

# codingOn x posco

K-Digital Training 스마트 팩토리 단기 8기

# 1. PLC 개요



## 과정

PLC 기초, PLC 프로그래밍

PLC 와 C# 연결 (LS electric 의 XG5000)

PLC 시뮬레이터(XG5000)를 이용한  
간단한 프로젝트와 문제풀이

PLC 프로젝트

※ 커리큘럼은 상황에 따라서 변동될 수 있습니다. 😊

# PLC 들어가기



# 1-1. PLC란?

# PLC란?

- **PLC(Programmable Logic Controller):** 산업 자동화 및 프로세스 제어를 목적으로 설계된 특수 목적의 컴퓨터.
- 입력 데이터를 읽고, 프로그램 된 논리로 처리하여, 출력 장치를 제어함.
- 원하는 기능은 주로 래더 다이어그램에 의해 프로그램되어 동작함.

# PLC란?

## ▪ PLC의 주요 특징

1. 내구성: 열, 진동, 먼지 등 산업 환경에 강함.
2. 유연성: 프로그램 수정으로 다양한 작업에 대응.
3. 실시간 제어: 입력 신호를 빠르게 처리하고 출력.

## ▪ PLC의 사용 분야

- 제조업, 자동차 산업, 발전 및 에너지, 충전 및 포장 시스템 등에 폭넓게 활용됨

# PLC 사용 산업 분야

분야	제어대상
식료산업	컨베이어 총괄 제어, 생산라인 자동 제어
제철, 제강 산업	작업장 하역 제어, 원료 수송 제어, 압연 라인 제어
섬유, 화학 공업	원료 수입 출하 제어, 직조 염색 라인 제어
자동차 산업	전송 라인 제어, 자동 조립 라인 제어, 도장 라인 제어
기계 산업	산업용 로봇 제어, 공작 기계 제어, 송·배수 펌프 제어
상하수도	정수장 제어, 하수 처리 제어, 송·배수 펌프 제어
물류 산업	자동 창고 제어, 하역 설비 제어, 반송 라인 제어
공장 설비	압축기 제어
공해 방지 사업	쓰레기 소각로 자동 제어, 공해 방지기 제어



## 1-2. PLC의 역사



# PLC의 역사



- 1967년 Medford 사에서 자동차 제조 회사 GM의 의뢰를 받아 **MODICON(Modular Digital Controller)**이라는 이름으로 장치를 개발 = **최초의 PLC**

# PLC의 역사

## ▪ GM에서 요구한 기능 조건

1. 프로그램 작성 및 변경이 용이하고 Sequence 변경을 현장에서 할 수 있을 것
2. 유지 보수가 용이할 것
3. 현장에 있어서 신뢰성이 릴레이 제어반 보다 높을 것
4. Size는 릴레이 제어반 보다 작을 것
5. 중앙 제어장치로 데이터 전송이 가능할 것
6. 릴레이 및 무접점 방식이 가능할 것
7. 입력은 AC 115V를 받을 수 있을 것
8. 출력은 AC 115V 2A 이상으로 솔레노이드 밸브, 모타의 기동, 조작이 가능할 것
9. 제어 시스템의 확장은 기본 시스템의 작은 변경으로 가능할 것
10. 프로그램의 확장은 최저 4Kbyte Word까지 가능할 것

# 릴레이와 PLC

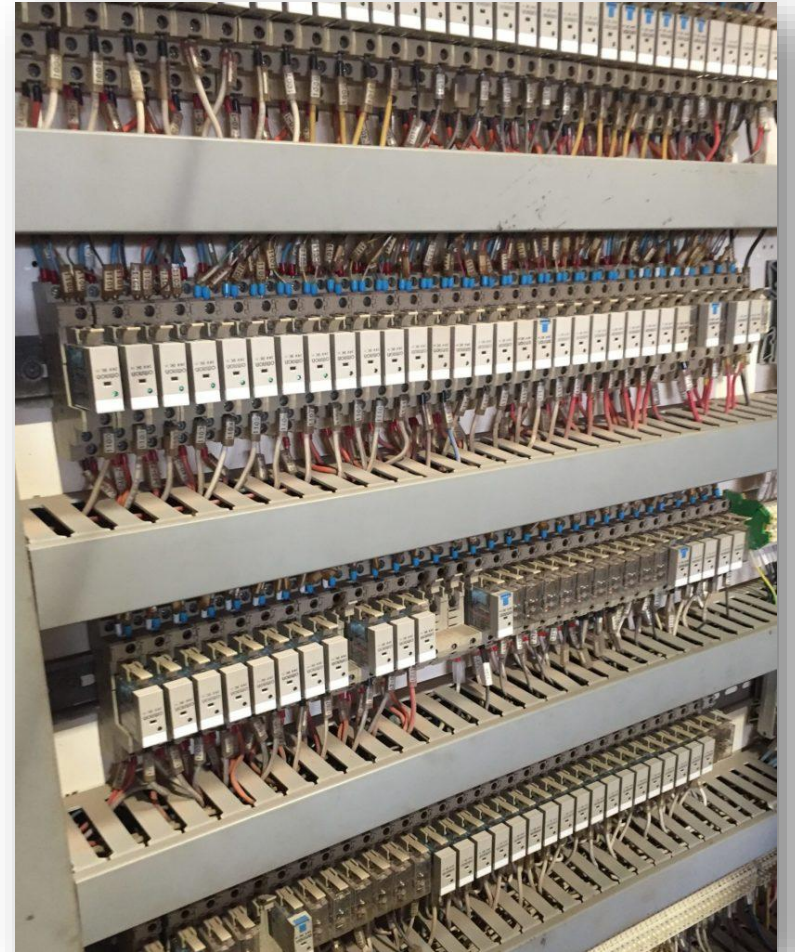
- 릴레이 : 전기 신호를 통해 다른 회로를 제어하는 스위치
- 릴레이 시퀀스
  - 릴레이를 이용하여 특정 순서로 기계나 장치를 제어하는 시스템
  - 릴레이의 ON/OFF 동작을 조합하여 제어 논리를 구성하고, 순차적으로 동작이 실행 되도록 설계





# 릴레이와 PLC

- 릴레이 시퀀스의 한계
  - 물리적인 배선으로 논리를 구현 → 공정이 변경되거나 새로운 제어 논리가 필요할 경우 기존 배선을 다시 설계하고 수정해야 함
  - 공정이 복잡해질수록 릴레이의 수가 급격히 증가 → 제어판 설계와 배선이 매우 복잡해짐.

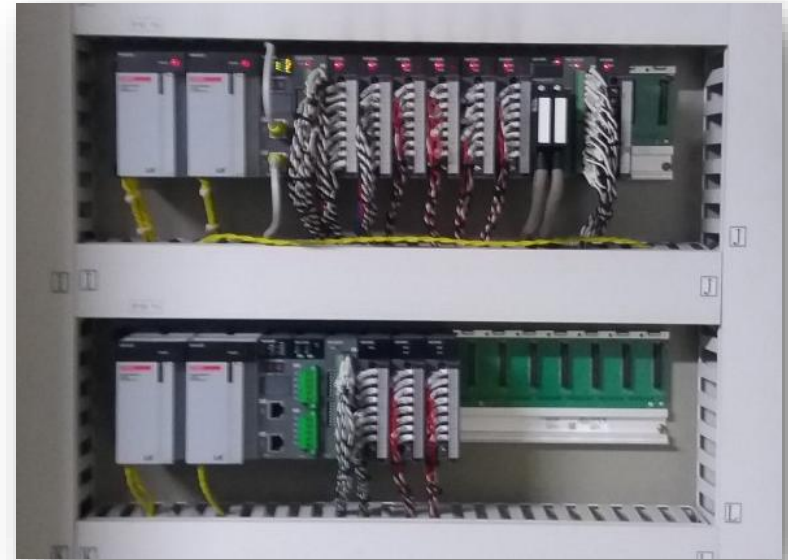


[https://dena.co.in/relay\\_logic\\_panels.php](https://dena.co.in/relay_logic_panels.php)

# 릴레이와 PLC

- PLC : Programmable Logic Controller  
= 논리 회로(≈릴레이 구성)

→ PLC는 릴레이를 프로그램 가능하도록 만든 것 → 릴레이를 대체



# 릴레이와 PLC

## ■ 하드 와이어드(Hard Wired Logic) – 릴레이만 사용

- PLC를 사용하지 않고 회로도를 제작 후 실제로 릴레이, 타이머, 카운터 등 전기 부품 및 전 기선을 연결하여 프로그램(기능들의 연속적인 작동)을 구성
- 제작이 어렵고 복잡하며, 수정이 어렵고, 문제 발생시 원인을 찾기가 어려움

## ■ 소프트 와이어드(Soft Wired Logic) – PLC 사용

- PLC를 사용하여 하드 와이어드의 기능을 컴퓨터 프로그램으로 제작하여 마이크로 컨트롤 러에서 프로그램을 구동

# 릴레이와 PLC

구분	릴레이 기반 제어 시스템	PLC (Programmable Logic Controller)
유연성	공정 변경 시 배선 수정 필요	프로그램 수정으로 공정 변경 가능
속도	릴레이 접점의 기계적 움직임으로 느림	전자식 신호 처리로 고속 제어 가능
신뢰성	접점 마모 및 유지보수 문제 발생	전자 부품 사용으로 높은 내구성과 안정성
설치 및 배선	복잡한 배선으로 설치와 수정이 어려움	간소화된 배선으로 설치 용이
확장성	릴레이 추가 필요, 확장 어려움	모듈 및 프로그램 확장으로 유연성 높음

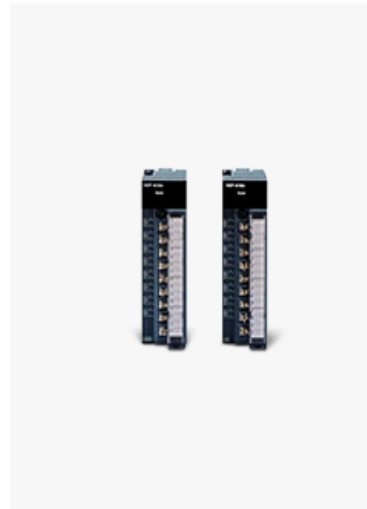
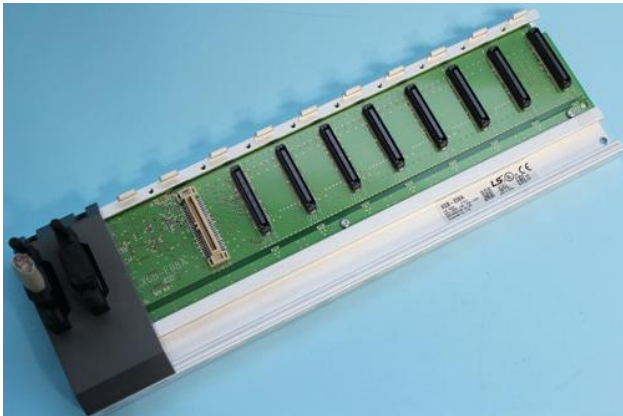


## 1-3. PLC의 종류



# 모듈형 PLC

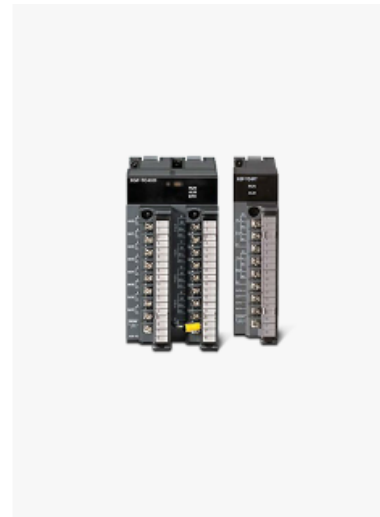
- 전원 입력, CPU, 신호 입력, 신호 출력, 통신 등 기능 단위로 모든 모듈이 따로 분리되어 있음
- 베이스 모듈에 필요한 모듈을 하나씩 장착하여 하나의 시스템을 구성



아날로그 입력



아날로그 출력



온도제어



모션제어

# 모듈형 PLC

- CPU에 따라 지원하는 프로그래밍 언어, 지원 용량, 처리 속도, 입/출력 점 (포트) 수가 달라짐

## 일반 범용 시퀀스 제어 CPU (IEC 언어지원)



### XGI-CPUS

- 프로그램 용량: 128KBytes
- 입출력 점수: 3,072점
- 입출력 디바이스 점수: 32,768점
- 처리속도: 28ns/step

## 고속 대용량 제어 CPU (LS전용 언어지원)



### XGK-CPUUN

- 프로그램 용량: 256Ksteps
- 입출력 점수: 6,144점
- 입출력 디바이스 점수: 65,536점
- 처리속도: 8.5ns/step

# 블록형 PLC

- 하나의 블록에 필수적인 기능들이 내장된 형태
- 모듈형에 비해 저렴함, 소형PLC라고도 함



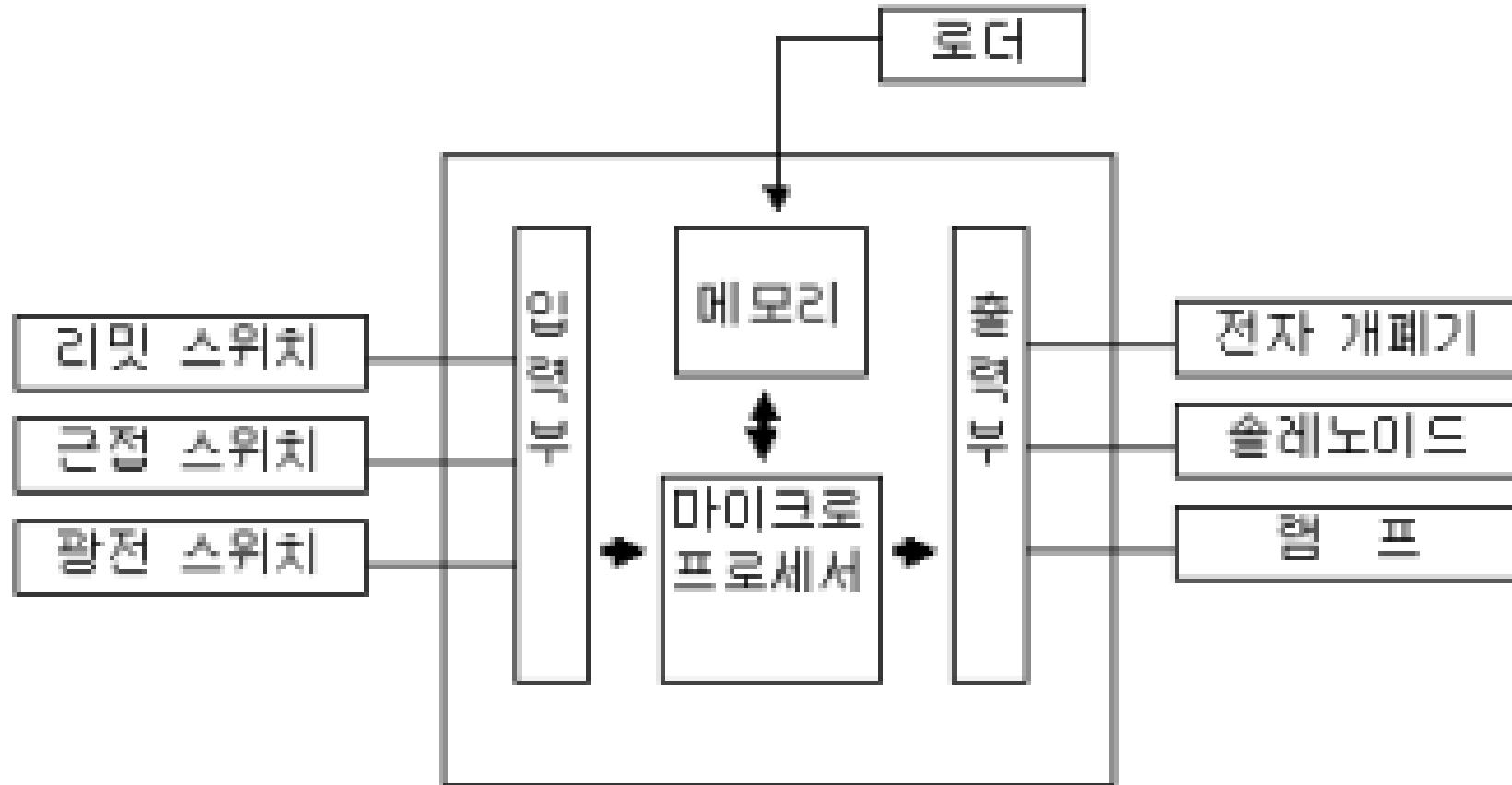
# 모듈형 vs 블록형

구분	모듈형 PLC (Modular PLC)	블록형 PLC (Block-Type PLC)
구조	CPU, 전원, I/O 모듈이 분리되어 개별 구성 가능	CPU, 전원, I/O가 하나의 단일 유닛으로 통합
확장성	모듈 추가로 확장 가능	확장이 어렵고, 입출력 고정
유연성	다양한 요구사항에 맞게 맞춤형 구성 가능	고정된 기능과 구성
적용 범위	대규모 시스템 및 복잡한 공정에 적합	소규모 자동화 시스템에 적합
설치 및 유지보수	설치 및 구성 복잡, 전문가 필요	설치 및 유지보수가 간단

## 1-4. PLC의 구조



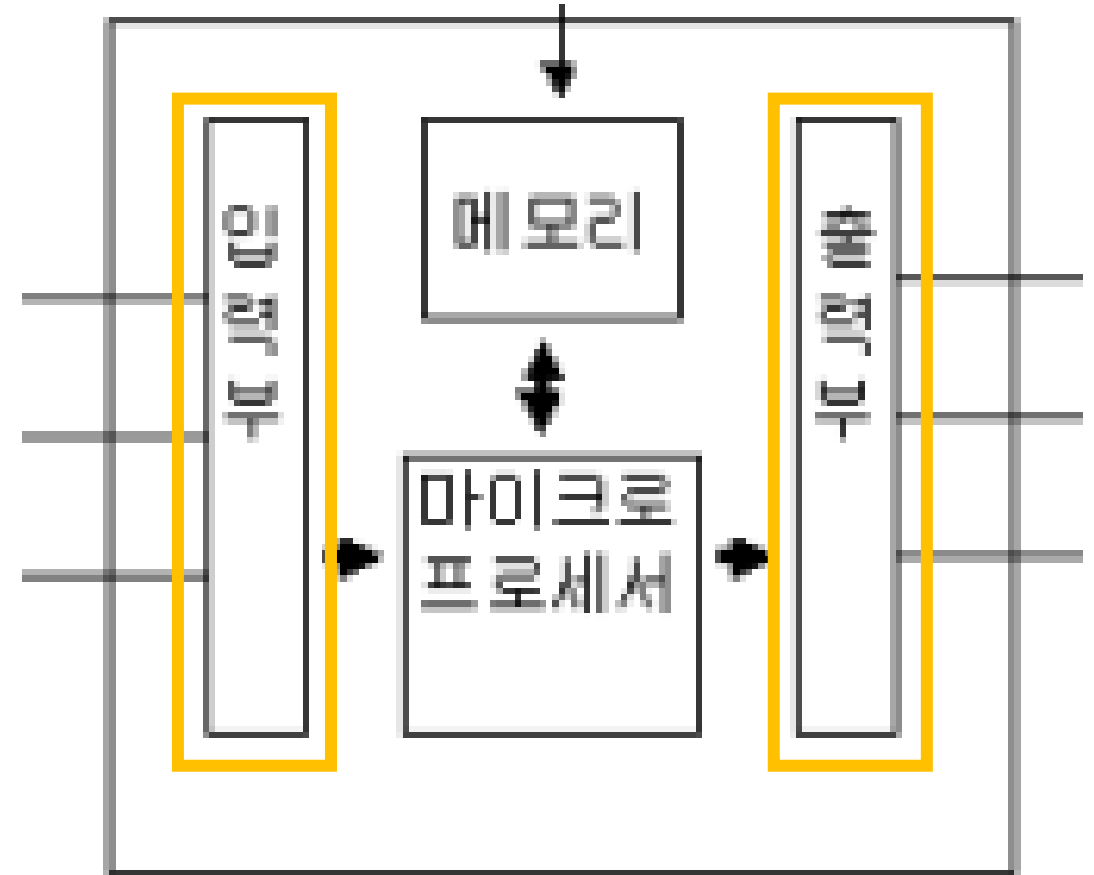
# PLC 구조



# PLC 구조

## 1. 입/출력부

- 외부 장치로부터 데이터를 받거나 외부 장치로 데이터를 보내기 위한 입/출력 메모리가 존재
- 물리적으로 전선을 통해 외부 장치와 연결되는 터미널(단자)이 존재
- 외부 장치와 PLC의 전압 차이가 존재하기에 외부 장치 연결 시 전압을 반드시 맞춰야 함





# PLC 구조: 입력부

- 외부 기기로부터 신호를 CPU의 연산부로 전달해주는 역할
  - 외부 입력 신호를 받아들이고, 이를 CPU가 처리할 수 있는 신호로 변환
- 입력 종류
  - DC 24v
  - AC 110v



# PLC 구조: 출력부

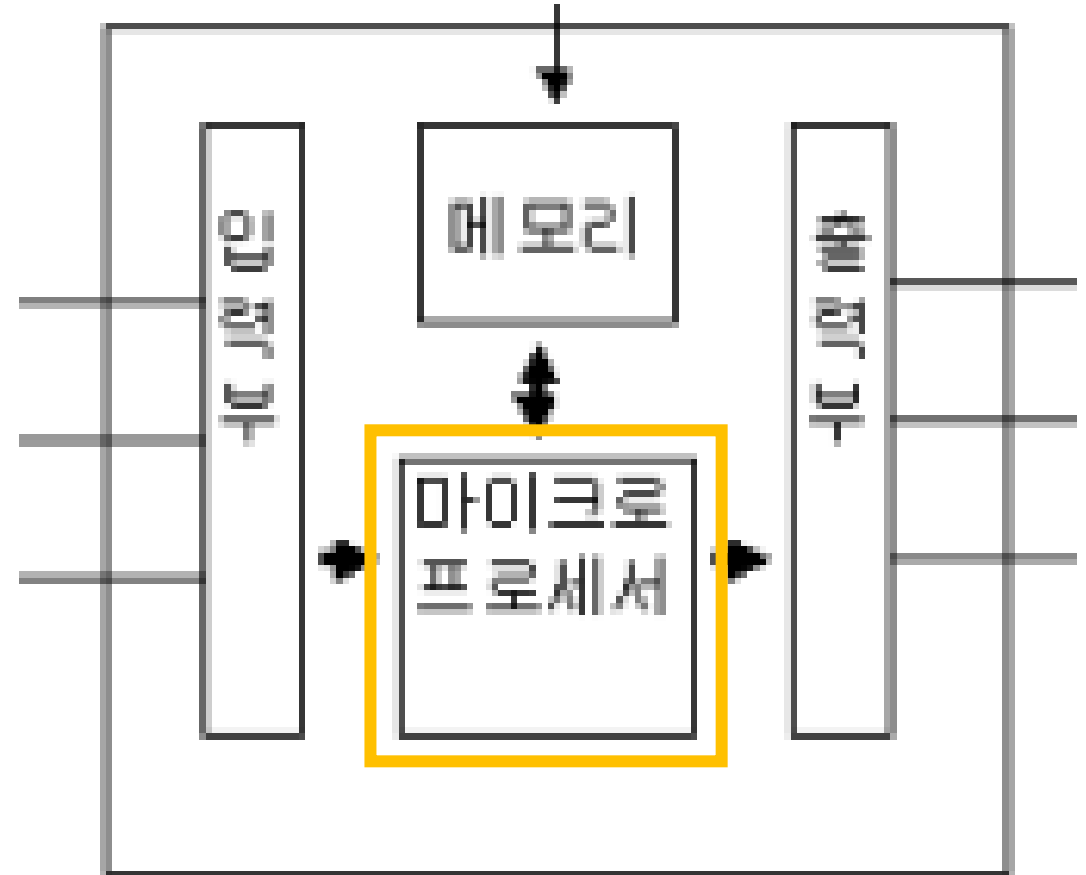
- 내부 연산의 결과를 외부의 기기에 전달하여 구동시키는 부분
- 출력 종류
  - 트랜지스터 출력
  - SSR 출력
  - 릴레이 출력



# PLC 구조: CPU

## 2. CPU(마이크로 프로세서)

- 메모리에 저장된 프로그램을 실행
- 메모리를 읽고 쓰기 가능
- 명령어에 따른 연산 수행
  - Step 이라고 부르는 명령어 처리 주기를 반복
  - 프로그램이 시작부터 끝까지 한 번 작동하는 것을 Scan이라고 함



# PLC 구조: 메모리

## 3. 메모리

### ■ 프로그램 메모리

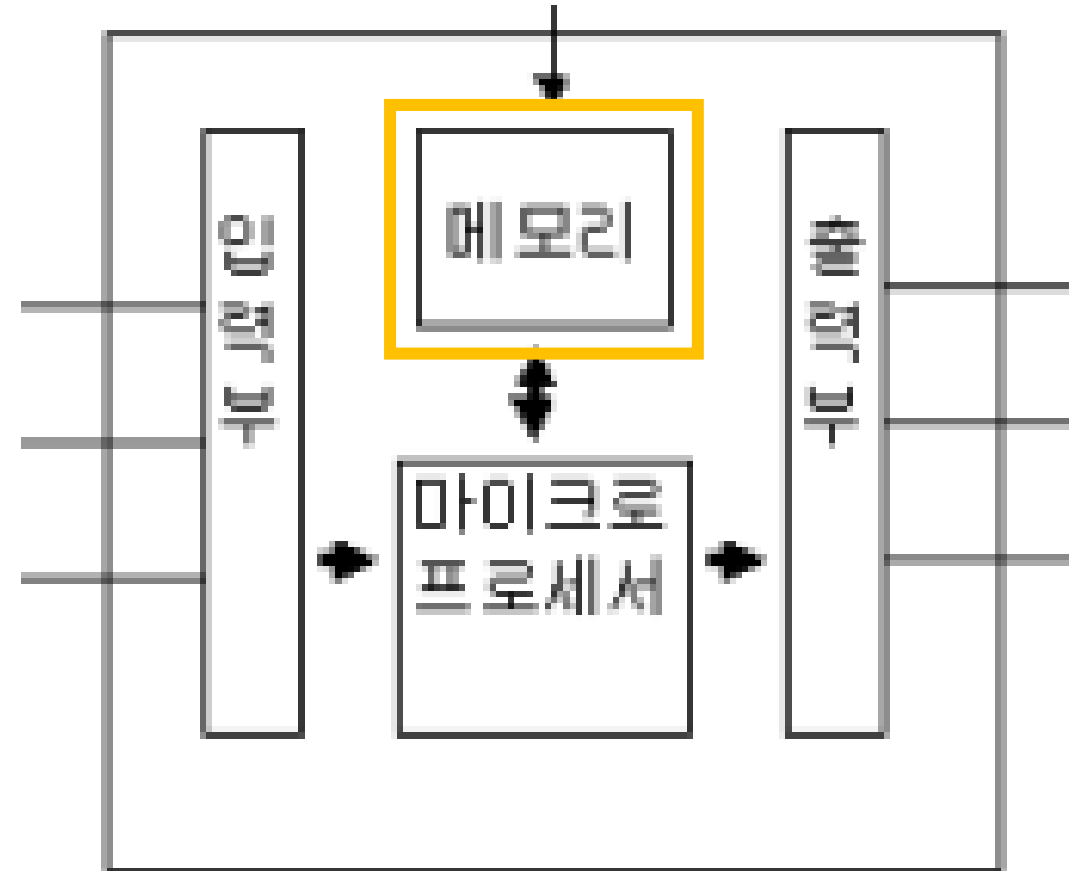
- 사용자가 작성한 프로그램을 저장

### ■ 데이터 메모리

- 프로그램에 따라 데이터를 저장 또는 변경

### ■ 시스템 메모리

- PLC 제조사에서 작성한 시스템 프로그램(운영체제)을 저장



# PLC 구조: 메모리

- ROM (read only memory)
    - 읽기 전용, 메모리 내용 변경 불가
    - 한 번 작성한 후 변경되지 않는, 시스템 관련 프로그램(펌웨어) 및 데이터 저장
    - 한 번 저장하면 전원이 끊어져도 메모리의 내용이 그대로 보존 (비휘발성)
-

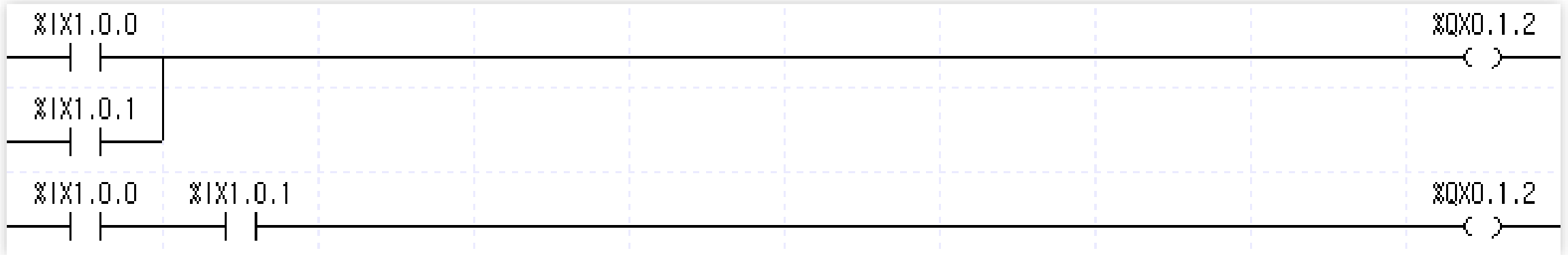
# PLC 구조: 메모리

- RAM (random access memory)
    - 메모리에 정보를 수시로 읽고 쓰기가 가능
    - 정보 일시저장 가능
    - 전원이 끊어지면 기억시킨 정보 내용 모두 상실(휘발성 메모리)
    - PLC의 데이터 영역은 RAM 영역에 저장
-

# 1-5. PLC의 동작 원리

# PLC 프로그래밍 언어

- IL, ST, Ladder, FBD, SFC 프로그래밍 방식을 사용
- 수업에서는 **Ladder(LD) 프로그래밍**을 사용
  - 대부분의 환경에서 LD 프로그래밍을 사용





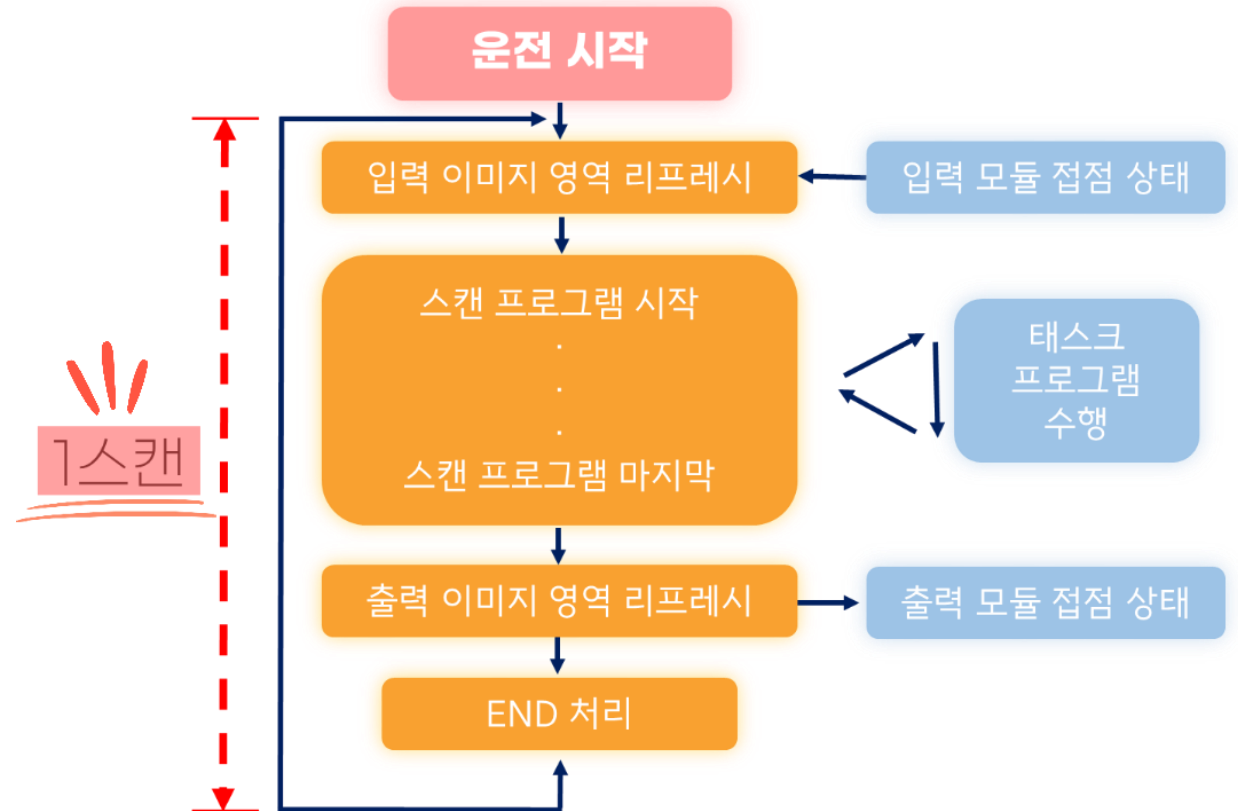
# PLC 동작원리

- LD 프로그래밍을 통해 입력 모듈의 입력 메모리, 출력 모듈의 출력 메모리(릴레이 코일), 내부 메모리를 제어할 수 있음
- 입력 모듈에 전기 신호(전압, 전류)가 들어오면 대부분의 모듈에서 자동으로 입력 메모리에 특정 값이 기록 됨
- 출력 메모리에 값을 쓰면 출력 코일에 전기가 흐르면서 릴레이가 가동, 출력 모듈에서 전기 신호가 출력 됨

# 연산 처리 과정

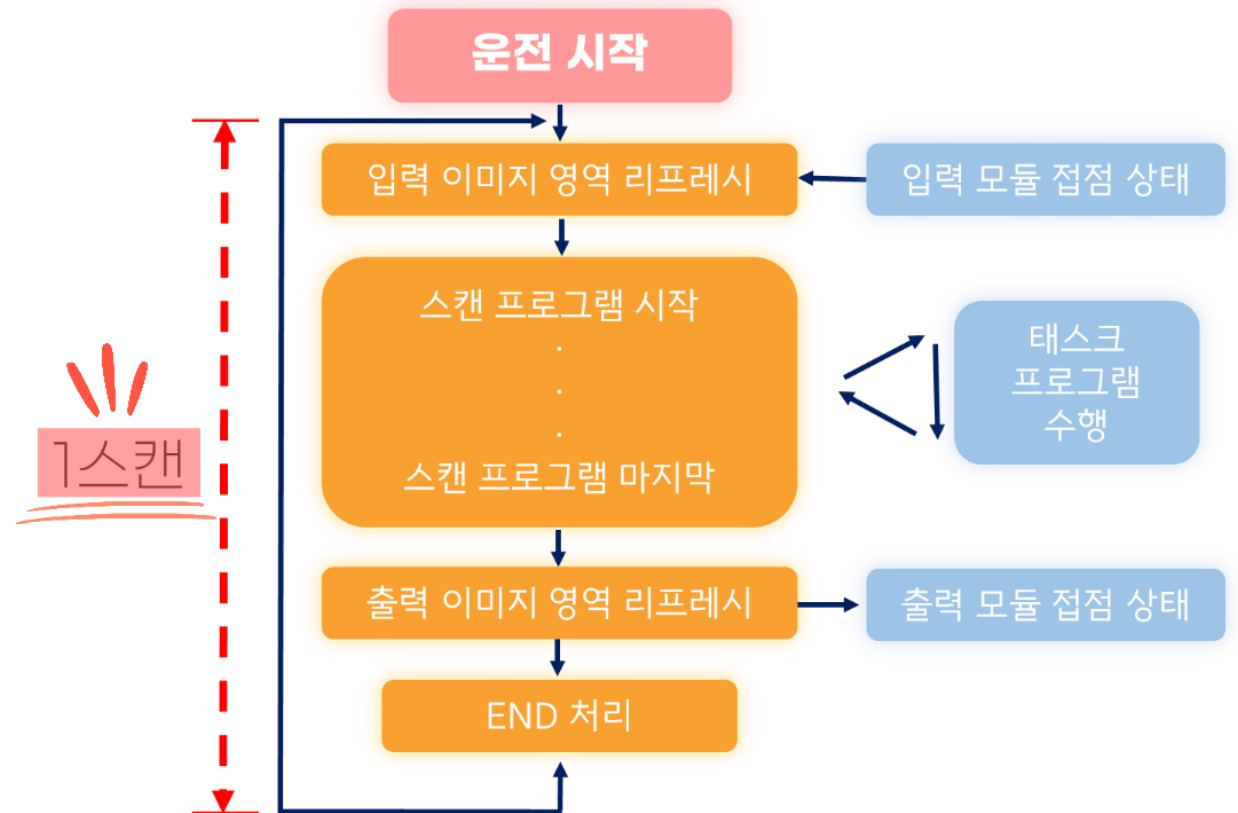
## ■ 스캔

- PLC가 입력 → 논리 연산 → 출력 단계를 순차적으로 반복하여 처리하는 작업 주기
- 작성한 프로그램이 전달하는 연산을 처음부터 끝까지 한 번 수행하는 것을 1스캔이라고 함  
(1스캔타임 = 1연산주기)



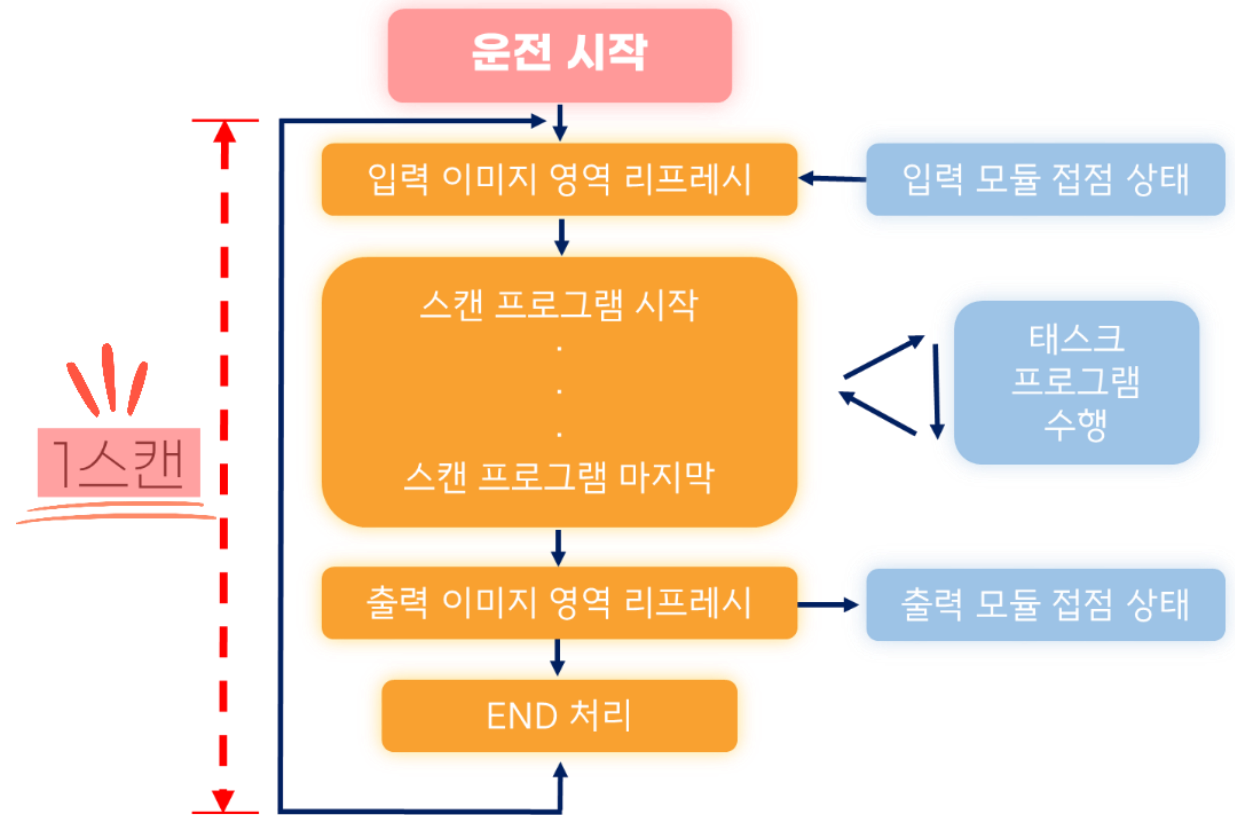
# 연산처리 과정

1. 입력 이미지 영역 리프레시
2. 스캔프로그램/태스크 프로그램
3. 출력이미지 영역 리프레시
4. END처리



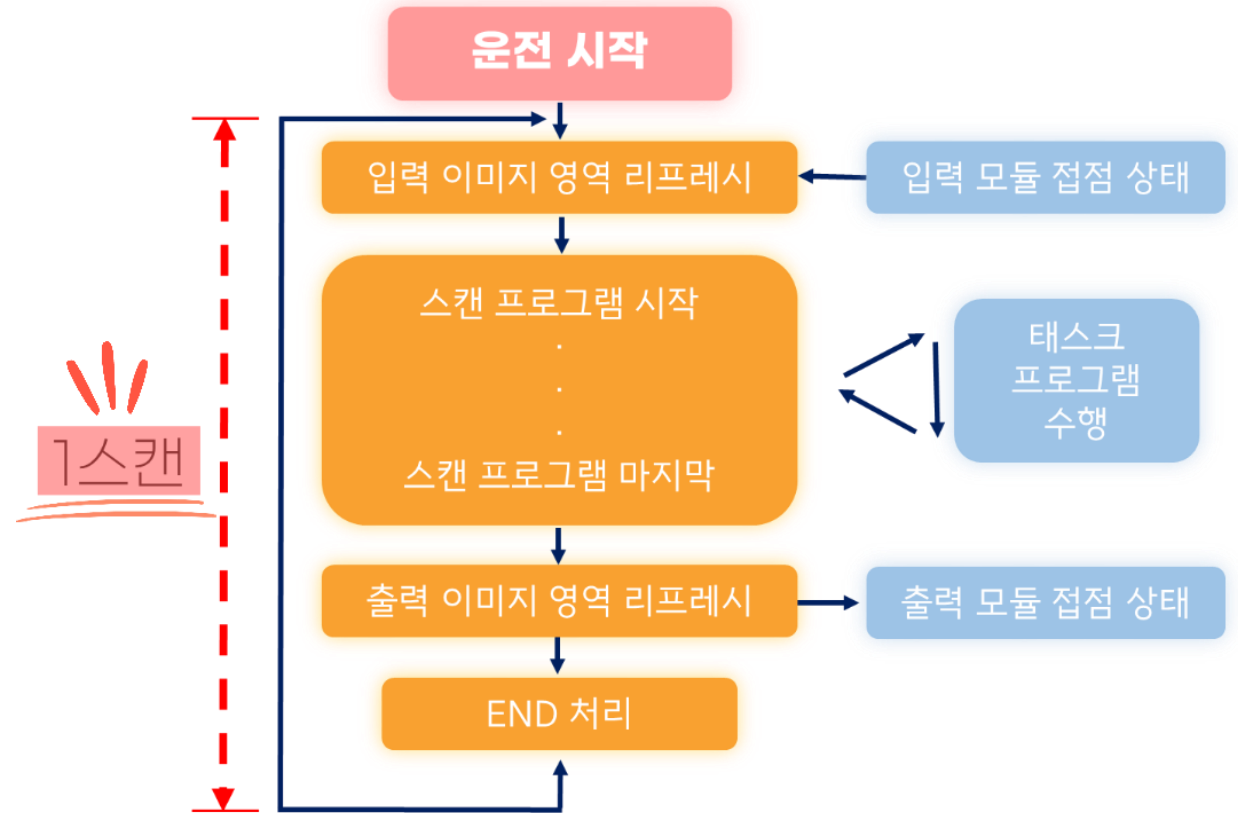
# 1) 입력 이미지 영역

- 입력 모듈로부터 읽어들이는 현재 입력 신호를 저장하는 메모리 공간.
- 입력 리프레시 : 입력 모듈로부터 입력되는 정보를 입력받아 입력 이미지 영역으로 복사  
→ 프로그램의 연산시 사용됨



## 2) Scan/Task 프로그램

- 이전 과정에서 저장한 정보를 바탕으로, 사전에 만들어 둔 프로그램 연산 수행
- 프로그램 연산 결과 → 메모리



## 2) Scan/Task 프로그램

- Scan Program

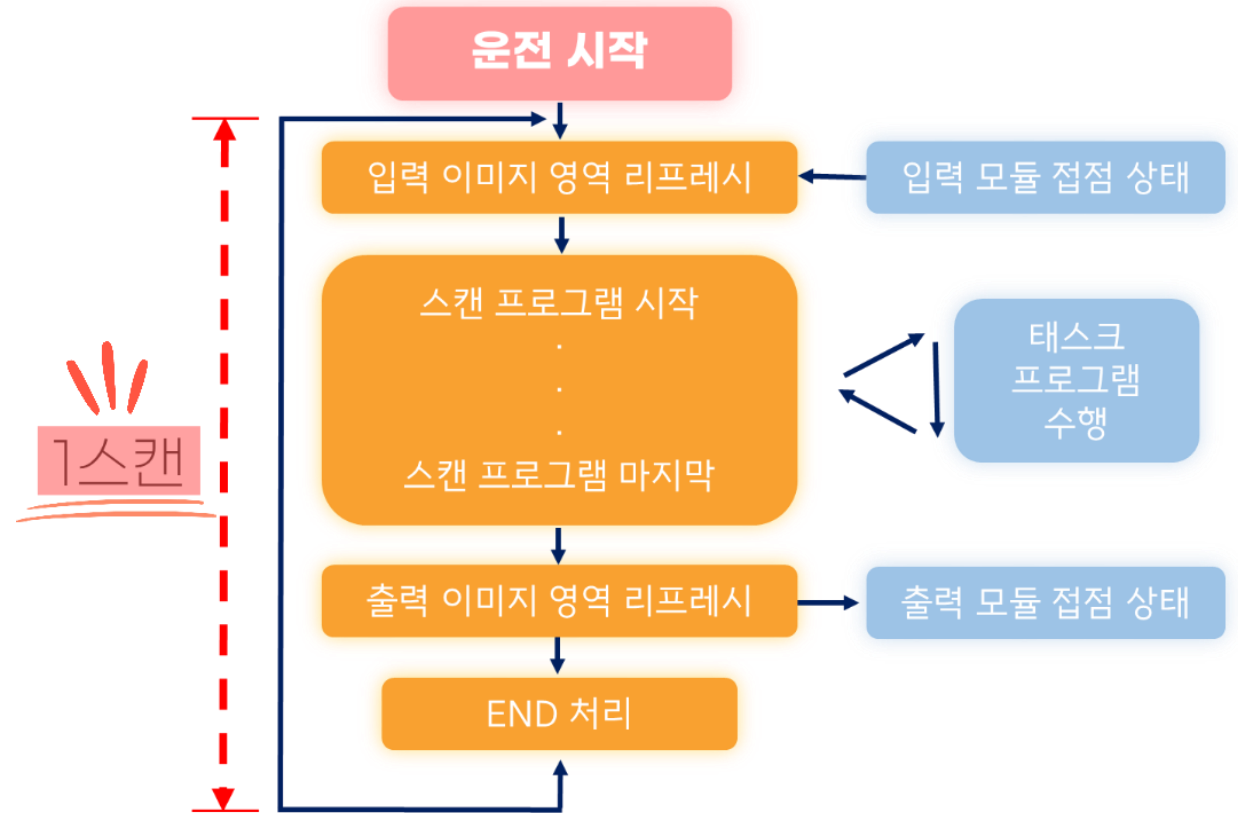
- PLC 의 CPU가 런 상태면 무조건 수행하는 프로그램

- Task Program

- 스캔 프로그램과 달리 실행 조건이 만족되면 실행하는 프로그램
    - Task Program 실행 시, Scan Program의 연산이 멈추고 실행
-

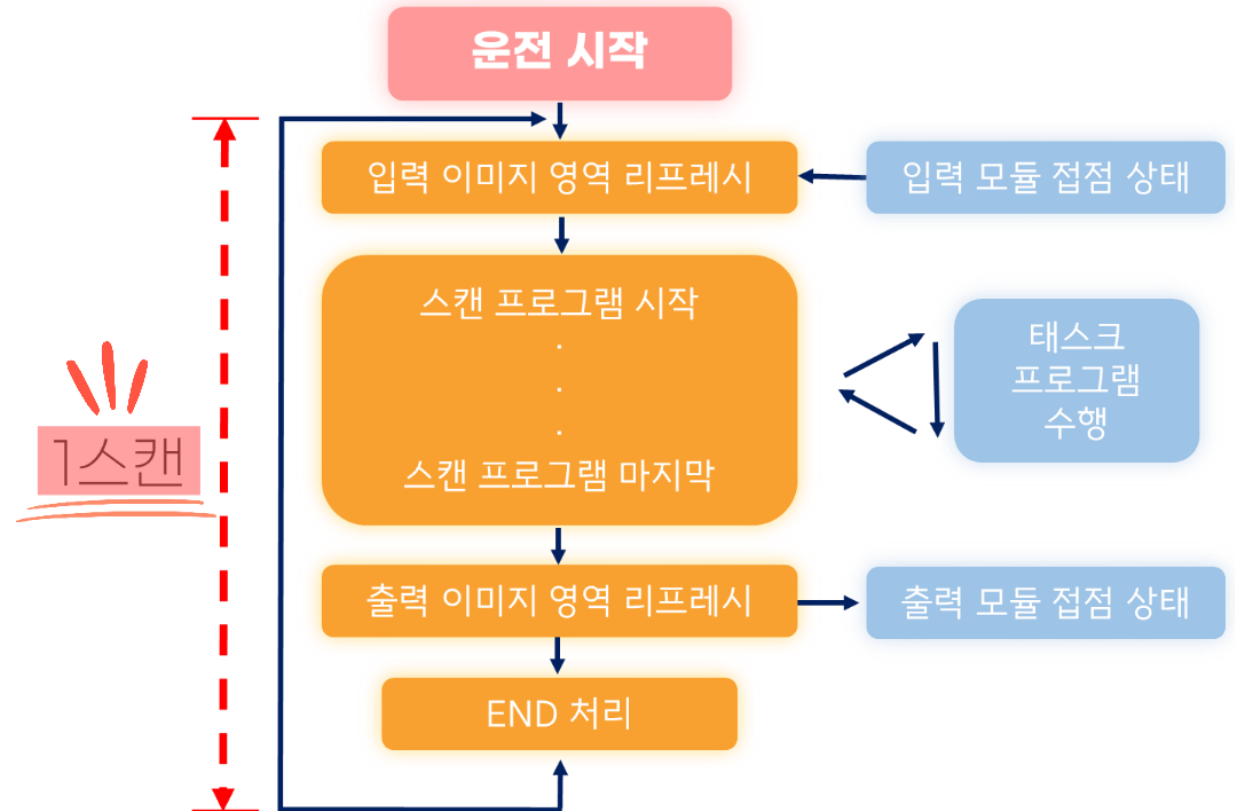
### 3) 출력 이미지 영역

- 사용자 프로그램 실행 결과를 저장하는 메모리 공간
  - 연산 과정에서 만들어진 결과는 바로 출력으로 보내지지 않고 출력 이미지 영역에 저장됨
- 출력 모듈을 통해 실제 장치를 제어하는 데 사용



## 4) 출력 이미지 영역: 자기 진단

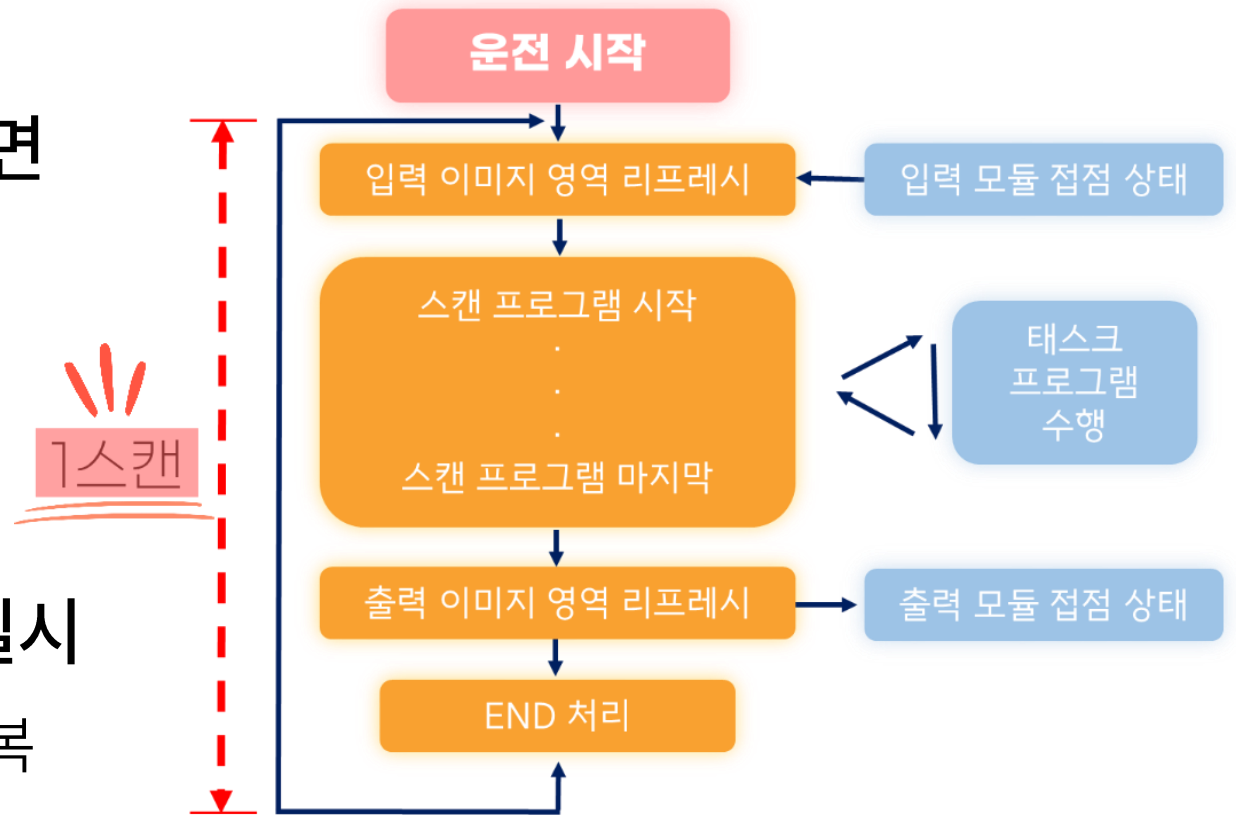
- 출력 이미지 영역에 저장하는 이유
  - 마지막 스텝 연산이 끝나고 나서 PLC 시스템 상에 오류가 있는지를 검사하고 오류가 없을 때만 출력을 내보냄
- 출력 이미지 영역에 저장 후 PLC의 CPU는 자기 시스템을 진단하여 시스템 상에 오류가 있다면 **에러 메시지** 발생





## 5) 출력 이미지 영역: END 처리

- 연산이 성공적으로 출력되고,  
자기 진단 결과 시스템에 오류가 없다면
- 이미지 영역에 저장된 데이터를  
출력 영역으로 복사,  
→ 실질적인 출력!
- END 처리 이후 다시 입력 리프레시 실시
  - 1 스캔 이후 다시 그 스캔을 반복하는 반복  
연산 수행



감사합니다

---