# PLC Code Generator - Design Document

## Overview

Desktop application that generates Rockwell Automation Studio 5000 ladder logic code (.L5K files) for conveyor systems.

**Tech Stack**: Python 3 + Tkinter GUI

## Architecture

```
GUI (gui.py)
    ↓
Conveyor Generator (generator.py)
    ↓
Section Classes → Generate Logic + Tags
    ↓
.L5K File Output
```

## Core Components

### Data Models

- **Context**: Stores `conveyor_name`, `num_conveyors`, `has_makeup_unit`
- **RoutineOutput**: Contains `name`, `logic`, and `tags` for each section

## Section Classes

Each section generates specific fault detection logic:

1. **PeJamSection** - Photo-eye jam detection with timers
2. **MotorFaultSection** - VFD motor fault monitoring
3. **DiscFaultSection** - Disconnect switch monitoring
4. **EStopFaultSection** - Emergency stop button logic

## Conveyor Class

- Coordinates all sections
- Deduplicates tags
- Assembles final L5K file with header/footer

## User Interface

```
┌──────────────────────────────────┐
│ Controller Name:    [_______]    │
│ Conveyor Name:      [_______]    │
│ Number of Conveyors:[_______]    │
│ ☐ Has Makeup Unit?                │
│ Filepath:           [_____] 📂   │
│                  [Generate]      │
└──────────────────────────────────┘
```

**Workflow**:

1. User fills inputs
2. Validates (non-empty, valid integer)
3. Calls `Conveyor.generate_plc_code_full()`
4. Writes to file
5. Shows success message

## Generated File Structure

```
Static Header (controller config, modules)
    ↓
Global TAG Block (all tags)
    ↓
PROGRAM Block
    ├── PE_Jam_Faults Routine
    ├── Motor_Faults Routine
    ├── Disc_Faults Routine
    ├── EStop_Faults Routine
    └── MainRoutine (calls all via JSR)
    ↓
Static Footer (task config)
```

## Key Logic Patterns

### Per-Conveyor Generation

```
for i in range(1, num_conveyors + 1):
    # Generate tags: F_LineA_01_PE_JAM, F_LineA_02_PE_JAM, etc.
    # Generate logic for each conveyor
```

### Tag Naming Convention

- `F_` = Fault flag
- `I_` = Input
- Format: `{Type}_{ConveyorName}_{Number}_{Element}`
- Example: `F_LineA_01_PE_JAM`

## Hardware Configuration

- Processor: 1756-L81E (ControlLogix)
- Ethernet Module: 1756-EN2T
- Timers: 2000ms delay for PE jam and motor faults
- Task Rate: 10ms continuous

## Adding New Sections

1. Create class inheriting from `Section`
2. Implement `render(ctx: Context) -> RoutineOutput`
3. Add to `Conveyor.__init__()` sections list

## Error Handling

- Empty field validation
- Integer conversion validation
- File write exception handling
- Error dialogs for all failures