








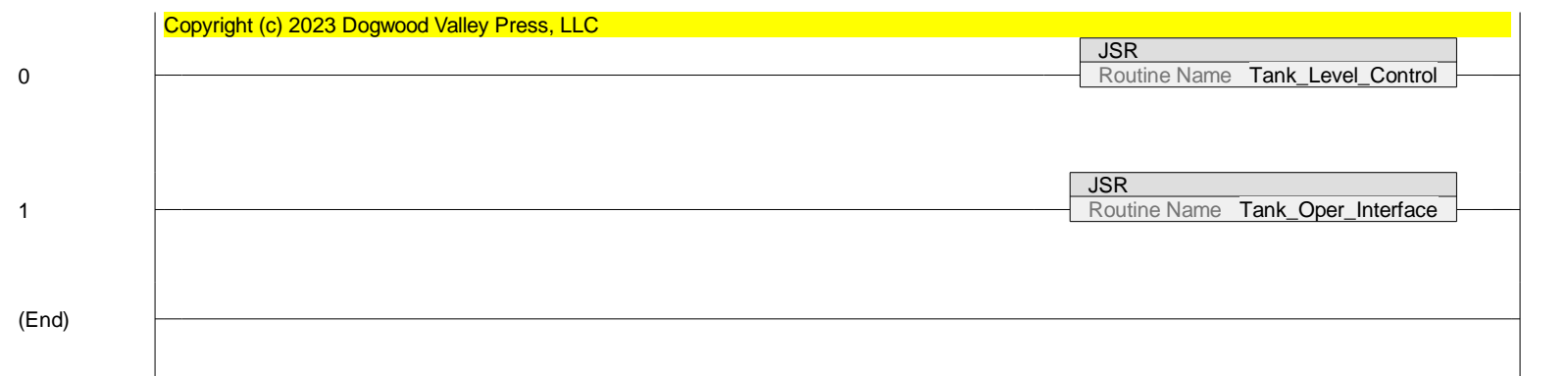


 **Controller Example\_12\_4** **Controller Fault Handler** **Power-Up Handler****Tasks** **MainTask** **MainProgram** **MainRoutine** **Tank\_Level\_Control** **Tank\_Oper\_Interface** **Unscheduled****Motion Groups** **Ungrouped Axes****Add-On Instructions****Data Types** **User-Defined** **Strings** **Add-On-Defined** **Module-Defined** **AB:1756\_DI:C:0** **AB:1756\_DI:I:0** **AB:1756\_DO:C:0** **AB:1756\_DO:I:0** **AB:1756\_DO:O:0** **AB:1756\_IF8\_Float:C:0** **AB:1756\_IF8\_Float:I:0** **AB:1756\_IF8\_Integer:C:0** **AB:1756\_IF8\_Integer:I:0** **AB:1756\_NII\_Struct:C:0****Trends****I/O Configuration** **1756 Backplane, 1756-A10** **[0] 1756-L71 Example\_12\_4** **[1] 1756-IB32/A din** **[2] 1756-OB16I dout** **[3] 1756-IF8 ana\_in**



```
1 (* Example 12.3 Simple Tank Level Control *)
2
3 (* Copyright (c) 2023 Dogwood Valley Press, LLC *)
4
5 (* Get tank level *)
6 LT428_Val := (LT428_MEAS/100.0)*(15-1)+1;
7
8 (* Level control: turn on when low, turn off when *)
9 (*   high. If not enabled, always turn off      *)
10 IF T428_Cntrl THEN
11     IF (LT428_Val < T428_Min) THEN
12         XV427_OPEN := 1;
13     END_IF;
14     IF (LT428_Val > T428_Max) THEN
15         XV427_OPEN := 0;
16     END_IF;
17 ELSE
18     XV427_OPEN := 0;
19 END_IF;
20
```

```
1 (* Example 12.4 Operator interface/alarm system for tank *)
2
3 (* Copyright (c) 2022 Dogwood Valley Press, LLC *)
4
5 (* Make sure T428_Min is at least 2.1 and no more than 13.4 *)
6 IF (T428_Min < 2.1) THEN T428_Min := 2.1; END_IF;
7 IF (T428_Min > 13.4) THEN T428_Min := 13.4; END_IF;
8
9 (* Calculate max level *)
10 T428_Max := T428_Min + 1.5;
11
12 (* Low level alarms *)
13 T428_LOLA := (LT428_Val < 4.0);
14 T428_Hrn_Act := (LT428_Val < 2.0);
15
16 (* Generate trigger when level falls below 2.0 *)
17 HrnOns.InputBit := T428_Hrn_Act;
18 OSRI( HrnOns);
19 Horn_Trig := HrnOns.OutputBit;
20 (* Acknowledge button transition *)
21 AckOns.InputBit := ALM_ACK;
22 OSRI(AckOns);
23
24 (* Trigger horn when level drops below 2.0 or *)
25 (* stays below 2 for 5 minutes after ack'ed. *)
26 IF (Horn_Trig) OR (Ack_Tmr.DN) THEN
27     T428_HORN := 1;
28 ELSE
29     IF AckOns.OutputBit THEN
30         T428_HORN := 0;
31     END_IF;
32 END_IF;
33
34 (* Time level staying below 2 after ack *)
35 IF (AckOns.OutputBit AND T428_Hrn_Act) THEN
36     Ack_Tmr_En := 1;
37 END_IF;
38 IF Ack_Tmr_En AND (NOT T428_Hrn_Act OR Ack_Tmr.DN) THEN
39     Ack_Tmr_En := 0;
40 END_IF;
41 Ack_Tmr.TimerEnable := Ack_Tmr_En;
42 Ack_Tmr.PRE := 30000; Ack_Tmr.EnableIn := 1;
43 TONR( Ack_Tmr);
44
```

<b>Example_12_4</b>	
Label does not exist .....	1
<b>MainTask</b>	
<b>MainProgram</b>	
<b>MainRoutine</b>	
Ladder Diagram .....	2
<b>Tank_Level_Control</b>	
Structured Text .....	3
<b>Tank_Oper_Interface</b>	
Structured Text .....	4