

기술

설비 제어 로직을 소프트웨어로 실행해 보면, 처음 틀리는 곳은 3군데입니다

방동걸 (주)웨이스 대표이사

2026년 9월 18일

공장 설비를 실제로 움직이는 것은 PLC라고 부르는 제어기입니다. 컨베이어를 언제 세우고 실린더를 언제 내릴지는 이 제어기 안의 프로그램이 정하고, 그 프로그램은 IEC 61131-3이라는 국제 표준이 정한 언어로 씁니다. 그중 현장에서 가장 많이 쓰이는 것이 전기 회로도처럼 생긴 래더 다이어그램입니다. 최근 이 프로그램을 제어기 밖에서, 그러니까 보통의 소프트웨어로 실행해 보려는 시도가 늘었습니다. 설비를 세우지 않고 프로그램을 시험하려는 이유도 있고, AI가 제어에 관여하기 시작하면서 명령이 실제로 나가기 전에 결과를 미리 보려는 이유도 있습니다. 제조 AI 운영체제 VEXPLOR를 만드는 웨이스도 이 실행 의미를 다루면서, 소프트웨어 쪽 구현이 제어기와 갈리는 자리가 늘 같은 3군데라는 것을 확인했습니다.

3군데는 전부 「언제」의 문제입니다. 논리식 자체는 어렵지 않습니다. 접점이 직렬이면 AND, 병렬이면 OR이고, 코일은 그 결과를 받습니다. 그런데 제어기는 이 식을 우리가 짐작하는 시점에 계산하지 않습니다. 이 글은 그 시점을 3군데로 나눠 적고, 마지막에 자기 코드가 제어기와 같은 결과를 내는지 직접 확인해 보는 5가지를 붙입니다.

01

PLC는 프로그램을 한 번 훑고 다시 처음으로 돌아옵니다

일반적인 프로그램은 이벤트가 오면 그때 함수가 불립니다. 버튼이 눌리면 버튼 처리 함수가 실행되는 식입니다. 제어기는 그렇게 동작하지 않습니다. 전원이 들어와 있는 동안 아래 세 단계를 쉬지 않고 되풀이합니다.

단계	하는 일	걸리는 시간
입력 읽기	센서 상태를 한꺼번에 복사	짧다
프로그램 계산	첫 줄부터 끝 줄까지	프로그램 길이에 따라
출력 쓰기	계산 결과를 한꺼번에 내보냄	짧다

이 한 바퀴를 스캔이라고 부르고, 한 바퀴에 걸리는 시간을 스캔 주기라고 합니다. 현장 설비에서는 보통 몇 밀리초에서 수십 밀리초 사이입니다. 여기서 중요한 것은 읽기와 쓰기가 계산과 분리돼 한 번씩만 일어난다는 점입니다. 프로그램이 100줄이든 1만 줄이든, 그 줄들이 보는 입력은 전부 이번 바퀴가 시작될 때 찍힌 같은 사진입니다.

이 사진을 입출력 이미지라고 부릅니다. 이름이 거창하지만 실제로는 센서 값을 복사해 둔 메모리입니다. 제어기가 이렇게 만들어진 이유는 분명합니다. 계산 도중에 입력이 바뀌면 같은 스캔 안에서 앞줄과 뒷줄이 서로 다른 세상을 보게 되고, 그러면 결과를 사람이 예측할 수 없습니다.

02

첫 번째로 틀리는 곳은 입력을 읽는 시점입니다

소프트웨어로 옮길 때 가장 먼저 어긋나는 것이 이 지점입니다. 보통 이렇게 짭니다. 접점을 계산할 때가 되면 그 순간의 센서 값을 읽어 옵니다. 읽어 오는 함수 하나를 만들어 두고 필요할 때마다 부르는 방식인데, 이렇게 하면 프로그램 앞부분과 뒷부분이 서로 다른 시점의 값을 보게 됩니다.

평소에는 차이가 드러나지 않습니다. 센서가 초당 몇 번 바뀌지 않으니 한 스캔 안에서 값이 달라질 일이 드물기 때문입니다. 문제는 값이 빠르게 바뀌는 순간에만 어긋난다는 점입니다. 광센서를 제품이 지나가는 순간, 비상정지 버튼이 눌리는 순간처럼 **가장 중요한 순간에만** 결과가 갈립니다. 그리고 그때는 이미 재현이 어렵습니다.

같은 이유가 출력에도 적용됩니다. 계산 도중에 코일이 켜졌다가 뒷줄에서 다시 꺼지는 프로그램이 있다면, 실제 제어기는 그 중간 상태를 밖으로 내보내지 않습니다. 한 바퀴가 끝난 뒤 최종 결과만 나갑니다. 계산할 때마다 바로 출력을 쓰는 코드는 실제 설비에서는 일어나지 않는 깜빡임을 만들어 냅니다.

고쳐야 할 곳은 한 군데입니다. 스캔이 시작될 때 입력 전체를 한 번 복사해 두고, 그 바퀴의 모든 계산은 복사본만 봅니다. 출력도 따로 모아 두었다가 바퀴 끝에서 한 번에 내보냅니다. 코드는 오히려 단순해집니다.

03

두 번째는 상승 검출을 이벤트로 만들면 어긋납니다

래더에는 상승 검출과 하강 검출이 있습니다. 신호가 꺼짐에서 켜짐으로 바뀐 그 순간에만 한 번 통과시키는 점점입니다. 버튼을 눌러 계수기를 1 올릴 때, 누르고 있는 동안 계속 올라가면 안 되므로 반드시 필요합니다.

이것을 소프트웨어로 옮길 때 흔히 이벤트로 만듭니다. 값이 바뀌면 콜백을 부르거나 큐에 넣는 방식입니다. 동작하는 것처럼 보이지만 제어기와는 다릅니다. 제어기의 상승 검출은 이벤트가 아니라 비교입니다. **이번 스캔의 값과 지난 스캔의 값을 비교해, 지난번이 꺼짐이고 이번이 켜짐이면 통과시킵니다.** 그래서 스캔과 스캔 사이에 신호가 켜졌다가 꺼지면 제어기는 그것을 아예 보지 못합니다.

차이가 드러나는 자리는 3곳입니다.

- **짧은 펄스:** 스캔 주기보다 짧게 왔다 간 신호를 제어기는 놓칩니다. 이벤트로 짰 코드는 그것을 잡아내고, 그래서 실제 설비보다 계수가 더 올라갑니다.
- **같은 스캔 안의 재사용:** 같은 상승 검출을 여러 줄에서 쓰면 제어기에서는 그 바퀴 내내 같은 값입니다. 이벤트로 짜면 먼저 소비한 줄만 통과합니다.
- **첫 스캔:** 지난 값이 없는 첫 바퀴에서 무엇을 지난 값으로 볼지 정해야 합니다. 여기를 정하지 않으면 전원을 넣는 순간 켜져 있던 신호가 상승으로 읽혀 설비가 한 번 움직입니다.

세 번째 항목은 실제로 위험한 자리입니다. 정비를 마치고 전원을 다시 넣는 순간에만 나타나기 때문에 시험 중에는 거의 보이지 않습니다.

04

세 번째는 타이머를 시계로 착각하는 것입니다

래더 타이머는 「3초 뒤에 켜짐」처럼 쓰지만, 제어기 안에서 3초는 시계가 아니라 스캔으로 셉니다. 스캔이 돌 때마다 지난 바퀴부터 흐른 시간을 누적하고, 그 누적값이 설정값을 넘으면 켜집니다. 그래서 타이머가 켜지는 시점은 **설정값 직후의 스캔 경계**이지 설정값 그 자체가 아닙니다.

설정	스캔 주기	실제로 켜지는 때
3,000 ms	10 ms	3,000~3,010 ms
3,000 ms	50 ms	3,000~3,050 ms
100 ms	50 ms	100~150 ms

위 두 줄은 오차가 1% 아래라 대개 문제가 되지 않습니다. 세 번째 줄이 문제입니다. 설정값이 스캔 주기의 두 배밖에 되지 않으면 오차가 최대 50%가 됩니다. 짧은 타이머를 쓰는 인터록에서 현장과 시험 결과가 안 맞는 원인이 대개 여기 있습니다.

소프트웨어로 옮길 때 흔한 실수는 실제 시각을 그대로 쓰는 것입니다. 시스템 시계를 읽어 경과를 재면 코드는 간단해지지만, 같은 프로그램을 같은 입력으로 다시 실행해도 결과가 매번 달라집니다. 빠른 컴퓨터에서는 통과하고 느린 컴퓨터에서는 실패하는 시험이 만들어집니다. 시험이 그때그때 다른 답을 내면 그 시험은 근거로 쓸 수 없습니다.

타이머는 시계가 아니라 스캔으로 셉니다



설정값은 스캔 경계와 무관한 자리에서 끝나지만, 타이머는 그다음 스캔 경계에서 켜집니다. 그 사이가 오차입니다

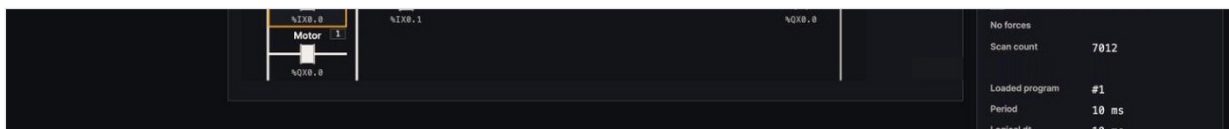
05

같은 입력에 같은 결과가 나와야, 그 실행을 근거로 쓸 수 있습니다

3가지를 한 줄로 묶으면 이렇습니다. 제어기의 시간은 초가 아니라 스캔으로 흐릅니다. 입력도 출력도 상승 검출도 타이머도 전부 스캔 경계에 맞춰 움직입니다. 그래서 소프트웨어로 옮길 때 지켜야 할 것은 논리식이 아니라 **바퀴의 구조**입니다.

이것을 지키면 따라오는 성질이 하나 있습니다. 시작 상태와 입력 순서가 같으면 몇 번을 실행해도 같은 결과가 나옵니다. 실행할 때마다 답이 달라지지 않는다는 것은 개발 편의의 문제가 아니라, 그 실행 결과를 **판단의 근거로 쓸 수 있느냐**의 문제입니다. 실행해 본 결과로 「이 명령은 안전하다」를 말하려면 그 실행이 재현돼야 합니다. 재현되지 않는 실행은 그냥 한 번 일어난 일입니다.

시계를 쓰지 않고 스캔 수로 시간을 세면 이 성질이 생깁니다. 난수를 쓰지 않고, 처리 순서를 스레드에 맡기지 않고, 첫 스캔의 초기값을 명시하면 나머지도 따라옵니다.



제어기 밖에서 실행 중인 래더 한 줄입니다. 실제 주기(Period)와 논리 시간(Logical dt)을 따로 들고 있어, 시간을 시계가 아니라 스캔으로 셉니다

06

자기 구현이 제어기와 같은지 직접 확인하는 5가지

읽는 것만으로는 자기 코드가 어느 쪽인지 알기 어렵습니다. 아래 5가지는 시험 프로그램 한 개로 확인할 수 있습니다.

- **하나.** 한 스캔 안에서 입력을 두 번 읽는 줄을 만들고, 그 사이에 입력을 바꿔 봅니다. 두 줄의 결과가 다르면 입출력 이미지가 없는 것입니다.
- **둘.** 코일을 켜다가 같은 스캔 뒷줄에서 끄는 프로그램을 실행하고, 출력 단자를 관찰합니다. 깜빡임이 보이면 출력을 즉시 쓰고 있는 것입니다.
- **셋.** 스캔 주기보다 짧은 펄스를 입력에 넣습니다. 계수가 올라가면 상승 검출이 이벤트로 구현된 것입니다.
- **넷.** 켜져 있는 입력을 둔 채로 실행을 다시 시작합니다. 첫 바퀴에서 상승으로 읽혀 설비가 움직이면 초기값이 정해지지 않은 것입니다.
- **다섯.** 같은 시험을 스캔 주기만 바꿔 두 번 실행합니다. 짧은 타이머가 든 프로그램의 결과가 달라지면, 그 차이가 실제 제어기에서도 그대로 일어납니다.

5가지 모두 통과하면 논리식이 아니라 시간 구조가 맞은 것이고, 그때부터 그 실행 결과를 다른 판단의 재료로 쓸 수 있습니다. 반대로 하나라도 걸리면 그 차이는 평소에 숨어 있다가 가장 빠르게 움직이는 순간에만 드러납니다.

WACE(웨이스)는 공장이 스스로 판단하고 멈추지 않게 하는 제조 AI 운영체제 VEXPLOR를 만드는 회사다.



원문 blog.vexplor.com/post/what-breaks-first-when-you-simulate-plc-logic

발행 2026년 9월 18일 · 방동걸 (주)웨이스 대표이사



원문



홈페이지



링크드인

회사 주식회사 웨이스 (WACE Inc.) · 대표이사 방동걸 ·
사업자등록번호 439-87-02852

주소 (18462) 경기 화성시 동탄구 동탄영천로 150, 에이동(A동)
1702·1706호 (영천동, 실리콘밸리동탄)

연락처 전화 070-4730-0629 · contact@wace.me

홈페이지 wace.me · linkedin.com/company/wace-inc