OCA (Object-Component Automation) Technical Briefing Document

1. Executive Summary

OCA (Object-Component Automation) is a next-generation industrial control framework adopting the latest C++20 standard. It realizes PLC-level robustness and deterministic behavior on general-purpose PCs and single-board computers (such as Raspberry Pi).

The traditional Industrial Automation (FA) industry has long suffered from structural issues such as dependence on specific manufacturers ("Vendor Lock-in") and programming models that lag more than 40 years behind the IT industry. OCA breaks these constraints by highly fusing the flexible expressiveness of IT technology (True Object-Orientation) with the reliability essential to OT technology.

Key Achievements and Advantages:

- **Ultra-High-Speed Control Cycles:** Achieves cycle times of **1.8µs to 10µs** on general-purpose OSs, overwhelming traditional hardware PLCs (which operate in the ms order).
- **Guaranteed Determinism:** Achieves long-term stable operation and minimal jitter through strict replication of the "Process Image" and a "Zero Allocation" design.
- **True Object-Oriented Development:** Enables the construction of highly reusable and sophisticated code assets by fully utilizing C++20 inheritance and polymorphism.
- **Hardware Agnostic:** Supports Windows, Linux, and Yocto Linux (Raspberry Pi), breaking away from the enclosure of existing proprietary hardware.
- **Standardized Redundancy:** Capable of configuring N-way redundant systems using only standard LAN connections without the need for dedicated communication cables.

---

2. Current Challenges in Industrial Control and OCA's Philosophy

2.1 Structural Contradictions in Existing PLC Systems

The current PLC industry is in a deeply conservative state regarding software technology, primarily due to the "enclosure" strategies employed by manufacturers to sell their own products (CPUs, I/O, devices).

- **Artificial Performance Throttling:** Manufacturers impose artificial limits on processing speed and tag counts for software PLCs to avoid competition with their own high-end hardware models.
- **"Pseudo" Object-Orientation:** IEC 61131-3 (Function Blocks, etc.) does not support inheritance or polymorphism, significantly lacking modern development efficiency.
- **The "PC is Unstable" Myth:** While general-purpose PCs handle massive calculations in the gaming industry and elsewhere, the widespread adoption of PC-based control in the FA industry has been hindered solely by concerns over jitter.

2.2 OCA's "Democratization of Performance"

OCA is a native framework free from manufacturers' political constraints, converting 100% of the processor's performance into control capability. This allows even a low-cost Raspberry Pi Zero 2 W to demonstrate performance (10µs cycle) equal to or exceeding that of high-end PLCs costing thousands of dollars.

---

3. OCA System Architecture

OCA adopts a robust multi-layer architecture that clearly separates Communication (I/O), Logic Computation, and Memory Management.

3.1 Strict Process Image and Data Flow

To structurally eliminate Race Conditions, the following three steps are enforced in every cycle:

1. **Snapshot (Input):** Atomically acquires all input data at the start of the cycle. Values are fixed during cycle execution.
2. **Logic (Computation):** Computes based on the fixed input values. Guarantees Read-After-Write integrity.
3. **Commit (Output):** After computation is complete, only the changes are transmitted to physical devices in a batch.

3.2 Zero Allocation Design

Dynamic memory allocation (`new` / `malloc`) is **strictly prohibited** during the control execution cycle. All necessary memory regions are pre-allocated during the startup phase. This eliminates sudden delays caused by memory fragmentation or OS memory searches, ensuring no performance degradation even over years of continuous operation.

3.3 Cooperative Interrupts

Forced interrupts by the OS carry the risk of data corruption. OCA is designed to accept interrupts only at logic breaks, ensuring determinism while maintaining dat a integrity.

---

4. Development Model and Programming

4.1 Abstraction via NetworkBase

Developers can enjoy complex background processing automatically simply by inherit ing the `NetworkBase` base class.

- **Automatic Handle Resolution:** Automatically converts string tag access to O (1) high-speed access.
- **Quality Management (Quality Bit):** Automatically propagates statuses such as communication disconnections.
- **Performance Monitoring:** Measures the execution time of each logic component with nanosecond precision.

4.2 Application of True Object-Orientation (OOP)

While existing PLC FBs (Function Blocks) are limited to simple instantiation, OCA fully applies C++20 "Inheritance" and "Polymorphism."

- **Encapsulation of Common Functions:** Centralizes exclusive control and error handling in the base class.
- **Asset Reuse:** Enables "Differential Programming," where modifications to a p arent class are immediately reflected in all derived classes.

4.3 Multi-Vendor Compatible Libraries

To lower the barrier to entry for existing PLC engineers, OCA comes standard with wrapper libraries that have signatures identical to the instruction names of major manufacturers (Siemens, Mitsubishi Electric, Omron, Keyence, etc.). This allows lo gic to be written in C++ using familiar commands like `P_TRIG` (Rising Edge Detectio n) and `TON` (Timer).

---

5. Performance and Platform

OCA minimizes OS-dependent code (less than 3% of the total), supporting multiple e nvironments with the same source code.

Platform	OS / Kernel	Cycle Time (Measured)	Use Case
Windows 11	Core i5 / Standard OS	4.0 µs	Development, HMI, High-Performance Computing
Ubuntu 22.04	Linux 5.15 (Virtual)	2.0 µs	Industrial PC (IPC)
Raspberry Pi ZERO 2 W	Yocto Linux	9.0 µs	Edge / Low-Cost Control

---

6. Safety and Reliability Design

Multiple defense measures are implemented against risks specific to PC control.

6.1 Double Watchdog Timer (WDT)

- **Internal WDT:** The logic engine monitors its own cycle time.
- **External WDT:** A separate thread (Coordinator) monitors the engine's liveness. Forces a stop in case of a freeze.

6.2 N-Way Redundancy System

Achieves mission-critical availability without dedicated hardware.

- Configurable using only LAN cable connections.
- **CRC32 Verification:** Detects corruption of synchronization data.
- **Automatic Failover:** When the Primary node goes down, the Standby node is immediately promoted.

6.3 Emergency Stop Sequence

Upon detecting an anomaly, the system rewrites all output tags to a predefined "Safe State," attempts up to 3 retry transmissions to devices, and then safely shuts down the system.

---

7. Engineering Environment: OCA User Logic Engineering Kit

A sophisticated development environment integrated with Visual Studio Code (VS Code) is provided.

- **Automatic Project Generation:** Builds CMake-based projects with a single click.
- **Modern Development Flow:** Features code completion via IntelliSense, static analysis, Git management, and AI coding assistance.
- **Dual Runtime:** Includes both Debug and Release environments, enabling simulation without actual hardware.
- **Deployment Pipeline:** Executes binary transfer, permission setting, and service restart on the target (Windows/Linux/Raspberry Pi) with one click.

---

8. Conclusion

OCA is the **"Third Option"** for the industrial sector, fusing the "flexibility and overwhelming computational power" of PCs with the "robustness and determinism" of PLCs.

By breaking away from the enclosure models of existing manufacturers and bringing generic C++20 technology into the world of control, OCA realizes agile development in manufacturing, advanced data processing (high-speed processing of image/waveform data), and overwhelming cost performance. It is not just a control engine, but a next-generation platform that liberates industrial control as a software technology.