VineRail Facility Control Specification: UML/SysML-Style Pipeline Model, PLC Hierarchy, and Safety Circuit Topology

Flyxion

January 4, 2026

Abstract

This document specifies the VineRail sorting and recycling pipeline as a hierarchical set of assemblies controlled by programmable logic controllers (PLCs) and safety PLCs, with distributed I/O, deterministic fieldbus networking, and explicit state-machine semantics for robotic traversal cycles, tool exchange, staging, recursive dismantling loops, and fluidized separation in a graded-sieve centrifuge. A UML/SysML-style model is provided via renderable text diagrams (PlantUML) and corresponding implementation-oriented PLC function-block interfaces. A facility-level "circuit diagram" is provided as a safety power-and-interlock topology with dual-channel emergency stop, safety gates, torque-off (STO) drive inhibit, and contactor feedback monitoring suitable for industrial certification workflows.

1 Scope and Assumptions

This specification covers the facility-level automation for a VineRail installation comprising overhead traversal lanes, docking/tool-change stations, staging tables and conveyors, hazard isolation cells, robotic-assisted dismantling cells, wash loops, and a kelp/baleen-inspired graded-sieve centrifuge (Centerfuge) for continuous bulk separation.

The robotic ball-bots are treated as *managed mobile resources* with bounded capabilities, whose safety-critical motion remains constrained to infrastructure-defined corridors. High-level task allocation is performed by supervisory control (SCADA/MES), while all safety interlocks, motion permissives, and actuator-level sequencing are implemented in PLC logic with hardwired safety paths.

2 System Decomposition into Hierarchical Assemblies

The facility is modeled as an assembly tree. Each assembly exposes a finite set of interfaces: power, safety, comms, discrete I/O, analog I/O, and a function-block API.

Top-Level Assemblies

Assembly ID	Description
A0	VineRail Facility (complete pipeline, safety envelope, network backbone, historian, operator interfaces).
A1	Overhead Locomotion Infrastructure (rails/cables, nodes, lane controllers, drop corridors).
A2	Docking & Tool-Change Stations (mechanical capture, tool magazines, charge rails, alignment fixtures).
A3	Staging & Presentation (conveyors, singulation, shallow-bed presentation surfaces, bin tipping).
A4	Identification & Hazard Screening (vision/RFID gates, IR/thermal gates, battery detection, diversion gates).
A5	Extraction & Dismantling Cells (magnetic extraction, constrained cutting/opening, component spreaders).
A6	Wash & Slurry Conditioning (wash loops, filters, contaminant traps, pH/ORP monitoring).
A7	Centerfuge Bulk Separation (rotor, graded sieves, compliant filtering, multi-stream discharge manifolds).
A8	Output Handling & Baling (fraction bins, conveyors, presses, QC sampling).
A9	Safety System (E-stops, gate switches, light curtains, safety PLC, STO network, safety relays).
A10	Supervisory Control (SCADA/HMI, MES interface, recipes, analytics, alarms, audit logs).

3 Control Architecture and PLC Hierarchy

3.1 Controller Roles

The control system is divided into four tiers.

Tier 0: Safety Control. A safety PLC executes all safety logic and drives STO permissives and safety contactors. Safety signals are dual-channel where required. Safety PLC communicates status to standard PLCs but remains operationally independent.

Tier 1: Machine PLCs. Standard PLCs control deterministic sequences for each major assembly (A1–A8). These PLCs execute state machines, interlocks, actuator sequencing, and diagnostic handling.

Tier 2: Cell Supervisors. A cell controller (could be PLC or industrial PC) coordinates multiple machine PLCs within a cell (e.g., dismantling + staging + diversion). It manages material flow buffers, prevents deadlock, and enforces recipe constraints.

Tier 3: SCADA/MES. Supervisory systems provide operator UI, recipes, trending, and reporting. They do not command safety functions directly and do not implement real-time motion sequencing.

3.2 Reference Hardware (Abstract)

This spec is vendor-neutral but assumes IEC 61131-3 capable PLCs and a safety PLC with certified safety I/O. Networking may be Profinet, EtherNet/IP, or EtherCAT for deterministic I/O; OPC UA for supervisory data exchange is assumed.

4 Tagging, Addressing, and Signal Conventions

Signals follow a structured naming scheme:

`[Assembly]::[Module]::[Device]::[Signal]`

Examples:

`A2::Dock01::ToolMag::PosOK`, `A7::Centerfuge::Rotor::SpeedRPM`, `A9::Safety::EST`

Discrete signals are Boolean. Analog signals include units and scaling. All devices include a diagnostic bitfield, a command word, and a status word.

5 UML/SysML-Style Model (PlantUML)

The following diagrams are provided as PlantUML text. Render them with any PlantUML tool to obtain visuals. These diagrams are *the UML document* for the pipeline.

5.1 Component Diagram: Facility Modules

Listing 1: PlantUML Component Diagram: VineRail Facility

```
@startuml
skinparam componentStyle rectangle
title VineRail Facility Components (A0)

component "A9 Safety System\n(Safety PLC + Safety IO)" as A9
component "A1 Overhead Locomotion\n(Lane PLCs + Node IO)" as A1
component "A2 Dock & Tool-Change\n(Station PLCs)" as A2
component "A3 Staging & Conveyance\n(Conveyor PLCs)" as A3
component "A4 ID & Hazard Screening\n(Gates PLCs)" as A4
component "A5 Extraction & Dismantling\n(Cell PLCs)" as A5
component "A6 Wash & Conditioning\n(Process PLC)" as A6
component "A7 Centerfuge Separation\n(Process PLC + Drives)" as A7
component "A8 Output Handling\n(Bins/Press PLC)" as A8
```

```
component "A10 Supervisory\n(SCADA/MES/Historian)" as A10

A9 -down-> A1 : Safety Permissive\nSTO/Zone Enable
A9 -down-> A2 : Safety Permissive\nDocking Enable
A9 -down-> A5 : Safety Permissive\nCut/Knife Enable
A9 -down-> A7 : Safety Permissive\nRotor Enable

A3 --> A4 : Material stream\n+ item IDs
A4 --> A5 : Diverted streams\n+ hazard flags
A5 --> A6 : Prepared fractions
A6 --> A7 : Conditioned slurry
A7 --> A8 : Separated fractions

A10 ..> A1 : Recipes, Alarms, KPIs
A10 ..> A3 : Setpoints, Schedules
A10 ..> A7 : Trend, Batch logs

@enduml
```

5.2 Deployment Diagram: PLC and Network Topology

Listing 2: PlantUML Deployment: Network and Controllers

```
@startuml
title Deployment: Controllers, Networks, and Safety Domains

node "Control Room" {
  node "SCADA Server\n(OPC UA Client)" as SCADA
  node "Historian" as HIST
  node "Engineering Workstation" as EWS
}

node "Industrial Network Backbone\n(VLAN: CTRL)" as NET {
}

node "Safety Network\n(Certified Safety Fieldbus)" as SAFENET {
  node "Safety PLC" as SPLC
  node "Safety IO Rack A9" as SIO
}

node "Machine Cell A1" {
  node "Lane PLC A1" as PLC_A1
  node "Remote IO A1" as RIO_A1
}

node "Machine Cell A2" {
```

```
  node "Station PLC A2" as PLC_A2
  node "Remote IO A2" as RIO_A2
}

node "Process Cell A7" {
  node "Process PLC A7" as PLC_A7
  node "VFD Drives (Rotor/Pumps)" as DRV
  node "Remote IO A7" as RIO_A7
}

SCADA -- NET
HIST -- NET
EWS -- NET

PLC_A1 -- NET
PLC_A2 -- NET
PLC_A7 -- NET

SPLC -- SAFENET
SIO -- SAFENET

SPLC ..> PLC_A1 : Safety status only
SPLC ..> PLC_A2 : Safety status only
SPLC ..> PLC_A7 : Safety status only

@enduml
```

5.3 Activity Diagram: Full Pipeline Flow with Recursion

Listing 3: PlantUML Activity: Recursive Sorting + Dismantling + Separation

```
@startuml
title Activity: VineRail Recursive Pipeline (Discrete + Bulk)

start
:Ingress staging (A3);
:Scan/ID gate (A4);
if (Hazard suspected?) then (yes)
  :Divert to hazard lane;
  :IR/Thermal verification;
  if (Confirmed hazard?) then (yes)
    :Isolate / safe handling;
    stop
  else (no)
    :Return to main stream;
  endif
```

```
endif

:Coarse sort / extraction (A5);

if (Composite/packaged?) then (yes)
  :Constrained opening (A5);
  :Spread contents (A5);
  :Re-scan newly exposed parts (A4);
  note right
    Recursion: parts re-enter earlier stages
    until entropy/heterogeneity below threshold.
  end note
  :Route parts to appropriate extraction;
endif

:Wash & conditioning (A6);
:Centerfuge graded separation (A7);
:Fraction outputs to bins/press (A8);
stop
@enduml
```

5.4 State Machine: Ball-Bot Traversal Cycle and Docking

Listing 4: PlantUML State: Ball-Bot Cycle (PLC-Managed Semantics)

```
@startuml
title State Machine: Ball-Bot Traversal Cycle

[*] --> Docked_Ready
Docked_Ready : Tool verified
Docked_Ready : Charge OK or sufficient SoC
Docked_Ready --> Acquire_Permissive : Dispatch command

Acquire_Permissive : Await lane enable + zone clear
Acquire_Permissive --> Traverse_Lane : Permissive granted

Traverse_Lane : Alternating attach-swing-release
Traverse_Lane --> Work_Zone : Arrive target node

Work_Zone : Execute unipurpose action
Work_Zone --> Return_To_Dock : Work complete

Return_To_Dock --> Dock_Approach
Dock_Approach --> Dock_Captured : Station capture OK
Dock_Captured --> Tool_Exchange : Exchange requested
Tool_Exchange --> Docked_Ready : Tool validated
```

```
state Fault {
  [*] --> Fault_Hold
  Fault_Hold --> Controlled_Descent : If attachment/power fault
  Controlled_Descent --> Await_Recovery
}
Traverse_Lane --> Fault : Fault detected
Work_Zone --> Fault : Fault detected
Dock_Approach --> Fault : Fault detected
@enduml
```

6 PLC Function-Block Specifications (IEC 61131-3 Interfaces)

This section specifies a minimal but complete set of PLC function blocks sufficient to implement the pipeline deterministically. The interfaces are written in pseudo-IEC notation.

6.1 Core Types

Listing 5: Core Types and Status Words

```
TYPE E_Mode : (MODE_OFF, MODE_MANUAL, MODE_AUTO, MODE_MAINT); END_TYPE
TYPE E_Severity : (SEV_INFO, SEV_WARN, SEV_ALARM, SEV_TRIP); END_TYPE

TYPE T_Alarm :
STRUCT
  Code : DINT;
  Severity : E_Severity;
  Active : BOOL;
  Latched : BOOL;
END_STRUCT
END_TYPE

TYPE T_DeviceStatus :
STRUCT
  Ready : BOOL;
  Running : BOOL;
  Fault : BOOL;
  Interlocked : BOOL;
  Permissive : BOOL;
  AlarmWord : DWORD;
END_STRUCT
END_TYPE
```

6.2 Safety Gateway FB

The safety PLC exposes read-only safety states to standard PLCs via a gateway mapping.

Listing 6: FB$_{SafetyGateway(read-only interface)}$

```
FUNCTION_BLOCK FB_SafetyGateway
VAR_INPUT
  ZoneEnable : BOOL; (* from safety PLC *)
  EstopOK : BOOL; (* all E-stops healthy *)
  GatesOK : BOOL; (* all safety gates healthy *)
  STO_OK : BOOL; (* drives in safe state when required *)
END_VAR
VAR_OUTPUT
  Permissive_A1_Lanes : BOOL;
  Permissive_A2_Docks : BOOL;
  Permissive_A5_Cutting : BOOL;
  Permissive_A7_Rotor : BOOL;
END_VAR
END_FUNCTION_BLOCK
```

6.3 Ball-Bot Lane Controller FB

Listing 7: FB$_{LaneController : dispatch + corridoroccupancy}$

```
FUNCTION_BLOCK FB_LaneController
VAR_INPUT
  Mode : E_Mode;
  SafetyPermissive : BOOL;
  ZoneClear : BOOL; (* from zone sensors / interlocks *)
  DockRequestDispatch : BOOL;
  BotAtDock : BOOL;
  BotAtWorkNode : BOOL;
  BotFault : BOOL;
END_VAR
VAR_OUTPUT
  DispatchGranted : BOOL;
  LaneEnable : BOOL;
  Alarm : T_Alarm;
END_VAR
END_FUNCTION_BLOCK
```

6.4 Tool-Change Station FB

Listing 8: FB$_{ToolChangeStation : mechanicalcapture + validation}$

```
FUNCTION_BLOCK FB_ToolChangeStation
VAR_INPUT
  Mode : E_Mode;
  SafetyPermissive : BOOL;
  BotDocked : BOOL;
  ToolPresent : BOOL;
  ToolID_Read : DINT;
  RequestToolID : DINT;
  MagazineReady : BOOL;
  ClampClosedFB : BOOL;
  ClampOpenFB : BOOL;
END_VAR
VAR_OUTPUT
  ExchangeOK : BOOL;
  ToolID_Confirmed : DINT;
  CmdClampClose : BOOL;
  CmdClampOpen : BOOL;
  Alarm : T_Alarm;
END_VAR
END_FUNCTION_BLOCK
```

6.5 Centerfuge Process FB (Rotor + Sieve Stages)

Listing 9: FB$_{C}entrifuge : rotorspeedcontrol + dischargecoordination$

```
FUNCTION_BLOCK FB_Centerfuge
VAR_INPUT
  Mode : E_Mode;
  SafetyPermissive : BOOL;
  InletFlow_OK : BOOL;
  RotorSpeed_RPM : REAL;
  RotorSpeed_SP : REAL;
  Vibration_OK : BOOL;
  Imbalance_OK : BOOL;
  SieveDeltaP_OK : BOOL;
END_VAR
VAR_OUTPUT
  RotorEnable : BOOL;
  RotorSpeedCmd_RPM : REAL;
  DischargeEnable : BOOL;
  Alarm : T_Alarm;
END_VAR
END_FUNCTION_BLOCK
```

7 PLC Logic Semantics: Key State Machines (Textual)

7.1 Station Tool Exchange (Structured Text)

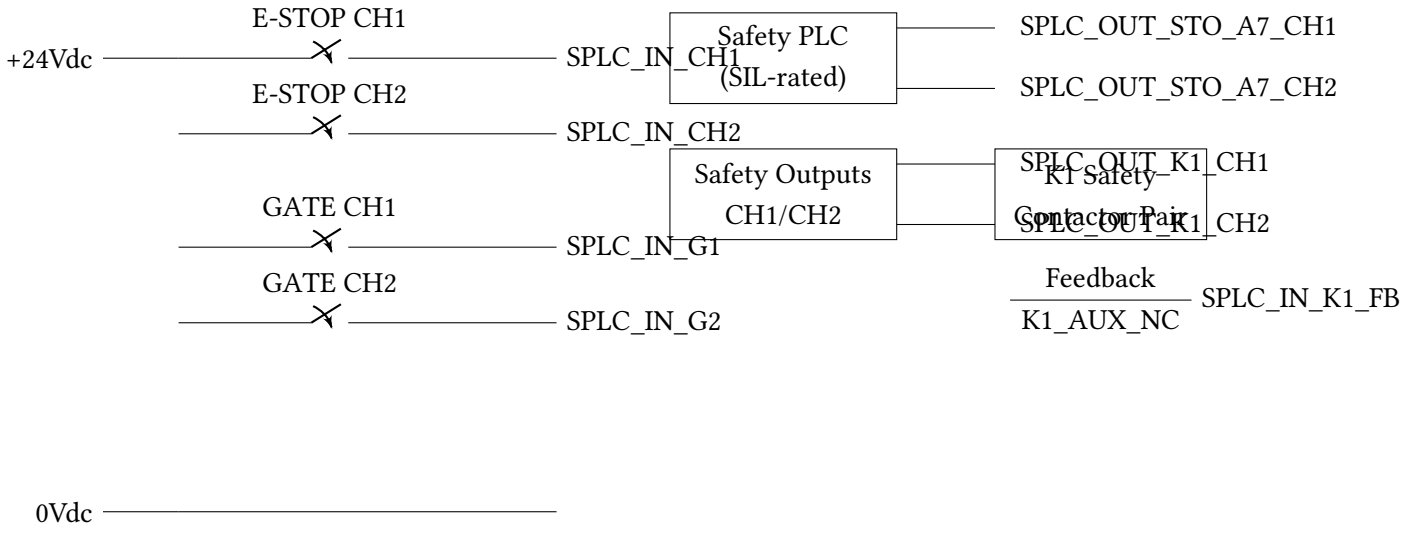
Listing 10: IEC 61131-3 ST pseudo-code: Tool Exchange State Machine

```
(* States: 0=IDLE, 10=CAPTURE, 20=UNCLAMP, 30=SWAP, 40=CLAMP, 50=VERIFY
   , 900=FAULT *)
IF Mode <> MODE_AUTO OR NOT SafetyPermissive THEN
 State := 0;
 CmdClampClose := FALSE;
 CmdClampOpen := TRUE;
END_IF

CASE State OF
0:
 ExchangeOK := FALSE;
 IF BotDocked AND MagazineReady AND (RequestToolID <> 0) THEN
  State := 10;
 END_IF

10: (* CAPTURE: assume external capture mechanism already engaged *)
 IF BotDocked THEN State := 20; END_IF

20: (* UNCLAMP current tool *)
 CmdClampOpen := TRUE;
 IF ClampOpenFB THEN State := 30; END_IF

30: (* SWAP: station actuates magazine selection; abstracted *)
 (* Assume ToolPresent becomes TRUE when new tool positioned *)
 IF ToolPresent THEN State := 40; END_IF

40: (* CLAMP new tool *)
 CmdClampOpen := FALSE;
 CmdClampClose := TRUE;
 IF ClampClosedFB THEN State := 50; END_IF

50: (* VERIFY *)
 ToolID_Confirmed := ToolID_Read;
 IF ToolID_Confirmed = RequestToolID THEN
  ExchangeOK := TRUE;
  State := 0;
 ELSE
  Alarm.Code := 4105; Alarm.Active := TRUE; Alarm.Latched := TRUE;
  State := 900;
 END_IF
```

```
900:
  CmdClampClose := FALSE;
  CmdClampOpen := TRUE;
END_CASE
```

8 Facility “Circuit Diagram”: Safety Power and Interlock Topology

This diagram is an abstraction of the safety circuit architecture, showing dual-channel E-stop, safety PLC, safety outputs to STO and contactors, and feedback monitoring. It is not an electrical construction drawing, but a system topology suitable for engineering review.



9 Full I/O Specification (Representative, Extensible)

This section provides a representative I/O list sufficient to implement the entire pipeline. The list is structured by assembly and device class. Expand by cloning rows per lane/station.

A9 Safety I/O (Safety PLC)

Tag	Type	Description
A9::Safety::ESTOP::CH1	DI(S)	Emergency stop channel 1 loop healthy.
A9::Safety::ESTOP::CH2	DI(S)	Emergency stop channel 2 loop healthy.
A9::Safety::GATE::CH1	DI(S)	Gate interlock channel 1 healthy.
A9::Safety::GATE::CH2	DI(S)	Gate interlock channel 2 healthy.
A9::Safety::LCURTAIN::OK	DI(S)	Light curtain clear.
A9::Safety::K1::FB	DI(S)	Safety contactor feedback (aux NC).

A9::Safety::STO::A7::CH1	DO(S)	STO channel 1 enable for Centerfuge drives.
A9::Safety::STO::A7::CH2	DO(S)	STO channel 2 enable for Centerfuge drives.
A9::Safety::ZONE::A1::EN	DO(S)	Zone enable for overhead lanes.
A9::Safety::ZONE::A5::EN	DO(S)	Zone enable for cutting/dismantling cell.

A7 Centerfuge Process I/O (Standard PLC)

Tag	Type	Description
A7::Centerfuge::Rotor::SpeedRPM	AI	Rotor speed feedback (RPM).
A7::Centerfuge::Rotor::SpeedSP	AO	Rotor speed setpoint (RPM).
A7::Centerfuge::Rotor::Vibration	AI	Vibration (mm/s or g).
A7::Centerfuge::Rotor::ImbalanceOK	DI	Imbalance relay OK.
A7::Centerfuge::Sieve::DeltaP	AI	Differential pressure across sieve manifold.
A7::Centerfuge::Inlet::Flow	AI	Inlet slurry flow (L/min).
A7::Centerfuge::Inlet::ValveCmd	DO	Inlet valve open/close.
A7::Centerfuge::Discharge::Frac1Cmd	DO	Fraction 1 discharge gate.
A7::Centerfuge::Discharge::Frac2Cmd	DO	Fraction 2 discharge gate.
A7::Centerfuge::Discharge::Frac3Cmd	DO	Fraction 3 discharge gate.
A7::Centerfuge::Pump::WashCmd	DO	Wash pump start/stop.

10 Implementation Notes: "UML-to-PLC" Mapping

The PlantUML diagrams define authoritative behavior. Each state in the bot and station state machines maps to a PLC integer state variable with explicit entry/exit actions. Each activity diagram transition maps to a routing predicate implemented as a PLC function returning a Boolean permissive. Each component in the deployment diagram maps to a PLC program organization unit (POU) with a corresponding IO map and function-block API.

The facility achieves determinism by constraining all real-time sequencing to PLC scan cycles. Supervisory systems provide recipes and setpoints, but do not manipulate safety interlocks or time-critical sequencing.

11 Appendix: Ladder-Logic Style Safety Permissive (Illustrative)

Listing 11: Illustrative LD semantics (expressed textually): Zone Enable

```
(* Equivalent to a rung: ZoneEnable = EstopOK AND GatesOK AND
   LightCurtainOK AND K1_FB *)
ZoneEnable := EstopOK AND GatesOK AND LightCurtainOK AND K1_FB;
Permissive_A1_Lanes := ZoneEnable;
Permissive_A2_Docks := ZoneEnable;
Permissive_A5_Cutting := ZoneEnable AND CuttingCellGuardsOK;
```

```
Permissive_A7_Rotor := ZoneEnable AND CenterfugeGuardsOK;
```

A Vendor-Specific Implementation Extensions (TIA Portal, Rockwell Studio 5000, Beckhoff TwinCAT, CODESYS)

This appendix extends the generic UML/PLC specification into vendor-realistic build packages. The goal is not to lock VineRail to a single ecosystem, but to demonstrate that the same architectural semantics map cleanly into the dominant industrial stacks. Each subsection provides a concrete control project skeleton, naming conventions, type definitions, program organization, and I/O mapping patterns consistent with certification workflows.

A.1 Siemens TIA Portal (S7-1500 / S7-1200) with PROFIsafe

The Siemens implementation is organized around a central S7-1500 PLC per major assembly cell, with ET200SP distributed I/O over PROFINET and PROFIsafe for safety I/O. Safety logic is implemented in an F-CPU (or a dedicated F-PLC) and exported to standard PLC programs via read-only safety status DBs. Motion and drives (Centerfuge rotor, wash pumps, conveyors) are controlled via SINAMICS drives where applicable, using standard telegrams on PROFINET with safe torque off enforced by PROFIsafe and hardwired safety contactors where required by risk assessment.

TIA Project Structure

The project is divided into PLC programs corresponding to assemblies A1–A8, with a safety program A9. Each assembly program exposes a primary data block (DB) containing commands, status, alarms, and diagnostics. Inter-assembly signals are exchanged via explicit interface DBs rather than global tags, enabling auditability and minimizing implicit coupling.

Data Types (UDTs) and Data Blocks (DBs)

Listing 12: TIA Portal: UDTs for Commands/Status/Alarms (SCL-style)

```
TYPE UDT_Alarm :
STRUCT
  Code : DINT;
  Severity : USINT; // 0=INFO,1=WARN,2=ALARM,3=TRIP
  Active : BOOL;
  Latched : BOOL;
END_STRUCT
END_TYPE

TYPE UDT_DeviceStatus :
STRUCT
  Ready : BOOL;
```

```
  Running : BOOL;
  Fault : BOOL;
  Interlock : BOOL;
  Permissive : BOOL;
  AlarmWord : DWORD;
END_STRUCT
END_TYPE

TYPE UDT_ToolChangeCmd :
STRUCT
  ReqToolID : DINT;
  ReqExchange : BOOL;
  ResetFault : BOOL;
END_STRUCT
END_TYPE

TYPE UDT_ToolChangeSts :
STRUCT
  ToolID_Confirmed : DINT;
  ExchangeOK : BOOL;
  State : INT;
  Status : UDT_DeviceStatus;
  Alarm : UDT_Alarm;
END_STRUCT
END_TYPE
END_TYPE
```

Listing 13: TIA Portal: Example DB Layout (Dock Station DB)

```
DATA_BLOCK "DB_A2_Dock01"
{ S7_Optimized_Access := 'TRUE' }
VAR
  Cmd : UDT_ToolChangeCmd;
  Sts : UDT_ToolChangeSts;

  IO_DI_BotDocked : BOOL;
  IO_DI_ToolPresent : BOOL;
  IO_DI_ClampClosed : BOOL;
  IO_DI_ClampOpen : BOOL;
  IO_AI_ToolID_Read : DINT;

  IO_DO_ClampClose : BOOL;
  IO_DO_ClampOpen : BOOL;
END_VAR
END_DATA_BLOCK
```

Function Blocks (FB) and Instance DBs

In TIA, each major function block (FB) is instantiated per station/lane with an instance DB. The state machines from the PlantUML states map directly to FB internal state integers with explicit transition guards. Alarms are raised only at transition points and latched until reset to match industrial expectation.

Listing 14: TIA Portal: FB$_{ToolChangeStation}(SCLpseudo-implementationskeleton)$

```
FUNCTION_BLOCK "FB_ToolChangeStation"
VAR_INPUT
  ModeAuto : BOOL;
  SafetyPerm : BOOL;
  BotDocked : BOOL;
  ToolPresent : BOOL;
  ToolID_Read : DINT;
  ReqToolID : DINT;
  MagazineReady : BOOL;
  ClampClosedFB : BOOL;
  ClampOpenFB : BOOL;
END_VAR
VAR_OUTPUT
  CmdClampClose : BOOL;
  CmdClampOpen : BOOL;
  ExchangeOK : BOOL;
  ToolID_Confirmed: DINT;
  State : INT;
  Fault : BOOL;
END_VAR
VAR
  AlarmCode : DINT;
  AlarmLatched : BOOL;
END_VAR

BEGIN
  IF NOT SafetyPerm OR NOT ModeAuto THEN
    State := 0;
    CmdClampClose := FALSE;
    CmdClampOpen := TRUE;
    ExchangeOK := FALSE;
    RETURN;
  END_IF;

  CASE State OF
    0:
      ExchangeOK := FALSE;
      Fault := FALSE;
```

```
    IF BotDocked AND MagazineReady AND (ReqToolID <> 0) THEN
      State := 10;
    END_IF;

  10:
    IF BotDocked THEN
      State := 20;
    END_IF;

  20:
    CmdClampOpen := TRUE;
    CmdClampClose := FALSE;
    IF ClampOpenFB THEN
      State := 30;
    END_IF;

  30:
    IF ToolPresent THEN
      State := 40;
    END_IF;

  40:
    CmdClampOpen := FALSE;
    CmdClampClose := TRUE;
    IF ClampClosedFB THEN
      State := 50;
    END_IF;

  50:
    ToolID_Confirmed := ToolID_Read;
    IF ToolID_Confirmed = ReqToolID THEN
      ExchangeOK := TRUE;
      State := 0;
    ELSE
      AlarmCode := 4105;
      AlarmLatched := TRUE;
      Fault := TRUE;
      State := 900;
    END_IF;

  900:
    CmdClampClose := FALSE;
    CmdClampOpen := TRUE;
    // Remain here until external reset sets ReqToolID=0 or resets
       AlarmLatched.
END_CASE;
```

```
END_FUNCTION_BLOCK
```

Safety (F-PLC) Pattern

The safety program computes zone enables and drive STO enables and exposes them as a read-only structure to standard PLC logic. This maps cleanly to the generic `FB_SafetyGateway`. The recommended Siemens pattern is dual-channel devices into F-DI modules, F-logic using standard Siemens safety blocks, and F-DO outputs to PROFIsafe STO and safety contactors with feedback.

A.2 Rockwell (Allen-Bradley) Studio 5000 (ControlLogix/CompactLogix) with CIP Safety

The Rockwell implementation organizes logic as programs per assembly with UDTs (User-Defined Types), AOIs (Add-On Instructions) for function blocks, and produced/consumed tags for inter-PLC exchange where multiple controllers are used. Safety uses GuardLogix with CIP Safety for safe I/O and safe drives, with additional hardwired safety contactors when required.

UDTs and AOIs

The UDT pattern mirrors the generic spec: command UDT, status UDT, and alarm UDT. Each station or lane owns an instance tag of its UDTs. AOIs encapsulate state machines and expose a consistent rung-level interface.

Listing 15: Rockwell: UDT definitions (conceptual)

```
UDT Alarm_t
  DINT Code;
  SINT Severity; // 0..3
  BOOL Active;
  BOOL Latched;

UDT DeviceStatus_t
  BOOL Ready;
  BOOL Running;
  BOOL Fault;
  BOOL Interlocked;
  BOOL Permissive;
  DWORD AlarmWord;

UDT ToolChangeCmd_t
  DINT ReqToolID;
  BOOL ReqExchange;
  BOOL ResetFault;

UDT ToolChangeSts_t
  DINT ToolID_Confirmed;
  BOOL ExchangeOK;
```

```
 DINT State;
 DeviceStatus_t Status;
 Alarm_t Alarm;
```

Listing 16: Rockwell: AOI ToolChangeStation (interface contract)

```
AOI ToolChangeStation_AOI
Inputs:
 BOOL AutoMode;
 BOOL SafetyPermissive;
 BOOL BotDocked;
 BOOL ToolPresent;
 DINT ToolID_Read;
 DINT ReqToolID;
 BOOL MagazineReady;
 BOOL ClampClosedFB;
 BOOL ClampOpenFB;

Outputs:
 BOOL CmdClampClose;
 BOOL CmdClampOpen;
 BOOL ExchangeOK;
 DINT ToolID_Confirmed;
 DINT State;
 Alarm_t Alarm;
```

Safety Integration

A best-practice pattern is to treat safety outputs (zone enables, STO enables, contactor coils) as *safe tags* originating in the safety task and only mirrored to standard logic for permissive display. The standard logic never energizes safety outputs directly; it requests enable states via non-safe tags that the safety task may grant only when safe conditions are true.

A.3 Beckhoff TwinCAT 3 (IEC 61131-3) with Safety over EtherCAT (FSoE)

TwinCAT maps particularly well to the function-block architecture because it is natively IEC 61131-3 with strong struct typing. The recommended implementation uses a PLC project per controller, a Global Variable List (GVL) defining I/O mappings, and function blocks instantiated per lane/station/cell. Safety is implemented on TwinSAFE terminals using FSoE with TwinSAFE logic, exporting safe status to the PLC for display and permissives.

GVL and FB Structure

Listing 17: TwinCAT: GVL for I/O mapping (conceptual)

```
VAR_GLOBAL
  (* Safety status mirrored from TwinSAFE *)
  gEstopOK : BOOL;
  gGatesOK : BOOL;
  gZoneEnable_A1 : BOOL;
  gZoneEnable_A7 : BOOL;

  (* Dock station I/O *)
  gDock01_BotDocked : BOOL;
  gDock01_ToolPresent : BOOL;
  gDock01_ClampClosedFB : BOOL;
  gDock01_ClampOpenFB : BOOL;
  gDock01_ToolID_Read : DINT;

  gDock01_CmdClampClose : BOOL;
  gDock01_CmdClampOpen : BOOL;
END_VAR
```

Listing 18: TwinCAT: FB signature aligns with generic spec

```
FUNCTION_BLOCK FB_ToolChangeStation
VAR_INPUT
  bAuto : BOOL;
  bSafetyPerm : BOOL;
  bBotDocked : BOOL;
  bToolPresent : BOOL;
  nToolID_Read : DINT;
  nReqToolID : DINT;
  bMagazineReady : BOOL;
  bClampClosedFB : BOOL;
  bClampOpenFB : BOOL;
END_VAR
VAR_OUTPUT
  bCmdClampClose : BOOL;
  bCmdClampOpen : BOOL;
  bExchangeOK : BOOL;
  nToolID_Confirmed : DINT;
  nState : INT;
END_VAR
END_FUNCTION_BLOCK
```

Determinism and Scan Semantics

TwinCAT allows explicit task cycle times. The intended pattern is a fast task for motion/drives and a standard task for state machines, while safety remains in TwinSAFE. This cleanly enforces the “PLC owns sequencing, safety owns permissives” rule from the main spec.

A.4 CODESYS (IEC 61131-3) with Vendor Safety

CODESYS implementations follow the same structure as TwinCAT but may be deployed on multiple hardware vendors (WAGO, Beckhoff runtime, Schneider variants, etc.). The recommended approach is to keep the same FB APIs, create per-assembly POUs, and treat safety permissives as read-only inputs exported by the safety system (vendor safety PLC or safety relay network).

B Concrete "Build Package" Artifacts

For any vendor, the deliverables that correspond to this specification are the same in spirit.

A complete build package consists of the PlantUML sources as authoritative behavior documentation; a tag dictionary exported from the PLC environment; per-assembly program modules; per-station instance data; an alarm catalogue mapping codes to text and severity; and a safety validation bundle consisting of wiring diagrams, device lists, and safety function test results. The UML state machine identifiers should match PLC state enumerations exactly, enabling straightforward traceability during commissioning and incident review.

C Facility-Wide "Circuit" Detailing: STO, Contactors, Feedback, and Zones

To extend the circuit diagram into a commissioning-ready topology, the facility is divided into safety zones, each with its own enable chain. The recommended arrangement is that A1 lanes, A2 docks, A5 cutting cells, and A7 centerfuge each constitute separate zones. Each zone has an independent STO enable output, a zone contactor pair if required by risk assessment, and a feedback loop verifying that contactors and STO channels are in the commanded state. Standard PLCs may request zone enable but cannot override safety PLC decisions.

The primary invariants are implemented at the safety layer as: emergency stop integrity, guard door integrity, presence sensing integrity, and feedback monitoring integrity. Only when all invariants are satisfied does the safety PLC assert the appropriate zone enable outputs. Standard PLCs treat these outputs as permissive inputs and must fail-closed by inhibiting actuation on any loss of permissive.

D Vendor-Specific I/O Export Patterns

In Siemens TIA, all I/O is mapped into optimized DBs or PLC tags and then aliased into assembly DB structures. In Rockwell, tags are scoped per program with alias tags for I/O modules, and UDT instances bind to tags by name. In TwinCAT, I/O mapping occurs through the I/O tree to GVL variables, which are then referenced by FB instances. In CODESYS, I/O mapping binds physical channels to global variables and is then encapsulated into FBs.

E Optional Extension: Drive Integration for Centerfuge (Generic Telegram Model)

For the centerfuge rotor, the process PLC controls a VFD with a speed setpoint and speed feedback. The speed control loop may be internal to the drive with PLC setpoint supervision, or PLC-based with a PID block. Safety torque off remains controlled exclusively by the safety system. Rotor enable in standard PLC logic is thus an operational command that is effective only when safety permissive is asserted. Vibration and imbalance signals are treated as hard interlocks that force controlled ramp-down and discharge inhibition.

F How to Convert This Appendix into a Real Project

If you choose a single vendor as your baseline, this appendix can be expanded into a literal project export: a Siemens TIA Portal folder layout with DB/FB numbering and comment headers, or a Rockwell Studio 5000 ACD design with AOIs and UDTs, or a TwinCAT solution with GVLs and FB libraries. The PlantUML text remains unchanged and serves as the behavioral contract. The PLC code is then forced to conform, state-by-state, to the UML, giving you a traceable "spec-to-implementation" pipeline.