

# PLC Programming Cook Book & Special Recipes

**Doug Norton & Currie Gardner**

ISA Meeting

May 31, 2007



*Setting the Standard for Automation™*

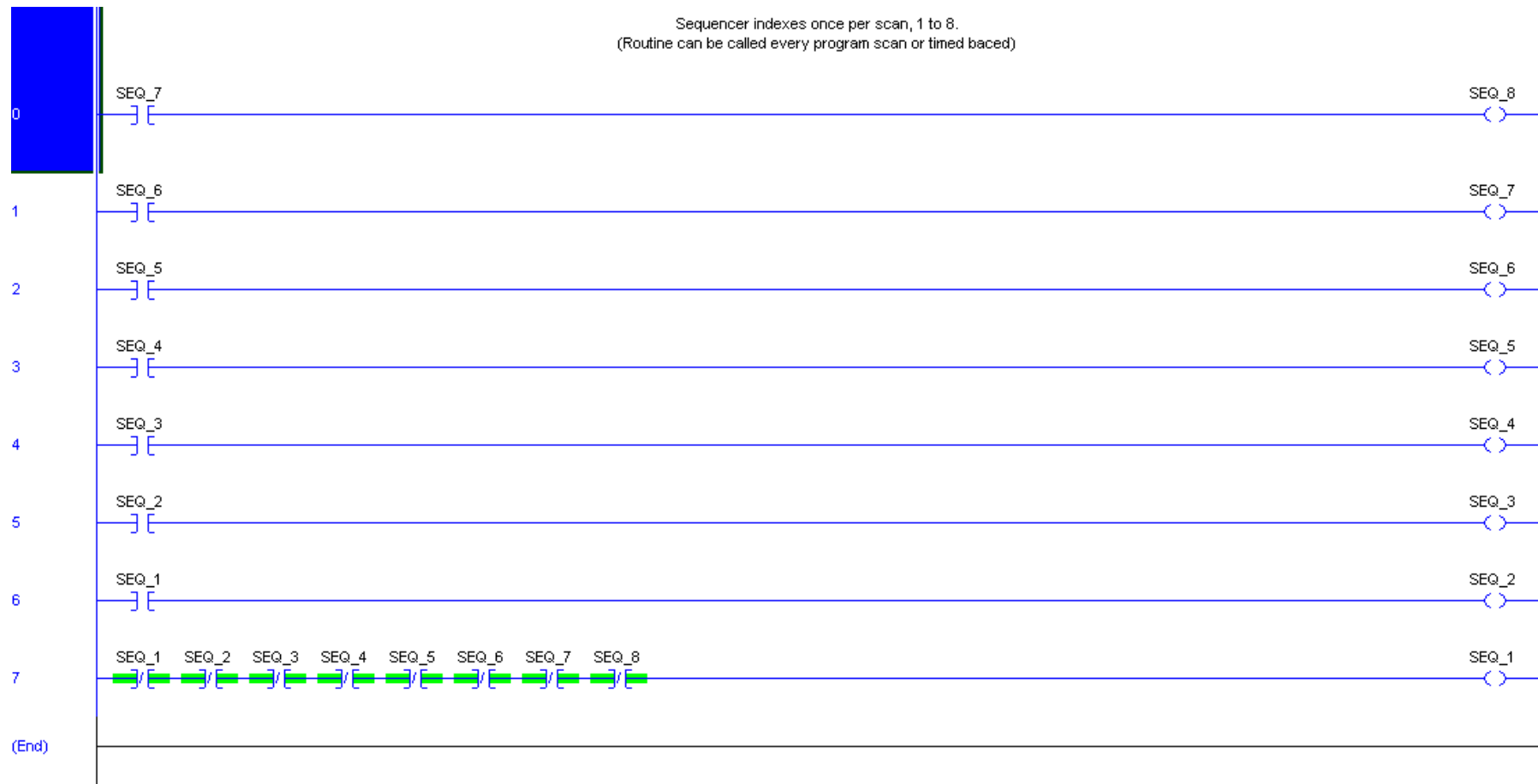


May 31, 2007

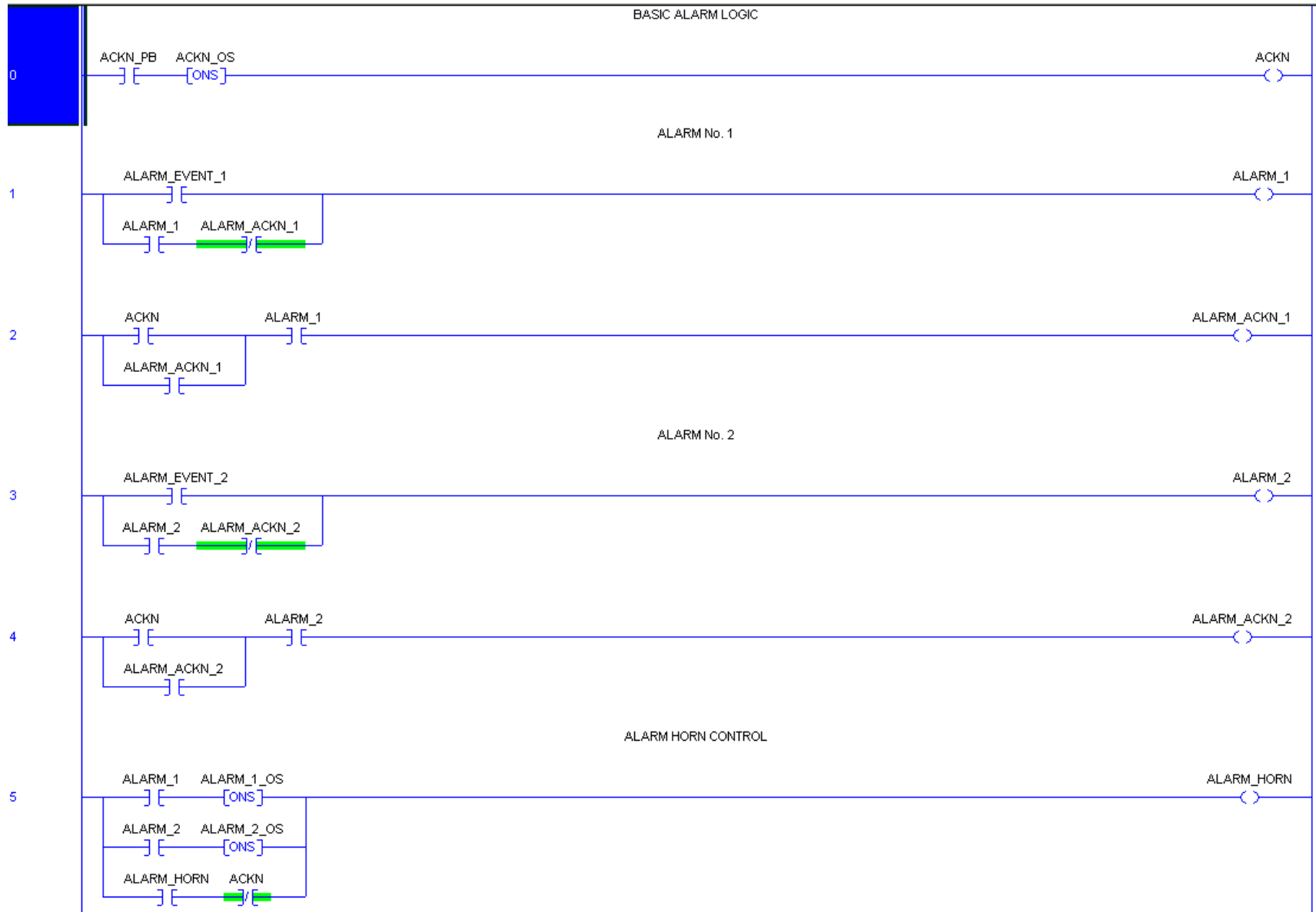
# Agenda

- Introduction
- Cook Book / Recipes
  - Basic Ladder Logic functions
    - Simple Sequencer
    - Basic Alarm Logic & Alarm Logic with FBs
    - Lamp Flasher
    - Process Alarms
    - Tank Volume Calculations
  - User Defined Function Blocks (DFBs)
  - Structured Variables / Tags & Arrays
    - Tank Level Scaling using Arrays & Structured Tags
    - Arrays / Structures in loop for calculations
  - Ladder Tank Level Control
- Questions & Discussion

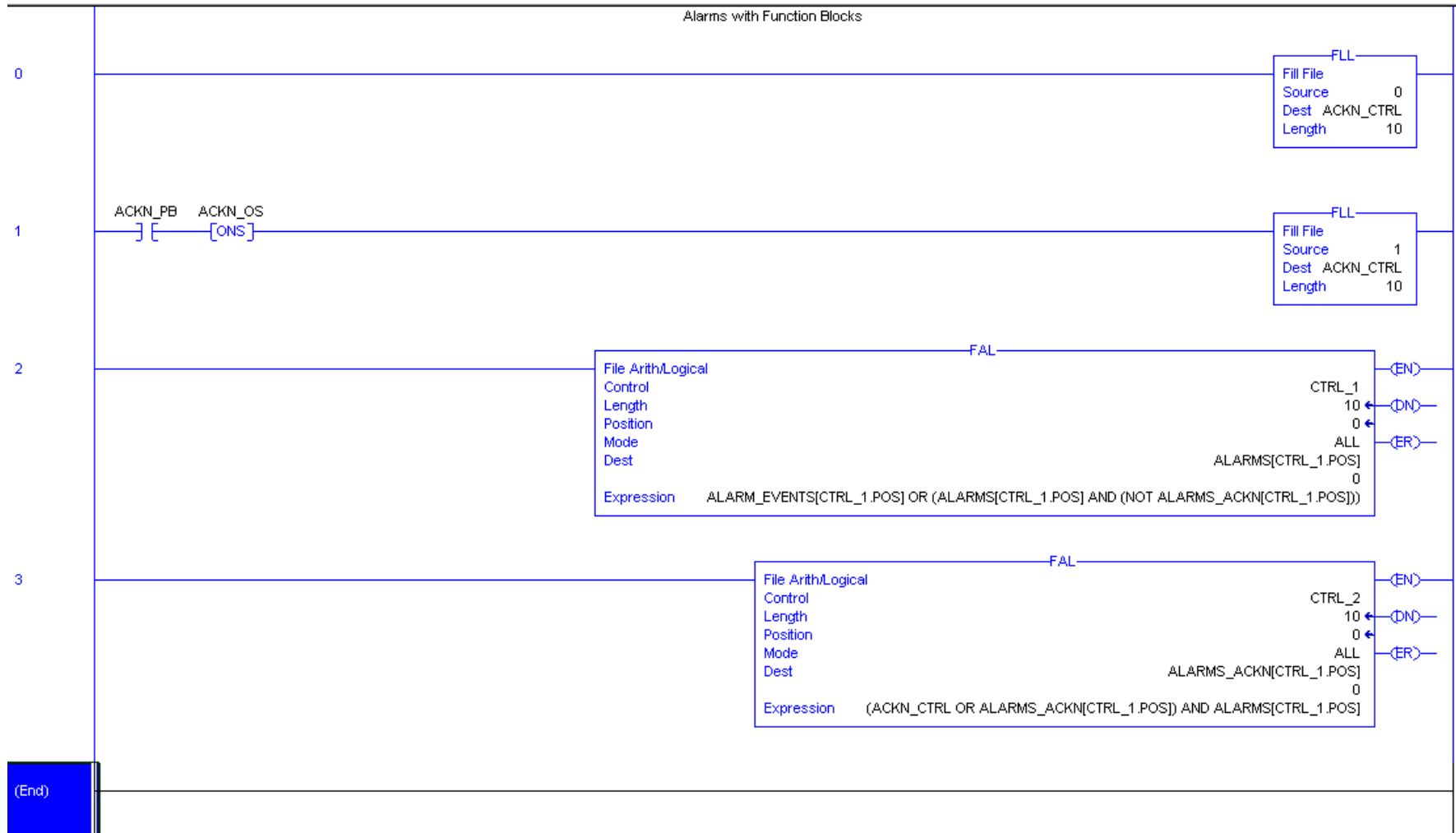
# Simple Eight Step Sequencer



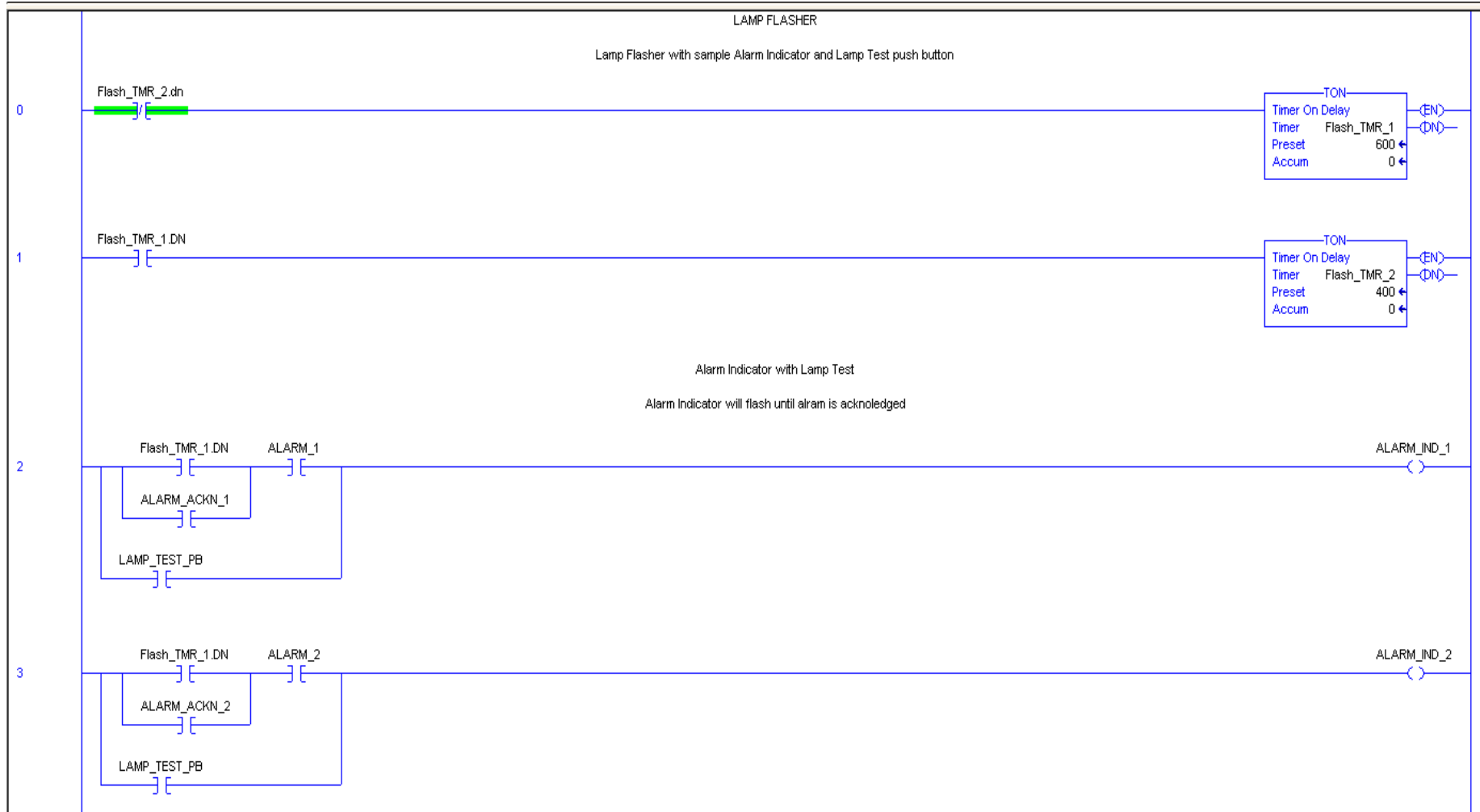
# Basic Alarm Logic



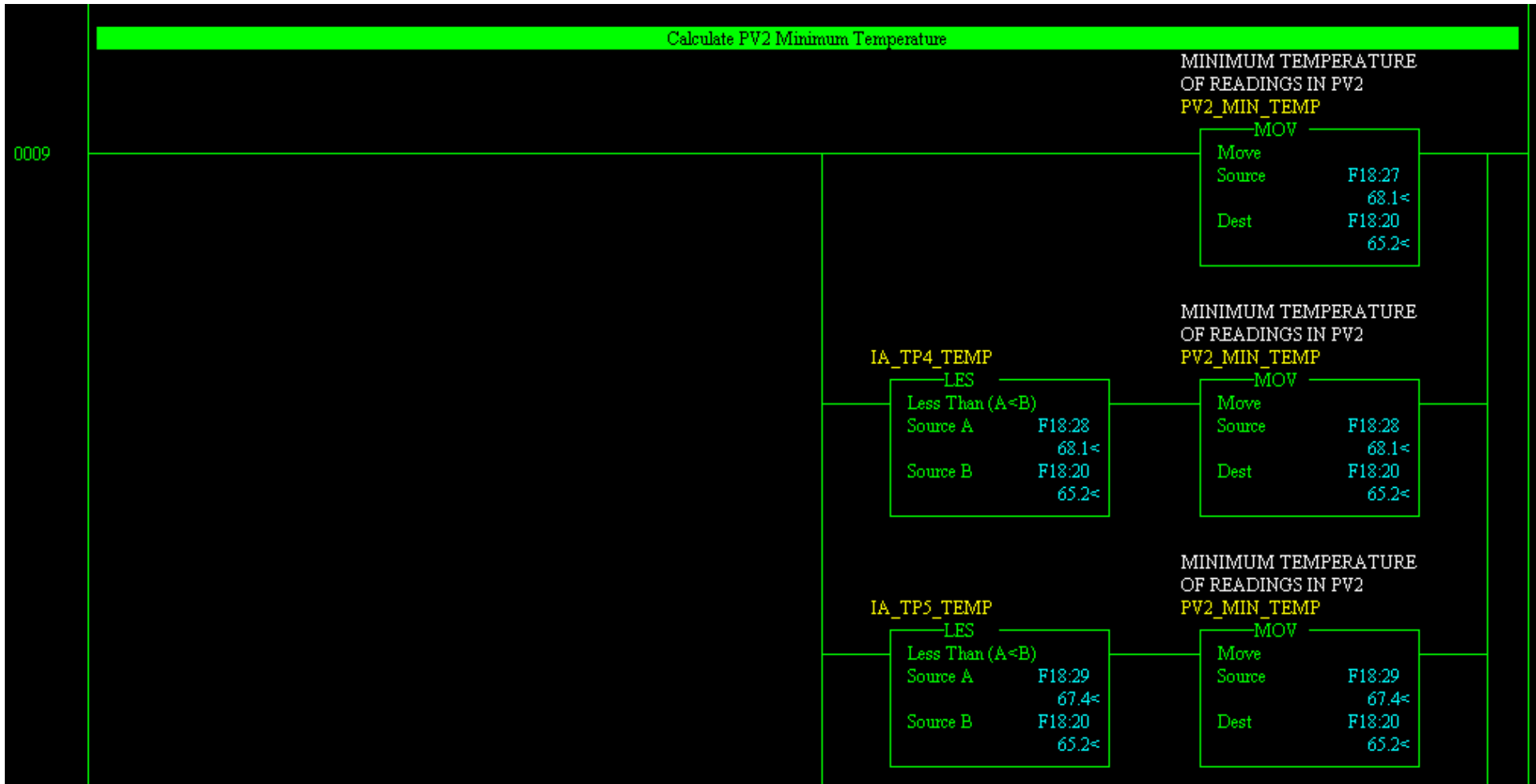
# Alarms with Function Blocks



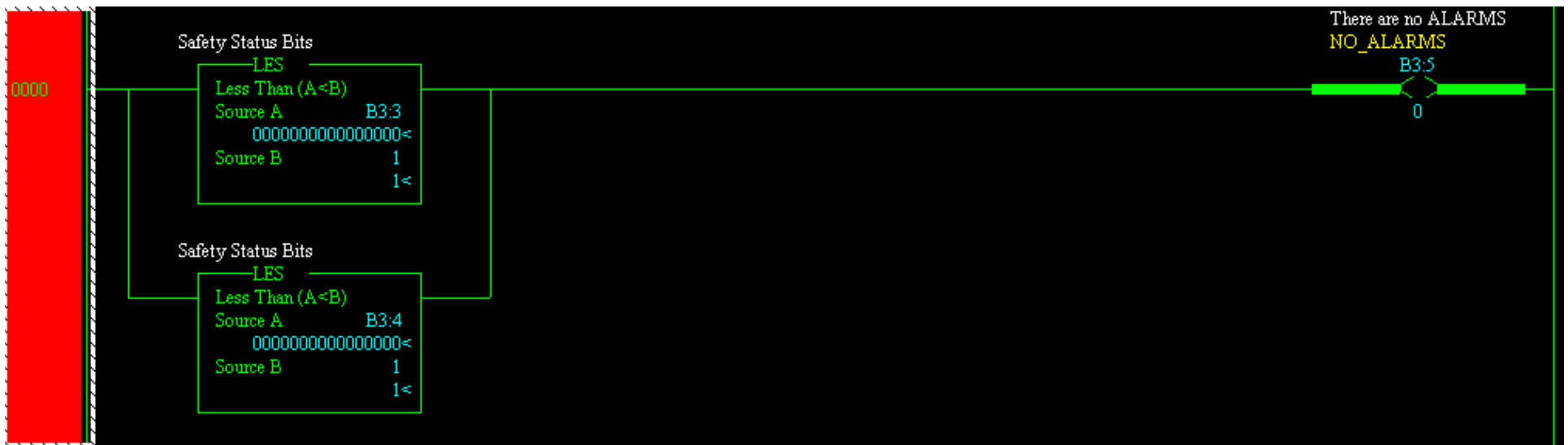
# Lamp Flasher & Alarm Indicator



# Minimum Value



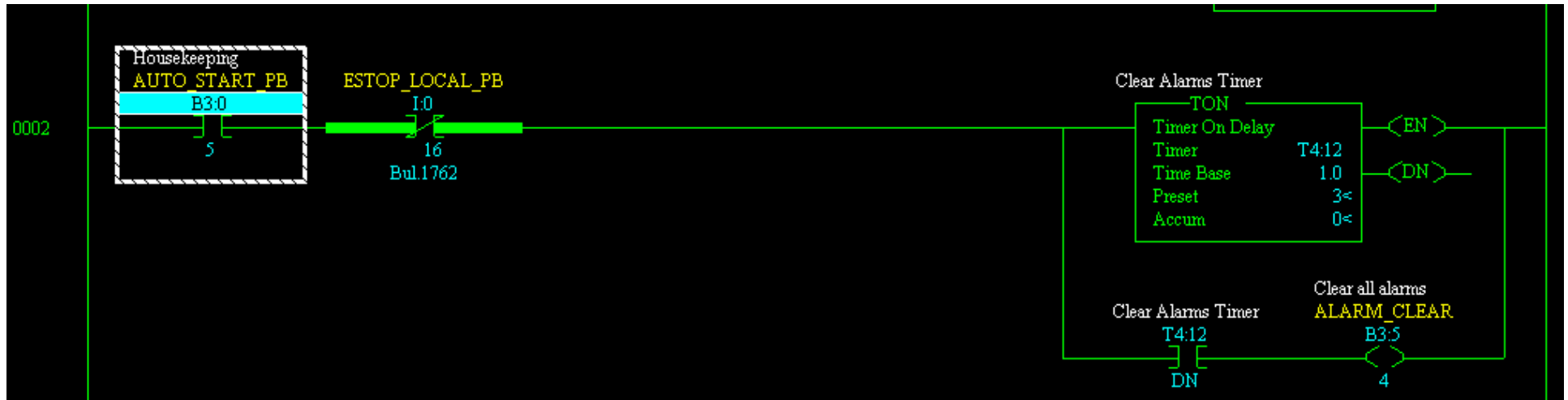
# Alarm Detection



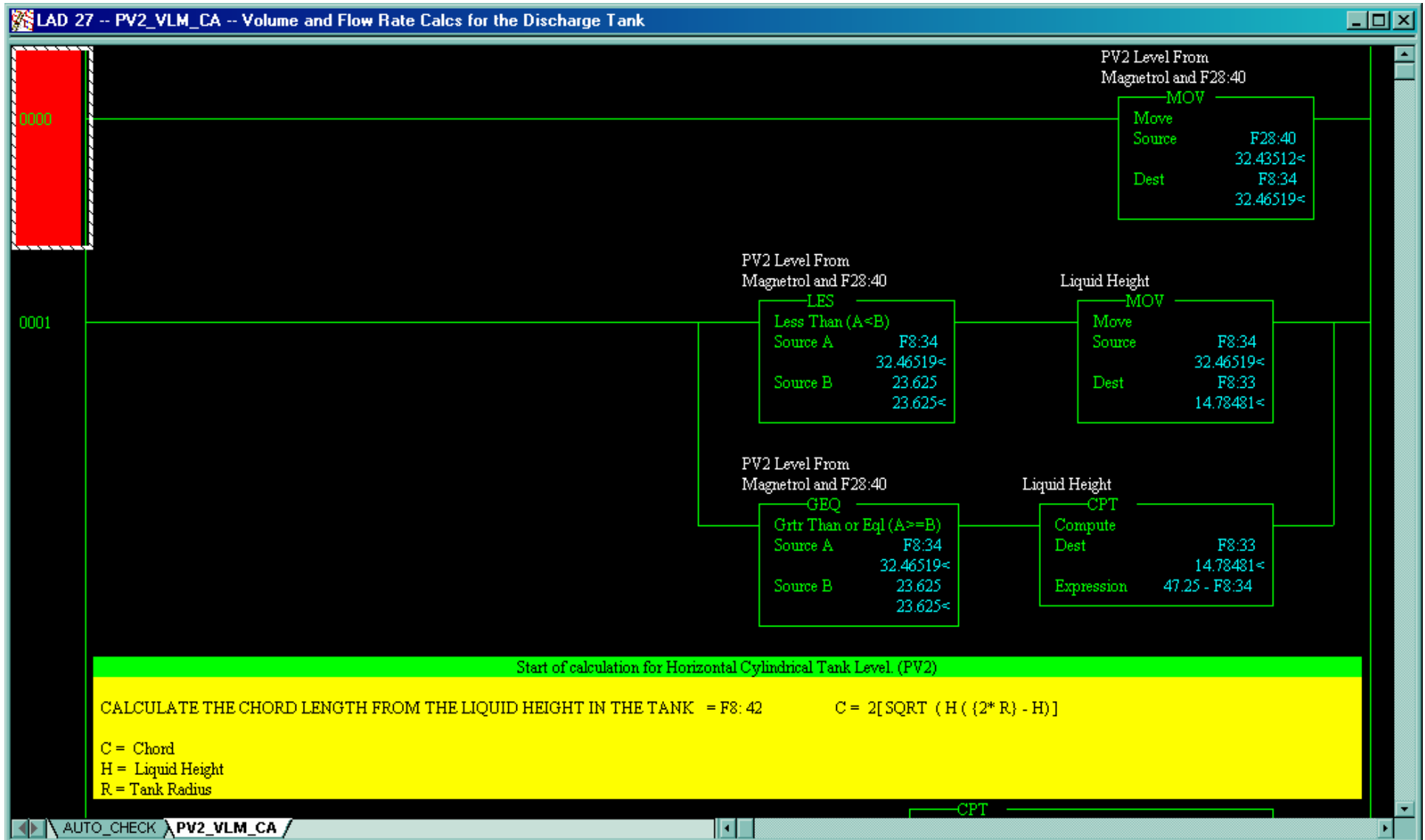
# Clear Alarms on First Pass



# More than one function per button



# Calculation of Volume of Horizontal Tank -1



# Tank continued -2

LAD 27 -- PV2\_VLM\_CA -- Volume and Flow Rate Calcs for the Discharge Tank

0002 R = Tank Radius

0003

CALCULATE THE ANGLE FORMED BY THE CHORD SEGMENT = F8:47      Angle = ARCOS (1 - (C SQR) / (2 \* R SQR))

C = Chord  
H = Liquid Height  
R = Tank Radius

0002

CPT  
Compute  
Dest F8:40  
479.9917<  
Expression F8:33 \* (( F8:31 \* 2.0 ) - F8:33 )

SQR  
Square Root  
Source F8:40  
479.9917<  
Dest F8:41  
21.90871<

CPT  
Compute  
Dest F8:42  
43.81743<  
Expression 2.0 \* F8:41

MUL  
Multiply  
Source A F8:42  
43.81743<  
Source B F8:42  
43.81743<  
Dest F8:43  
1919.967<

AUTO\_CHECK PV2\_VLM\_CA

# Tank -3

LAD 27 -- PV2\_VLM\_CA -- Volume and Flow Rate Calcs for the Discharge Tank

0003

CALCULATE THE ANGLE FORMED BY THE CHORD SEGMENT = F8:47      Angle = ARCOS (1 - (C SQR) / (2 \* R. SQR))

C = Chord  
H = Liquid Height  
R = Tank Radius

```

graph TD
    MUL1[MUL: Multiply] --> F8_43[F8:43]
    MUL2[MUL: Multiply] --> F8_44[F8:44]
    GRT[GRT: Greater Than A>B] --> F8_45[F8:45]
    CPT[CPT: Compute] --> F8_46[F8:46]
    ACS[ACS: Arc Cosine] --> F8_47[F8:47]
    DEG[DEG: Radians to Degrees] --> F8_47
  
```

MUL: Multiply  
Source A: F8:42, 43.81743<  
Source B: F8:42, 43.81743<  
Dest: F8:43, 1919.967<

MUL: Multiply  
Source A: F8:31, 23.625<  
Source B: F8:31, 23.625<  
Dest: F8:44, 558.1406<

GRT: Greater Than (A>B)  
Source A: F8:44, 558.1406<  
Source B: 0.0, 0.0<

CPT: Compute  
Dest: F8:45, -0.7199671<  
Expression: 1.0 - ( F8:43 / ( 2.0 \* F8:44 ) )

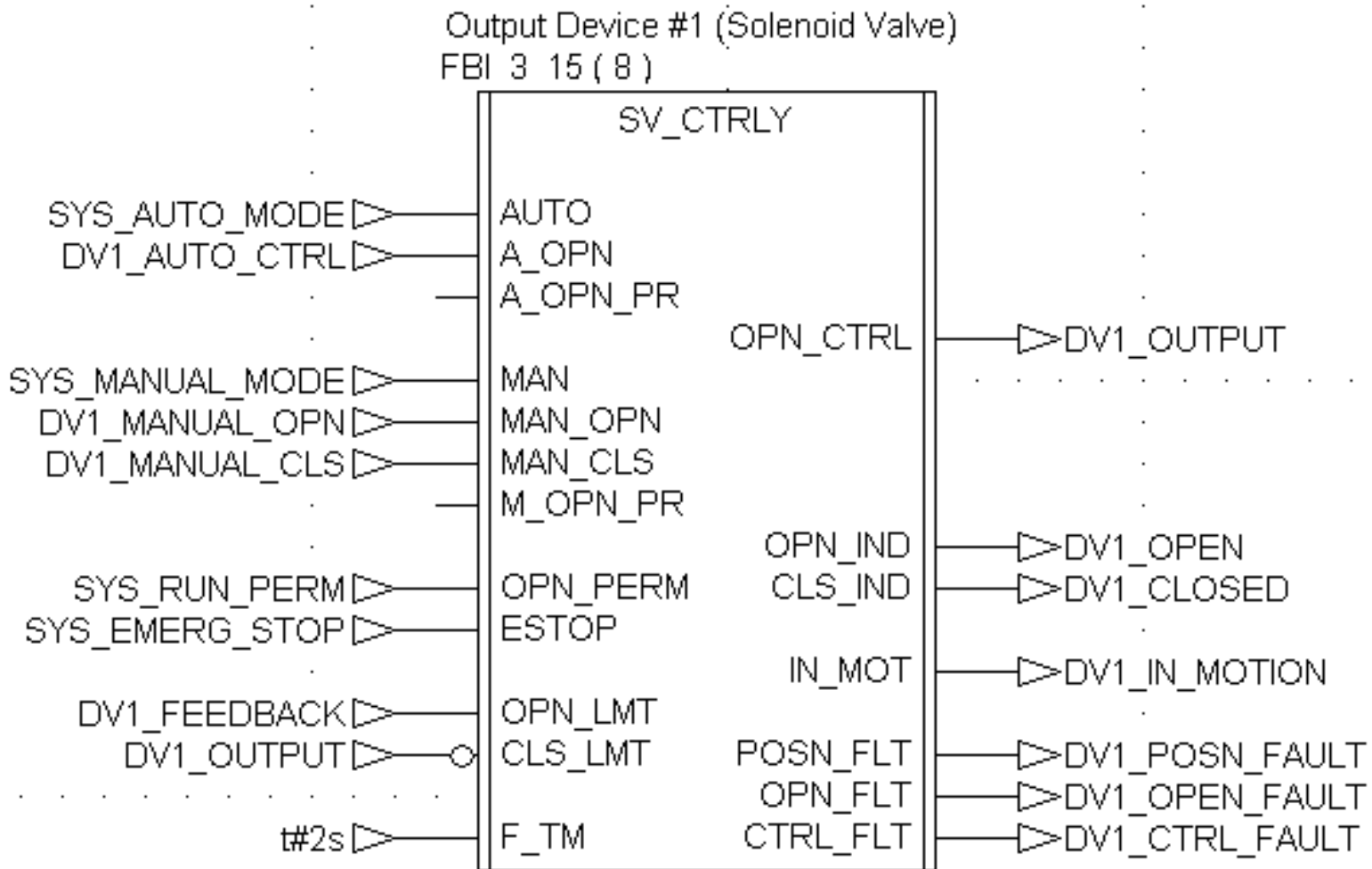
ACS: Arc Cosine  
Source: F8:45, -0.7199671<  
Dest: F8:46, 2.374551<

DEG: Radians to Degrees  
Source: F8:46, 2.374551<  
Dest: F8:47, 136.8118<

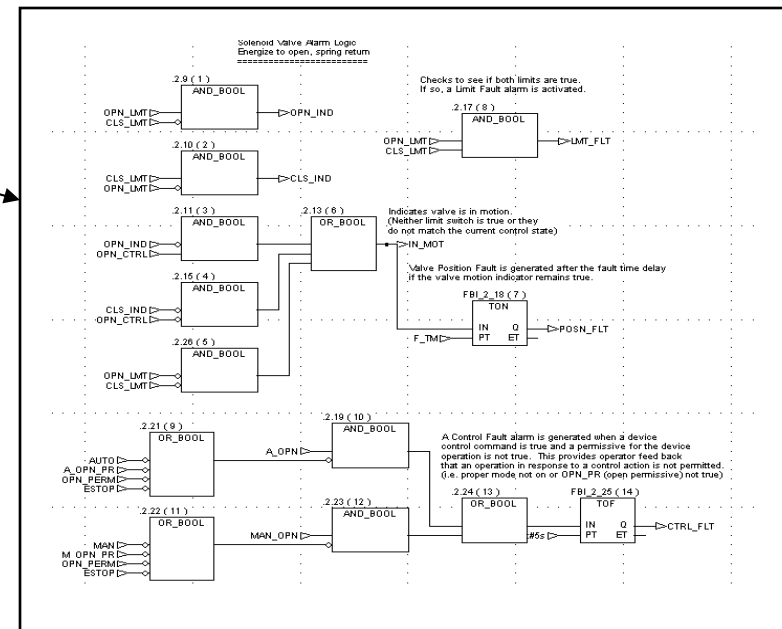
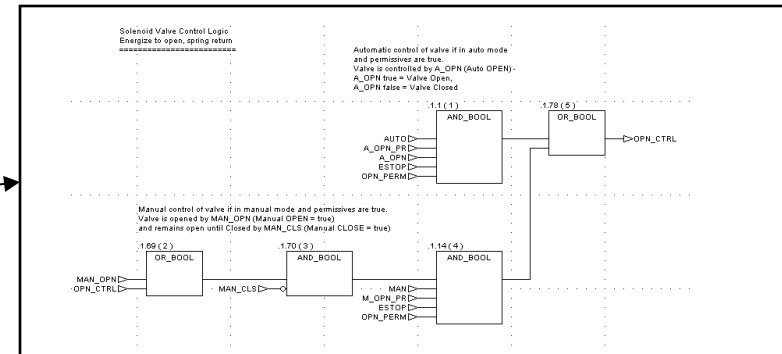
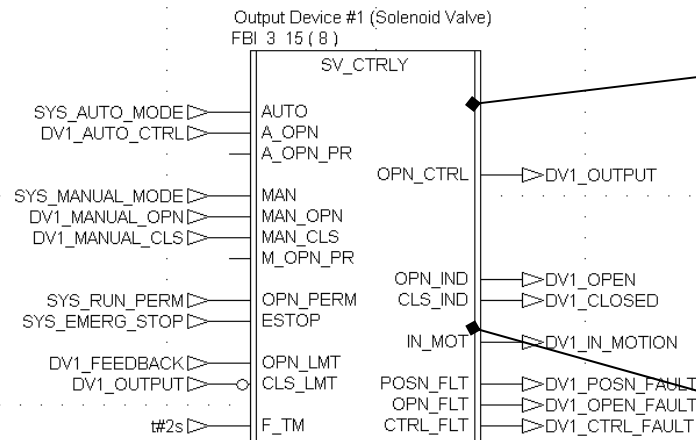
AUTO\_CHECK PV2\_VLM\_CA



# DFB - Solenoid Valve Control (single coil)



# DFB - Solenoid Valve Control (single coil)



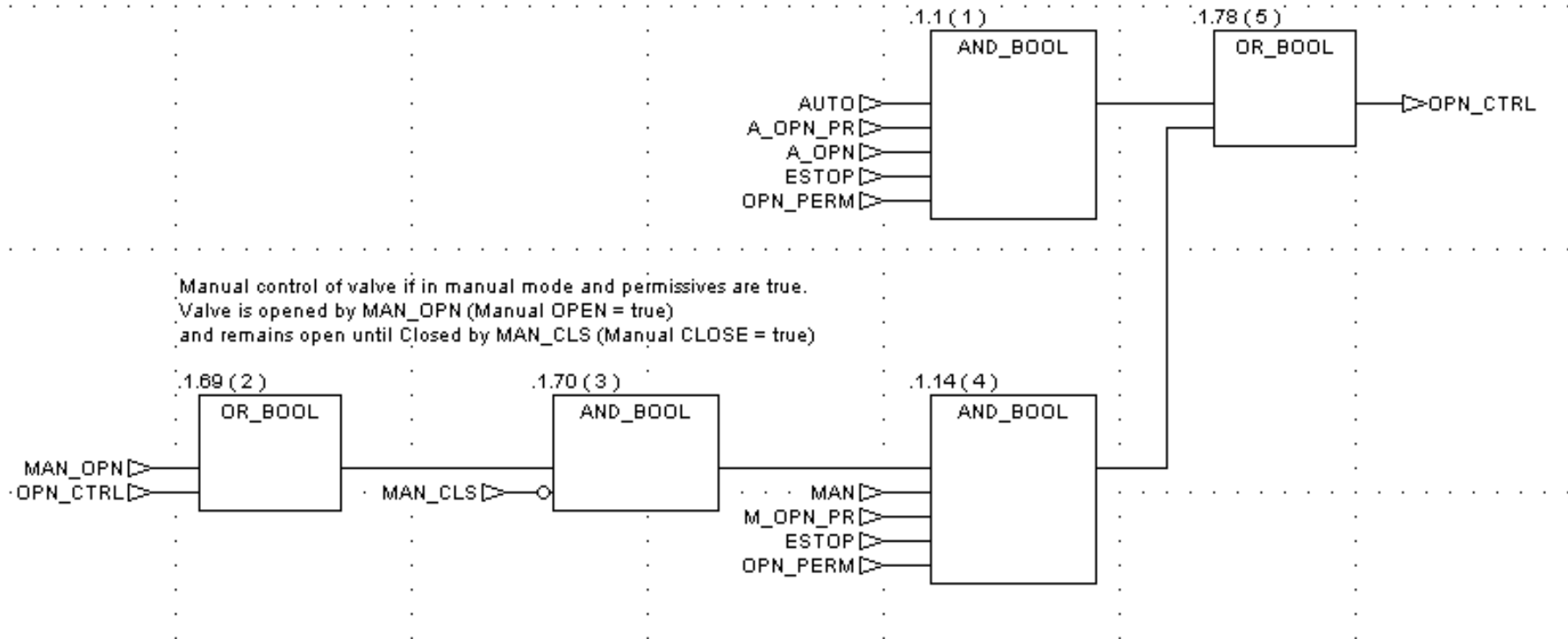
**A DFB can have many program sections of different language types and encapsulate standard repetitive logic**

# DFB - Solenoid Valve - Control Logic

Solenoid Valve Control Logic  
Energize to open, spring return  
=====

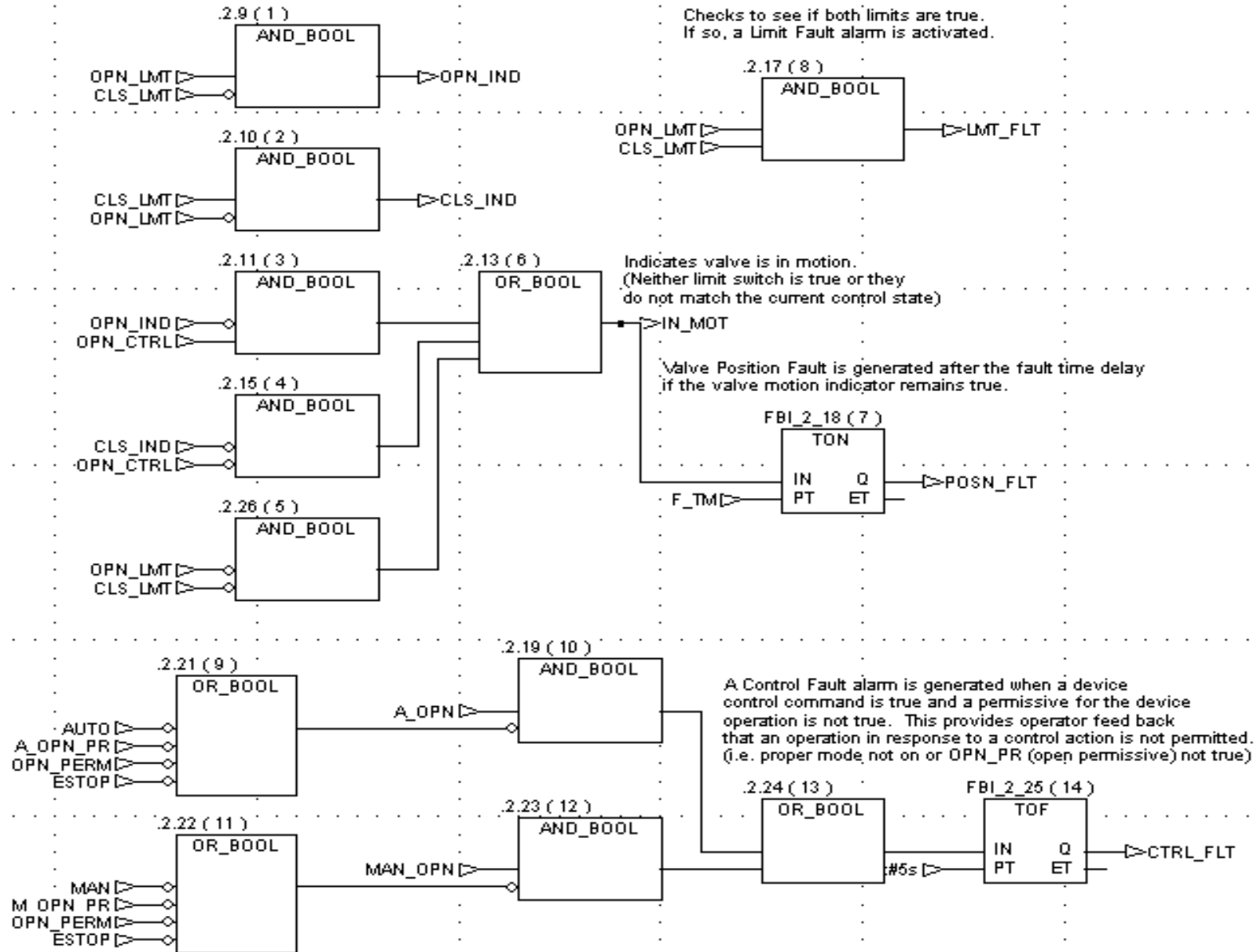
Automatic control of valve if in auto mode  
and permissives are true.  
Valve is controlled by A\_OPN (Auto OPEN) -  
A\_OPN true = Valve Open,  
A\_OPN false = Valve Closed

Manual control of valve if in manual mode and permissives are true.  
Valve is opened by MAN\_OPN (Manual OPEN = true)  
and remains open until Closed by MAN\_CLS (Manual CLOSE = true)



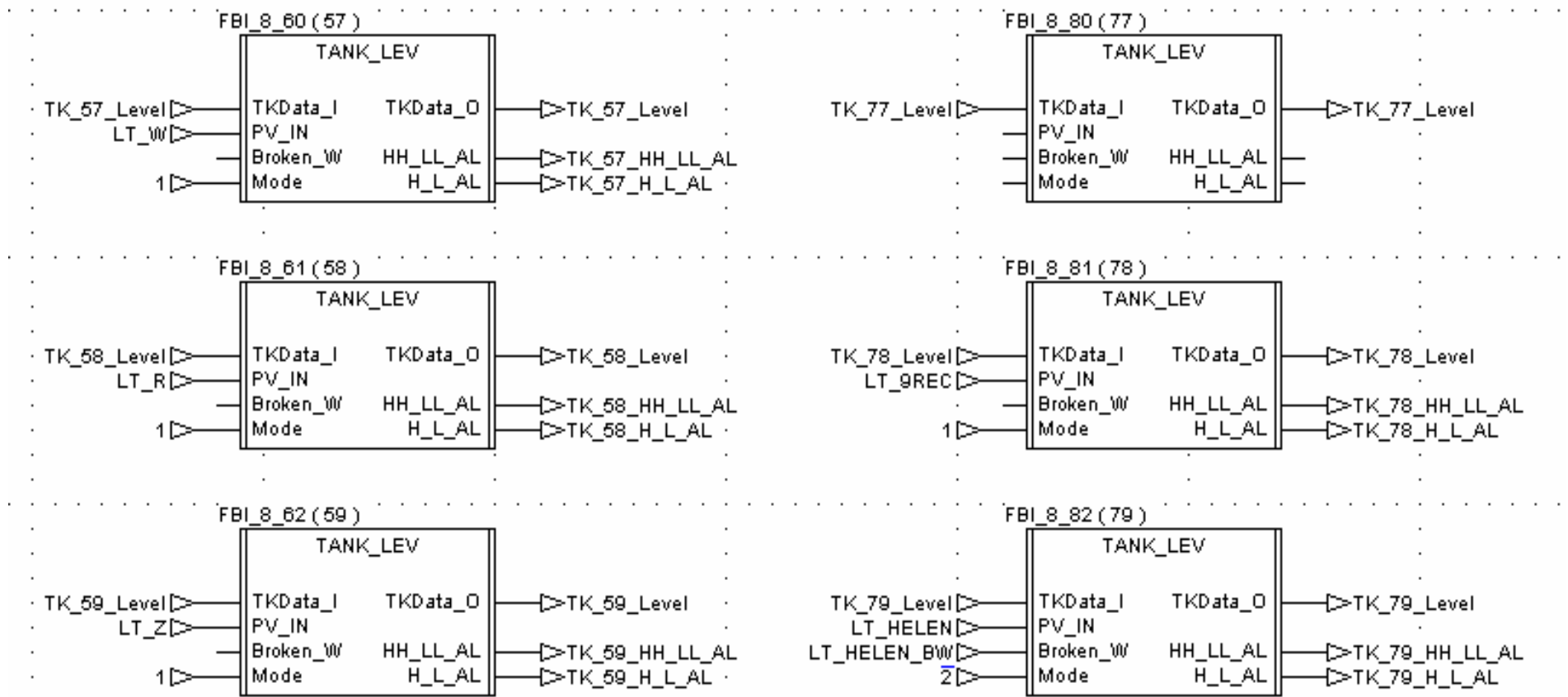
# DFB - Solenoid Valve - Alarm Logic

Solenoid Valve Alarm Logic  
 Energize to open, spring return  
 =====





# Tank Level Scaling using Arrays & Structured tags



```

analogPU: STRUCT
  PV: INT;      (* Analog Process Variable - current value X 10 *)
  LL: INT;      (* Analog PU - Low Low Alarm Set Point *)
  LO: INT;      (* Analog PU - Low Alarm Set Point *)
  HI: INT;      (* Analog PU - High Alarm Set Point *)
  HH: INT;      (* Analog PU - High High Alarm Set Point *)
  AL: WORD;     (* Analog PU - Alarm Word *)
                  (* Bit 0 = Low Low Alarm
                  *   Bit 1 = Low Alarm
                  *   Bit 2 = High Alarm
                  *   Bit 3 = High High Alarm
                  *   Bit 4 = Broken Wire Alarm, if available *)

END_STRUCT;

tankData: STRUCT
  MT: DINT;     (* Material currently in Tank *)
  DBID: DINT;   (* Tank Data Base ID *)
  PV: INT;      (* Tank Process Variable - current value *)
  LL: INT;      (* Analog PU - Low Low Alarm Set Point *)
  LO: INT;      (* Analog PU - Low Alarm Set Point *)
  HI: INT;      (* Analog PU - High Alarm Set Point *)
  HH: INT;      (* Analog PU - High High Alarm Set Point *)
  AL: WORD;     (* Analog PU - Alarm Word *)
                  (* Bit 0 = Low Low Alarm
                  *   Bit 1 = Low Alarm
                  *   Bit 2 = High Alarm
                  *   Bit 3 = High High Alarm
                  *   Bit 4 = Broken Wire Alarm, if available *)

  SG: INT;      (* Specific Gravity of Material in Tank - X 1000 *)
  CAP: INT;     (* Capacity (volume) of Tank *)
  DIA: INT;     (* Tank Diameter for Volume calculation X 10 *)
  HGT: INT;     (* Tank Height for Volume calculation X10 *)
  BOS: INT;     (* Bubbler Off Set for Volume calculation *)
  TOS: INT;     (* Tank Volume Off Set for Tank Dip calculation *)
  HOS: INT;     (* Height Off Set for Volume calculation X10 *)
  DIP: INT;     (* Calculated Tank expected Dip Volume X10 *)

END_STRUCT;

Tank: ARRAY [1..110] OF tankData;  (* Array of 110 units for Tank Data *)

```



# Arrays in loop for calculations

```
»»»»»»»»»»»» Structured Text Start ««««««««««««
(* Procedure checks each still for its current Crude, Product & Bottoms Tank Numbers
  Then the current Specific Gravities are assigned to the appropriate Tank data base
  for use in tank level calculations with the head pressure *)

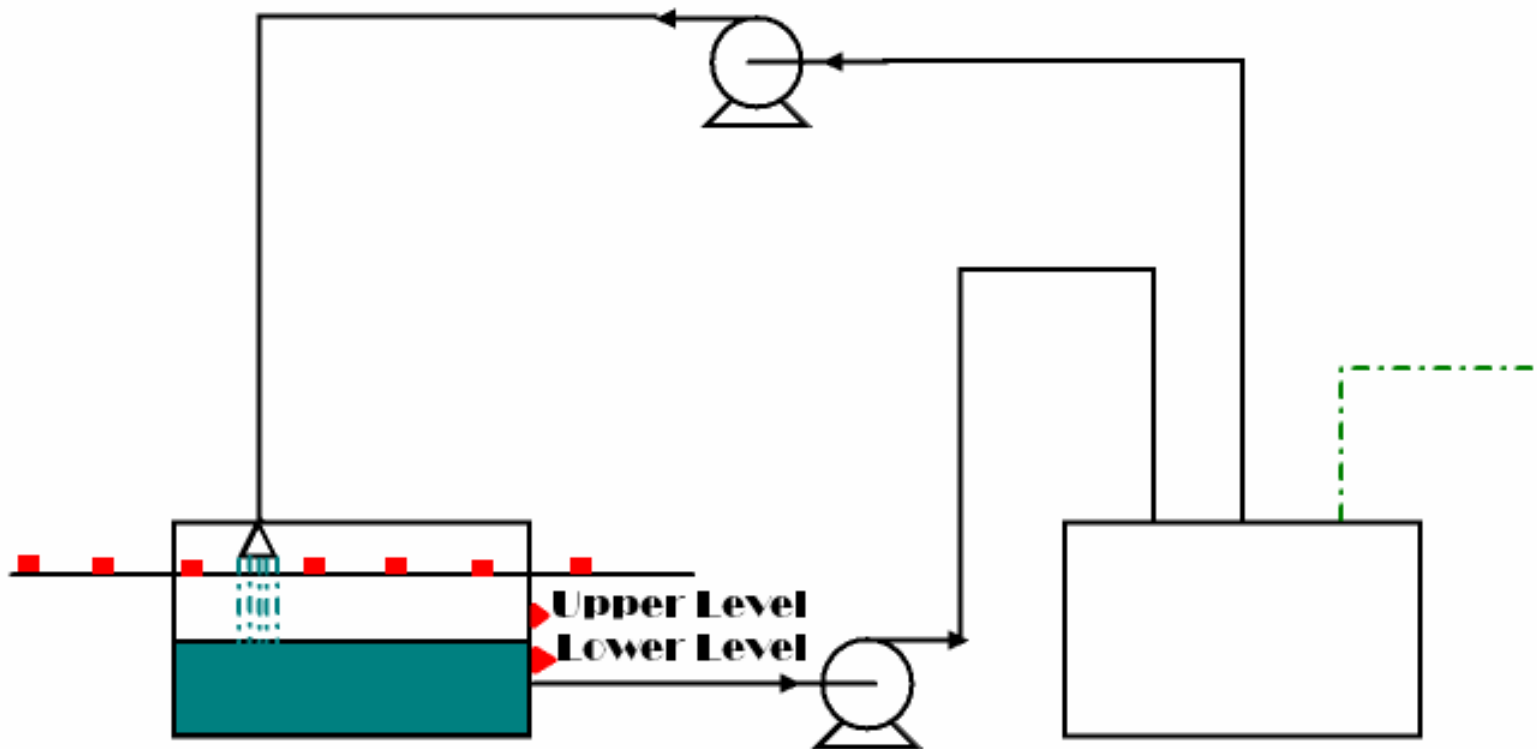
FOR Still_No := 1 TO 9 DO
  IF Still_Data[Still_No].ProductTankID > 0 AND (Still_Data[Still_No].ProductSG > 0 AND
    Still_Data[Still_No].ProductSG <= 110) THEN
    P_Tank_No := Still_Data[Still_No].ProductTankID;
    Tank_Data[P_Tank_No].SG := Still_Data[Still_No].ProductSG;
  END_IF;

  IF Still_Data[Still_No].CrudeTankID > 0 AND (Still_Data[Still_No].CrudeSG > 0 AND
    Still_Data[Still_No].CrudeSG <= 110) THEN
    C_Tank_No := Still_Data[Still_No].CrudeTankID;
    Tank_Data[C_Tank_No].SG := Still_Data[Still_No].CrudeSG;
  END_IF;

  IF Still_Data[Still_No].BottomTankID > 0 AND (Still_Data[Still_No].BottomsSG > 0 AND
    Still_Data[Still_No].BottomsSG <= 110) THEN
    B_Tank_No := Still_Data[Still_No].BottomTankID;
    Tank_Data[B_Tank_No].SG := Still_Data[Still_No].BottomsSG;
  END_IF;

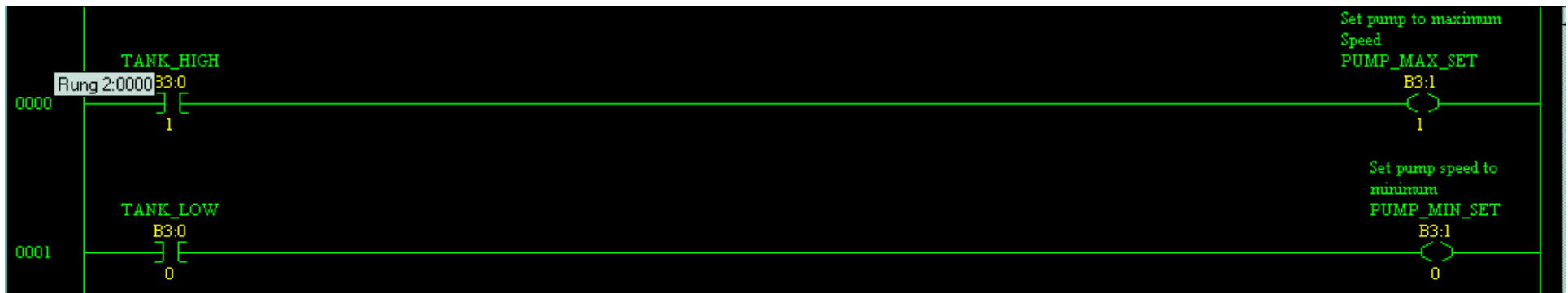
END_FOR;
»»»»»»»»»»»» Structured Text End ««««««««««««
```

# Controlling Tank Level



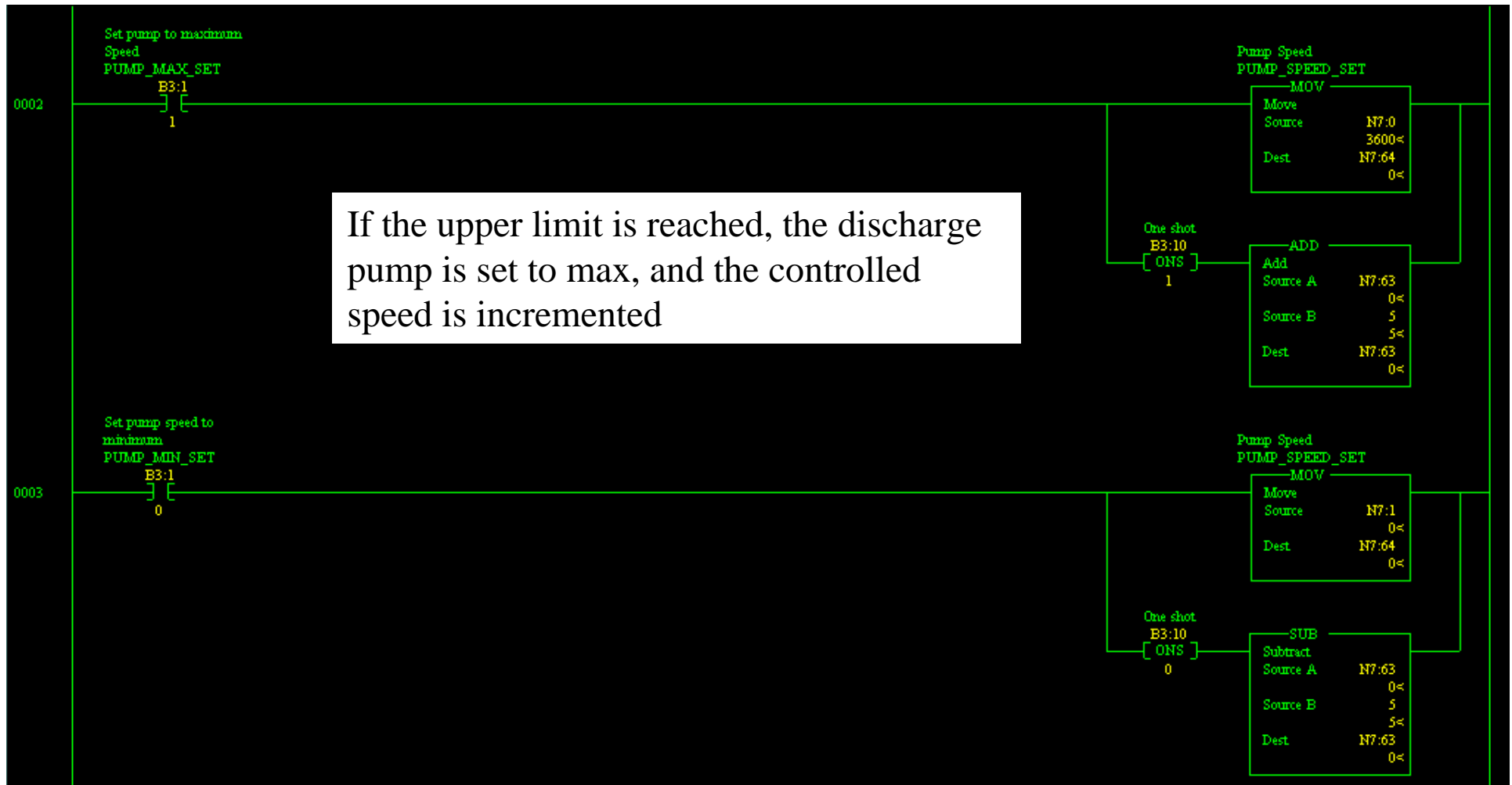
Useful when analog signal for tank level is too costly, or impractical because of process conditions.

# Detect Tank Level

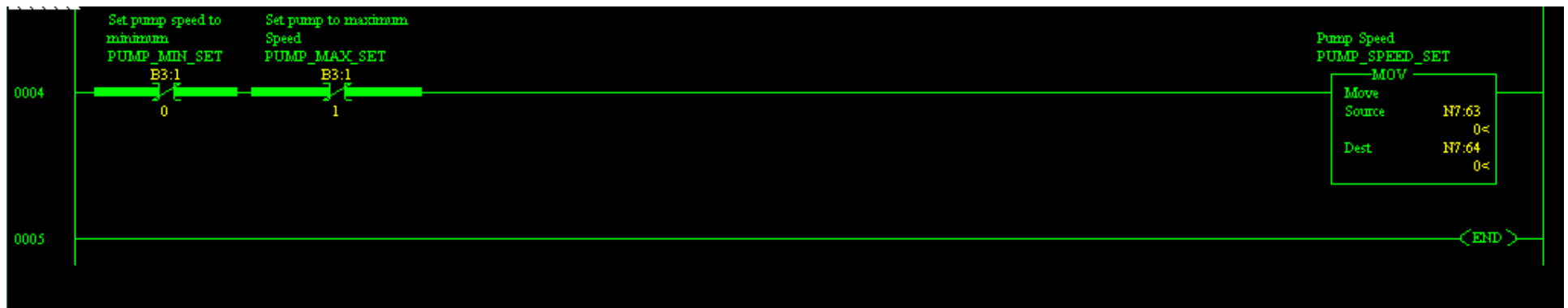


Fluid level is detected by proximity sensors, and sets appropriate program bits.

# Set Pump Speed when Outside Limits

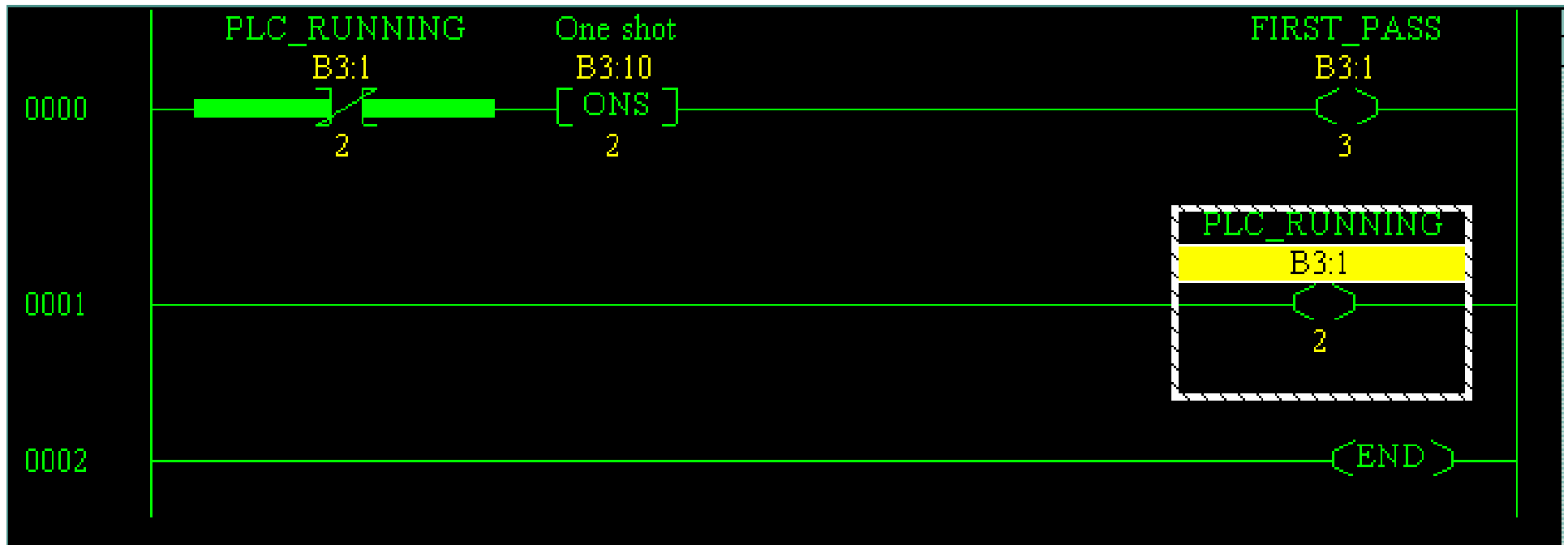


# Set Pump Speed when between limits



When the fluid level is between the limits, the pump speed is set to the adjusted speed, and the normal operation of the pump continues until the next time one of the fluid level limits are reached again.

# Capture First Pass



# Questions & Discussions

## **Contact Info:**

**Doug Norton** (647) 295-2051

PLC Systems Integrators Inc.

[doug.norton@plcsystems.net](mailto:doug.norton@plcsystems.net)

**Currie Gardner** (416) 505-5530

Rendrag Solutions Inc.

[cgardner@rendragsolutions.com](mailto:cgardner@rendragsolutions.com)