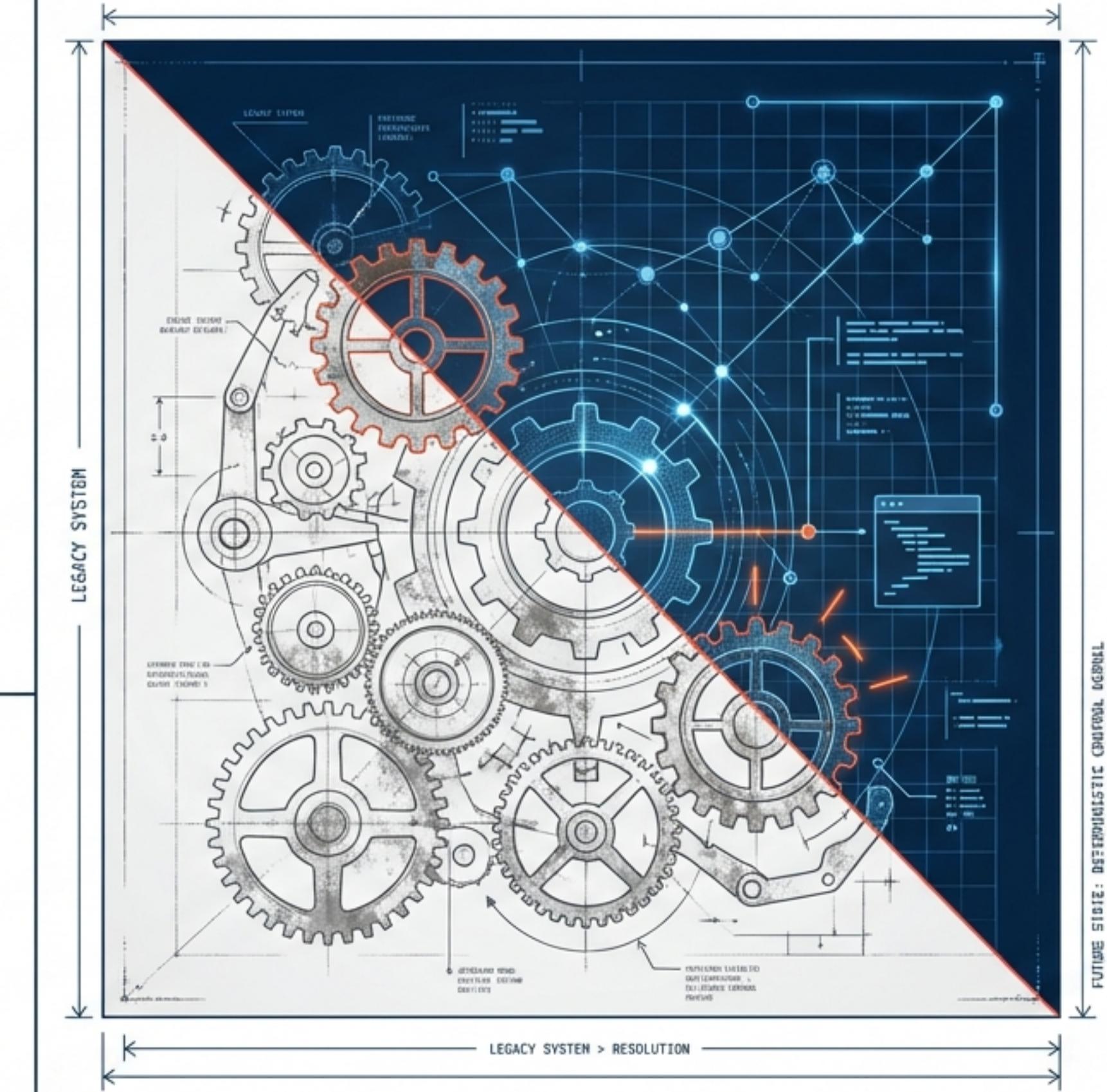


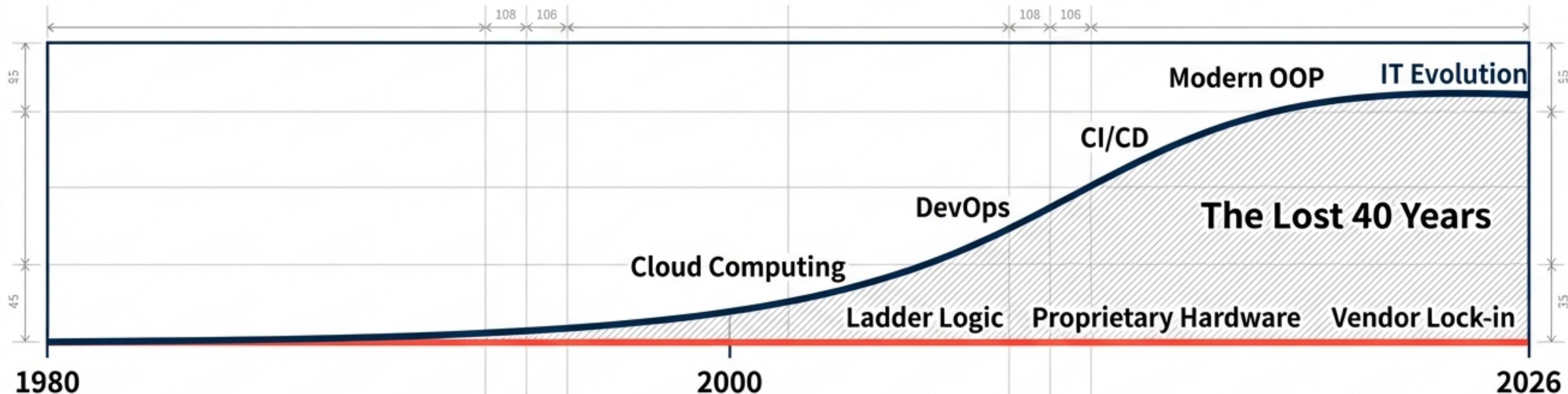
# Closing the 40-Year Gap

制御システムは「40年遅れた  
ソフトウェア技術」ではないことを  
証明する

- 汎用プロセッサ × C++20 ×  
真のオブジェクト指向
- 高級言語技術者の理想を実装した、  
次世代の決定論的制御カーネル



# なぜ、あなたの工場の制御技術は「1980年代」で止まっているのか？



## ハードウェアの呪縛

言語仕様や接続機器の縛りにより、汎用CPUの処理能力の1%も引き出せていない。

## 技術の断絶

コピペ量産とスパゲッティコードが常態化し、ソフトウェア工学の恩恵を受けられない。

## エンジニアの葛藤

高級言語を知る技術者ほど、既存PLCの非効率な実装に絶望している。

「信頼性」を人質に取られたこの停滞を、**技術**で打破する。

# 「PCは信頼できない」という神話の崩壊

Myth	Reality
PCはOSのジッタ（揺らぎ）で制御が不安定になる。	適切なカーネル設計とLinux（Yocto）活用により、マイクロ秒オーダーの定周期性を確保可能。
専用ハードウェアPLCしか高速動作できない。	汎用CPUの演算能力はPLC用ASICを数百倍凌駕している。

## Case Study

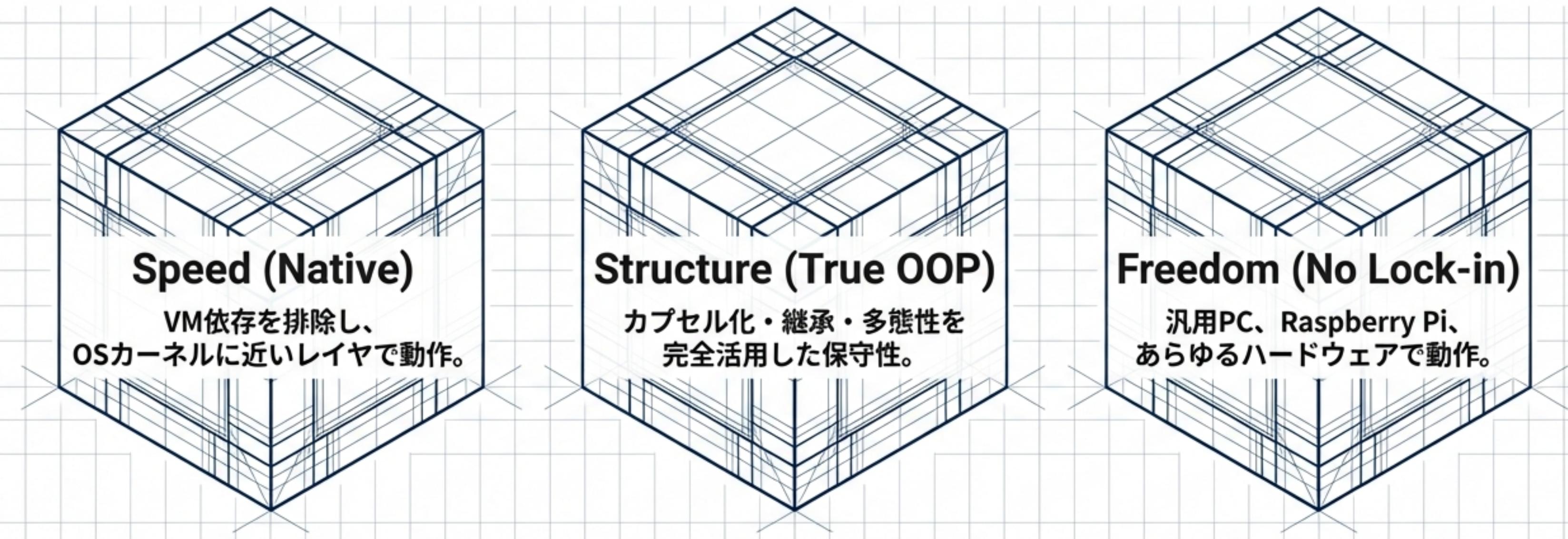
### Inconvenient Truth: Siemens S7-1500 Software Controller

ソフトウェア版にも関わらず、ハードウェア販売を守るために「処理速度」「メモリ」「タグ数」に人工的な制限（キャップ）が設けられている。

遅いのはPCの性能限界ではなく、メーカーの都合である。

# OCA (Object-Component Automation) とは何か？

既存のソフトPLCのとは一線を画す、C++20ネイティブの制御カーネル。  
PLCの堅牢性を、現代のソフトウェア工学で正しく再構築したプラットフォーム。

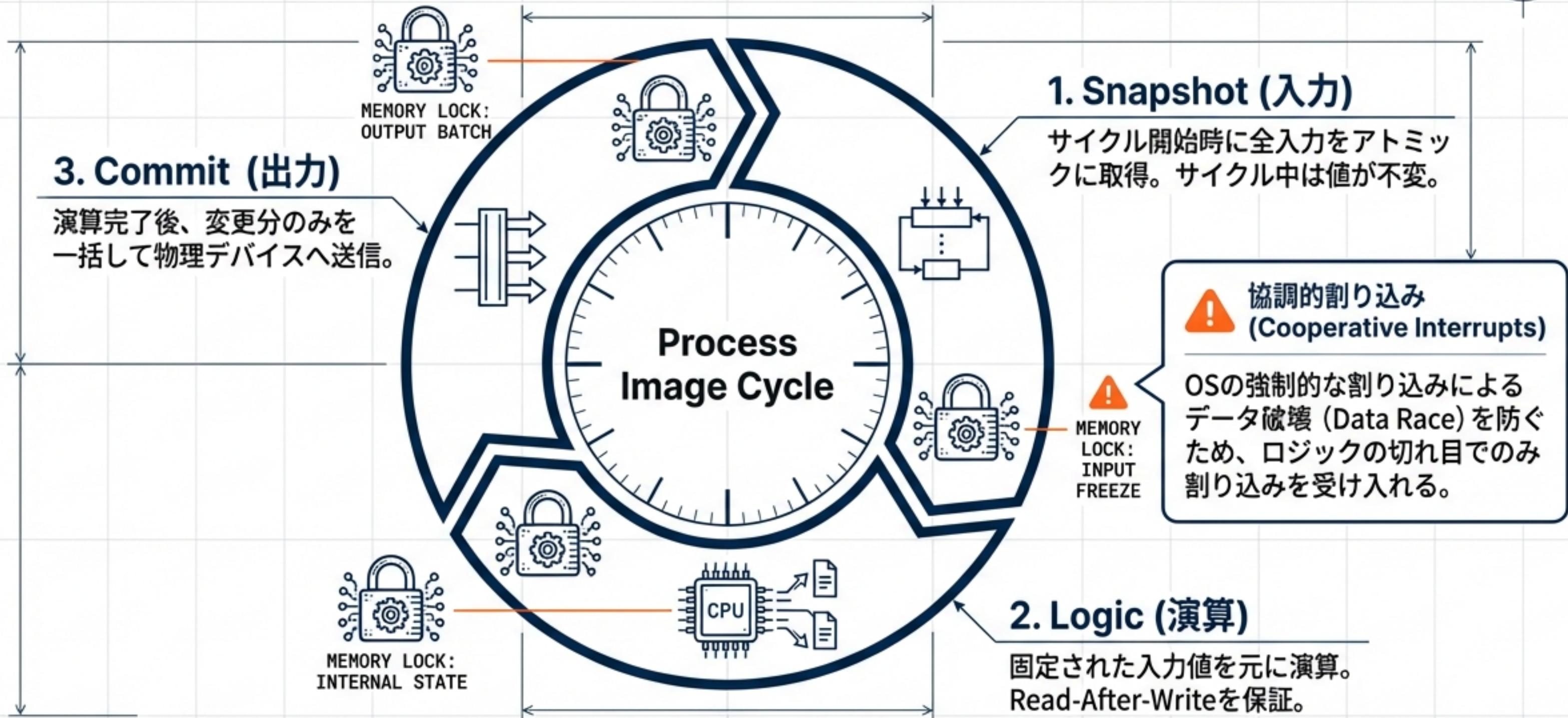


**CODESYS / SoftPLC**  
= 中間言語ランタイム (VM)

VS

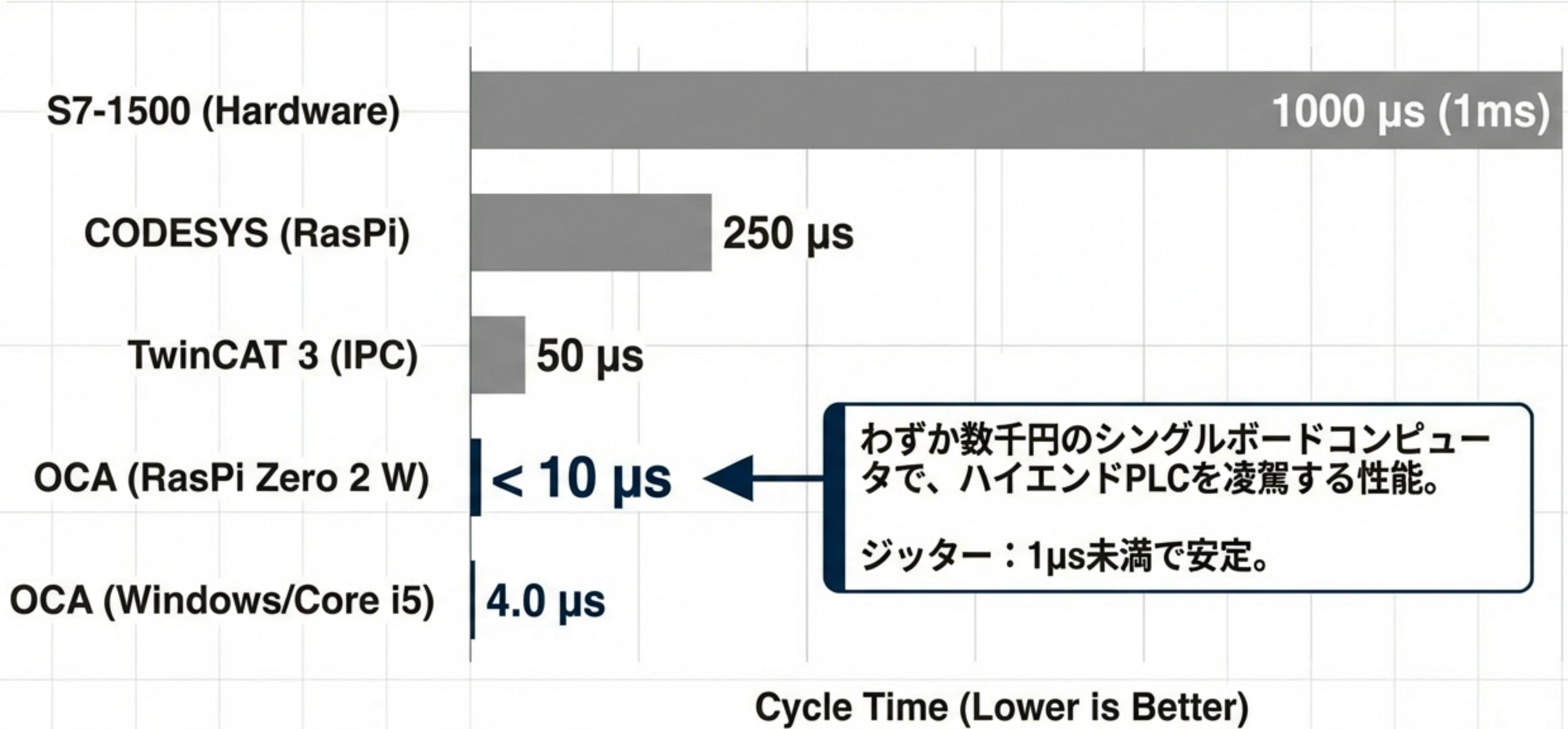
**OCA = C++20 ネイティブバイナリ  
(最適化された機械語)**

# 「決定論的動作 (Determinism)」の構造的強制



スキャン実行中にデータが変動する問題を構造的に排除。

# 汎用プロセッサの真価を解放する：圧倒的なベンチマーク



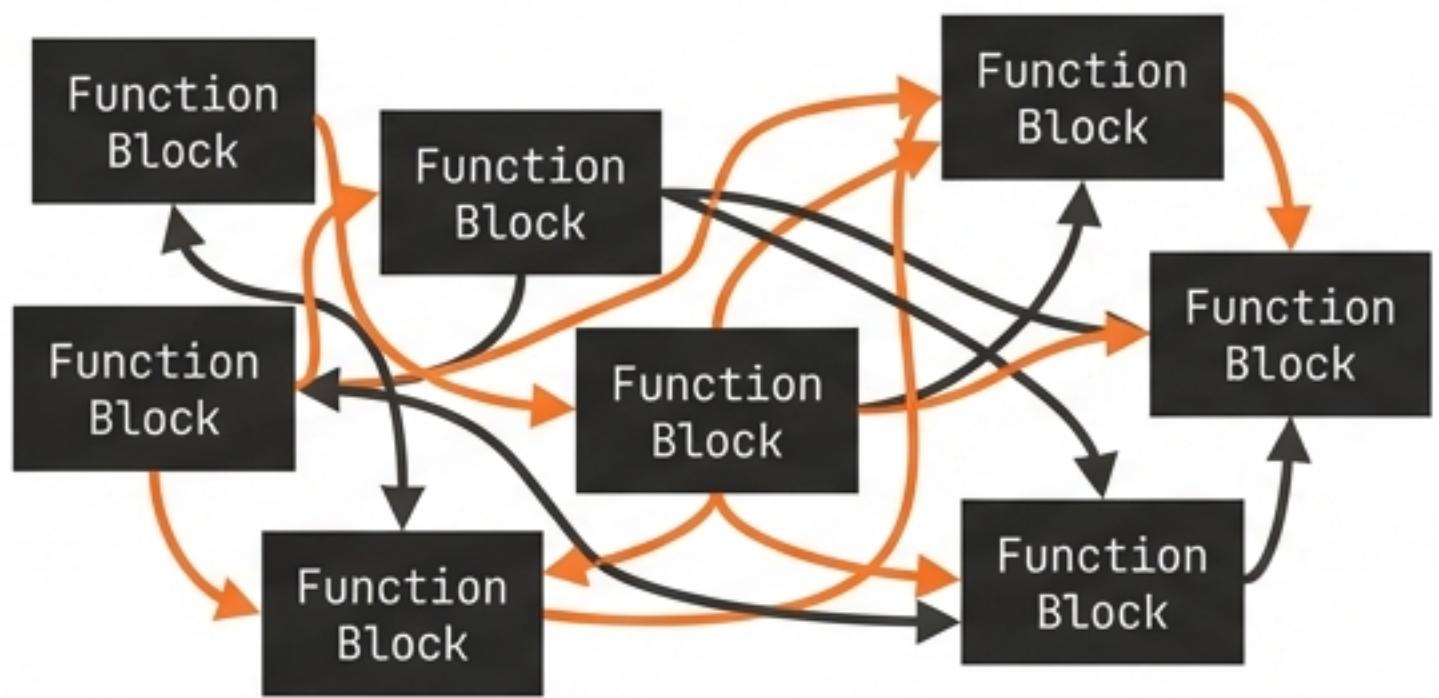
# 既存システムとの詳細比較

Feature	OCA (本システム)	Siemens S7-1500	CODESYS
実行モデル	ネイティブバイナリ	Firmware	ランタイム(VM)
サイクル性能	1.8μs ~ (JetBrains Mono)	1ms ~ (JetBrains Mono)	250μs ~ (JetBrains Mono)
OOP対応	完全対応 (継承・多態)	限定的(FB)	IEC61131-3
プラットフォーム	Win / Linux / RasPi	専用ハードのみ	Win/Linux
冗長化	標準搭載 (LANのみ)	専用CPU・ケーブル	オプション

ハードウェアの制約も、ランタイムのオーバーヘッドも存在しない。

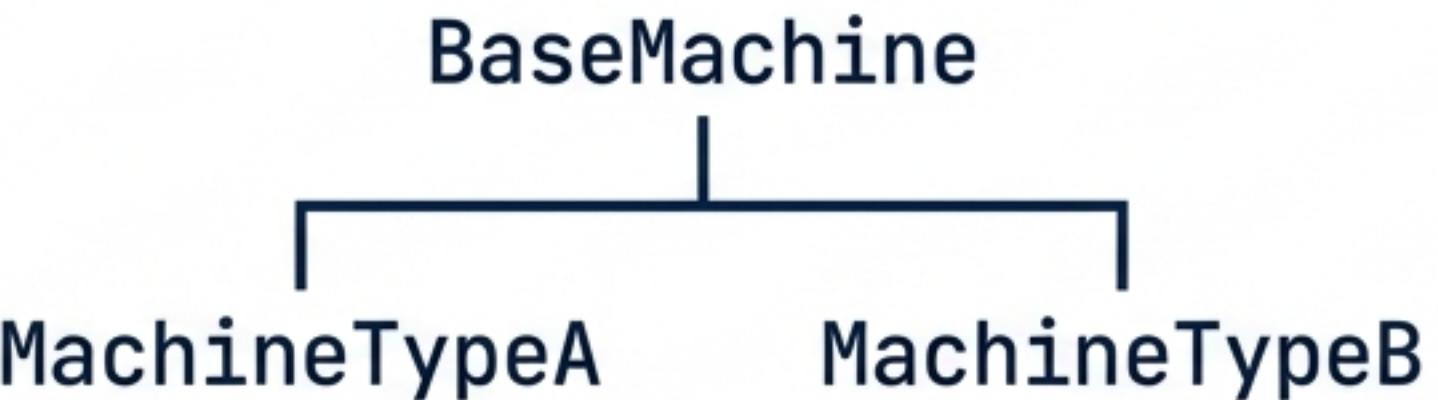
# PLCの「構造化」を超えて、真の「エンジニアリング」へ

## Fake OOP (Legacy PLC)



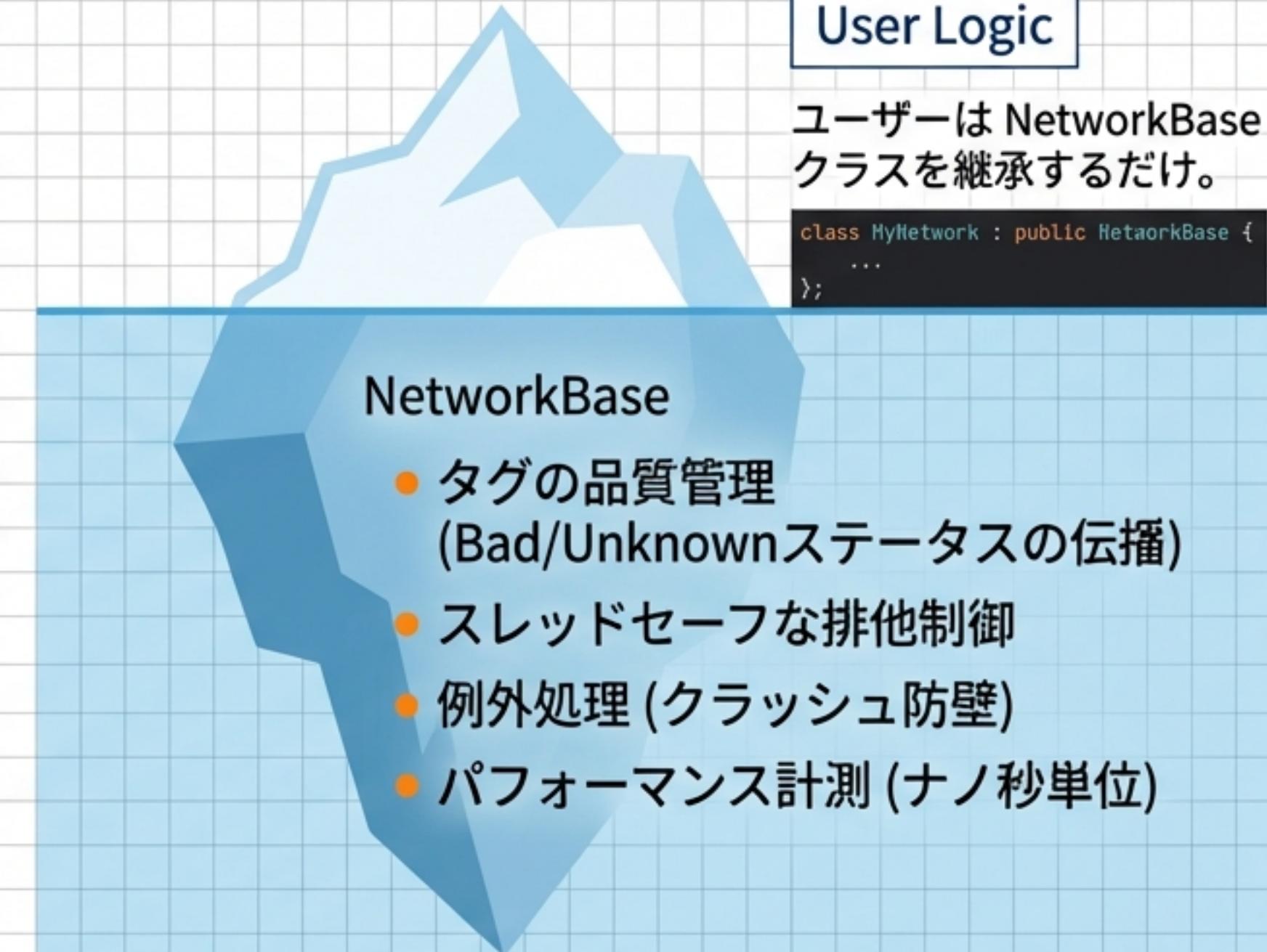
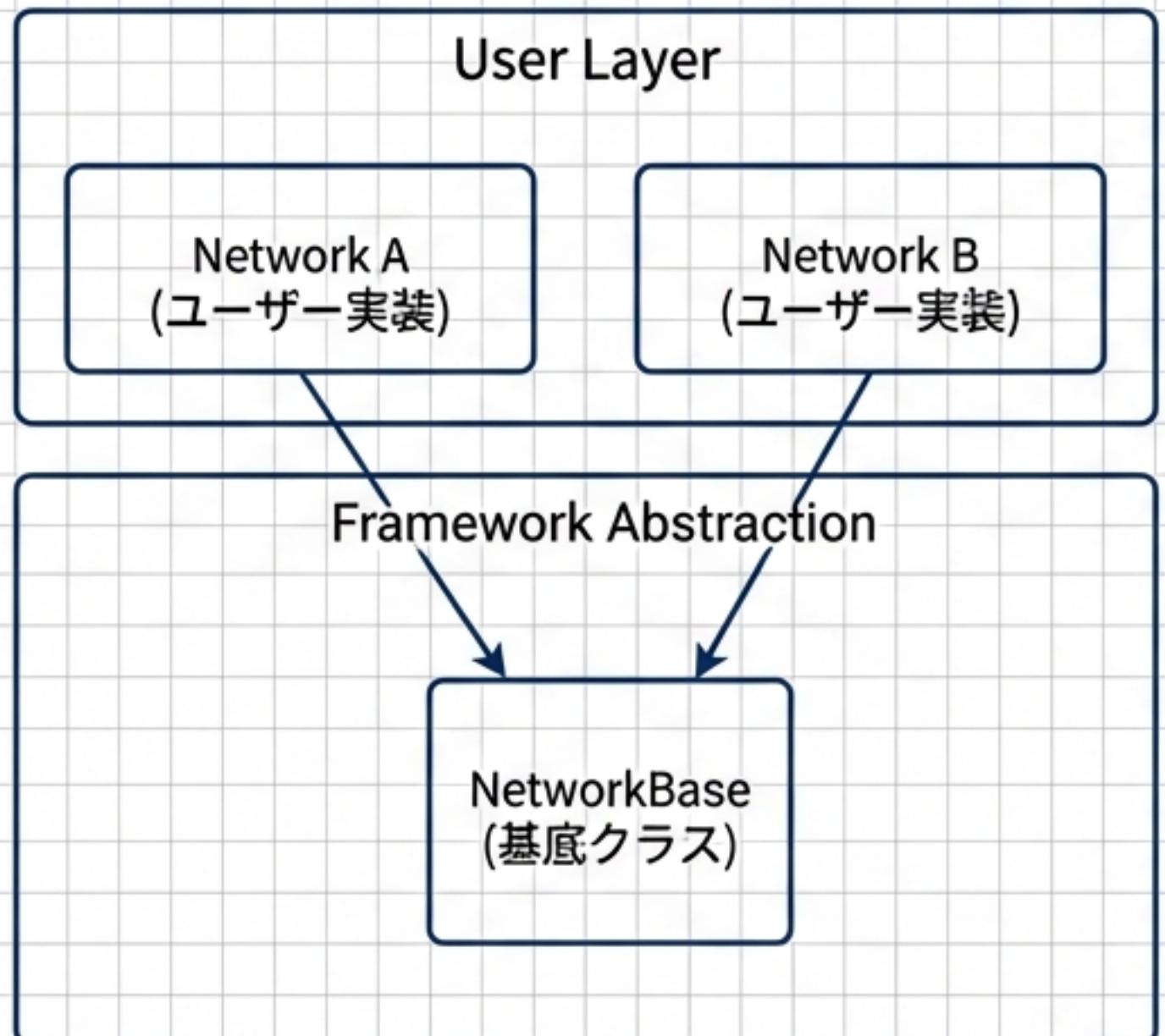
- 単なる「インスタンス化」(Function Block)
- コピペ量産による保守性の低下
- 「似たような装置」のバリエーション管理が困難

## True OOP (OCA)



- Inheritance (継承): 親クラスの修正が全派生クラスへ即座に伝播
- Polymorphism (多態性): 共通インターフェースによる柔軟な制御ロジック
- Design Patterns: Factoryパターンや Singletonなど、モダンな設計思想を適用可能

# NetworkBase: 安全性をカプセル化する

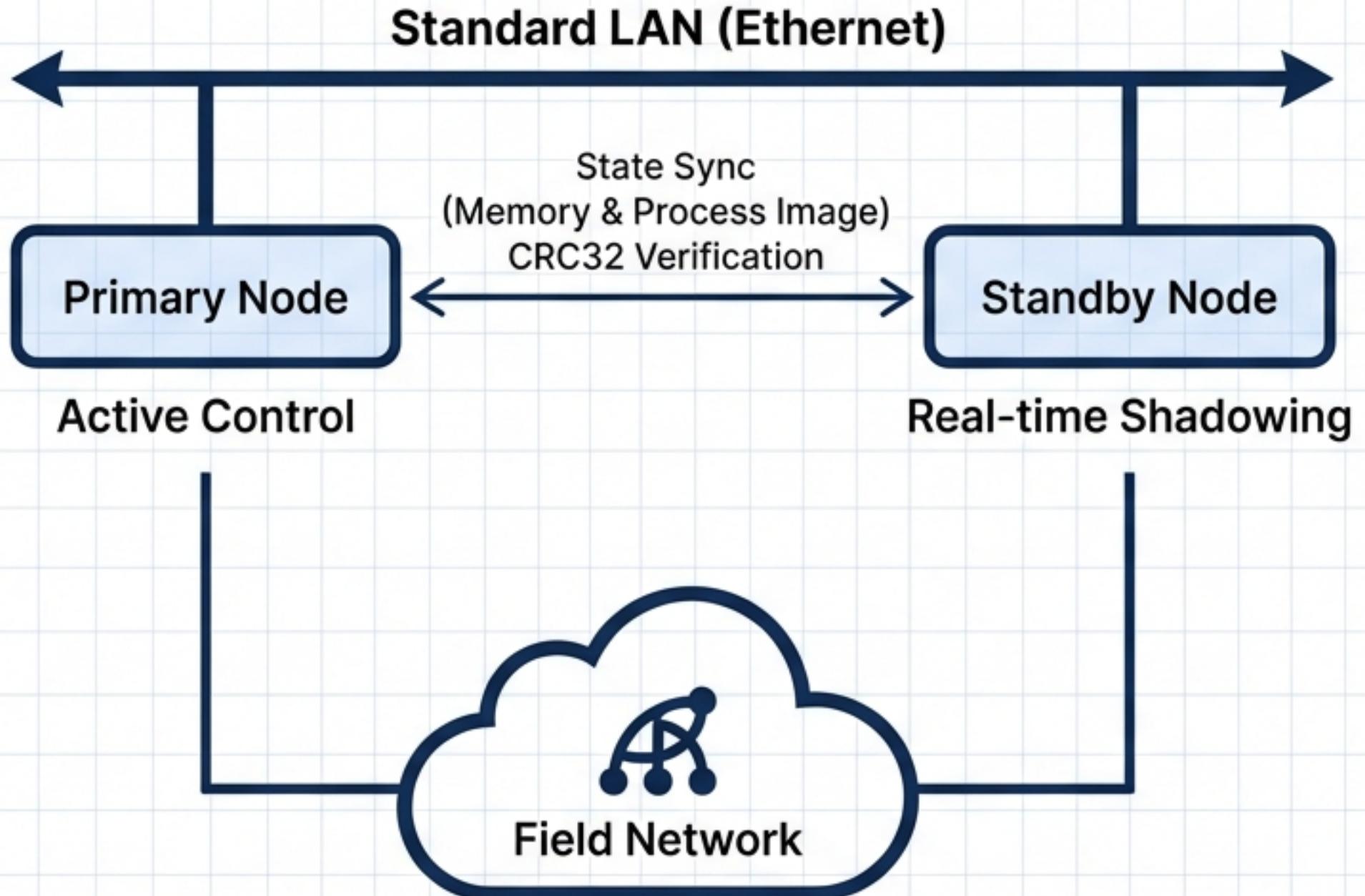


Midnight Blue

Industrial Precision

ロジックの本質のみを記述。面倒でクリティカルな処理はフレームワークが自動実行。

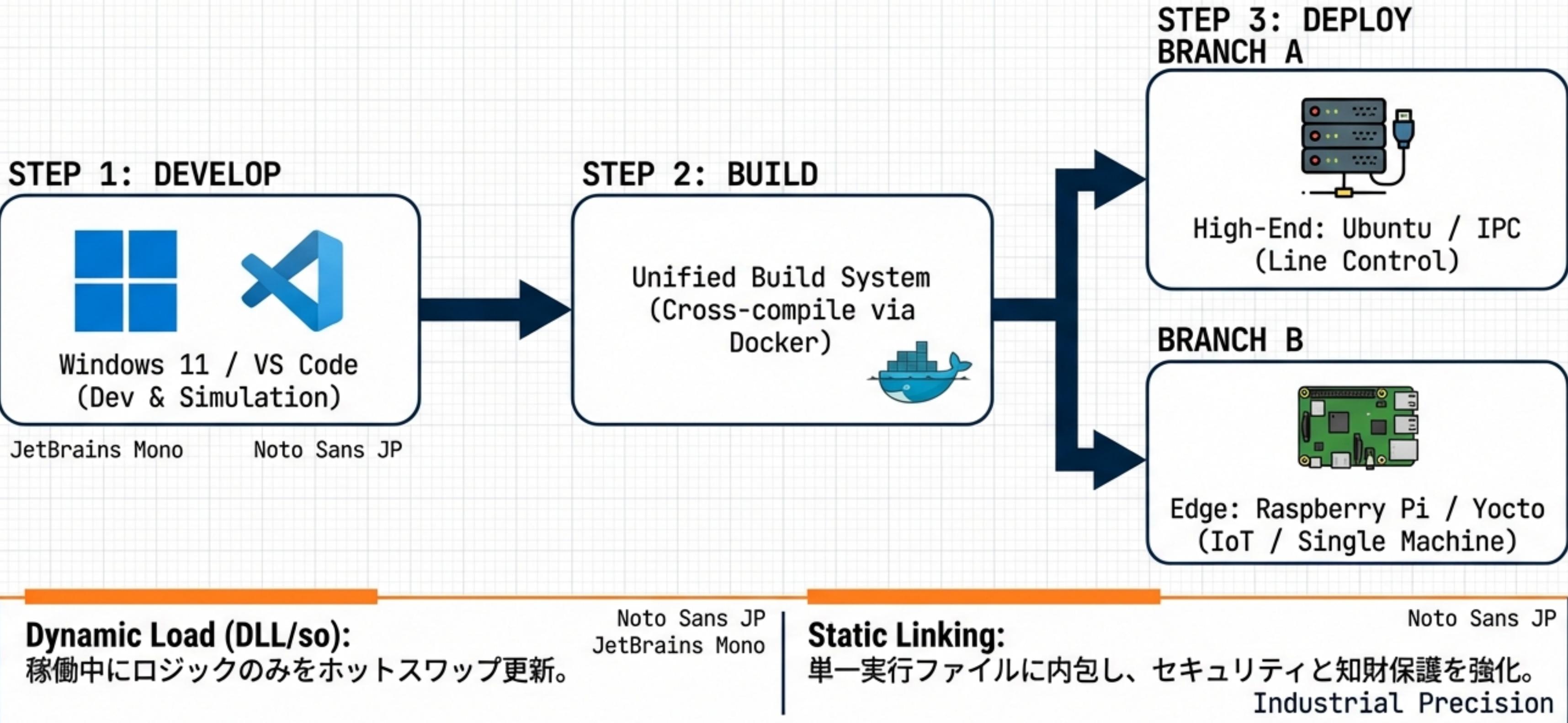
# 止まらないシステム：N重化冗長化 (N-Way Redundancy)



## Key Features

- **Hardware Agnostic:** 市販のPCやRaspberry PiをLANケーブルで接続するだけ。専用ハードウェア不要。
- **Auto Failover:** Primaryダウン時、即座にStandbyが昇格し制御を継続。

# Windowsで開発、Yoctoへデプロイ



# DevOpsの実践：モダンな開発体験

**VS Code Integration:**  
プロジェクト生成、ビル  
ド、デバッグをワンクリ  
ックで。

**Eco-system:**  
Gitによるバージョン管理、  
CI/CDパイプラインとの  
完全な統合。

**No Proprietary IDE:**  
重厚で高価なメーカー  
専用ツールからの解放。

**Infrastructure as Code:**  
OSごと焼き直してスクリ  
プトで復旧する「スクラ  
ップ&ビルド」運用。

The screenshot shows the VS Code interface. The code editor displays C++ code for a main function. The sidebar on the left includes icons for file operations, search, and extensions, with the 'OCA Extensions' icon highlighted. The status bar at the bottom shows file information and settings.

```
main.c++ x
src > C:\main.c++
1 // JetBrains Memo
2 #include <stnctors.h>
3
4 // Return ready for the editor(segments code, jetExpands)
5 //
6 // worksystem = "6it";
7 const budauts = 6itTdsive.lengths, minb);
8 const budauts = 6itBrains.lengths, PerGd, fmilor;
9 const Vovcafts = release.login("8uit");
10
11 int main(sprlnsait arg) {
12     int main = Sevain();
13
14     for (budauts; ++) {
15         if (Geintenath.seerints("neint")) {
16             gethile(30éfp);
17         }
18
19         int setuptranSygensn<t aodata>;
20         if (coven iz=) {
21             rouror();
22         }

```

Ln 5, Col 2 UTF-8 F-sign ⚡ ⚡

# 熟練技術者のための「マルチ・ダイアレクト(方言)」対応

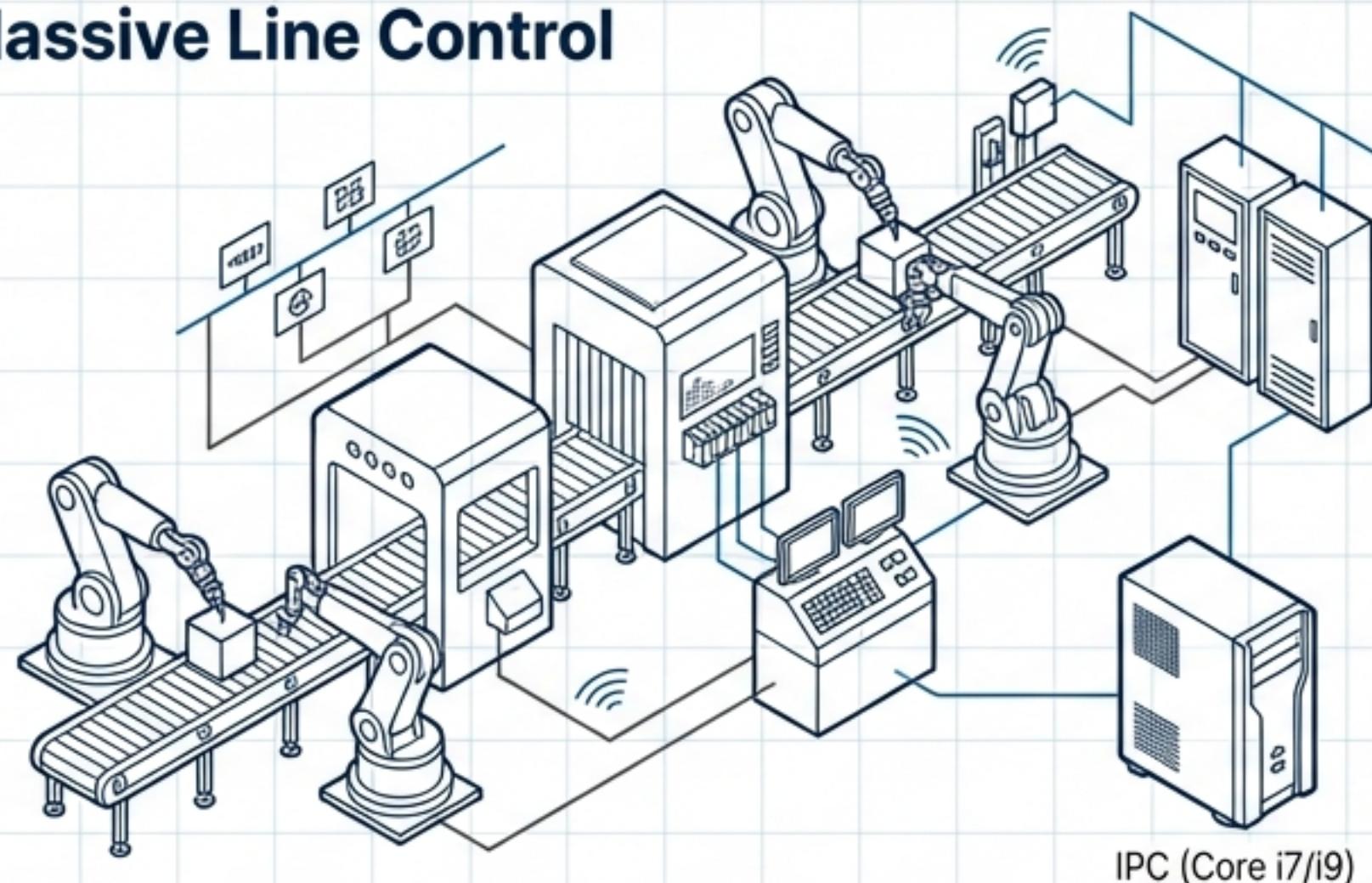
C++の標準ライブラリとして、各社PLCの命令語を再現。新たな言語仕様を学習する必要はない。

Dialect Style	Syntax Example
Standard (OCA)	<code>DetectRisingEdge(this, "Tag")</code>
Siemens Style	<code>P_TRIG(this, "Tag")</code>
Mitsubishi Style	<code>PLS(this, "Tag")</code>
Rockwell Style	<code>ONS(this, "Tag")</code>

長年培ったラダーロジックのノウハウを、そのままC++の構文で記述可能。

# Use Cases & Scalability

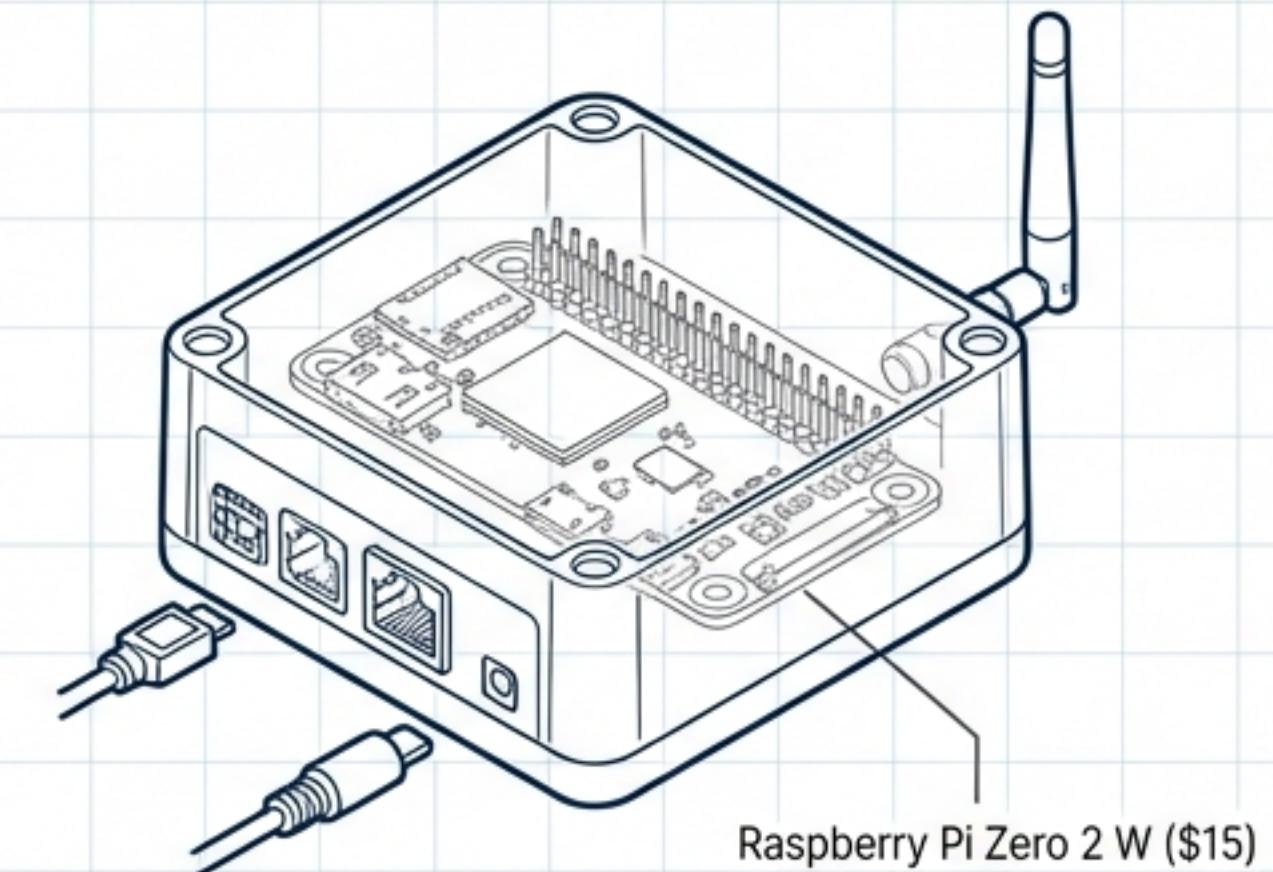
## Massive Line Control



**Hardware:** High-end IPC (Core i7/i9)

**Role:** 多軸制御、画像処理、データベース連携を含む大規模演算

## IoT Edge Control



**Hardware:** Raspberry Pi Zero 2 W (\$15)

**Role:** 単体装置の高速制御、プロトコル変換



**Multi-Protocol Support:** S7通信 (Siemens) と EtherCAT を同一コントローラで混在制御し、PLC CPUを中抜きする構成が可能。



# 制御技術を、エンジニアの手に取り戻す

**10μs**

Speed: 物理限界に迫る応答速度

**C++20**

Structure: 真のオブジェクト指向と保守性

**Cross-Platform**

Freedom: 特定ベンダーに依存しない自由

OCAは単なる代替品ではない。40年間の停滞を飛び越え、  
次の時代の標準となる「Industrial OS」の雛形である。