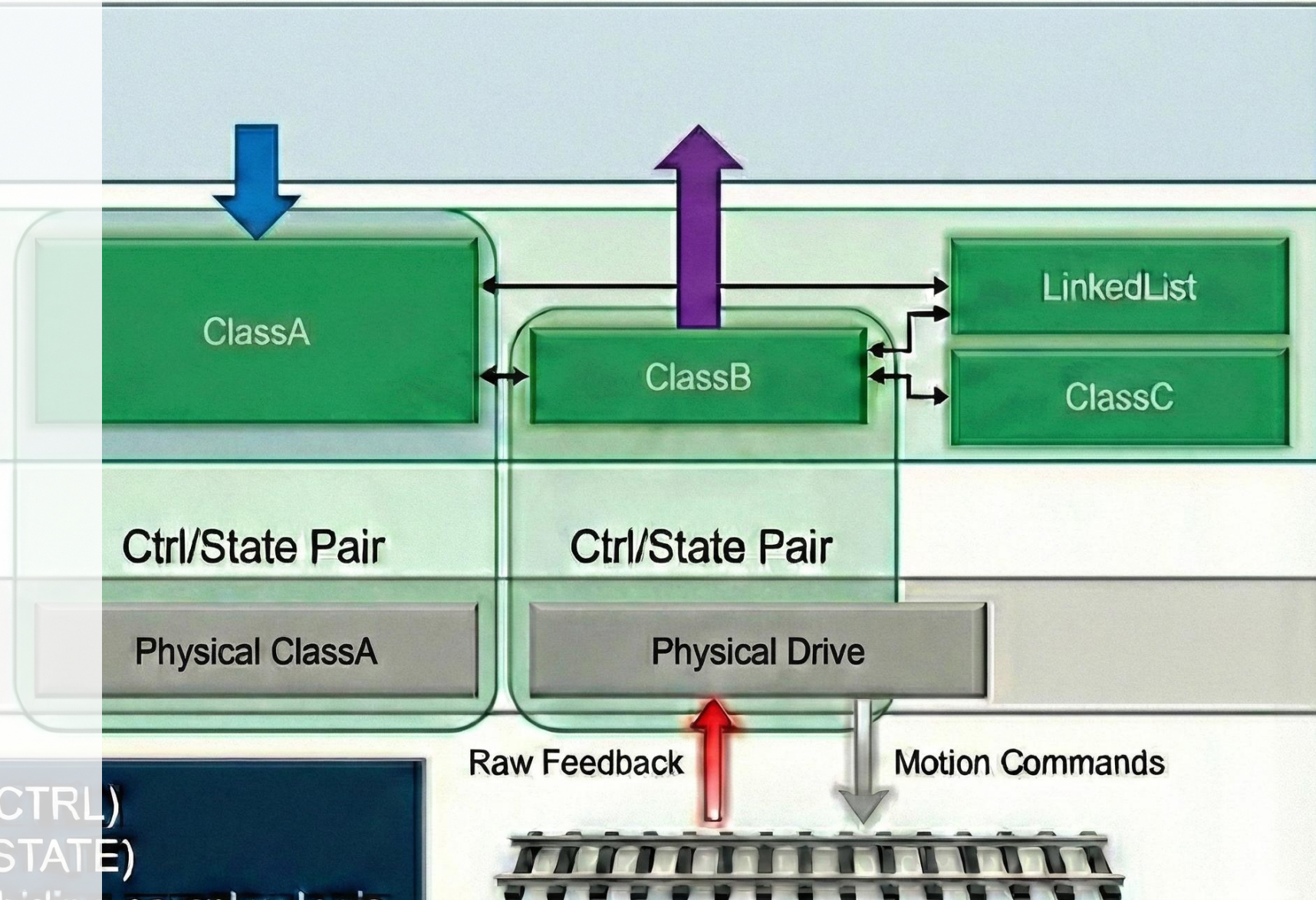


1. Motivation
2. Design



Ctrl = Command Input (ST_*_CTRL)
 State = Status Output (ST_*_STATE)
 Facade = Simplified Interface

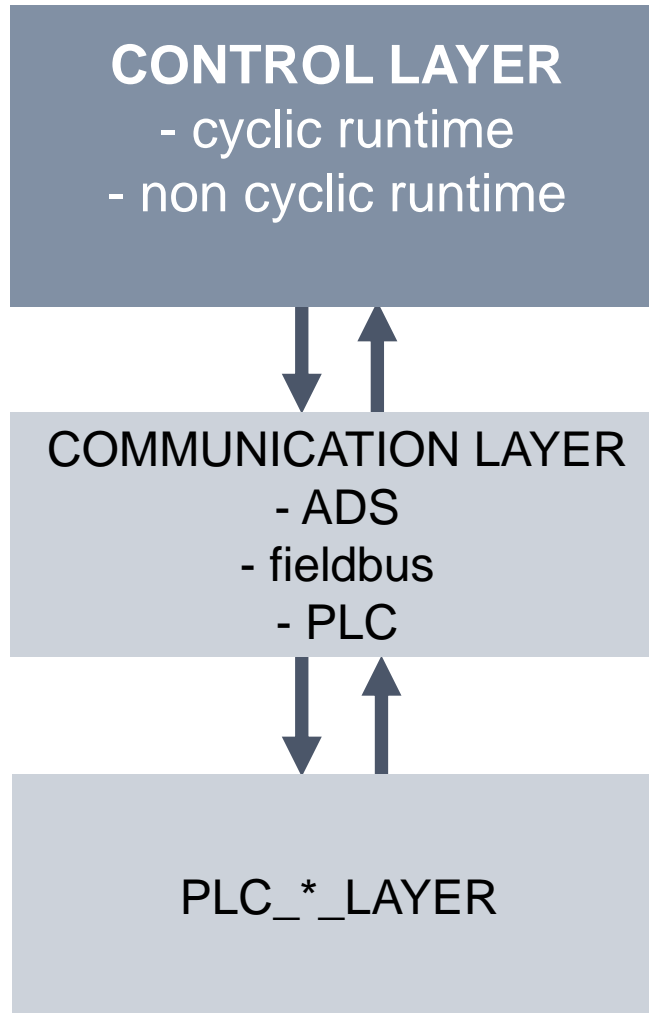
■ Motivation

- Unified means of communication between entities
 - Higher level logic is preserved and shielded from lower-level changes
- Agnostic to means of communication
 - Datafields are mappable to any cyclic or non-cyclic information transport
 - Hard real time, TCP/IP, ADS
- Asynchronous, Atomic flow of information
 - Class instances are communicating via LinkedList implementation
- Reducing software latency
 - Sequential case execution enables zero frame latency
 - LinkedList implementation enables atomic information transport
 - Information arrives ahead of state of the physical layer
- Message-System with narrative syntax
 - Use of enumerations enables logs that are instantly readable for the human eye

▪ Design

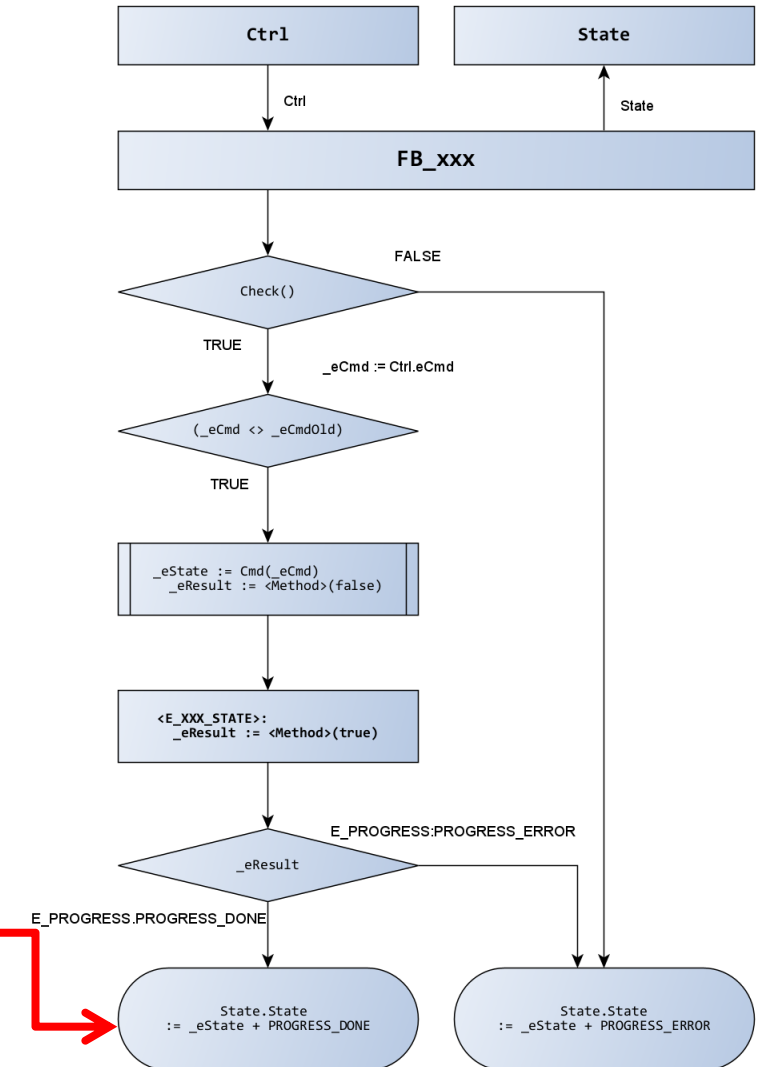
- Compliant to SOLID and ACID principles
 - Built in self similarity of approach enables fractal design choices
 - Due to strong Separation of Concerns the layered approach is suitable for distributed HRT systems
- Encapsulated state machines with generic pattern
 - Generic pattern is applicable to existing code segments
 - Code segments may be reused.
 - Generic pattern enables secure execution
 - State isolation enables to prevent race conditions on a structural level
- Use of base classes with cyclic wrappers (Ctrl/state)
- Cross communication via Interface pointers

- **Design:**
- Transparency of communication layer
- Code base shall remain independent of control layer
- Configurable Options for specific libraries / TC functions
- Balanced load for configurable options in machine layout
- Stable cpu use for XFC applications

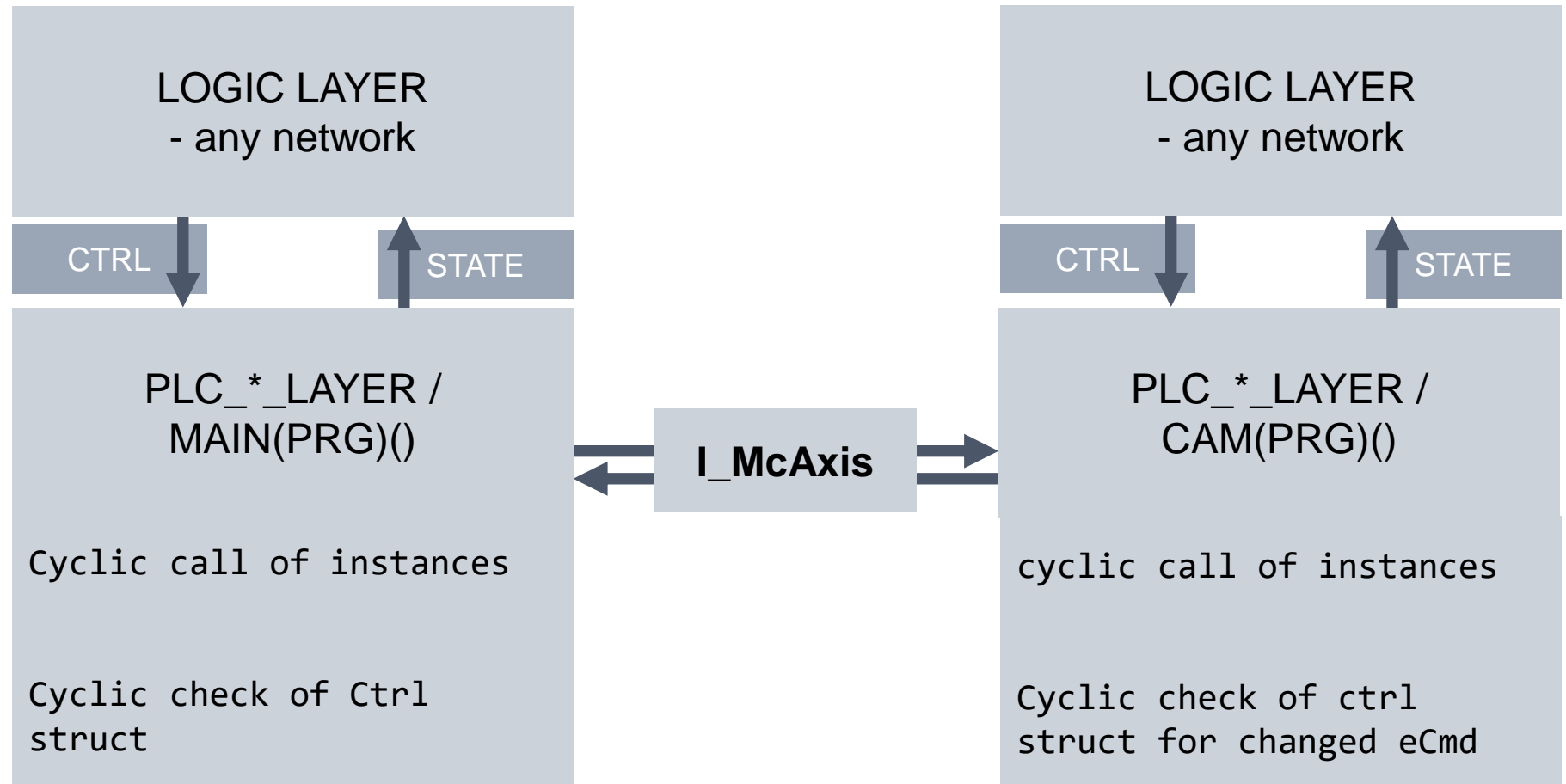


■ Design:

- State / Ctrl structures
 - Establishes unified access
 - Commands can simply be 'dropped' into Ctrl datafield
 - Enables asynchronous communication with PLC_*_LAYER (e.g. C# via ADS, C/C++ via ADS, ADS over MQTT, ...)
 - Enables cyclic communication with PLC_*_LAYER since structures can easily be mapped onto any cyclic fieldbus TwinCAT supports (EtherCAT, Profinet, CanOpen, EAP, ...)
 - State is updated by PLC_*_LAYER, so you can move on doing other stuff, come back and check completion/error.
- State feedback for cyclic class wrappers
 - **Always** combined with E_PROGRESS
 - You can filter your response by a simple modulo division
 - The result may be your entry point for your reaction to State.



▪ Software Design Example: Separation of Concerns



▪ SOLID DESIGN

- **S** — Single Responsibility Principle
 - a class should have one, and only one, reason to change.
- **O** — Open/Closed Principle
 - software entities should be open for extension but closed for modification.
- **L** — Liskov Substitution Principle
 - Objects of a superclass shall be replaceable with objects of its subclasses without breaking the application.
- **I** — Interface Segregation Principle
 - users should not be forced to depend upon interfaces that they do not use.
 - a software entity only sees the methods it actually needs to the work.
- **D** — Dependency Inversion Principle
 - Depend upon abstractions, not concretions.

▪ ACID DESIGN

- **A — Atomicity**
 - A transaction is an "all-or-nothing" unit of work.
- **C — Consistency**
 - A transaction must bring the system from one valid state to another valid state.
- **I — Isolation**
 - Concurrent transactions should not interfere with each other.
- **D — Durability**
 - Once a transaction has been committed, it will remain so, even in the event of power loss, crashes, or errors.