mcdb

次世代産業用制御プラットフォーム「Object-Component Automation（OCA）」ブリーフィングドキュメント

1. エグゼクティブサマリー

OCA（Object-Component Automation）は、最新のC++20規格を採用し、汎用PCやシングルボードコンピュータ（Raspberry Pi等）上でPLCレベルの堅牢性と決定論的動作を実現する次世代の産業用制御フレームワークである。

従来の産業用制御（FA）業界は、特定メーカーへの依存（ベンダーロックイン）や、IT業界から40年以上遅れたプログラミングモデルといった構造的課題を抱えてきた。OCAは、IT技術の柔軟な表現力（真のオブジェクト指向）と、OT技術に不可欠な信頼性を高度に融合させることで、これらの制約を打破する。

主要な成果と優位性:

- **超高速な制御周期:** 汎用OS上でありながら、1.8µs〜10µsという、従来のハードウェアPLC（msオーダー）を圧倒するサイクルタイムを実現。
- **決定論的動作の保証:** 「プロセスイメージ」の厳格な再現と「ゼロ・アロケーション」設計により、長期間の安定稼働とジッタの極小化を達成。
- **真のオブジェクト指向開発:** C++20による継承・多態性の完全活用により、再利用性の高い洗練されたコード資産の構築が可能。
- **ハードウェア選定の自由:** Windows、Linux、Yocto Linux（Raspberry Pi）をサポートし、既存の専用ハードウェアによる囲い込みから脱却。
- **標準搭載の冗長化:** 専用通信ケーブルを必要とせず、LAN接続のみでN重化冗長システムを構成可能。

---

2. 産業用制御における現状の課題とOCAの理念

2.1 既存PLCシステムの構造的矛盾

現在のPLC業界は、メーカー各社が自社製品（CPU、I/O、デバイス）を販売するための「囲い込み」を意図的に行っており、ソフトウェア技術において極めて保守的な状況にある。

- **性能の人工的抑制:** メーカーは自社の上位機種との競合を避けるため、ソフトウェアPLCに対して処理速度やタグ数に人工的な制限を設けている。
- **「なんちゃって」オブジェクト指向:** IEC 61131-3（ファンクションブロック等）では継承や多態性が実現できず、現代的な開発効率を著しく欠いている。
- **「PCは不安定」という神話:** 汎用PCがゲーム産業等で膨大な演算をこなしている一方で、FA業界ではジッタ（揺らぎ）のみを理由にPC制御の普及が妨げられてきた。

2.2 OCAによる「性能の民主化」

OCAはメーカーの政治的制約を受けないネイティブ・フレームワークであり、プロセッサの性能を100%制御に転化する。これにより、数千円のRaspberry Pi Zero 2 Wであっても、数百万円クラスのハイエンドPLCと同等以上の性能（10µs周期）を発揮させることが可能である。

---

3. OCAのシステムアーキテクチャ

OCAは、通信（I/O）、ロジック演算、メモリ管理を明確に分離した堅牢な多層アーキテクチャを採用している。

3.1 厳格なプロセスイメージとデータフロー

データ競合（Race Condition）を構造的に排除するため、以下の3ステップを毎サイクル強制する。

1. **Snapshot（入力）:** サイクル開始時に全入力データをアトミックに取得。サイクル実行中は値が固定される。
2. **Logic（演算）:** 固定された入力値を元に演算。Read-After-Write（書き込み直後の読み出し）の整合性を保証。
3. **Commit（出力）:** 演算完了後、変更分のみを一括して物理デバイスへ送信。

3.2 ゼロ・アロケーション設計

制御実行サイクル中における動的メモリ確保（new / malloc）を一切禁止している。必要なメモリ領域は起動時にすべてプリロケーション（事前確保）される。これにより、メモリの断片化（フラグメンテーション）やOSのメモリ探索に伴う突発的な遅延を排除し、年単位の連続稼働でもパフォーマンスが劣化しない。

3.3 協調的割り込み（Cooperative Interrupts）

OSによる強制的な割り込みはデータ破壊のリスクを伴う。OCAはロジックの切れ目でのみ割り込みを受け入れる設計となっており、データの整合性を保ちながら定周期性を確保する。

---

4. 開発モデルとプログラミング

4.1 NetworkBase による抽象化

開発者は `NetworkBase` 基底クラスを継承するだけで、複雑な裏側の処理を自動的に享受できる。

- **ハンドル自動解決:** 文字列タグアクセスをO(1)の高速アクセスへ自動変換。
- **品質管理（Quality Bit）:** 通信断線等のステータスを自動で伝播。
- **パフォーマンス監視:** 各ロジックの実行時間をナノ秒精度で計測。

4.2 真のオブジェクト指向（OOP）の適用

既存PLCのFB（ファンクションブロック）が単なるインスタンス化に留まるのに対し、OCAはC++20の「継承」と「多態性」を完全適用している。

- **共通機能の隠蔽:** 排他制御やエラーハンドリングを基底クラスに集約。
- **資産の再利用:** 親クラスの修正を全派生クラスへ即座に反映させる「差分プログラミング」が可能。

4.3 マルチベンダー互換ライブラリ

既存のPLCエンジニアの参入障壁を下げるため、主要メーカー（Siemens, 三菱電機, オムロン, キーエンス等）の命令名と同一のシグネチャを持つラッパーライブラリを標準装備している。これにより、`P_TRIG`（立ち上がり検出）や`TON`（タイマー）といった慣れ親しんだ命令を使用してC++で記述できる。

---

5. パフォーマンスとプラットフォーム

OCAはOS依存コードを最小限（全体の3%以下）に抑えており、同一のソースコードで複数の環境をサポートする。

プラットフォーム	OS / カーネル	サイクルタイム（実測値）	用途
Windows 11	Core i5 / 標準OS	4.0 µs	開発・HMI・高性能演算
Ubuntu 22.04	Linux 5.15（仮想環境）	2.0 µs	産業用PC（IPC）
Raspberry Pi ZERO 2 W	Yocto Linux	9.0 µs	エッジ・低コスト制御

---

6. 安全性と信頼性設計

PC制御特有のリスクに対し、多重の防御策を講じている。

6.1 二重ウォッチドッグタイマー（WDT）

- **内部WDT:** ロジックエンジンが自らサイクル時間を監視。
- **外部WDT:** 別スレッド（Coordinator）がエンジンの死活を監視。フリーズ時に強制停止。

6.2 N重化冗長システム（N-Way Redundancy）

専用ハードウェアなしでミッションクリティカルな可用性を実現。

- LANケーブル接続のみで構成可能。
- **CRC32検証:** 同期データの破損を検知。
- **自動フェイルオーバー:** Primaryダウン時、Standbyが即座に昇格。

6.3 緊急停止シーケンス

異常検知時には、全出力タグを定義済みの「安全値（Safe State）」に書き換え、デバイスに対して最大3回のリトライ送信を行った後、システムを安全にシャットダウンする。

---

7. エンジニアリング環境：OCA User Logic Engineering Kit

Visual Studio Code（VS Code）と統合された高度な開発環境が提供される。

- **プロジェクト自動生成:** CMakeベースのプロジェクトをワンクリックで構築。
- **モダンな開発フロー:** IntelliSenseによる補完、静的解析、Git管理、AIコーディング支援。
- **デュアルランタイム:** Debug/Releaseそれぞれの環境を内包し、実機なしでのシミュレーションが可能。
- **デプロイメントパイプライン:** ワンクリックでターゲット（Windows/Linux/Raspberry Pi）へのバイナリ転送、権限設定、サービス再起動を実行。

---

8. 結論

OCAは、PCの持つ「柔軟性と圧倒的な計算能力」と、PLCの持つ「堅牢性と決定論的動作」を融合させた、産業界における「第3の選択肢」である。

既存メーカーの囲い込みモデルから脱却し、汎用的なC++20技術を制御の世界に持ち込むことで、製造現場におけるアジャイルな開発、高度なデータ処理（画像・波形データの高速処理）、および圧倒的なコストパフォーマンスを実現する。これは単なる制御エンジンではなく、産業用制御をソフトウェア技術として解放する次世代のプラットフォームである。