PLC Review

• Standard Symbols
  – Normally Open relay:
  – Normally Closed relay:

• Simple Logic:
  – AND
  – OR
  – NOT
PLC Programming

• Simple Combinatorial Logic
• State logic -- latching
• Sequential logic

Simple Combinatorial Logic

• Key steps:
  – Identify inputs and outputs
  – For each output, determine expression for what makes it true
  – Construct ladder logic to reflect expression
Combinatorial Example

If “Oven On” switch is activated (S=1) and oven door closed (DLS=1) and Temperature below threshold (TLS = 0) then heat is on (H=1)

If oven is on (H=1)
  or ( temp > thresh (TLS = 1) and door is closed (DLS=1) )
then fan is on (F=1)

If oven is on (H=1)
  or light switch is on (LS=1)
then light is on (L=1)

State Logic -- Latching

- Key steps:
  - Identify inputs and outputs. Consider outputs to be state variables
  - For each state variable \( X \),
    - determine expression \( \text{Reset}_X \) of what makes state go false
    - determine expression \( \text{Set}_X \) of what makes state go true
  - Construct ladder logic to reflect the following expression:
    \[
    \overline{(\text{Reset}_X)} (\text{Set}_X + X) \rightarrow X
    \]
    (If you have difficulty converting \( \text{Reset}_X \) into \( \text{Reset}_X \), then you may use a different rung to calculate \( \text{Reset}_X \), then invert it in the rung for \( X \)
Example

Tank System:
• S1 is turned on whenever level is below LLS switch (LLS=0)
• S1 is turned off when level above ULS switch (ULS =1)

Example

• Motor Control
  – Turn on motor when On-switch is pressed
  – Turn off motor when Off-switch is pressed, or when temperature sensor is high.
Demorgan’s Laws

- Demorgan's laws are useful for negating an expression:

\[
\overline{A + B} = \overline{A} \cdot \overline{B} \\
\overline{A \cdot B} = A + \overline{B}
\]

- To describe a reset which is a negated expression, we can either apply Demorgan’s laws or we can create a new rung to create an intermediate signal.

---

Sequential Logic -- Cascade method

- Sequential Logic has steps.
  - The functions that drive the outputs differ depending upon which step is being done

- Use extra state variables to represent which step is active
  - Draw state diagram representing steps, indicating what causes changes in the states
  - Establish “latching” rung for each step

- Define Output rungs that depend on the steps
- Initialization rung may be required
Basic format of cascade method program

Cascade method: Revisit of Motor example

- Turn on motor when on-switch is pressed
- Turn off motor when off switch is pressed or when temperature sensor is high.
Example problem….

- Problem... what if temperature sensor is high and start switch is on?

Cascade Method: revisit of tank method

Tank System:
- Two states: filling and not filling
- Start filling whenever level is below LLS switch (LLS=0)
- Go to “not filling” state when level above ULS switch (ULS =1)
Sequential Logic -- Example

- An industrial oven has four states, Off, Preheat, Superheat, and CoolDown. The oven starts in the Off state. If start is pushed,
- then it enters the Preheat state and turns the heater on. It remains in the preheat state until the temperature sensor $T_1$ becomes true, at which time it then enters the Superheat state. In this state, it continues to heat, but locks the oven door. Once temperature sensor $T_2$ becomes true, then it moves to state CoolDown, where the heater turns off. The door remains locked, until the sensor $T_1$ becomes false, at which point the OFF state is entered and the door is unlocked.
- There is a stop switch also, but it only works during the preheat state, and returns the system to off.

Industrial Oven -- state diagram
Industrial Oven -- Ladder Program

Industrial oven – ladder continued
Cascade method critique

Cascade method:
• Advantages:
  – States are explicit – helps in debugging and maintenance.
  – Helpful when we have states without unique outputs. (example: supersuperheat).
  – Methodical and relatively simple.
• Disadvantages:
  – Program is larger than necessary in some cases
  – Extra state variables are bits in memory which may be limited resource.
  – May depend on operation method of PLC and sequence of rungs (depends on PLC mfg.)
    • (if updates states while running program, then could “lose” state.)

More advanced structures

• Concurrent parallel paths:
Built-in Sequencer Functions

Timers

Example in SLC500 (Allen-Bradley/Rockwell)

- Timer begins timing when rung goes true
- Timer always reset whenever timer times out
- Counts (using accum) to preset value
- Sets DN when Accum = preset
- Timing in 1 second or 100ths of second
Timer example: Buzzer

- If overtemperature sensor is high, then buzz for 15 seconds.

Timer example:

- Automatic lubrication system: after machine runs for fixed time, we activate a lube solenoid and reset the timer.

Note: T4:0 will be true for just one scan! (so solenoid must run on single pulse)
Counters

- Counter similar to timer.
- Increments counter on each false-to-true transition.

Counter example: Wrapper

- Count four products before activating wrapper.
- Ladder Diagrams: Advantages:
  - Graphical – easy for simple logic
  - Easy for maintenance and diagnosis
  - Language understandable by floor personnel

- Ladder Diagrams: Problems:
  - Note suited to structured programming
  - Poor reuse of logic
  - Poor data structuring
  - Limited support for complex sequencing
  - Limited execution control
  - Cumbersome arithmetic

- Efforts to offer better programming options: IEC 61131-3
  - Standard on programming languages for PLC’s

- IEC = “International Electrotechnical Commission”
- IEC 61131-3:
  - Encourages structured programming
  - Strong Data Typing
  - Execution control
  - Sequence control
  - Data Structures
  - Multiple languages defined
• Program Organization Unit (POU): Three types
  – Function
    • Traditional function – no internal data
  – Function Block
    • Like objects: data and operation
  – Program
    • Top level: accesses I/O, coordinates POU’s

• Code of POU is either:
  – Instruction list
  – Structured text
  – Ladder diagram
  – Function block diagram

• Pieces tied together using sequential function chart

---

**Sequential Function Charts**

Explicit representation of sequencing and concurrency
Ties together function blocks and other program elements into sequences.

Basic Elements:
- Steps: Active/Inactive
- Transitions
- Actions