

② LD 프로그램 윈도우에서 원하는 위치로 마우스를 옮긴 후에 왼쪽 단추를 누른다.

③ 아이콘 를 선택하고 작성된 서브루틴 LD를 더블 클릭 하여 레이블 명을 “ABC”라고 입력한다.

(2) END 명령어

① 프로그램의 끝 행에 마우스를 두고 (행2)를 더블 클릭 한다.

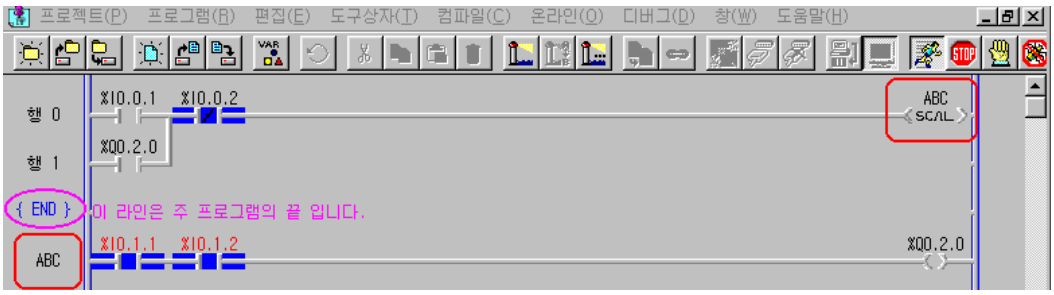
② 레이블 / 렁 설명문 / 주 프로그램 끝 표시 대화상자에서 “주프로그램 끝표시”를 선택한다.

(3) 레이블

① END 명령어 이 후에 불러오기를 하고자하는 행에 마우스를 두고 더블 클릭 한다.

② 레이블 / 렁 설명문 / 주 프로그램 끝 표시 대화상자에서 레이블을 선택한다.

③ 레이블 추가 대화상자에서 레이블 명을 “ABC”로 정의한다.(최대 16자)

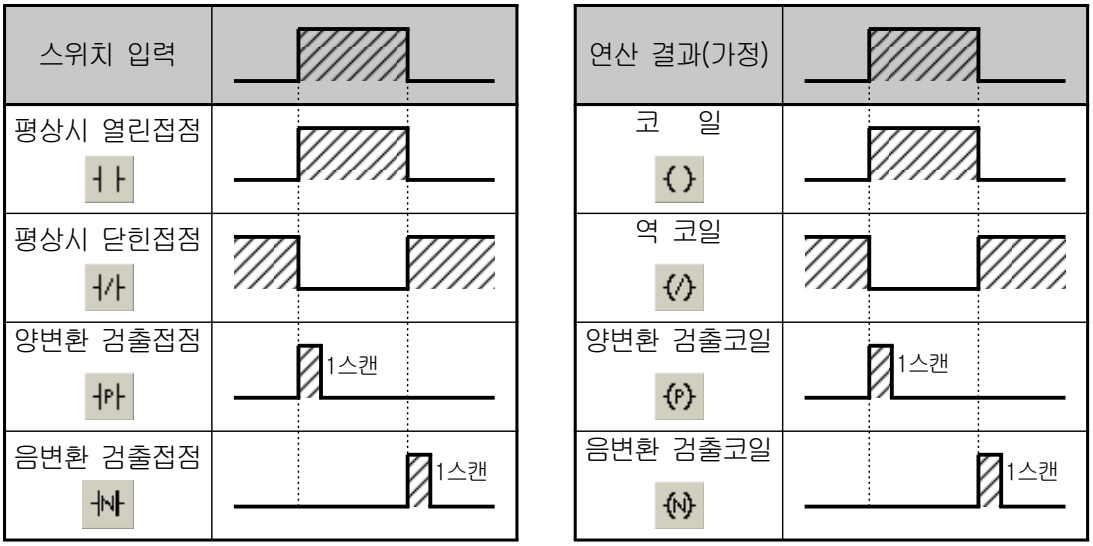


(4) 프로그램 설명

회로에서와 같이 %I0.1.1과 %I0.1.2의 접점이 동작하더라도 %Q0.2.0이 동작하지 않음을 볼 수 있다. 그 이유는 END 명령어 이후에 존재하기 때문에 서브루틴 부름이 실행되지 않으면 동작할 수 없다.

본 회로에서는 %I0.0.1 접점이 동작하면 서브루틴이 ABC를 부르며, %I0.1.1과 %I0.1.2 접점이 동작 후 출력 %Q0.2.0이 동작한다.

16) 연산에 따른 타임차트

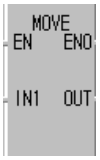
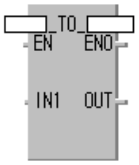
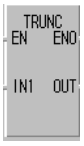


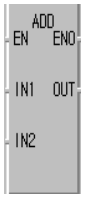
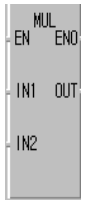
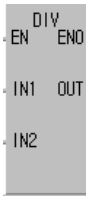
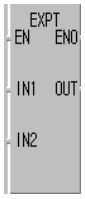
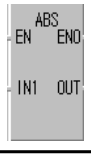
연산에 따른 입·출력 도구 타임차트

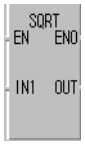
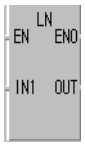
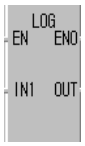
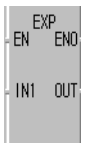
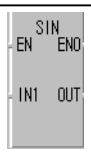
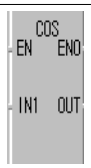
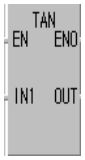
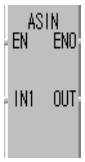
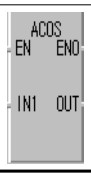
7-2. 시퀀스 연산자

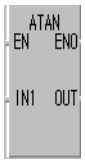
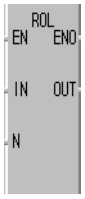
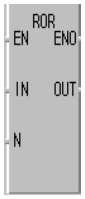
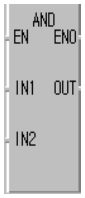
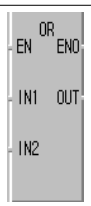
구분	명 령 어	기 호	기 능 설 명	비 고
시퀀스 연산자	평상시 열린접점		접점 연산	A 접점
	평상시 닫힌접점		접점 연산	B 접점
	양변환 검출접점		상승 에지에서 1 Scan On 접점	
	음변환 검출접점		하강 에지에서 1 Scan On 접점	
	코일		연산결과 출력	
	역 코일		연산결과 반전출력	
	양변환 검출코일		상승에지에서 1 Scan On 접점	
	음변환 검출코일		하강에지에서 1 Scan On 접점	
	셋팅 코일		연산결과 세트출력	
	리셋 코일		연산결과 리셋출력	
	점프		레이블 위치로 점프	
	프로그램 종료		Return을 만나면 현재 프로그램 종료	
			END 명령어를 만나면 현재 프로그램 종료	

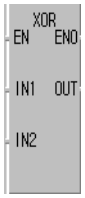
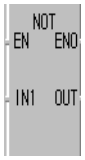
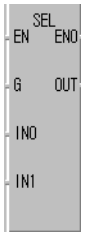
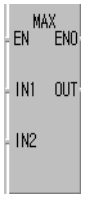
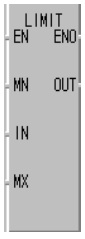
7-3. 펑션 일람

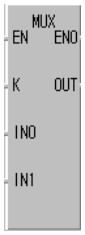
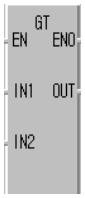
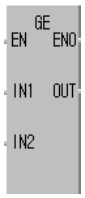
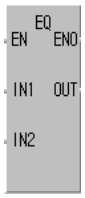
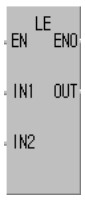
구분	명 령 어	기 호	기 능 설 명	비 고
전송펑션	MOVE		데이터 전송 IN 1 : 전송원(모든 형식) OUT : 전송선(모든 형식)	
변환펑션	***_ TO_***		데이터 형식 변환 펑션 IN 1 : 전송원 OUT : 전송선 변환 명령 펑션의 종류 SINT_TO_INT외 14종 INT_TO_SINT 외 14종 DINT_TO_SINT 외 14종 LINT_TO_SINT 외 14종 USINT_TO_SINT 외 14종 UINT_TO_SINT 외 14종 UDINT_TO_SINT 외 14종 ULINT_TO_SINT 외 14종 BYTE_TO_SINT 외 14종 WORD_TO_SINT 외 14종 DWORD_TO_SINT 외 14종 LWORD_TO_SINT 외 14종 BCD_TO_SINT 외 7종 REAL_TO_SINT 외 7종 LREAL_TO_SINT 외 7종 STRING_TO_SINT 외18종 NUM_TO_STRING TIME_TO_UDINT 외 2종 DATE_TO_UINT 외 2종 TOD_TO_UDINT 외 2종 DT_TO_DATE 외 2종	LINT ULINT LWORD REAL LREAL 관련펑션은 GM1,GM2 만 가능
	TRUNC		실수를 정수로 변환 IN 1 : 전송원(REAL, LREAL) OUT : 전송선(DINT, LINT)	GM1,GM2 전용

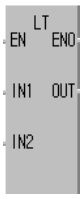
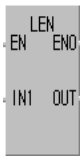
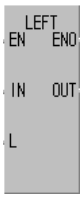
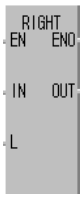
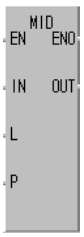
구분	명 령 어	기 호	기 능 설 명	비 고
수치연산평션	ADD		덧셈 펑션 IN1 ~ IN8 : 더할 값(Any_INT) OUT : 결과값(Any_INT)	
	SUB		뺄셈 펑션 IN1 : 연산원(Any_INT) IN2 : 뺄 값(Any_INT) OUT : 결과값(Any_INT)	
	MUL		곱셈 펑션 IN1 ~ IN8 : 곱할 값(Any_INT) OUT : 결과값(Any_INT)	
	DIV		나눗셈(몫) IN1 : 연산원(Any_INT) IN2 : 나눌값(Any_INT) OUT : 몫(Any_INT)	
	MOD		나눗셈(나머지) IN1 : 연산원(Any_INT) IN2 : 나눌 값(Any_INT) OUT : 나머지 값(Any_INT)	
	EXPT		지수 연산 IN1 : 정수(Any_REAL) IN2 : 지수(Any_REAL) OUT : 결과값(Any_REAL)	GM1,GM2 전용
	ABS		절대값 IN1 : 정수(Any_INT) OUT : 결과값(Any_INT)	

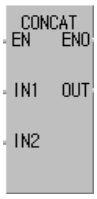
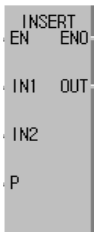
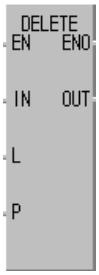
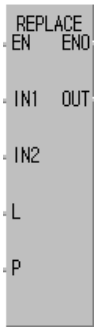
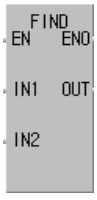
구분	명 령 어	기 호	기 능 설 명	비 고
수치연산평션	SQRT		제곱근10 IN1 : 연산원(Any_REAL) OUT : 결과값(Any_REAL)	GM1,GM2 전용
	LN		자연 대수 IN1 : 연산원(Any_REAL) OUT : 결과값(Any_REAL)	GM1,GM2 전용
	LOG		상용 대수 IN1 : 연산원(Any_REAL) OUT : 결과값(Any_REAL)	GM1,GM2 전용
	EXP		자연 지수 IN1 : 연산원(Any_REAL) OUT : 결과값(Any_REAL)	GM1,GM2 전용
삼각평션	SIN		싸인 연산 IN1 : 연산원(Any_REAL) OUT : 결과값(Any_REAL)	GM1,GM2 전용
	COS		코싸인 연산 IN1 : 연산원(Any_REAL) OUT : 결과값(Any_REAL)	GM1,GM2 전용
	TAN		탄젠트 연산 IN1 : 연산원(Any_REAL) OUT : 결과값(Any_REAL)	GM1,GM2 전용
	ASIN		아크 싸인 연산 IN1 : 연산원(Any_REAL) OUT : 결과값(Any_REAL)	GM1,GM2 전용
	ACOS		아크 코싸인 연산 IN1 : 연산원(Any_REAL) OUT : 결과값(Any_REAL)	GM1,GM2 전용

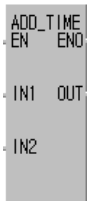
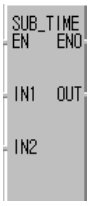
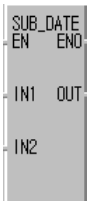
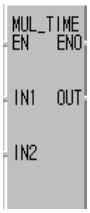
구분	명 령 어	기 호	기 능 설 명	비 고
삼각펑션	ATAN		아크 탄젠트 연산 IN1 : 연산원(Any_REAL) OUT : 결과값(Any_REAL)	GM1,GM2 전용
이동펑션	SHL		비트열 왼쪽으로 이동 IN : 전송원(Any_BIT) N : 이동할 비트 수 (INT) OUT : 전송선(Any_BIT)	
	SHR		비트열 오른쪽으로 이동 IN : 전송원(Any_BIT) N : 이동할 비트 수 (INT) OUT : 전송선 (Any_BIT)	
회전명령	ROL		비트열 왼쪽으로 이동 IN : 전송원(Any_BIT) N : 이동할 비트 수 (INT) OUT : 전송선 (Any_BIT)	
	ROR		비트열 오른쪽으로 이동 IN : 전송원(Any_BIT) N : 이동할 비트 수 (INT) OUT : 전송선 (Any_BIT)	
논리연산	AND		논리곱 IN1 ~ IN8 : 연산원(Any_BIT) OUT : 결과값(Any_BIT)	
	OR		논리합 IN1 ~ IN8 : 연산원(Any_BIT) OUT : 결과값(Any_BIT)	

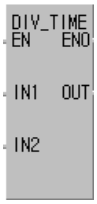
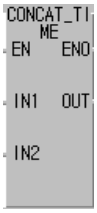
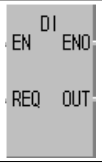
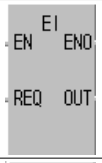
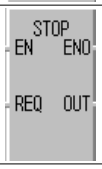
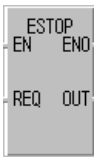
구분	명 령 어	기 호	기 능 설 명	비 고
논리연산	XOR		배타적 논리합 IN1 ~ IN8 : 연산원(Any_BIT) OUT : 결과값(Any_BIT)	
	NOT		논리 반전 IN1, IN2 : 연산원(Any_BIT) OUT : 결과값(Any_BIT)	
선택펑션	SEL		2개중 선택 G : 출력 선택(BOOL) IN1 : G가 Off일 경우 선택값(Any) IN1 : G가 On일 경우 선택값(Any) OUT : 출력값(Any)	
	MAX		최대값 구하기 IN1 ~ IN8 : 선택값(Any_NIT) OUT : 최대 출력값(Any_INT)	
	MIN		최소값 구하기 IN1 ~ IN8 : 선택값(Any_NIT) OUT : 최소 출력값(Any_INT)	
	LIMIT		상하한 제한값 MN : 하한값(Any_NIT) IN : 전송원(Any_INT) MX : 상한원 (Any_NIT) OUT : 출력값(Any_INT)	

구분	명 령 어	기 호	기 능 설 명	비 고
선택펑션	MUX		최대 7개중 선택 K : 선택 입력 번호 IN0 : 전송원 0번(Any) IN1 : 전송원 1번(Any) IN2 : 전송원 2번(Any) IN3 : 전송원 3번(Any) IN4 : 전송원 4번(Any) IN5 : 전송원 5번(Any) IN6 : 전송원 6번(Any) OUT : 출력값(Any)	
비교펑션	GT(>)		비교 펑션 IN1 ~ IN8 : 비교 데이터(Any) OUT : 출력(BOOL) IN1>IN2>.....IN7>IN8의 조건 성립시 OUT 출력 On	
	GE(≥)		비교 펑션 IN1 ~ IN8 : 비교 데이터(Any) OUT : 출력(BOOL) IN1≥IN2≥.....IN7≥IN8의 조건 성립시 OUT 출력 On	
	EQ(=)		비교 펑션 IN1 ~ IN8 : 비교 데이터(Any) OUT : 출력(BOOL) IN1=IN2=.....IN7=IN8의 조건 성립시 OUT 출력 On	
	LE(≤)		비교 펑션 IN1 ~ IN8 : 비교 데이터(Any) OUT : 출력(BOOL) IN1≤IN2≤.....IN7≤IN8의 조건 성립시 OUT 출력 On	

구분	명 령 어	기 호	기 능 설 명	비 고
비교펑션	LT(<)		비교 펑션 IN1 ~ IN8 : 비교 데이터(Any) OUT : 출력(BOOL) IN1<IN2<.....<IN7<IN8의 조건 성립시 OUT 출력 On	
	NE(≠)		비교 펑션 IN1 , IN2 : 비교 데이터(Any) OUT : 출력(BOOL) IN1≠IN2의 조건 성립시 OUT 출력 On	
문자열 펑션	LEN		문자열 길이 IN1 : 문자열 입력(STRING) OUT : 문자열 길이(INT)	
	LEFT		문자열 왼쪽 부분 전송 IN : 문자열 입력(STGING) L : 문자열 길이(INT) OUT : 문자열 출력(STRING)	
	RIGHT		문자열 오른쪽 부분 전송 IN : 문자열 입력(STRING) L : 문자열 길이(INT) OUT : 문자열 출력(STRING	
	MID		문자열 중간 부분 전송 IN : 문자열 입력(STRING) L : 문자열 길이(INT) P : 문자열 선두 위치(INT) OUT : 문자열 출력(STRING	

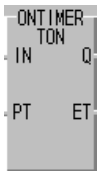
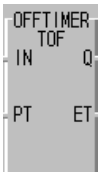
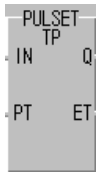
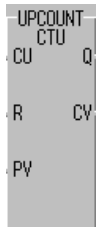
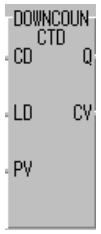
구분	명 령 어	기 호	기 능 설 명	비 고
문자열 펑션	CONCAT		문자열 연결 입력 문자열을 순서대로 연결 IN1 ~ IN8 : 문자열(STRING) OUT : 문자열 출력(STRING)	
	INSERT		문자열 삽입 IN1 : 문자열 입력(STRING) IN2 : 삽입할 문자열(STRING) P : 문자열 선두 위치(INT) OUT : 문자열 출력(STRING)	
	DELETE		문자열 삭제 IN1 : 문자열 입력(STRING) L : 삭제할 문자열(INT) P : 문자열 선두 위치(INT) OUT : 문자열 출력(STRING)	
	REPLACE		문자열 대체 IN1 : 문자열 입력(STRING) IN2 : 대체할 문자열(STRING) P : 문자열 선두 위치(INT) OUT : 문자열 출력(STRING)	
	FIND		문자열 찾기 IN1 : 문자열 입력(STRING) IN2 : 검색할 문자열(STRING) OUT : 문자열 출력(INT))	

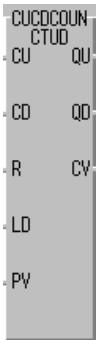
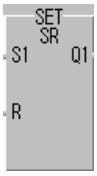
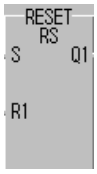
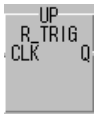
구분	명 령 어	기 호	기 능 설 명	비 고
날짜 시간 펑션	ADD_TIME		시간 더하기 IN1 : 시각 또는 시간 (TIME, TOD, TD) IN2 : 더할 시간(TIME) OUT : 결과 시각 또는 시간 (TIME, TOD, TD)	
	SUB_TIME		시간 빼기 IN1 : 시각 또는 시간 (TIME, TOD, TD) IN2 : 뺄 시간(TIME) OUT : 결과 시각 또는 시간 (TIME, TOD, TD)	
	SUB_DATE		날짜 빼기 IN1 : 날짜(DATE) IN2 : 뺄 날짜(DATE) OUT : 결과 시간(TIME)	
	SUB_TOD		시각 빼기 IN1 : 시각(TIME OF DAY) IN2 : 뺄 시각(TIME OF DAY) OUT : 결과 시간(TIME)	
	SUB_DT		날짜 시각 빼기 IN1 : 시각(DATE&TIME) IN2 : 뺄 시각(DATE&TIME) OUT : 결과 시간(TIME)	
	MUL_TIME		시간 곱하기 IN1 : 입력 시간(TIME) IN2 : 곱할 값(INT) OUT : 결과 시간(TIME)	

구분	명 령 어	기 호	기 능 설 명	비 고
날짜시간펑션	DIV_TIME		시간 나누기 IN1 : 입력 시간(TIME) IN2 : 나눌 값(INT) OUT : 결과 시간(TIME)	
	CONCAT_TIME		날짜와 시각 연결 IN1 : 입력 날짜(DATE) IN2 : 입력 시각(TOD) OUT : 결과 날짜 시각(DT)	
시스템 제어 펑션	DI		인터럽트 금지 REQ : 금지 요구(BOOL) OUT : 금지 확인(BOOL)	
	EI		인터럽트 허가 REQ : 허기 요구(BOOL) OUT : 허가 확인(BOOL)	
	STOP		PLC 비상 정지 요구 REQ : 정지 요구(BOOL) OUT : 정지 확인(BOOL)	
	ESTOP		PLC 비상 정지 요구 REQ : 정지 요구(BOOL) OUT : 정지 확인(BOOL)	
	DIREC_IN		입력 데이터 즉시 갱신 BASE : 베이스 모듈 번호 SLOT : 입력 모듈 슬롯 위치 MASK_L : 하위 32Bit중 갱신 하지 않을 Bit 지정(DWORD) MASK_H : 상위 32Bit중 갱신 하지 않을 Bit 지정(DWORD) OUT : 실행 완료(BOOL)	GM5 제외

구분	명 령 어	기 호	기 능 설 명	비 고
시스템 제어 펑션	DIREC_IN5	DIREC_IN5 EN ENO MODL OUT MASK	입력 데이터 즉시 갱신 MODL : 입력 모듈 번호 MASK : 하위 32Bit중 갱신하지 않을 Bit 지정(DWORL)	GM5 전용
	DIREC_O	DIREC_O EN ENO BASE OUT SLOT MASK_L MASK_H	출력 데이터 즉시 갱신 BASE : 베이스 모듈 번호 SLOT : 출력 모듈 슬롯 위치 MASK_L : 하위 32Bit중 갱신하지 않을 Bit 지정(DWORD) MASK_H : 상위 32Bit 중 갱신하지 않을 Bit 지정(DWORD) OUT : 실행 완료(BOOL)	GM5 제외
	DIREC_OUT5	DIREC_OUT5 EN ENO MODL OUT MASK	출력 데이터 즉시 갱신 MODL : 출력 모듈 번호 MASK : 하위 32Bit중 갱신하지 않을 Bit 지정(DWORD) OUT : 실행 완료(BOOL)	GM5 전용
	WDT_RST	WDT_RST EN ENO REQ OUT	위치 독 타이머 리셋 REQ : 리셋 요구(BOOL) OUT : 실행 완료(BOOL)	

7-4. 펑션 블록 일람

구분	명 령 어	기 호	기 능 설 명	비 고
타이머 펑션 블록	TON		On 딜레이 타이머 IN : 동작 개시 신호(BOOL) PT : 설정 시간(TIME) Q : 출력(BOOL) ET : 현재값	
	TOF		Off 딜레이 타이머 IN : 동작 개시 신호(BOOL) PT : 설정 시간(TIME) Q : 출력(BOOL) ET : 현재값	
	TP		펄스 타이머 IN : 동작 개시 신호(BOOL) PT : 설정 시간(TIME) Q : 출력(BOOL) ET : 현재값	
카운터 펑션 블록	CTU		가산 타이머 CU : 펄스 입력(BOOL) R : 현재 값 리셋(BOOL) PV : 설정 값(INT) Q : 출력(BOOL) CV : 현재값(INT)	
	CTD		감산 카운터 CD : 펄스 입력(BOOL) LD : 설정 값 Read(BOOL) PV : 설정 값(INT) Q : 출력(BOOL) CV : 현재 값(INT)	

구분	명 령 어	기 호	기 능 설 명	비 고
카운터펑션블록	CTUD		가감산 카운터 CU : 가산 펄스 입력(BOOL) CD : 감산 펄스 입력(BOOL) R : 현재 값 리셋(BOOL) LD : 설정 값 Read(BOOL) PV : 설정 값(INT) QU : 가산 카운트 출력(BOOL) QD : 감산 카운트 출력(BOOL) CV : 현재 값(INT)	
	SEMA		시스템 자원 제어(Semaphore) CLAIM : 자원 독점 요구(BOOL) RELEASE : 자원 해방(BOOL) BUSY : 자원 취득 불가(BOOL)	
펑션블록	SR		Set 우선 쌍안정(Bistable) S1 : Set신호(BOOL) R : Reset신호(BOOL) Q1 : 출력(BOOL)	
	RS		Reset 우선 쌍안정(Bistable) S : Set신호(BOOL) R1 : Reset신호(BOOL) Q1 : 출력(BOOL)	
	R_TRIG		상승 에지 검출 CLK : 입력(BOOL) Q : 출력(BOOL)	
	F_TRIG		하강 에지 검출 CLK : 입력(BOOL) Q : 출력(BOOL)	

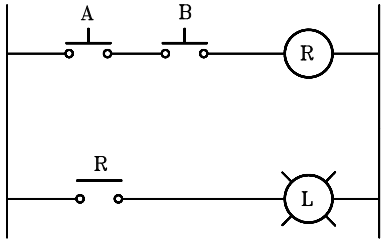
8장. 시퀀스 기본 회로 및 LD 예제 프로그램

8-1. AND 회로

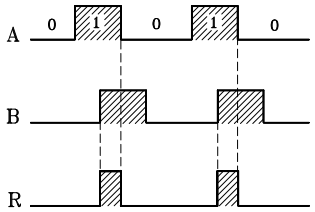
여러 개의 입력이 있을 때 모든 입력이 존재할 때에만 출력이 나타나는 회로를 AND회로라고 하며 직렬 스위치 회로와 같다.

[그림 8-1]은 두 개의 입력 A와 B가 모두 ON일 때에만 릴레이 코일 R이 여자 되고 R접점이 닫혀 램프가 점등되는 AND회로이다.

이와 같은 직렬회로는 한 대의 프레스에 여러 명의 작업자가 함께 작업할 때, 안전을 위해 각 작업자마다 프레스 기동용 누름버튼을 설치하여 모든 작업자가 스위치를 누를 때에만 동작 되도록 하는 경우에 적용된다. 또 기계의 각 부분이 소정의 위치까지 진행되지 않으면 다음 동작으로 이행을 금지하는 경우 등 그 응용 범위가 넓은 회로이다.



(a) 릴레이 회로

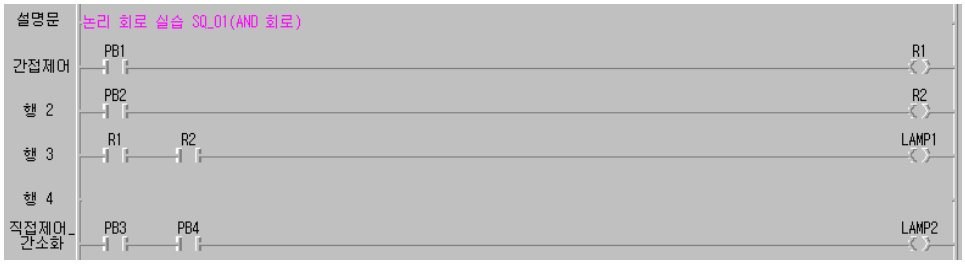


(b) 타임 차트

입 력		출 력
A	B	R
0	0	0
0	1	0
1	0	0
1	1	1

(c) 진리표

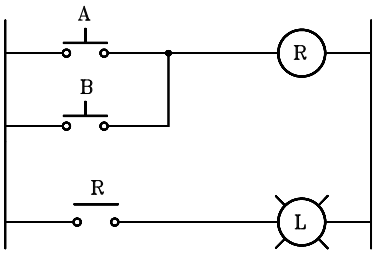
[그림 8-1] AND 회로



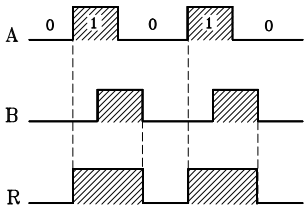
8-2. OR 회로

OR회로는 여러 개의 입력신호 중 하나 또는 그 이상의 신호가 ON되었을 때 출력을 내는 회로로서 병렬회로라고 한다.

[그림 8-2]에서 누름버튼 스위치 A가 눌려지거나, 아니면 B가 눌려져도, 또는 A와 B가 동시에 눌려져도 릴레이 R이 동작되어 램프가 점등된다.



(a) 릴레이 회로

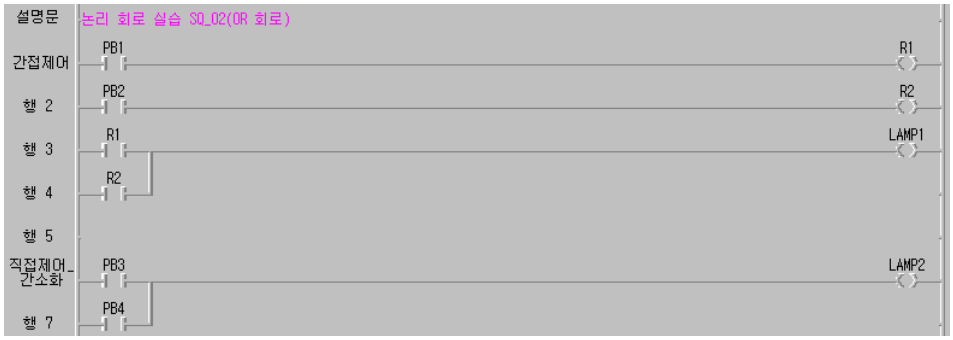


(b) 타임 차트

입 력		출 력
A	B	R
0	0	0
0	1	1
1	0	1
1	1	1

(c) 진리표

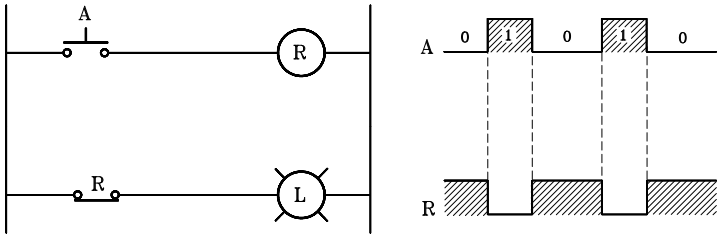
[그림 8-2] OR 회로



8-3. NOT 회로

NOT 회로는 출력이 입력의 반대가 되는 회로로서 입력이 0이면 출력이 1이고 입력이 1이면 출력이 0이 되는 부정회로이다.

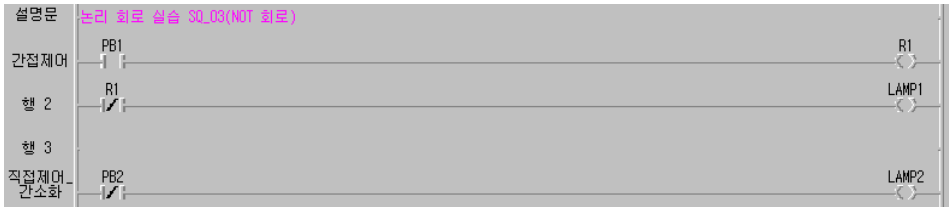
[그림 8-3]은 릴레이의 b접점을 이용한 NOT 회로로서 누름버튼 스위치 A가 눌려 있지 않은 상태에서는 램프가 점등되어 있고, 누름버튼 스위치 A가 눌려지면 R접점이 열려 램프가 소등하는 NOT회로이다.



입 력	출 력
A	R
0	1
1	0

(a) 릴레이 회로　　(b) 타임 차트　　(c) 진리표

[그림 8-3] NOT 회로

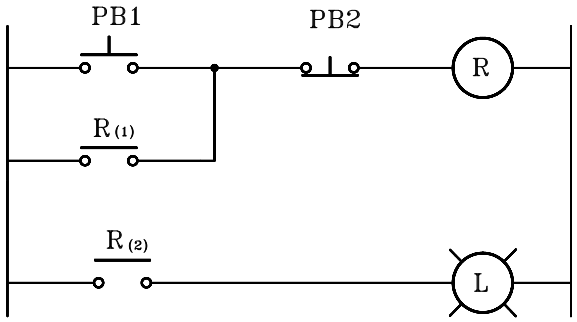


8-4. 자기 유지 회로

릴레이의 기능 중에는 메모리 기능이 있다고 앞서 설명하였다. 이 릴레이의 메모리 기능이란 릴레이는 자신의 접점으로 자기유지 회로를 구성하여 동작을 기억시킬 수 있다는 것이다. 그림 8-4는 릴레이의 자기유지 회로이며, 자기유지 접점 R(1)은 누름버튼 스위치 PB1에 병렬로 접속한다.

동작원리는 누름버튼 스위치 PB1을 누르면 릴레이가 동작되고, 접점 R(1)과 R(2)가 동시에 닫혀 램프가 점등한다. 이때 누름버튼 스위치 PB1에서 손을 떼도 전류는 R(1)접점과 PB2를 통해 코일에 계속 흐르므로 동작 유지가 가능하다. 즉 PB1이 복귀하여도 R(1) 접점에 의해 R의 동작회로가 유지된다.

자기 유지의 해제는 누름버튼 스위치 PB2를 누르면 R이 복귀되고 접점 R(1)과 R(2)가 열려 회로는 초기 상태로 되돌아간다.



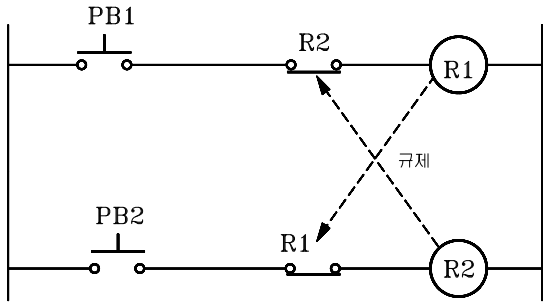
[그림 8-4] 자기유지 회로

8-5. 인터록(inter - lock) 회로

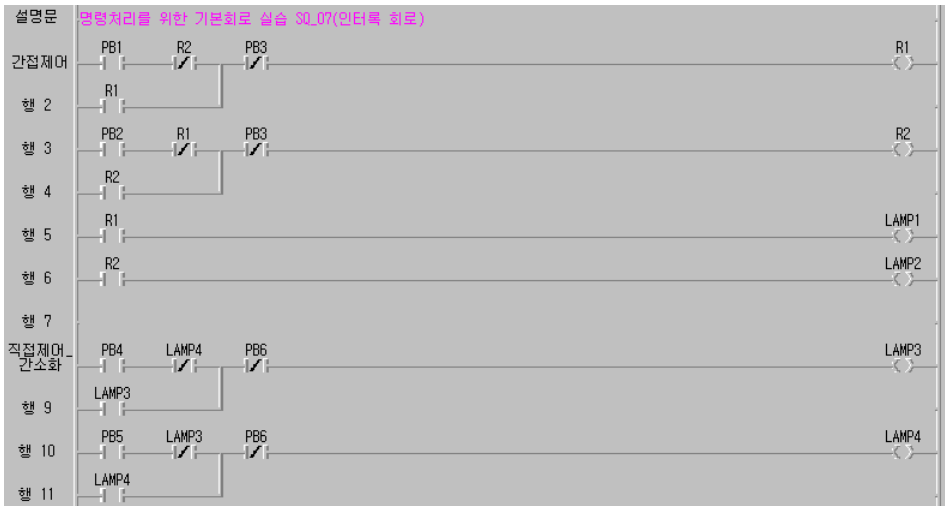
기기의 보호나 작업자의 안전을 위해 기기의 동작상태를 나타내는 접점을 사용하여 관련된 기기의 동작을 금지하는 회로를 인터록 회로라 하며, 다른 말로 선행 동작 우선 회로 또는 상대 동작 금지 회로라고도 한다.

인터록은 릴레이의 b접점을 상대측 회로에 직렬로 연결하여 어느 한 릴레이가 동작중일 때에는 관련된 다른 릴레이는 동작할 수 없도록 규제한다.

[그림 8-5]는 누름버튼 스위치 PB1이 ON되어 R1릴레이가 동작하면 PB2가 눌려져도 R2릴레이는 동작할 수 없다. 또한 PB2가 먼저 입력되어 R2가 동작하면 R1릴레이는 역시 동작 할 수 없다.

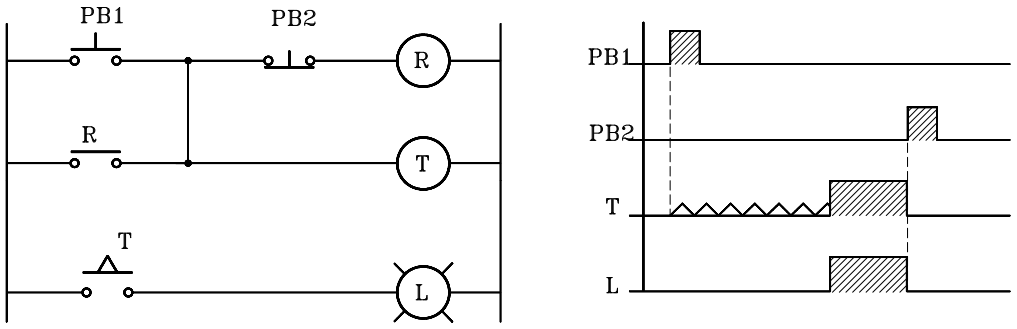


[그림 8-5] 인터록 회로



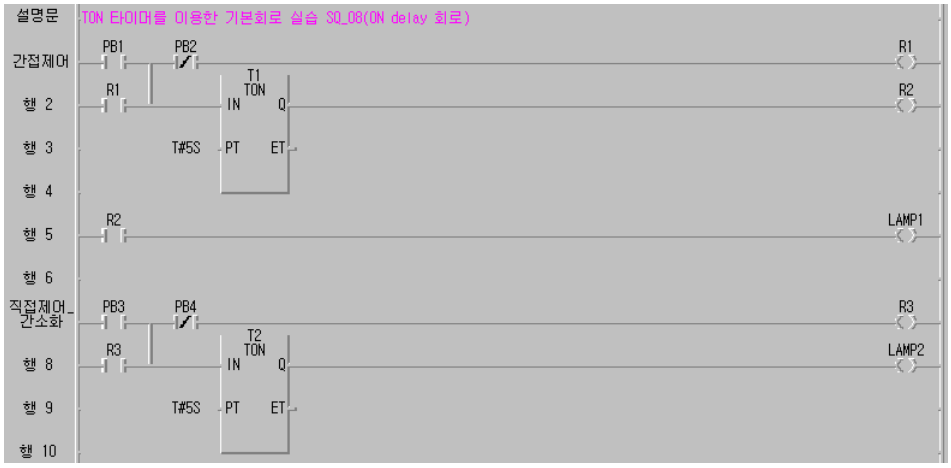
8-6. 온 딜레이(ON - delay) 회로

입력 신호를 준 후에 곧바로 출력이 나타나지 않고 계획된 시간만큼 늦게 출력이 나타나도록 설계한 회로를 지연회로라 하며, 지연회로에는 ON시간 지연 회로와 OFF시간 지연 회로가 있다. [그림 8-6[은 ON시간 지연 작동 회로로서 누름버튼 스위치 PB1을 누르면 타임 릴레이가 동작하기 시작하여 미리 설정해 둔 시간이 경과되면 타이머 접점이 닫혀 램프가 점등되며, 누름버튼 스위치 PB2를 누르면 타임 릴레이가 복귀하고 타이머 접점도 열려 램프가 소등 되는 회로이다.



(a) 회로도 (b) 타임 차트

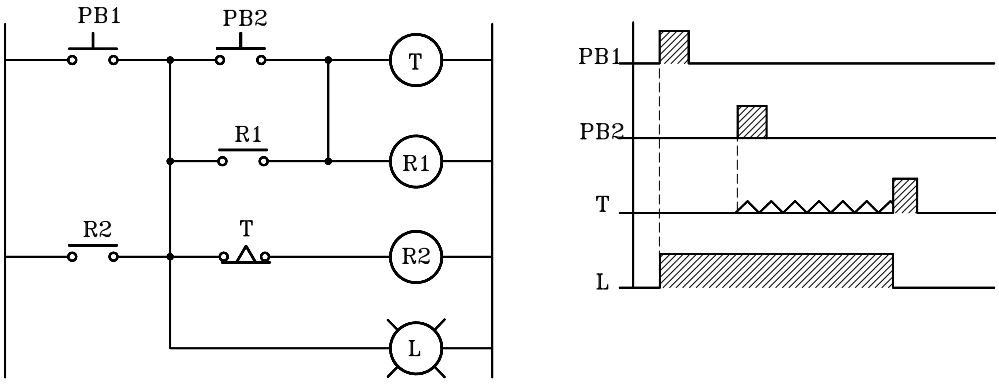
[그림 8-6] 온 딜레이 회로



8-7. 오프 딜레이(OFF - delay) 회로

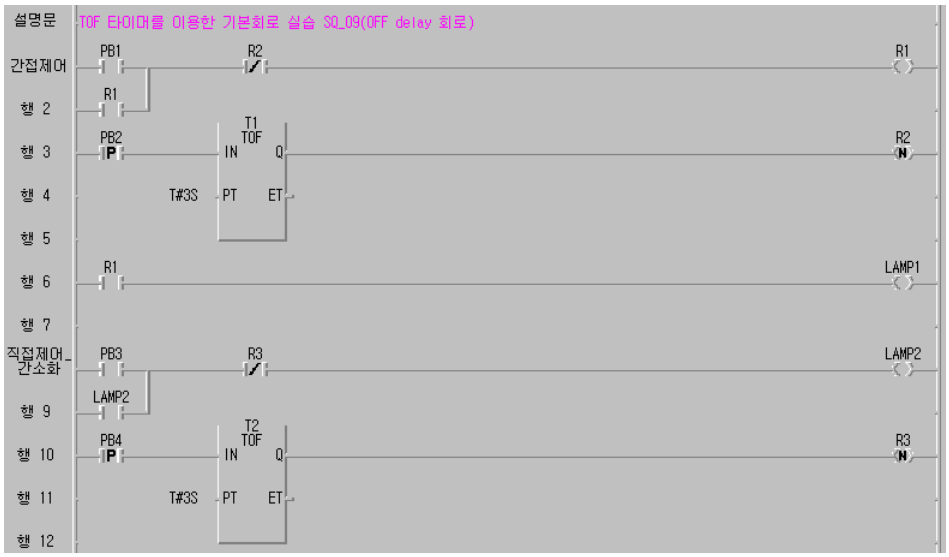
오프 딜레이 회로는 복귀신호가 주어지면 출력이 곧바로 복귀되지 않고, 계획된 시간 후에 부하가 개방되는 회로로서 ON딜레이 타이머의 b접점을 이용하거나, OFF딜레이 타이머의 a접점을 이용하여 회로를 구성할 수 있다.

[그림 8-7]은 OFF딜레이 회로의 일례로 누름버튼 스위치 PB1을 누르면 램프가 점등되고 PB2를 누르면 곧바로 램프가 소등되지 않고 타이머에 설정된 시간 후에 소등되는 오프 딜레이 회로이다.



(a) 회로도 (b) 타임 차트

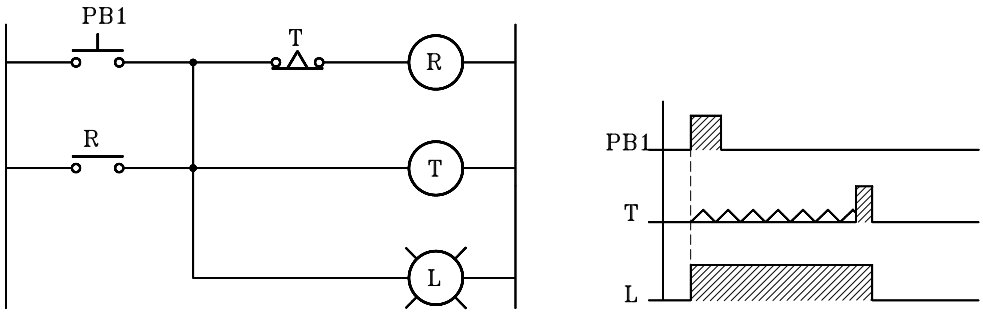
[그림 8-7] 오프 딜레이 회로



8-8. 일정시간 동작 회로(one shot)

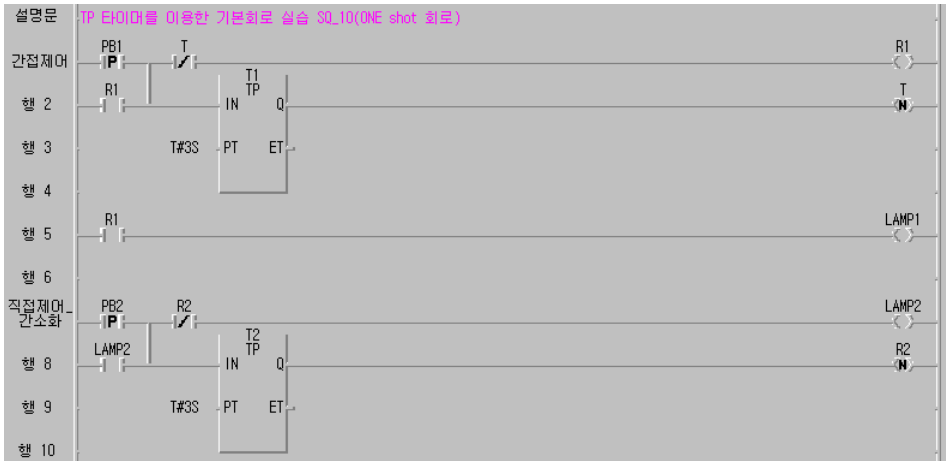
이 회로는 누름버튼 스위치 등의 입력이 주어지면 부하가 동작하기 시작하여 타이머에 설정된 시간이 경과되면 스스로 복귀하는 회로이다.

[그림 8-8]이 이 회로의 일예로, 누름버튼 스위치 PB1을 누르면 릴레이 코일 R1이 여자 되어 자기 유지되고, 램프가 점등됨과 동시에 타이머가 동작하기 시작한다. 타이머에 설정된 시간이 경과되면 타이머 b접점이 개방되어 램프가 소등되는 회로이다. 이와 같은 일정 시간 동작회로는 가정에서 현관 출입문 등에서 이용되고 있다.



(a) 회로도 (b) 타임 차트

[그림 8-8] 일정시간 동작 회로



< 부 록 > 용 어 설 명

용 어	정 의	비 고
모 듈 (Module)	시스템을 구성하는 일정한 기능을 가진 표준화된 요소로서 베이스에 삽입되도록 조립된 입·출력 보드와 같은 장치	CPU모듈 전원모듈, 입·출력모듈 등
유 닛 (Unit)	PLC 시스템의 동작상에서 최소 단위가 되는 모듈 또는 모듈의 집합체이며, 다른 모듈 또는 모듈의 집합체와 접속되어 PLC 시스템을 구성하는 것	기본유닛, 증설유닛
PLC 시스템 (PLC System)	PLC와 주변장치로 이루어지는 시스템으로 사용자 프로그램에 의하여 제어가 가능하도록 구성된 것	
콜드 리스타트 (Cold Restart)	모든 데이터(입·출력 이미지 영역, 내부 레지스터, 타이머, 카운터 등의 변수 및 프로그램)를 자동 또는 수동에 의하여 정해진 상태로 초기화 한 후 PLC 시스템 및 사용자 프로그램을 다시 시동하는 것	
웜 리스타트 (Warm Restart)	전원 Off 발생을 사용자 프로그램에 통지하는 기능을 가지고, 전원 Off가 발생한 후 사용자가 사전에 정한 데이터 및 사용자 프로그램에 따라 다시 시동하는 것	
핫 리스타트 (Hot Restart)	전원 Off 가 발생한 후 최대 허용 시간 이내에 PLC 시스템이 모두 데이터를 그 이전의 상태로 복귀시켜 다시 시동하는 것	
입·출력 이미지 영역	입·출력 상태를 유지하기 위하여 설정된 CPU 모듈의 내부 메모리 영역	
워치독 타이머 (Watchdog Timer)	프로그램의 미리 정해진 실행 시간을 감시하고 규정 시간 내에 처리가 완료되지 않을 때 경보를 발생하기 위한 타이머	
펑 션 (Function)	4칙 연산, 비교연산 등과 같이 연산 결과를 명령어 내부에 기억하지 않고 입력에 대한 연산 결과를 즉시 출력하는 연산 단위	
펑션블록 (Function Block)	타이머, 카운터, 등과 같이 명령어 내부에 연산 결과를 기억하여 여러 스캔에 걸쳐 기억된 연산 결과를 이용하는 연산 단위	
직접변수	이름, 타입을 별도로 선언하지 않고 사용하는 변수로 %I, %Q, %M 영역이 이 변수에 해당 함	%IX0.0.2 %QW1.2.1 %MD1234 등

용 어	정 의	비 고
네임드 변수 (Named 변수)	사용자가 이름, 타입 등을 선언하고 사용하는 변수로 '스위치0' = %i0.0.2, '결과값'=%MD1234 등과 같이 선언하면 %IX0.0.2와 %MD1234 대신에 '스위치0'과 '결과값'의 이름으로 프로그래밍 할 수 있음	
GMWIN	프로그램 작성, 편집, 컴파일 및 디버그 기능을 수행하는 GLOFA-GM 시리즈용 로더	
싱크(Sink)입력	입력 신호가 ON 될 때 스위치로부터 PLC 입력 단자로 전류가 유입되는 방식	
소스(Source) 입력	입력 신호가 ON 될 PLC 입력 단자로부터 스위치로 전류가 유입되는 방식	
싱글 출력	PLC 출력 접점이 ON 될 때 부하에서 출력 단자로 전류가 유입되는 방식	
소스 출력	PLC 출력 접점이 ON 될 때 출력 단자로부터 부하단자로 전류가 유입되는 방식	
다운로드	PLC에 프로그램을 쓸 때 업 - 로드 프로그램을 선택한다.	
업로드	메뉴 프로젝트 - PLC로부터 열기를 선택한다. PLC로부터 업-로드 파일을 읽어서 프로젝트, 프로그램 파일, 사용자 라이브러리를 생성한 후 프로젝트를 연다. GMWIN에 이미 같은 이름의 프로젝트나 프로그램이 있을 경우 Overwrite 시키거나, 사용자가 지정한 다른 디렉토리 또는 다른 이름으로 저장한다.	
리소스 글로벌 변수(Resource global variable)	리소스 글로벌 변수에서 정의한 변수는 리소스 안의 어떤 프로그램에서도 사용할 수 있다. 즉 프로그램과 프로그램간에 공유할 데이터들을 리소스 글로벌 변수에서 정의해 둔다. 프로그램에서 이 변수를 사용하기 위해서는 변수를 정의할 때 변수형을 VAR_EXTERNAL로 선언하여야 한다.	
디버깅 (Debugging)	디버깅이란 사용자가 작성한 PLC 프로그램이 정상적으로 동작하도록 프로그램의 오류를 찾아 제거하는 것	

용 어	정 의	비 고
메이크	프로젝트에 속해 있는 프로그램 중 컴파일 할 필요가 있는 프로그램만 컴파일 한 후 실행 파일을 생성한다.	
변수	변수란 프로그램 안에서 사용하는 데이터이며 값을 가지고 있다. 변수는 PLC의 입력이나 출력, 내부 메모리 등과 같이 변할 수 있는 대상을 가리킨다.	
인스턴스	인스턴스라는 것은 펑션 블록에서 사용하는 변수들의 집합이다. 펑션 블록은 내부에서 사용하는 변수뿐 아니라 출력 값도 자체에서 보관하여야 하므로 데이터 메모리를 가지고 있어야 하며 바로 그것이 인스턴스다.	
컨피그레이션 (Configuration)	컨피그레이션은 하나의 PLC 시스템을 의미한다. 하나의 PLC 시스템은 베이스와 CPU 모듈, I/O 모듈, 특수 모듈 등으로 이루어진다. 보통 하나의 PLC 시스템은 한 개의 CPU 모듈을 가지고 있다. GM1은 하나의 PLC 시스템에 최대 4개의 CPU 모듈을 장착할 수 있다.	
타임 차트 (Time Chart)	프로그램에서 선언된 변수, 글로벌 변수, I, Q, M영역, 시스템 플래그 중에서 타입이 BOOL로 선언된 것에 대해 시간에 따른 ON/OFF 상태를 모니터링 할 수 있다. BOOL 이 아닌 변수는 위의 변수 모니터링에서 모니터링 해야 한다.	
태스크 (Task)	프로그램을 실행시킬 조건을 태스크라고 한다. 태스크 정의에는 프로그램 기동 조건 지정과 우선순위 지정이 있다.	