I have added notes to document what had to be inserted or modified to move from Lab 2 to Lab 3.

Added Routines:

- Communication (rungs to transfer data from PLC_1 to PLC_2)
- Vision (new vision sensors detects blue and green bases and lids)

The main routine contains unconditional jumps to Communication, Vision, Lab3B and Indicators.

If the upstream PLC is not connected or not ready, just change the emitters to send parts automatically instead of via the PLC interface).

Auto, Manual and Cycle Rungs.
It should be noted that Cycle Start after Cycle Stop may only work smoothly if the Emitters are controlled, otherwise some jams and collisions may occur.
The Lab 2 instruction suggests keeping the same Automatic Program as the Lab 1 to keep the focus on Manual Mode Operation and the addition of Position Indication.
Lab 3 requires a few changes to separate Blue and Green components.

0
Auto_PB <BOOL_IN_14> | Auto_Man_Reset <BOOL_IN_11> | Factory_Running <BOOL_IN_15> | E_Stop <BOOL_IN_13> | Manual_Mode_Lt <BOOL_OUT_13> | Auto_Mode_Lt <BOOL_OUT_11>
Auto_Mode_Lt <BOOL_OUT_11>
Cycle_Running_Lt <BOOL_OUT_9>

1
Manual <BOOL_IN_16> | Auto_Man_Reset <BOOL_IN_11> | Factory_Running <BOOL_IN_15> | E_Stop <BOOL_IN_13> | Auto_Mode_Lt <BOOL_OUT_11> | Manual_Mode_Lt <BOOL_OUT_13>
Manual_Mode_Lt <BOOL_OUT_13>

2
Cycle_Stop_PB <BOOL_IN_12> | Cycle_Running_Lt <BOOL_OUT_9> | Cycle_Stop_Request_Lt <BOOL_OUT_10>
Cycle_Stop_Request_Lt <BOOL_OUT_10>

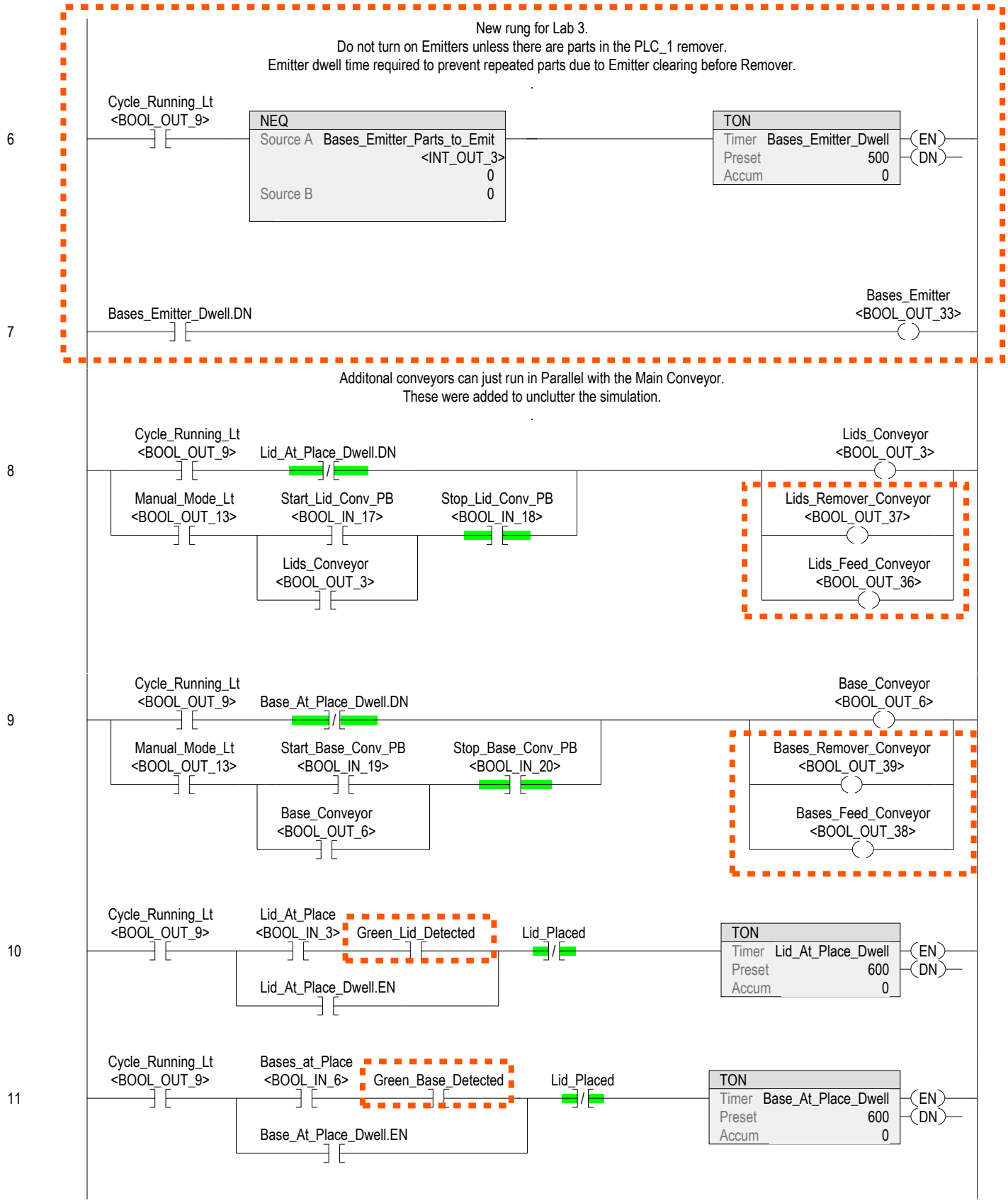
The intent of the Lab was to add Automatic & Manual Modes as well as Indicators & Lights without major changes to the original Automatic Code.
The logic after this point is the same as the original solution with two slight changes. One to allow an indicator to turn on (see Rung 8) and another for smoother operation (See Rung 19). Both are not necessary for Automatic Operation to function.

3
Cycle_Start_PB <BOOL_IN_10> | Auto_Mode_Lt <BOOL_OUT_11> | Cycle_Stop_Request_Lt <BOOL_OUT_10> | Factory_Running <BOOL_IN_15> | E_Stop <BOOL_IN_13> | Cycle_Running_Lt <BOOL_OUT_9>
Cycle_Running_Lt <BOOL_OUT_9>
Part_Clear

New rung for Lab 3.
Do not turn on Emitters unless there are parts in the PLC_1 remover.
Emitter dwell time required to prevent repeated parts due to Emitter clearing before Remover.

4
Cycle_Running_Lt <BOOL_OUT_9>
NEQ
Source A Lids_Emitter_Part_to_Emit <INT_OUT_4>
0
Source B 0
TON
Timer Lids_Emitter_Dwell
Preset 500
Accum 0
EN
DN

5
Lids_Emitter_Dwell.DN
Lids_Emitter <BOOL_OUT_34>


New rung for Lab 3.
Do not turn on Emitters unless there are parts in the PLC_1 remover.
Emitter dwell time required to prevent repeated parts due to Emitter clearing before Remover.
6
Cycle_Running_Lt
<BOOL_OUT_9>
NEQ
Source A Bases_Emitter_Parts_to_Emit
<INT_OUT_3>
0
Source B 0
TON
Timer Bases_Emitter_Dwell
Preset 500
Accum 0
EN
DN
7
Bases_Emitter_Dwell.DN
Bases_Emitter
<BOOL_OUT_33>
Additonal conveyors can just run in Parallel with the Main Conveyor.
These were added to unclutter the simulation.
8
Cycle_Running_Lt
<BOOL_OUT_9>
Lid_At_Place_Dwell.DN
Manual_Mode_Lt
<BOOL_OUT_13>
Start_Lid_Conv_PB
<BOOL_IN_17>
Stop_Lid_Conv_PB
<BOOL_IN_18>
Lids_Conveyor
<BOOL_OUT_3>
Lids_Conveyor
<BOOL_OUT_3>
Lids_Remover_Conveyor
<BOOL_OUT_37>
Lids_Feed_Conveyor
<BOOL_OUT_36>
9
Cycle_Running_Lt
<BOOL_OUT_9>
Base_At_Place_Dwell.DN
Manual_Mode_Lt
<BOOL_OUT_13>
Start_Base_Conv_PB
<BOOL_IN_19>
Stop_Base_Conv_PB
<BOOL_IN_20>
Base_Conveyor
<BOOL_OUT_6>
Base_Conveyor
<BOOL_OUT_6>
Bases_Remover_Conveyor
<BOOL_OUT_39>
Bases_Feed_Conveyor
<BOOL_OUT_38>
10
Cycle_Running_Lt
<BOOL_OUT_9>
Lid_At_Place
<BOOL_IN_3>
Green_Lid_Detected
Lid_Placed
Lid_At_Place_Dwell.EN
TON
Timer Lid_At_Place_Dwell
Preset 600
Accum 0
EN
DN
11
Cycle_Running_Lt
<BOOL_OUT_9>
Bases_at_Place
<BOOL_IN_6>
Green_Base_Detected
Lid_Placed
Base_At_Place_Dwell.EN
TON
Timer Base_At_Place_Dwell
Preset 600
Accum 0
EN
DN

Added Item Detected Dwell as a way to ensure the Lowered Indicator has time to light up on the panel. The timing of the advance was too fast in the previous version.

12
Item_Detected <BOOL_IN_2>
TON Timer Item_Detected_Dwell Preset 1000 Accum 0 (EN) (DN)

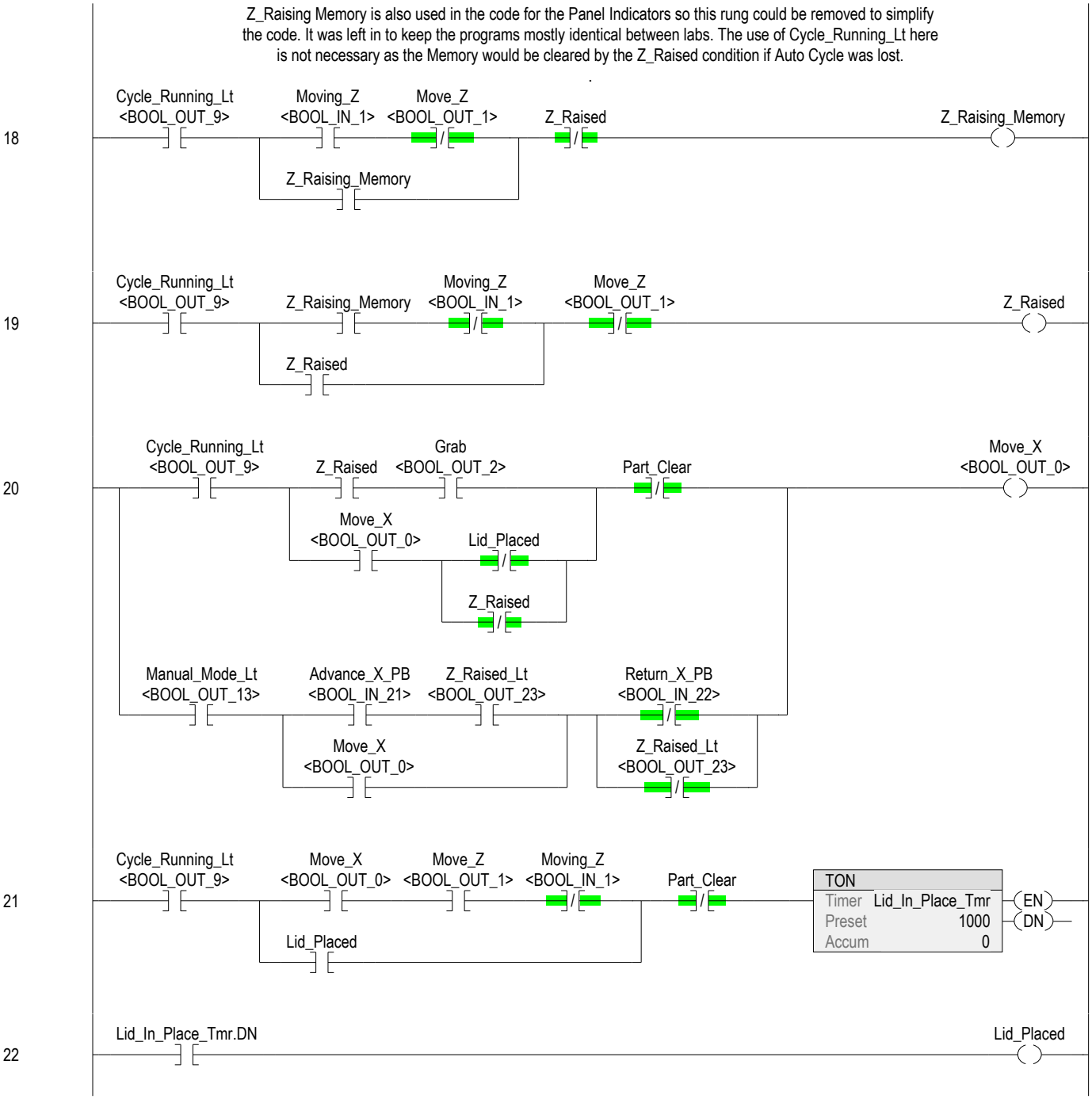
13
Cycle_Running_Lt <BOOL_OUT_9> Item_Detected_Dwell.DN Lid_Placed <BOOL_IN_3> Grab <BOOL_OUT_2>

14
Cycle_Running_Lt <BOOL_OUT_9> Lid_At_Place_Dwell.DN Grab <BOOL_OUT_2> Green_Lid_Detected Clamp_Lids <BOOL_OUT_4>
Manual_Mode_Lt <BOOL_OUT_13> Clamp_Lid_PB <BOOL_IN_25> Unclamp_Lid_PB <BOOL_IN_26>
Clamp_Lids <BOOL_OUT_4>

15
Cycle_Running_Lt <BOOL_OUT_9> Base_At_Place_Dwell.DN Green_Base_Detected Clamp_Bases <BOOL_OUT_7>
Manual_Mode_Lt <BOOL_OUT_13> Clamp_Base_PB <BOOL_IN_27> Unclamp_Base_PB <BOOL_IN_28>
Clamp_Bases <BOOL_OUT_7>

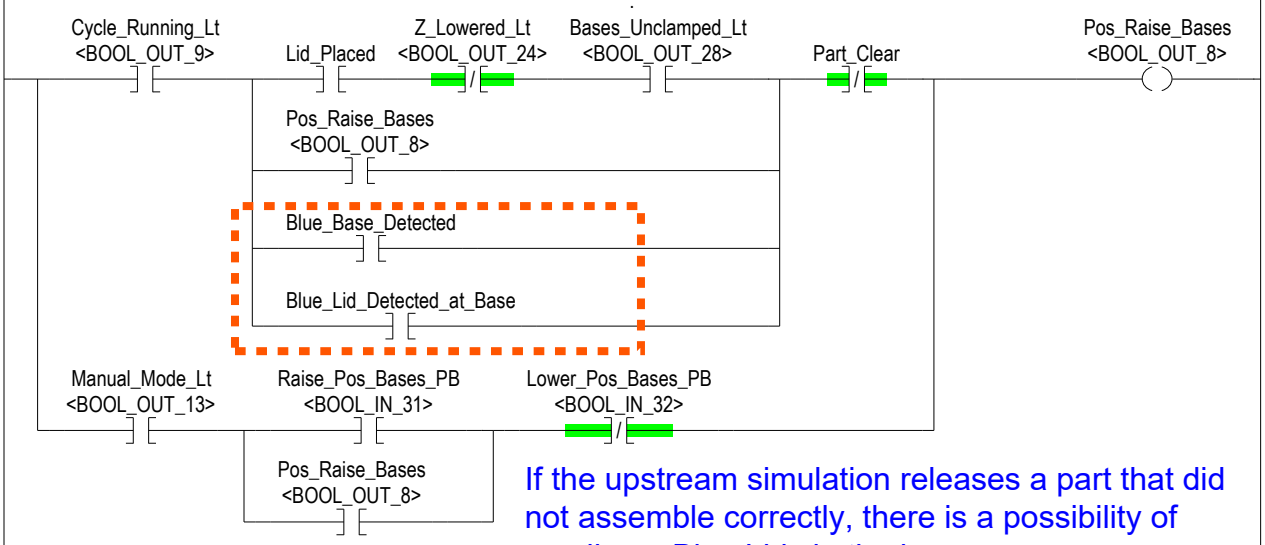
16
Move_X <BOOL_OUT_0>
TON Timer Advance_X_Dwell Preset 200 Accum 0 (EN) (DN)

17
Cycle_Running_Lt <BOOL_OUT_9> Lid_Clamped <BOOL_IN_4> Base_Clamped <BOOL_IN_7> Grab <BOOL_OUT_2> Move_Z <BOOL_OUT_1>
Moving_X Advance_X_Dwell.DN <BOOL_IN_0> Lid_Placed <BOOL_IN_3>
Manual_Mode_Lt <BOOL_OUT_13> Lower_Z_PB <BOOL_IN_24> Raise_Z_PB <BOOL_IN_23>
Move_Z <BOOL_OUT_1>


Z_Raising Memory is also used in the code for the Panel Indicators so this rung could be removed to simplify the code. It was left in to keep the programs mostly identical between labs. The use of Cycle_Running_Lt here is not necessary as the Memory would be cleared by the Z_Raised condition if Auto Cycle was lost.
18
Cycle_Running_Lt <BOOL_OUT_9>
Moving_Z <BOOL_IN_1>
Move_Z <BOOL_OUT_1>
Z_Raised
Z_Raising_Memory
Z_Raising_Memory
19
Cycle_Running_Lt <BOOL_OUT_9>
Z_Raising_Memory
Moving_Z <BOOL_IN_1>
Move_Z <BOOL_OUT_1>
Z_Raised
Z_Raised
20
Cycle_Running_Lt <BOOL_OUT_9>
Z_Raised
Grab <BOOL_OUT_2>
Part_Clear
Move_X <BOOL_OUT_0>
Move_X <BOOL_OUT_0>
Lid_Placed
Z_Raised
Manual_Mode_Lt <BOOL_OUT_13>
Advance_X_PB <BOOL_IN_21>
Z_Raised_Lt <BOOL_OUT_23>
Return_X_PB <BOOL_IN_22>
Move_X <BOOL_OUT_0>
Z_Raised_Lt <BOOL_OUT_23>
21
Cycle_Running_Lt <BOOL_OUT_9>
Move_X <BOOL_OUT_0>
Move_Z <BOOL_OUT_1>
Moving_Z <BOOL_IN_1>
Part_Clear
TON
Timer Lid_In_Place_Tmr
Preset 1000
Accum 0
EN
DN
Lid_Placed
22
Lid_In_Place_Tmr.DN
Lid_Placed

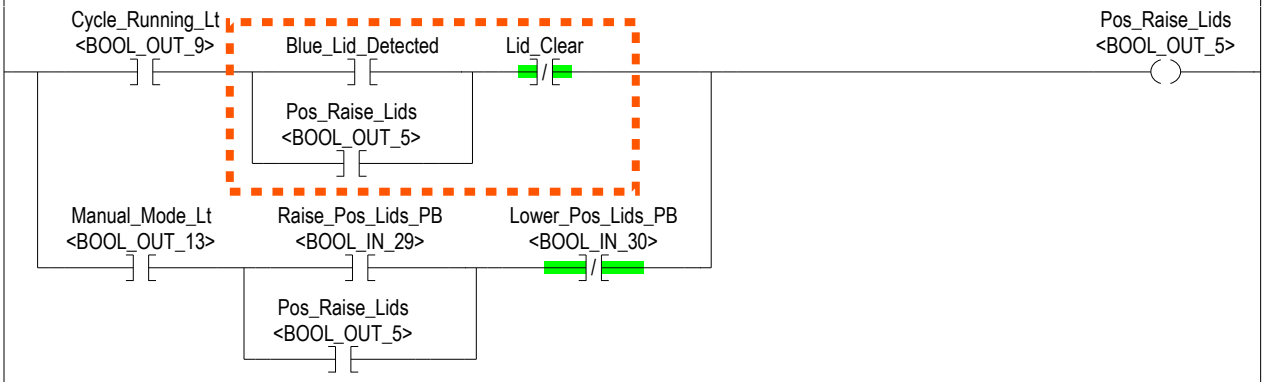
Added Unclamped Bases Lt. to Raise the Positioner as a way to slightly modify the program to run more realistically athough this change is not required.

23



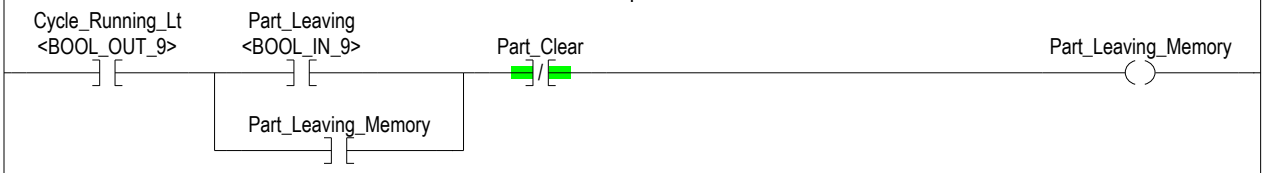
If the upstream simulation releases a part that did not assemble correctly, there is a possibility of sending a Blue Lid via the base conveyor.

24

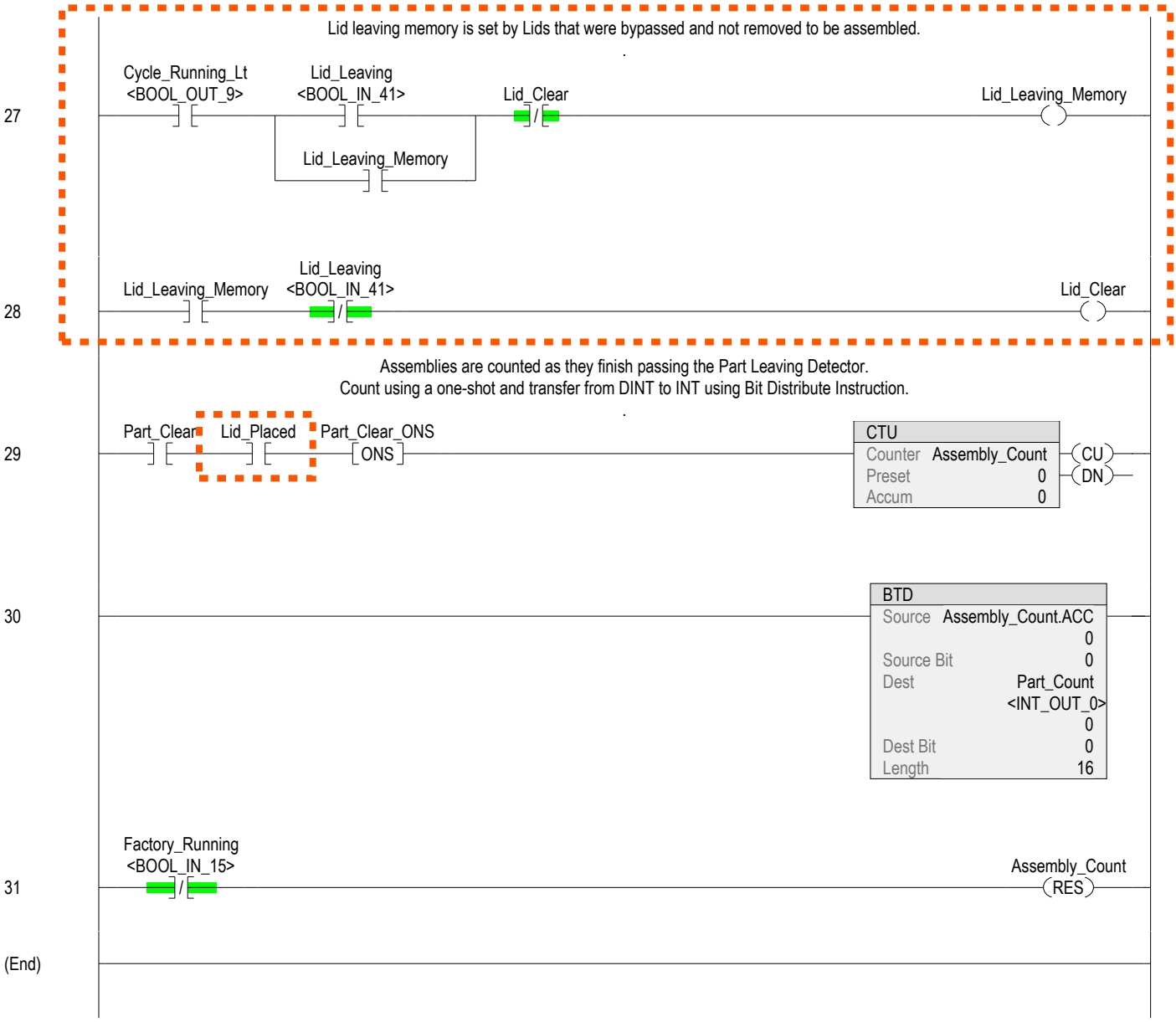


Part leaving memory can be set by a base that was not assembled or a complete assembly.

25



26


Lid leaving memory is set by Lids that were bypassed and not removed to be assembled.
27
Cycle_Running_Lt
<BOOL_OUT_9>
Lid_Leaving
<BOOL_IN_41>
Lid_Clear
Lid_Leaving_Memory
Lid_Leaving_Memory
28
Lid_Leaving_Memory
Lid_Leaving
<BOOL_IN_41>
Lid_Clear
Assemblies are counted as they finish passing the Part Leaving Detector.
Count using a one-shot and transfer from DINT to INT using Bit Distribute Instruction.
29
Part_Clear
Lid_Placed
Part_Clear_ONS
ONS
CTU
Counter Assembly_Count
Preset 0
Accum 0
CU
DN
30
BTD
Source Assembly_Count.ACC
0
Source Bit 0
Dest Part_Count
<INT_OUT_0>
0
Dest Bit 0
Length 16
31
Factory_Running
<BOOL_IN_15>
Assembly_Count
RES
(End)

Position Indication: Lab instructions were to add Lights and Pushbuttons with the intend of not altering the Automatic Code (for ease of implementation).

.

Conveyor Running Light can be run directly from the Lids and Bases Conveyor Output. This would be similar to a real machine since the Output would only seal in if the Physical Contactor energized (ie: overloads not tripped).

.

0

Lids_Conveyor <BOOL_OUT_3> — Lids_Conv_Running_Lt <BOOL_OUT_17>

1

Base_Conveyor <BOOL_OUT_6> — Bases_Conv_Running_Lt <BOOL_OUT_19>

X Axis Position Indicators.
The techniques used to detect the Position of the Axis is somewhat unique to the simulation. X_Advancing_Memory or X_Returning_Memory only turns ON when the simulation returned the "Moving" bit back to the PLC. Moving is used for both directions, however, since Move_X determines the direction of motion, the combination of the two bits can be utilized. Once the X Axis is no longer Moving, the Directional Moving Memory is used to turn ON the correct panel light.

.

2

Move_X <BOOL_OUT_0> Moving_X <BOOL_IN_0> X_Adanced_Lt <BOOL_OUT_21> Factory_Running <BOOL_IN_15> — X_Advancing_Memory

X_Advancing_Memory

3

X_Advancing_Memory Moving_X <BOOL_IN_0> Move_X <BOOL_OUT_0> Factory_Running <BOOL_IN_15> — X_Advanced_Lt <BOOL_OUT_21>

X_Adanced_Lt <BOOL_OUT_21>

4

Move_X <BOOL_OUT_0> Moving_X <BOOL_IN_0> X_Returned_Lt <BOOL_OUT_22> Factory_Running <BOOL_IN_15> — X_Returning_Memory

X_Returning_Memory

In order to detect the axis position on "power up", a detection Limit was added. Returning Memory is redundant but left in to demonstrate both methods. It should be noted that, if the Switch was removed, the X-Axis would have to be Advanced and then Returned (in Auto or Manual) for the Returned Light to turn ON.

.

5

X_Returning_Memory Moving_X <BOOL_IN_0> Move_X <BOOL_OUT_0> Factory_Running <BOOL_IN_15> — X_Returned_Lt <BOOL_OUT_22>

X_Returned_Prox <BOOL_IN_33>

X_Returned_Lt <BOOL_OUT_22>

Z Axis Position Indicators.
Same as X Axis.

6
Move_Z <BOOL_OUT_1> | Moving_Z <BOOL_IN_1> | Z_Lowered_Lt <BOOL_OUT_24> | Factory_Running <BOOL_IN_15> | Z_Lowering_Memory
Z_Lowering_Memory

7
Z_Lowering_Memory | Moving_Z <BOOL_IN_1> | Move_Z <BOOL_OUT_1> | Factory_Running <BOOL_IN_15> | Z_Lowered_Lt <BOOL_OUT_24>
Z_Lowered_Lt <BOOL_OUT_24>

Using Z_Raising_Memory_2 so as to not conflict with Automatic Code.

8
Move_Z <BOOL_OUT_1> | Moving_Z <BOOL_IN_1> | Z_Raised_Lt <BOOL_OUT_23> | Factory_Running <BOOL_IN_15> | Z_Raising_Memory_2
Z_Raising_Memory_2

9
Z_Raising_Memory_2 | Moving_Z <BOOL_IN_1> | Move_Z <BOOL_OUT_1> | Factory_Running <BOOL_IN_15> | Z_Raised_Lt <BOOL_OUT_23>
Z_Raised_Lt <BOOL_OUT_23>
Z_Raised_Prox <BOOL_IN_40>

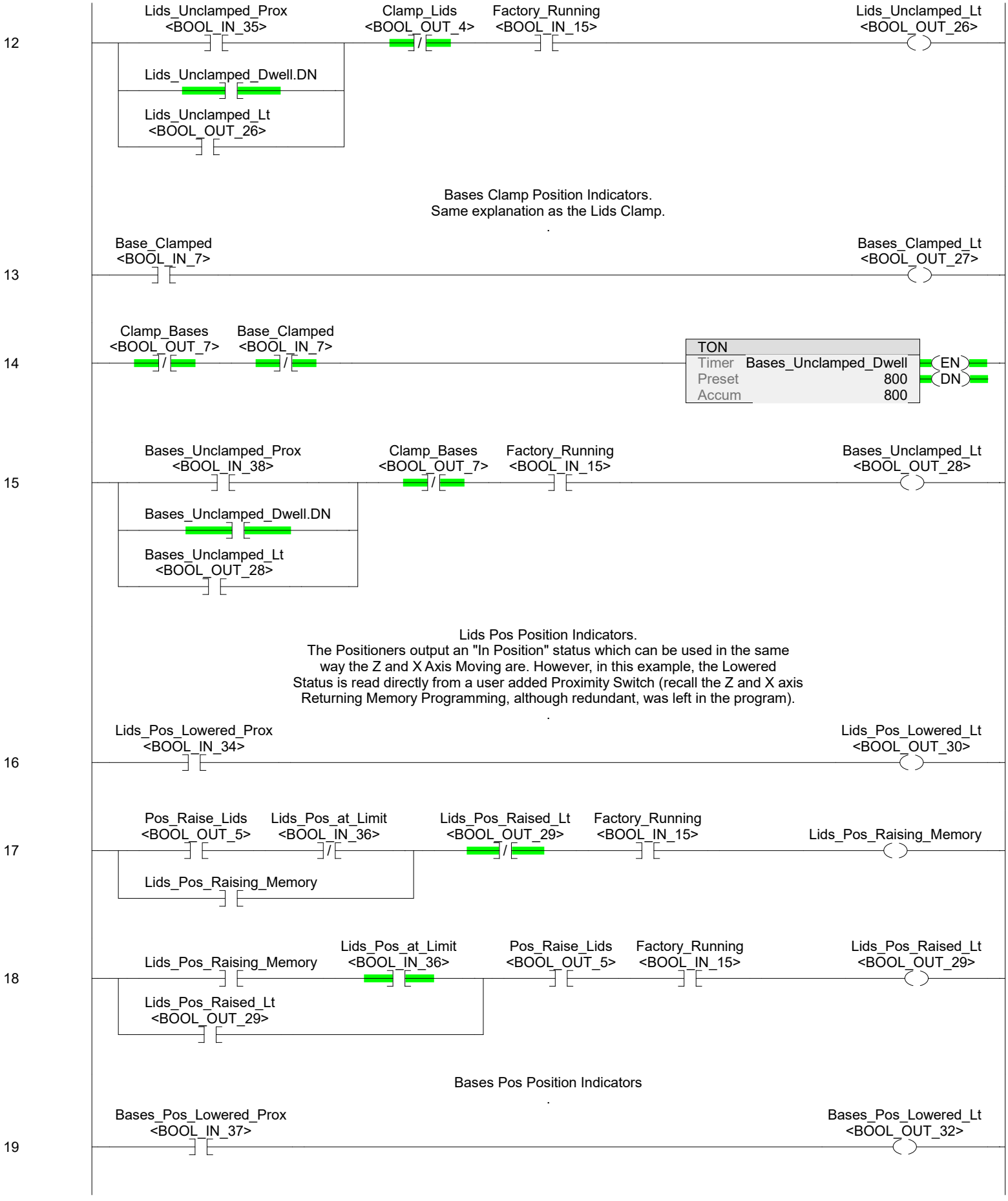
Lids Clamp Position Indicators.
The Clamped operation has an input from the Simulation that represents "Clamped" status. This is similar using a Pressure Switch on the Pneumatic Feed Line to detect when the part has been Captured allowing for parts of various sizes to be used.

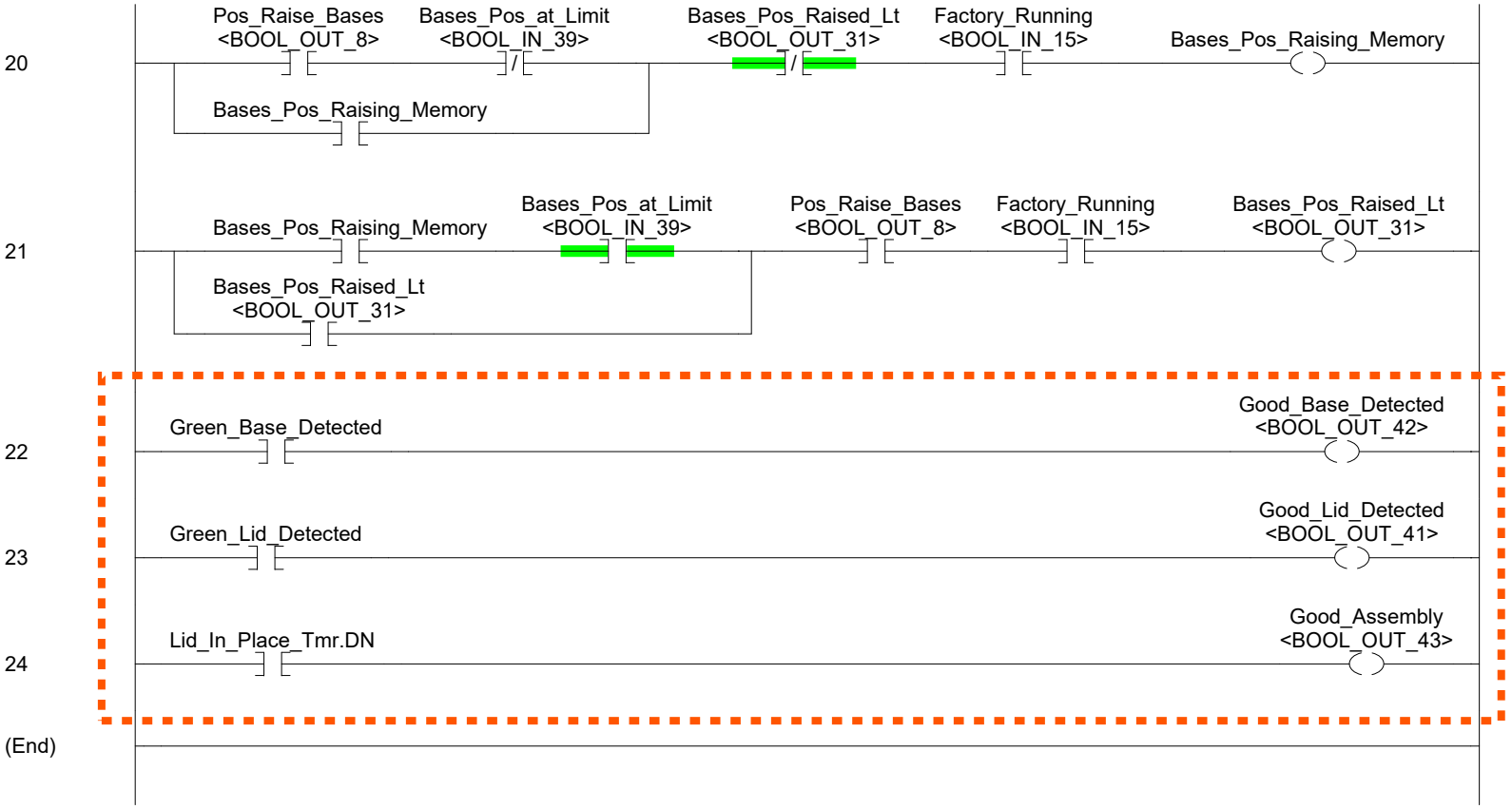
10
Lid_Clamped <BOOL_IN_4> | Lids_Clamped_Lt <BOOL_OUT_25>

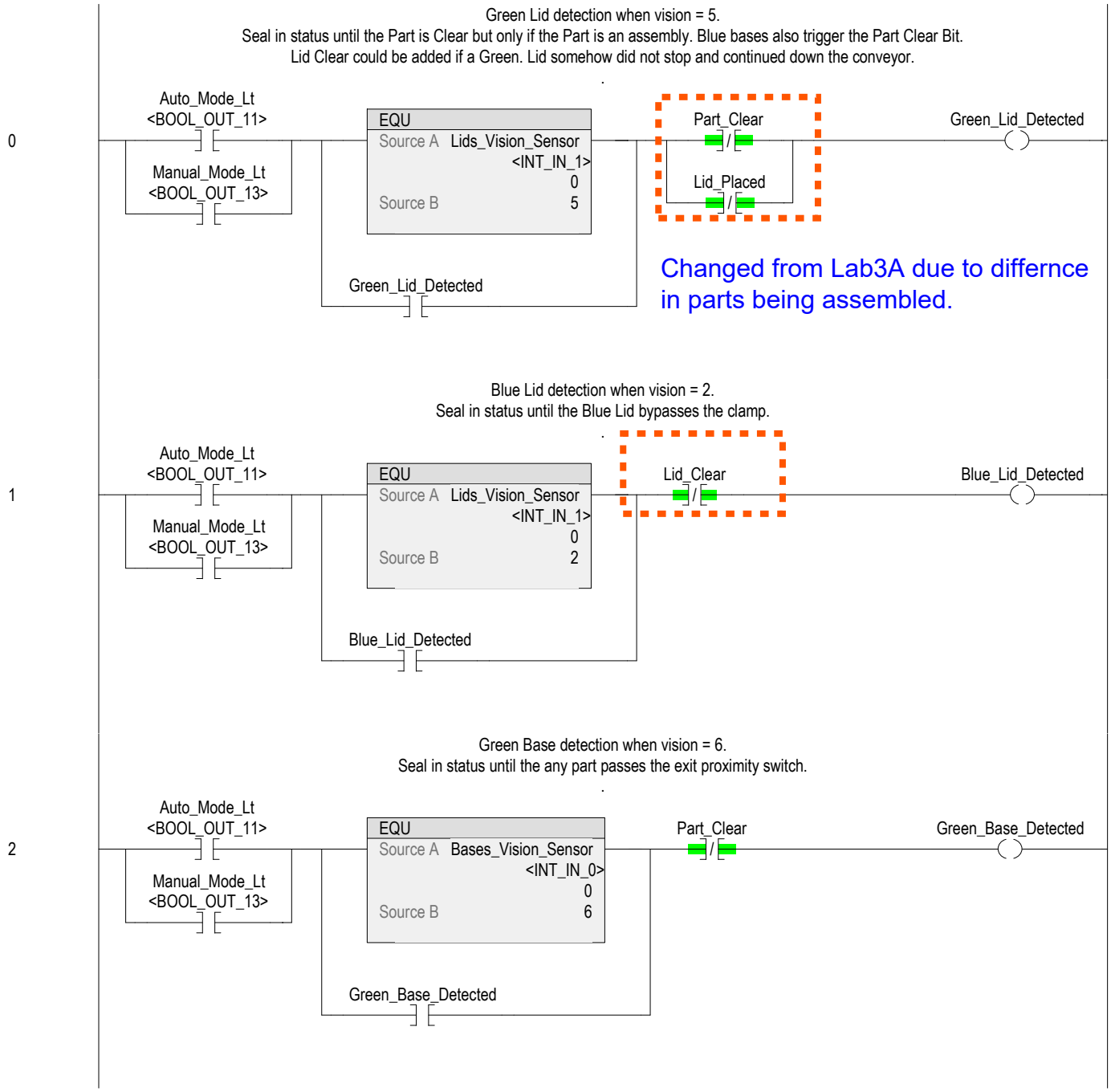
Since the Simulation Scene does not come standard with an Unclamped Proximiaty Switch and to avoid adding a switch both the Lowered and Raised Position of the Positioner, a timer is used as an "Assumption" of motion completion.

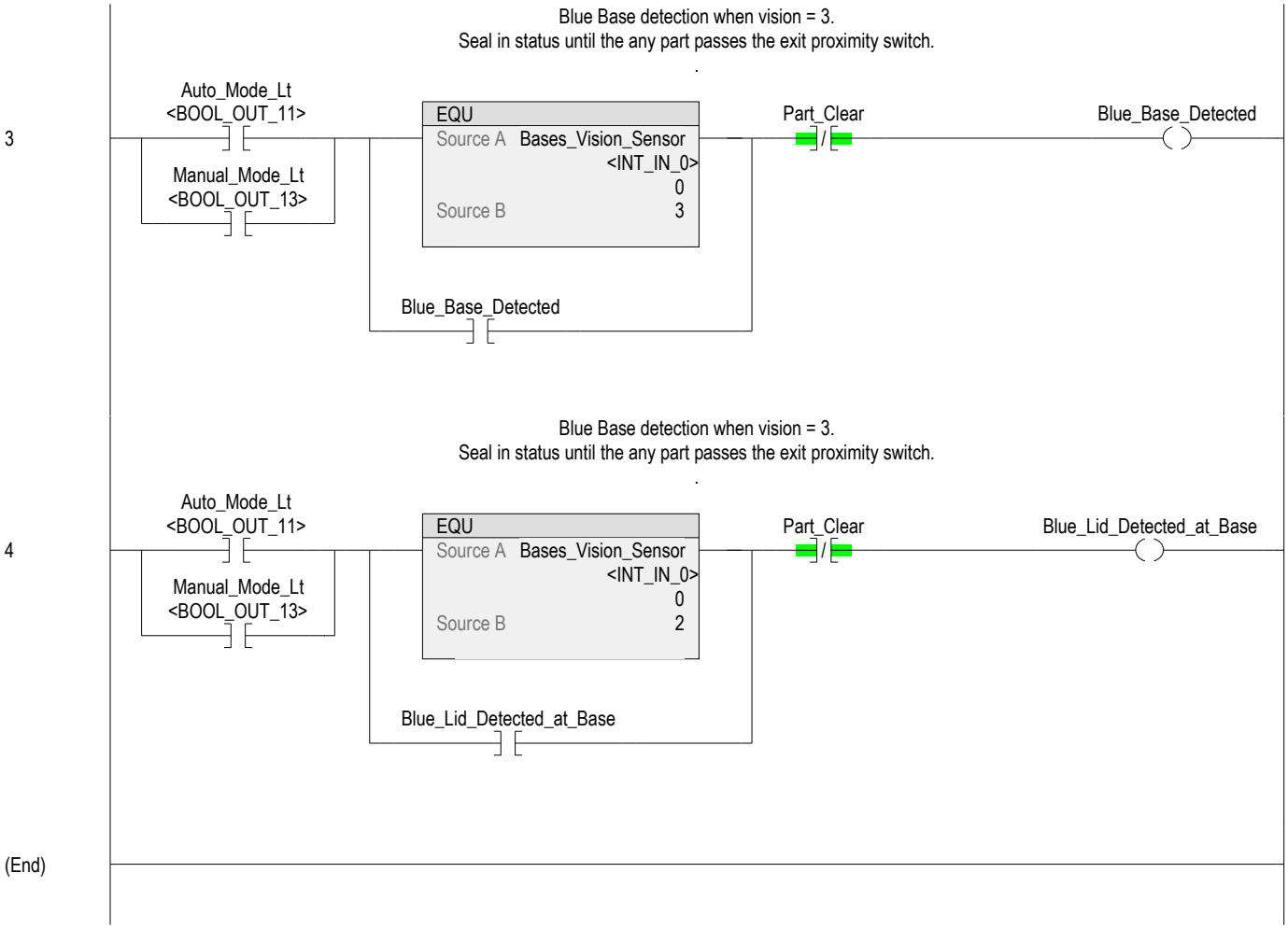
11
Clamp_Lids <BOOL_OUT_4> | Lid_Clamped <BOOL_IN_4>

TON		
Timer	Lids_Unclamped_Dwell	EN
Preset	800	DN
Accum	800	


12
Lids_Unclamped_Prox
<BOOL_IN_35>
Clamp_Lids
<BOOL_OUT_4>
Factory_Running
<BOOL_IN_15>
Lids_Unclamped_Lt
<BOOL_OUT_26>
Lids_Unclamped_Dwell.DN
Lids_Unclamped_Lt
<BOOL_OUT_26>
Bases Clamp Position Indicators.
Same explanation as the Lids Clamp.
13
Base_Clamped
<BOOL_IN_7>
Bases_Clamped_Lt
<BOOL_OUT_27>
14
Clamp_Bases
<BOOL_OUT_7>
Base_Clamped
<BOOL_IN_7>
TON
Timer Bases_Unclamped_Dwell
Preset 800
Accum 800
EN
DN
15
Bases_Unclamped_Prox
<BOOL_IN_38>
Clamp_Bases
<BOOL_OUT_7>
Factory_Running
<BOOL_IN_15>
Bases_Unclamped_Lt
<BOOL_OUT_28>
Bases_Unclamped_Dwell.DN
Bases_Unclamped_Lt
<BOOL_OUT_28>
Lids Pos Position Indicators.
The Positioners output an "In Position" status which can be used in the same
way the Z and X Axis Moving are. However, in this example, the Lowered
Status is read directly from a user added Proximity Switch (recall the Z and X axis
Returning Memory Programming, although redundant, was left in the program).
16
Lids_Pos_Lowered_Prox
<BOOL_IN_34>
Lids_Pos_Lowered_Lt
<BOOL_OUT_30>
17
Pos_Raise_Lids
<BOOL_OUT_5>
Lids_Pos_at_Limit
<BOOL_IN_36>
Lids_Pos_Raised_Lt
<BOOL_OUT_29>
Factory_Running
<BOOL_IN_15>
Lids_Pos_Raising_Memory
Lids_Pos_Raising_Memory
18
Lids_Pos_Raising_Memory
Lids_Pos_at_Limit
<BOOL_IN_36>
Pos_Raise_Lids
<BOOL_OUT_5>
Factory_Running
<BOOL_IN_15>
Lids_Pos_Raised_Lt
<BOOL_OUT_29>
Lids_Pos_Raised_Lt
<BOOL_OUT_29>
Bases Pos Position Indicators
19
Bases_Pos_Lowered_Prox
<BOOL_IN_37>
Bases_Pos_Lowered_Lt
<BOOL_OUT_32>


20
Pos_Raise_Bases
<BOOL_OUT_8>
Bases_Pos_at_Limit
<BOOL_IN_39>
Bases_Pos_Raised_Lt
<BOOL_OUT_31>
Factory_Running
<BOOL_IN_15>
Bases_Pos_Raising_Memory
Bases_Pos_Raising_Memory
21
Bases_Pos_Raising_Memory
Bases_Pos_at_Limit
<BOOL_IN_39>
Pos_Raise_Bases
<BOOL_OUT_8>
Factory_Running
<BOOL_IN_15>
Bases_Pos_Raised_Lt
<BOOL_OUT_31>
Bases_Pos_Raised_Lt
<BOOL_OUT_31>
22
Green_Base_Detected
Good_Base_Detected
<BOOL_OUT_42>
23
Green_Lid_Detected
Good_Lid_Detected
<BOOL_OUT_41>
24
Lid_In_Place_Tmr.DN
Good_Assembly
<BOOL_OUT_43>
(End)


Green Lid detection when vision = 5.
Seal in status until the Part is Clear but only if the Part is an assembly. Blue bases also trigger the Part Clear Bit.
Lid Clear could be added if a Green. Lid somehow did not stop and continued down the conveyor.
0
Auto_Mode_Lt
<BOOL_OUT_11>
Manual_Mode_Lt
<BOOL_OUT_13>
EQU
Source A Lids_Vision_Sensor
<INT_IN_1>
0
Source B 5
Part_Clear
Lid_Placed
Green_Lid_Detected
Green_Lid_Detected
Changed from Lab3A due to differnce
in parts being assembled.
Blue Lid detection when vision = 2.
Seal in status until the Blue Lid bypasses the clamp.
1
Auto_Mode_Lt
<BOOL_OUT_11>
Manual_Mode_Lt
<BOOL_OUT_13>
EQU
Source A Lids_Vision_Sensor
<INT_IN_1>
0
Source B 2
Lid_Clear
Blue_Lid_Detected
Blue_Lid_Detected
Green Base detection when vision = 6.
Seal in status until the any part passes the exit proximity switch.
2
Auto_Mode_Lt
<BOOL_OUT_11>
Manual_Mode_Lt
<BOOL_OUT_13>
EQU
Source A Bases_Vision_Sensor
<INT_IN_0>
0
Source B 6
Part_Clear
Green_Base_Detected
Green_Base_Detected


Blue Base detection when vision = 3.
Seal in status until the any part passes the exit proximity switch.
3
Auto_Mode_Lt
<BOOL_OUT_11>
Manual_Mode_Lt
<BOOL_OUT_13>
EQU
Source A Bases_Vision_Sensor
<INT_IN_0>
0
Source B 3
Part_Clear
Blue_Base_Detected
Blue_Base_Detected
Blue Base detection when vision = 3.
Seal in status until the any part passes the exit proximity switch.
4
Auto_Mode_Lt
<BOOL_OUT_11>
Manual_Mode_Lt
<BOOL_OUT_13>
EQU
Source A Bases_Vision_Sensor
<INT_IN_0>
0
Source B 2
Part_Clear
Blue_Lid_Detected_at_Base
Blue_Lid_Detected_at_Base
(End)

Send the value of the emitted part back to the upstream PLC to confirm commuication
and that a part has start to leave the Emitter. If the Simulation is not running, the confirmation will not be sent.

0

MOV
Source Bases_53_to_52
0
Dest Bases_Emitter_Parts_to_Emit
<INT_OUT_3>
0

Send to value of the Part in the Remover to the Downstream PLC.

1

MOV
Source Bases_Emitter_Parts_to_Emit
<INT_OUT_3>
0
Dest Bases_52_to_53
0

Send the data from the upstream PLC to the Emitter.

2

MOV
Source Lids_53_to_52
0
Dest Lids_Emitter_Part_to_Emit
<INT_OUT_4>
0

Once the downstream PLC recieved the part, the data is sent back so that part can be removed.

3

MOV
Source Lids_Emitter_Part_to_Emit
<INT_OUT_4>
0
Dest Lids_52_to_53
0

(End)