CHAPTER 6 SOLVED PROBLEMS

General instructions for the problems:

Using the function chart approach, write a ladder logic program for the application. Implement it for one of the following PLC ladder logic languages

ControlLogix/CompactLogix, **or**

Micro800, **or**

MicroLogix, **or**

SLC-500

If any part of the operation is ambiguous, write down your additional assumptions.

The physical inputs, physical outputs, and internal tags/symbols for each problem are given in the problem. **DO NOT** assign any more physical inputs!

Your solution should include the following:

1. Function chart of the process, showing the transitions between steps and the outputs active (or **on**) during each step.
2. Specify the PLC processor used.
3. Ladder logic diagram (with comments). For consistency among the different PLCs, use only tags/symbols in the ladder logic. Use instructions consistent with the PLC processor.
4. Table listing additional internal memory (tags/symbols) used and a brief description of their use. For the ControlLogix, CompactLogix, and Micro800 processors, list the internal tags/variables and the data type. For the MicroLogix/SLC-500 processors, list the internal variables/symbols and the associated memory address.

SP6-1. Carton Sealer Control. Implement the program for the following station that folds and seals a corrugated cardboard box.

Figure SP6.1 shows two views of a station that folds and glues a corrugated cardboard box. Assume the open boxes are already filled with product. Upon startup, the conveyor motor is **on** until the photoelectric eye senses a box in the station. When a box is in the station, the conveyor is stopped, and the box is closed and sealed by the following procedure. Two pneumatic rams (FRONT and BACK) are extended to push the two end flaps down, a glue sprayer is activated for 1 second, and then two more pneumatic rams (LEFT and RIGHT) are extended to push the two side flaps down. After a 5 second wait, all of the pneumatic rams are retracted, and the conveyor motor is turned **on**. The sealed box is moved out of the station, and the conveyor continues to run until a new box moves into the station. Thus, the operation repeats. Assume there is a gap between the boxes, so that the photoelectric eye will also sense that there is no box in the station after the sealed box is moved out.

Each ram is a single-action linear pneumatic cylinder controlled by one output. Once an output is energized, the ram extends and keeps moving as long as power is applied (turned

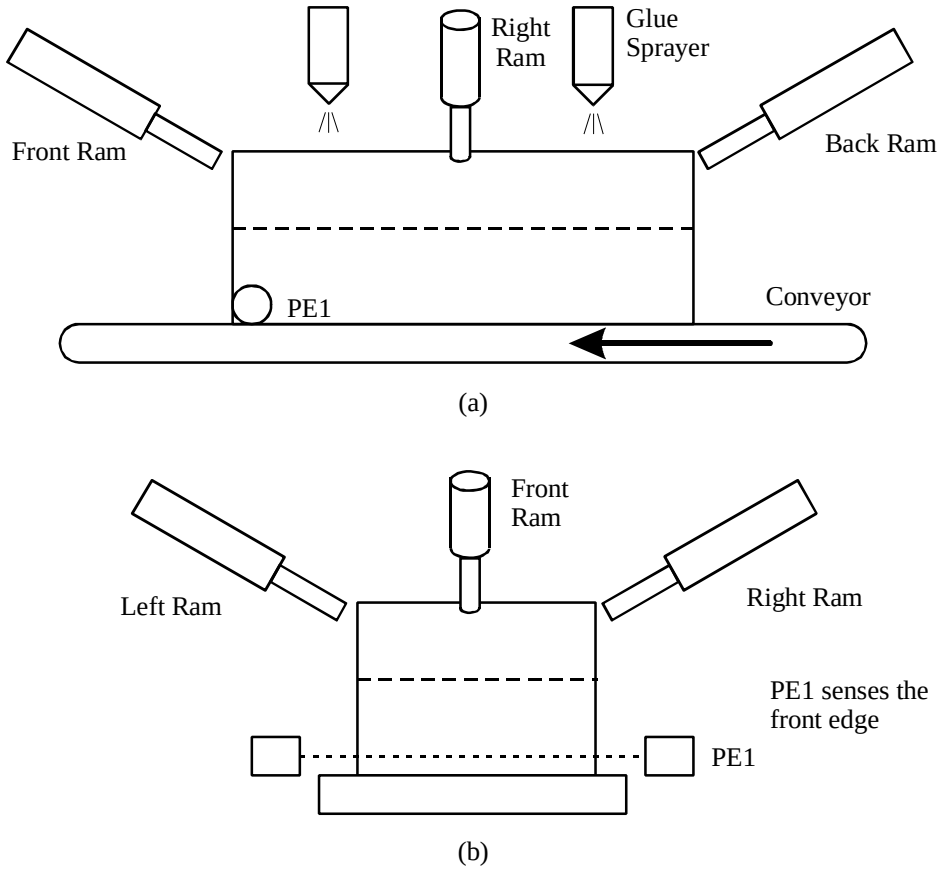


Figure SP6.1. Box sealing station: *(a)* view from left side; *(b)* view from front.

on) or a mechanical stop is reached. The ram retracts when power is removed (turned **off**). Each ram has a limit switch to detect when a ram is fully extended (out). There is no limit switch to detect when a ram is retracted. Assume that 2 seconds is sufficient time to retract a ram when its control is turned **off**.

When the start switch is pressed (turned **on**) for the first time only, the station assumes there is no box in the station, and waits for the photoelectric eye to detect the first box and perform the operation cycle continuously. Pressing the start switch when the mechanism is already running must have no effect. When the stop switch is pressed (turned **off**), the operation should stop (pause), but only when the conveyor belt is moving. The operation MUST not be stopped while the box flaps are being pushed down or being glued. Pressing the start switch while the operation of the station is paused causes the station to resume its suspended operation.

There is a RESET_PB switch that when **on**, restarts the operation. When RESET_PB is **on**, the internal state is set so that the ladder logic program assumes there is no corrugated box at the station. RESET_PB must be ignored if the operation is running. RESET_PB only has effect when the operation is already paused.

Assume the following physical inputs and outputs.

Tag/Var./Symbol	Description
START_PB	Start push button, N. O., **on** when starting.
STOP_PB	Stop push button, N. C., **off** when stopping.
RESET_PB	Reset push button, N. O., **on** when restoring station to initial state.
PE1	Photoelectric sensor, **off** (open) when box is in station.
FRONT_ELS	Front flap ram extended limit switch, **on** when extended.
BACK_ELS	Back flap ram extended limit switch, **on** when extended.
RIGHT_ELS	Right flap ram extended limit switch, **on** when extended.
LEFT_ELS	Left flap ram extended limit switch, **on** when extended.
CONV_MOTOR	Conveyor motor control, **on** to move conveyor.
GLUE_SPRAY	Glue sprayer control, **on** to spray glue.
FRONT_EXT	Front flap ram extend direction control, **on** to extend ram.
BACK_EXT	Back flap ram extend direction control, **on** to extend ram.
RIGHT_EXT	Right flap ram extend direction control, **on** to extend ram.
LEFT_EXT	Left flap ram extend direction control, **on** to extend ram.

The addresses associated with the physical inputs and outputs are:

Tag/Var./Symbol	ControlLogix	Micro800	MLogix	SLC-500
START_PB	Local:1:I.Data.0	_IO_EM_DI_00	I:0/0	I:01/0
STOP_PB	Local:1:I.Data.1	_IO_EM_DI_01	I:0/1	I:01/1
RESET_PB	Local:1:I.Data.2	_IO_EM_DI_02	I:0/2	I:01/2
PE1	Local:1:I.Data.3	_IO_EM_DI_03	I:0/3	I:01/3
FRONT_ELS	Local:1:I.Data.4	_IO_EM_DI_04	I:0/4	I:01/4
BACK_ELS	Local:1:I.Data.5	_IO_EM_DI_05	I:0/5	I:01/5
RIGHT_ELS	Local:1:I.Data.6	_IO_EM_DI_06	I:0/6	I:01/6
LEFT_ELS	Local:1:I.Data.7	_IO_EM_DI_07	I:0/7	I:01/7
CONV_MOTOR	Local:2:O.Data.0	_IO_EM_DO_00	O:0/0	O:2/0
GLUE_SPRAY	Local:2:O.Data.1	_IO_EM_DO_01	O:0/1	O:2/1
FRONT_EXT	Local:2:O.Data.2	_IO_EM_DO_02	O:0/2	O:2/2
BACK_EXT	Local:2:O.Data.3	_IO_EM_DO_03	O:0/3	O:2/3
RIGHT_EXT	Local:2:O.Data.4	_IO_EM_DO_04	O:0/4	O:2/4
LEFT_EXT	Local:2:O.Data.5	_IO_EM_DO_05	O:0/5	O:2/5

SP6-2. Batch Process Control. Implement the program for the following batch process that mixes two chemicals.

A diagram of the equipment is shown in Figure SP6.2. When the start switch is pressed (turned **on**) for the first time to start the operation, the tank is filled with Ingredient A up to Level A. After a 2 second wait, the tank is filled with Ingredient B up to Level B. Assume Level B is higher than Level A. After another 2 second wait, the stirrer motor is turned **on**

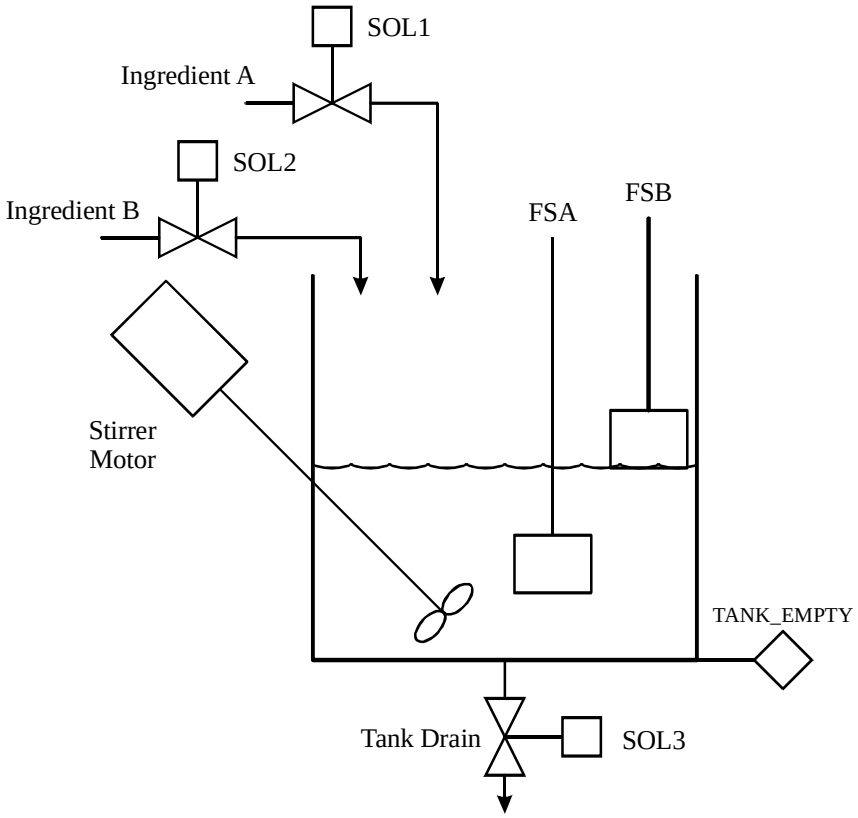


Figure SP6.2. Batch process.

for 10 minutes. Then the tank is emptied. The TANK_EMPTY sensor must be continuously **on** for 1 minute to ensure that the tank is completely empty. After the tank is empty, the process operation is finished and does not start again until the start push button is pressed. The operation does not repeat.

Pressing the start switch when the operation is already running must have no effect. When the stop switch is pressed (turned **off**) the operation should pause at the current step and all outputs must be **off**. All timer values must be retained during pause. Pressing the start switch while the operation of the station is paused causes the station to resume its suspended step.

A separate override switch is provided to drain the remainder of the material in the tank if the operation has been paused. The override switch should be ignored if the operation is not already paused. When the override switch is pressed to empty the tank, the emptying operation should continue until the TANK_EMPTY sensor is **on** for 1 minute to ensure that the tank is completely empty. The operation cannot be restarted until the tank emptying is complete.

Assume the following physical inputs and outputs.

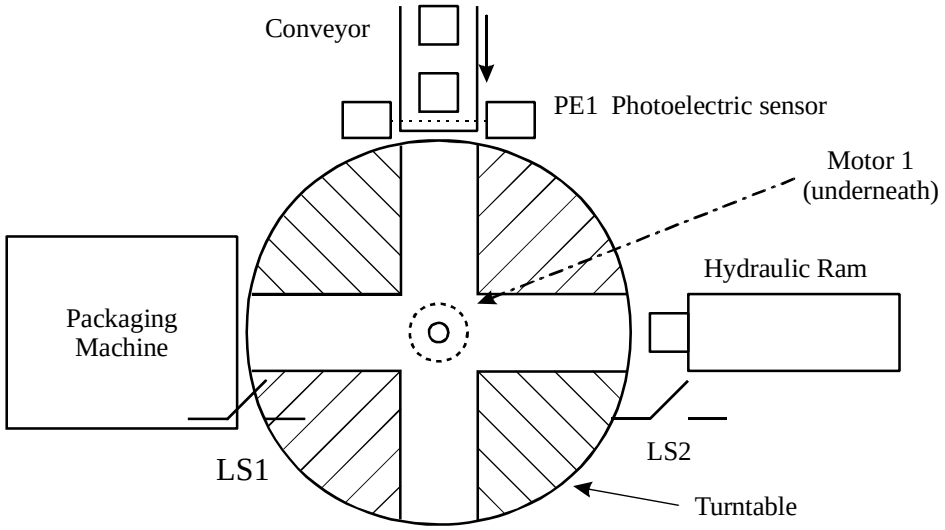
Tag/Var./Symbol	Description
START_PB	Start push button, N. O., **on** when starting.
STOP_PB	Stop push button, N. C., **off** when stopping.
FSA	Float level A sensor, N. O., **on** (closed) when level of material in tank is at least level A.
FSB	Float level B sensor, N. O., **on** (closed) when level of material in tank is at least level B.
TANK_EMPTY	Tank empty sensor, on (closed) when tank is empty.
OVERRIDE	Outlet override, N. O., **on** (closed) to start tank draining.
SOL1	Ingredient A fill solenoid, **on** to allow Ingredient A to flow into tank.
SOL2	Ingredient B fill solenoid, **on** to allow Ingredient B to flow into tank.
SOL3	Tank outlet solenoid, **on** to empty tank.
STIRRER	Stirrer motor control, **on** to run stirrer.

The addresses associated with the physical inputs and outputs are:

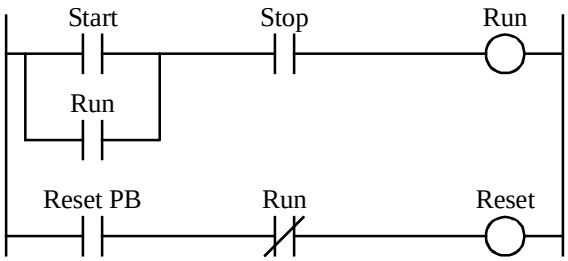
Tag/Var./Symbol	ControlLogix	Micro800	MLogix	SLC-500
START_PB	Local:1:I.Data.0	_IO_EM_DI_00	I:0/0	I:1/0
STOP_PB	Local:1:I.Data.1	_IO_EM_DI_01	I:0/0	I:1/1
FSA	Local:1:I.Data.2	_IO_EM_DI_02	I:0/2	I:1/2
FSB	Local:1:I.Data.3	_IO_EM_DI_03	I:0/3	I:1/3
TANK_EMPTY	Local:1:I.Data.4	_IO_EM_DI_04	I:0/4	I:1/4
OVERRIDE	Local:1:I.Data.5	_IO_EM_DI_05	I:0/5	I:1/5
SOL1	Local:2:O.Data.0	_IO_EM_DO_00	O:0/0	O:2/0
SOL2	Local:2:O.Data.1	_IO_EM_DO_01	O:0/1	O:2/1
SOL3	Local:2:O.Data.2	_IO_EM_DO_02	O:0/2	O:2/2
STIRRER	Local:2:O.Data.3	_IO_EM_DO_03	O:0/3	O:2/3

SP6-3. Parts Transfer Station Control. Implement the program for the following station that transfers parts from a conveyor to a packaging machine.

Figure SP6.3*a* shows the layout of a station that transfer parts from a conveyor to a packaging machine. In summary, 6 parts are loaded onto the turntable, the table is turned 90°, a hydraulic ram is extended to push the parts into a packaging machine, and the ram is retracted. Parts are placed on the belt conveyor by another machine. The parts move down the conveyor and then onto the turntable. Vertical walls on the turntable make sure the parts move straight in the turntable. The passage of parts to the turntable is detected by a photoelectric sensor, PE1, that turns OFF when a part interrupts the beam. PE1 turns OFF just as the part moves on to the turntable. After 6 parts are on the turntable, the turntable is rotated counterclockwise 90° by activating the turntable motor, MOTOR1, for 1 second. After the table is rotated, the hydraulic ram is extended by turning on SOL1 until limit switch LS1 closes. This operation pushes the parts into the packing machine. The ram is then retracted, by turning off SOL1, until limit switch LS2 closes. While the turntable is being rotated and the ram is moving, the belt conveyor motor, MOTOR2, must be stopped.



(a)



(b)

Figure SP6.3. Parts transfer station; *(a)* equipment; *(b)* ladder logic for Run and Reset.

There is an internal coil, Run, that is **on** when the operation is enabled. The Run internal coil is set by another part of the ladder logic. When the Run coil is **off**, all counter values must be retained, and the ladder logic program must remain in the state in which the Run coil changed from **on** to **off**. If the Run coil becomes **off** when moving the turntable, MOTOR1 must continue to run and transition to the next step when the time is complete. There is another internal coil, Reset, that when **on**, restarts the operation. The Run internal coil is set by another part of the ladder logic. When Reset is **on**, internal counters and timers are reset, and the internal state is set so that the ladder logic program assumes no parts are on the turntable. You may assume that the Reset can only be **on** when Run is **off**. The ladder logic for the Run and Reset coils is shown in Figure SP6.3*b*.

Assume the following physical inputs, physical outputs, and internal coils.

Tag/Var./Symbol	Description
PE1	Photoelectric sensor, **off** (open) when part passes.
LS1	Ram extended limit switch, **on** (closed) when ram is extended

LS2	Ram retracted limit switch, **on** (closed) when ram is retracted
SOL1	Pneumatic ram extension solenoid control, **on** to extend ram, **off** retracts ram.
MOTOR1	Turntable motor control, **on** to turn turntable
MOTOR2	Belt conveyor motor control, **on** to run conveyor
Run	Internal coil, **on** when operation enabled to run (set by another part of the ladder logic)
Reset	Internal coil, **on** to reset operation (set by another part of the ladder logic)

The addresses associated with the physical inputs and outputs are:

Tag/Var./Symbol	ControlLogix	Micro800	MLogix	SLC-500
PE1	Local:1:I.Data.0	_IO_EM_DI_00	I:0/0	I:1/0
LS1	Local:1:I.Data.1	_IO_EM_DI_01	I:0/1	I:1/1
LS2	Local:1:I.Data.2	_IO_EM_DI_02	I:0/2	I:1/2
SOL1	Local:2:O.Data.0	_IO_EM_DO_00	O:0/0	O:2/0
MOTOR1	Local:2:O.Data.1	_IO_EM_DO_01	O:0/1	O:2/1
MOTOR2	Local:2:O.Data.2	_IO_EM_DO_02	O:0/2	O:2/2

The internal tag/symbol data types or addresses are:

	CLogix	Micro800	MLogix/SLC500
Tag/Var./Symbol	Data Type	Data Type	Address
Run	BOOL	BOOL	B3/1
Reset	BOOL	BOOL	B3/2

SP6-4. Bag Sealing Station Control. Implement the program for the following station that seals plastic bags.

Figure SP6.4 shows the layout of a station that seals plastic bags. Assume the bags are already filled with product. Upon startup, the conveyor motor is ON until the photoelectric "eye" senses a bag in the station. When a bag is in the station, the conveyor is stopped, and the bag is sealed by the following steps:

1. A mechanism using air cylinders is used to push two heated bars together (one bar on each side of the bag),
2. The bars are held together for 1 second, and
3. The bars are moved apart.

The mechanism used to move the heated bars is driven by a double-action linear pneumatic cylinder controlled by two outputs. Once a direction output is energized, the mechanism moves and keeps moving as long as power is applied (turned **on**). The mechanism stops at its current position when power is removed (turned **off**). The mechanism will not move if both opposing directions are energized simultaneously (e.g., out and in). Limit switch LS1 is **on** when the two heated bars are together. Limit switch LS2 is **on** when the bars have been moved sufficiently far apart. When the bag-sealing operation is complete (bars have been moved apart), the conveyor motor is turned **on**. The sealed bag is moved out of the station, and the conveyor continues to run until a new bag moves into the station. Thus, the operation repeats. Assume there is a gap between the bags, so that the

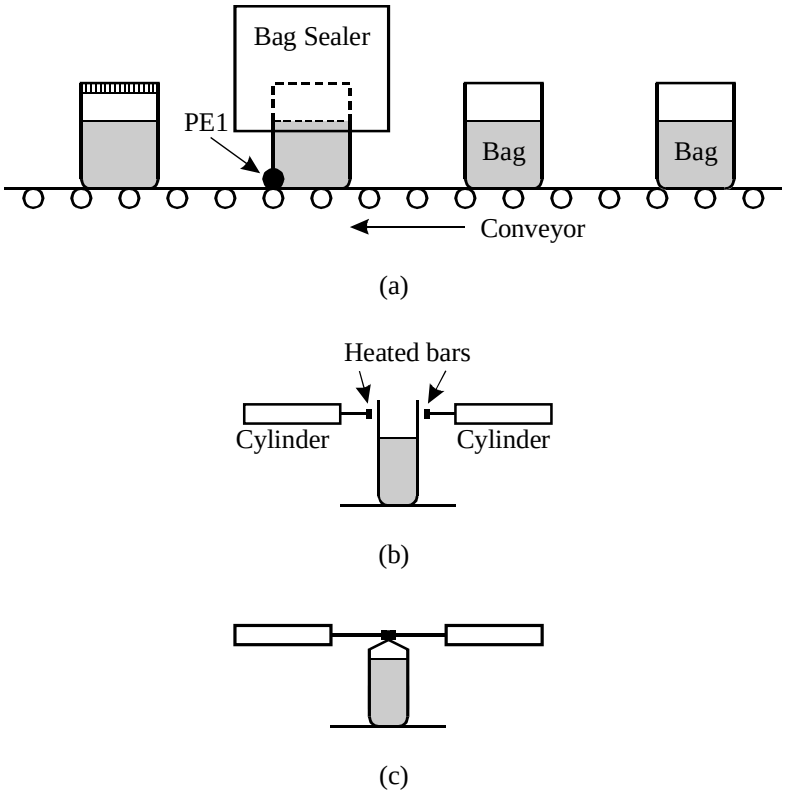


Figure SP6.4. Bag sealing station; *(a)* front view; *(b)* view from side, bars apart; *(c)* view from side, bars together.

photoelectric "eye" will also sense that there is no bag in the station after the sealed bag is moved out.

When the start switch is pressed (turned **on**) for the first time only, the station assumes there is no bag in the station, and waits for the photoelectric "eye" to detect the first bag and perform the operation cycle continuously. Pressing the start switch when the mechanism is already running must have no effect. When the stop switch is pressed (turned **off**), the operation should stop (pause), but only when the conveyor belt is moving. The operation MUST not be stopped while the bars are being held together or when they are being moved. Pressing the start switch while the operation of the station is paused causes the station to resume its suspended operation.

There is a RESET_PB switch that when **on**, restarts the operation. When RESET_PB is **on**, the internal state is set so that the ladder logic program assumes there is no plastic bag at the station. RESET_PB must be ignored if the station is not already paused. When the station is paused and RESET_PB is pressed and then released, the next press of the start switch is treated as the first time the start switch is pressed.

Assume the following physical inputs and outputs.

Tag/Var./Symbol	Description
START_PB	Start push button, N. O., **on** when starting.
STOP_PB	Stop push button, N. C., **off** when stopping.
RESET_PB	Reset push button, N. O., **on** when restoring station to initial state.
PE1	Photoelectric sensor, **off** (open) when bag is in station.
LS1	Limit switch, **on** (closed) when two heated bars are together
LS2	Limit switch, **on** (closed) when two heated bars sufficiently far apart
MOTOR	Conveyor motor control, **on** to move conveyor
BAR_IN	Heated bar mechanism in control, **on** to move bars together
BAR_OUT	Heated bar mechanism out control, **on** to move bars apart

The addresses associated with the physical inputs and outputs are:

Tag/Var./Symbol	ControlLogix	Micro800	MLogix	SLC-500
START_PB	Local:1:I.Data.0	_IO_EM_DI_00	I:0/0	I:1/0
STOP_PB	Local:1:I.Data.1	_IO_EM_DI_01	I:0/1	I:1/1
RESET_PB	Local:1:I.Data.2	_IO_EM_DI_02	I:0/2	I:1/2
PE1	Local:1:I.Data.3	_IO_EM_DI_03	I:0/3	I:1/3
LS1	Local:1:I.Data.4	_IO_EM_DI_04	I:0/4	I:1/4
LS2	Local:1:I.Data.5	_IO_EM_DI_05	I:0/5	I:1/5
MOTOR	Local:2:O.Data.1	_IO_EM_DO_00	O:0/1	O:2/1
BAR_IN	Local:2:O.Data.3	_IO_EM_DO_03	O:0/3	O:2/3
BAR_OUT	Local:2:O.Data.4	_IO_EM_DO_04	O:0/4	O:2/4

P6-5. Erbia Elevator Control. Using the function chart approach, implement the program for the following elevator that moves cans of erbia (a metal) powder from the first floor to the second floor of the factory.

The memo below describes the operation of the erbia elevator. A simplified drawing of the elevator is shown in Figure SP6.5.

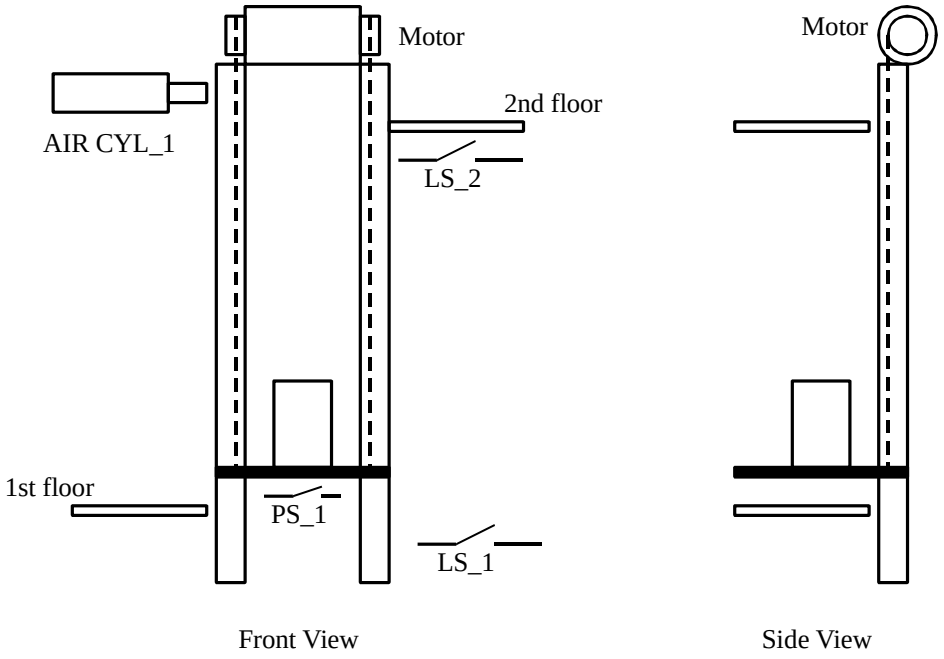
To: A. Doe

From: B. Smith

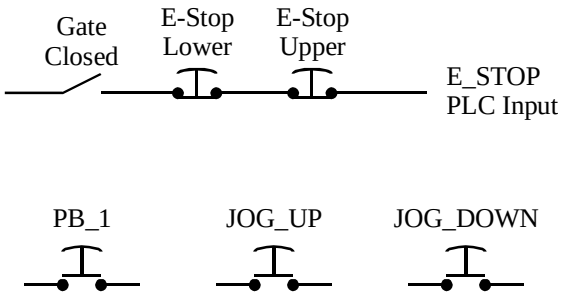
Subject: Controls for Erbia Elevators

We need to design an elevator to move cans of erbia from the first floor to the second floor. Refer to the drawing of the elevators for switch reference numbers. The general operation is described as follows:

1. The platform is at the lower position so that LS_1 is actuated.
2. Press PB_1 switch. The motor starts in forward and drives the elevator to the up position where LS_2 is actuated. The motor stops.
3. AIR_CYL_1 is actuated and drives out for a limited time. Then it reverses to the retracted position.
4. PS_1 (off) indicates that the elevator is clear.



(a)



(b)

Figure SP6.5. Erbia elevator: *(a)* equipment; *(b)* safety-related and operator switches..

5. The motor starts in reverse to lower the platform to the bottom to activate LS_1. The motor stops.
6. The jog buttons drive the motor up or down between the limit switches, but not when in automatic control. These switches are intended for moving the elevator after an emergency stop and for drive troubleshooting. They should be ignored if the motor is under automatic control.

Notes to operation memo:

1. The two motor controls (MOTOR_UP and MOTOR_DOWN) connect to a motor controller that drives the motor.
2. Only one output, AIR CYL_1, is used to control the pneumatic cylinder (ram) that pushes the can off the elevator. When it is **on**, the ram moves out to its fully extended position and remains there (as long as AIR CYL_1 stays **on**). When **off**, the ram moves to the retracted position, if not already there. There are no switches to detect when the ram is fully extended or retracted. Assume that 10 seconds are required to fully extend the ram and 15 seconds will assure that it is fully retracted when AIR CYL_1 is turned OFF.
3. Pressing the JOG_UP switch should have no effect if the platform is in the upper position (top) or when the JOG_DOWN switch is already pressed. Also, pressing the JOG_DOWN switch should have no effect if the platform is in the lower position (bottom) or when the JOG_UP switch is already pressed. Both switches should have no effect if the elevator is moving between floors because the operator pressed the PB_1 switch.
4. Since the gate closed switch and the two emergency stop switches are wired in series to one PLC input, the ladder logic can only see them as one switch, E_STOP. When either E-stop switch is pressed (E_STOP input is **off**), the elevator should immediately stop, and the JOG_UP and JOG_DOWN switches must be used to move the elevator. When the E_STOP input is **off**, the JOG_UP and JOG_DOWN switches should be ignored.
5. Do not be concerned with how the cans are placed on the platform when it is in the lower position.

Assume the following physical inputs and outputs.

Tag/Var./Symbol	Description
LS_1	Limit switch that closes (**on**) when elevator platform is in the lower position
LS_2	Limit switch that closes (**on**) when elevator platform is in the upper position
PS_1	Limit switch that closes (**on**) when a can is on the elevator platform
PB_1	N. O. push button, **on** when starting the elevator operation.
E_STOP	N. C. push button, **off** when stopping.
JOG_UP	N. O. push button, **on** to jog (manually move) platform up
JOG_DOWN	N. O. push button, **on** to jog move platform down
AIR_CYL_1	Pneumatic ram extension solenoid, **on** to extend ram, **off** causes ram to retract
MOTOR_UP	Elevator motor control, **on** to cause elevator platform to move up, **off** has no effect on platform position
MOTOR_DOWN	Elevator motor control, **on** to cause elevator platform to move down, **off** has no effect on platform position

The addresses associated with the physical inputs and outputs are:

Tag/Var./Symbol	ControlLogix	Micro800	MLogix	SLC-500
LS_1	Local:1:I.Data.0	_IO_EM_DI_00	I:0/0	I:1/0
LS_2	Local:1:I.Data.1	_IO_EM_DI_01	I:0/1	I:1/1
PS_1	Local:1:I.Data.2	_IO_EM_DI_02	I:0/2	I:1/2
PB_1	Local:1:I.Data.3	_IO_EM_DI_03	I:0/3	I:1/3
E_STOP	Local:2:I.Data.0	_IO_EM_DI_04	I:1/0	I:2/0
JOG_UP	Local:2:I.Data.1	_IO_EM_DI_05	I:1/1	I:2/1
JOG_DOWN	Local:2:I.Data.2	_IO_EM_DI_06	I:1/2	I:2/2
AIR_CYL_1	Local:3:O.Data.0	_IO_EM_DO_00	O:0/0	O:3/0
MOTOR_UP	Local:3:O.Data.1	_IO_EM_DO_01	O:0/1	O:3/1
MOTOR_DOWN	Local:3:O.Data.2	_IO_EM_DO_02	O:0/2	O:3/2

SP6-6. Transfer Station Control. Using the function chart approach, implement the program for the following station that groups and transfers parts from one conveyor to another conveyor.

Figure SP6.6 shows the layout of a station that transfers items from one conveyor to another. The items are long and actually hang over the sides of the conveyor so that they can be easily picked up and transferred to the other conveyor. In summary, four items are collected at the end of the inbound conveyor, and a mechanism executes the following moves to transfer the items to the outbound conveyor:

Move up 5 seconds (to lift group of items),

Move right to the right position,

Move down to the lower position (to put items on outbound conveyor),

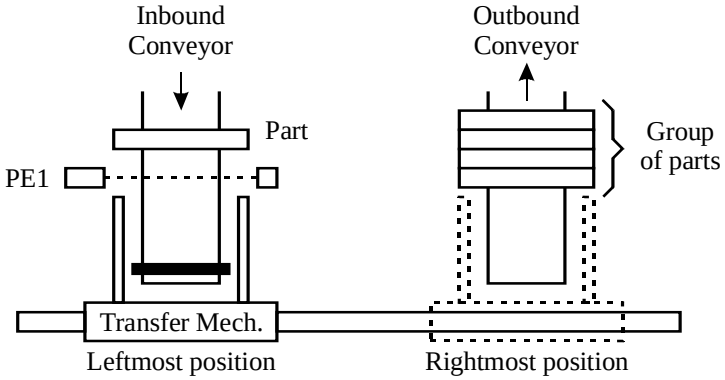
Move left to the left position.

The directions assume a person is looking up the inbound conveyor and down the outbound conveyor. In the lower position, the arms can move horizontally without interference (crashing into the conveyors). The operation repeats.

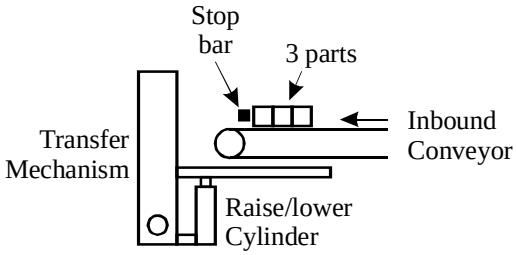
Photoelectric sensor PE1 is used to detect the items as they near the end of the inbound conveyor. PE1 turns **off** as each part passes. There is a bar above the conveyor against which the items collect before they are transferred to the other conveyor. PE1 detects the fourth item (PE1 turns **off**) just before the item hits the three already stopped, and does not turn back **on** until the group of items has been moved off the inbound conveyor.

Limit switches LSRT and LSLT determine the right and left positions, respectively, of the transfer mechanism. Limit switch LSDN determines the lower (down) position of the transfer mechanism. There is no limit switch that determines the upper position of the transfer mechanism.

The horizontal and vertical movements of the transfer mechanism are each driven by a double-action linear pneumatic cylinder — a raising/lowering unit and a rightward/leftward unit. Each cylinder is controlled by two outputs. Once a direction output is energized, the mechanism moves and keeps moving as long as power is applied (turned **on**). The mechanism stops at its current position when power is removed (turned **off**). The mechanism will not move if both opposing directions are energized simultaneously (e.g., left and right).



(a)



(b)

Figure P6.6. Transfer station: *(a)* top view; *(b)* side view.

The inbound conveyor motor control, IN_MOTOR must be **on** whenever the station is running. The inbound conveyor is not turned off while the transfer mechanism is moving. Assume the items are spaced far enough that the transfer mechanism returns to the inbound conveyor before the next item comes down the conveyor. Your ladder has no control over the outbound conveyor motor.

Upon initial startup, assume there are no items present at the end of the inbound conveyor. If the stop switch is pressed at any time, the station operation should pause, except when the transfer mechanism is being lowered to place the items on the outbound conveyor. The operation **must not** pause when the mechanism is being lowered (otherwise the parts may not remain as a group if the conveyor catches one that is just barely touching the conveyor). When the start switch is pressed while the operation is paused, the station should resume the suspended step. When the station is paused, all active outputs are turned off.

A separate reset switch is provided which resets any internal states so that when the start switch is pressed, no items are assumed present at the collection bar. The reset switch should be ignored if the station is running. The operator is responsible for using the four jog switches to move the transfer mechanism back to the starting position. The program must

ensure that the operator must hold down the reset push button (station paused) while manipulating the jog switches.

Assume the tolerance on all timer values is 0.1 seconds.

Assume the following physical inputs and outputs.

Tag/Var./Symbol	Description
START_PB	Start push button, N. O., **on** when starting.
STOP_PB	Stop push button, N. C., **off** when stopping.
RESET_PB	Reset push button, N. O., **on** when restoring station to initial state.
PE1	Photoelectric sensor, **off** (open) as item passes (see above).
LSDN	Limit switch that closes (**on**) when transfer mechanism is in the lower position
LSRT	Limit switch that closes (**on**) when transfer mechanism is in the rightmost position
LSLT	Limit switch that closes (**on**) when transfer mechanism is in the leftmost position
JOG_UP	Jog up button, N. O., **on** to move mechanism up when reset.
JOG_DN	Jog down button, N. O., **on** to move mechanism down when reset.
JOG_LT	Jog left button, N. O., **on** to move mechanism left when reset.
JOG_RT	Jog right button, N. O., **on** to move mechanism right when reset.
UP_CYL	Raising control, **on** to raise mechanism, **off** has no effect.
DN_CYL	Lowering control, **on** to lower mechanism, **off** has no effect.
LT_CYL	Left motion control, **on** to move mechanism left, **off** has no effect.
RT_CYL	Right motion control, **on** to move mechanism right, **off** has no effect.
IN_MOTOR	Inbound conveyor motor control, **on** to move conveyor.

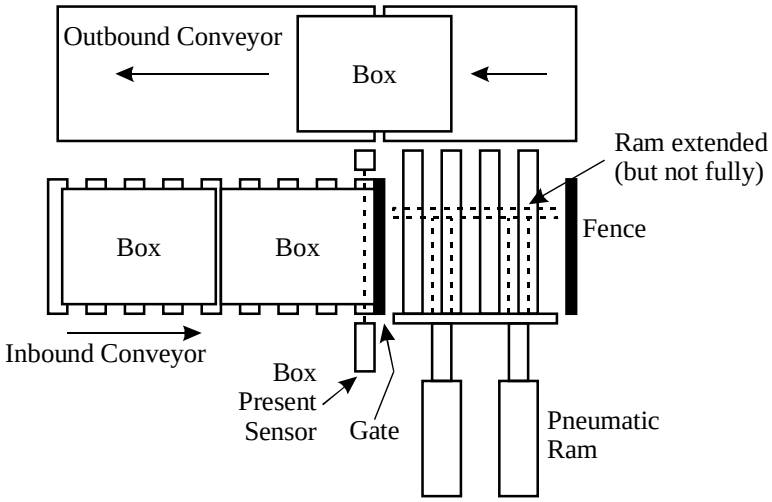
The addresses associated with the physical inputs and outputs are:

Tag/Var./Symbol	ControlLogix	Micro800	MLogix	SLC-500
START_PB	Local:2:I.Data.0	_IO_EM_DI_00	I:0/0	I:2/0
STOP_PB	Local:2:I.Data.1	_IO_EM_DI_01	I:0/1	I:2/1
RESET_PB	Local:2:I.Data.2	_IO_EM_DI_02	I:0/2	I:2/2
PE1	Local:2:I.Data.3	_IO_EM_DI_03	I:0/3	I:2/3
LSDN	Local:2:I.Data.4	_IO_EM_DI_04	I:0/4	I:2/4
LSRT	Local:2:I.Data.5	_IO_EM_DI_05	I:0/5	I:2/5
LSLT	Local:2:I.Data.6	_IO_EM_DI_06	I:0/6	I:2/6
JOG_UP	Local:2:I.Data.7	_IO_EM_DI_07	I:0/7	I:2/7
JOG_DN	Local:2:I.Data.8	_IO_EM_DI_08	I:0/8	I:2/8
JOG_LT	Local:2:I.Data.9	_IO_EM_DI_09	I:0/9	I:2/9
JOG_RT	Local:2:I.Data.10	_IO_EM_DI_10	I:0/10	I:2/10
UP_CYL	Local:3:O.Data.0	_IO_EM_DO_00	O:0/0	O:3/0

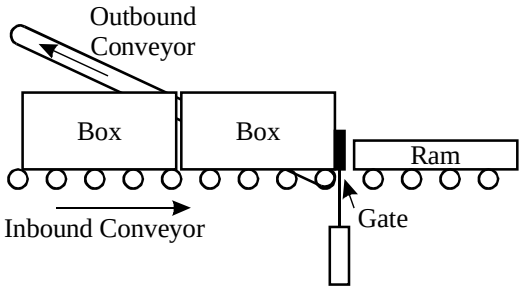
DN_CYL	Local:3:O.Data.1	_IO_EM_DO_01	O:0/1	O:3/1
LT_CYL	Local:3:O.Data.2	_IO_EM_DO_02	O:0/2	O:3/2
RT_CYL	Local:3:O.Data.3	_IO_EM_DO_03	O:0/3	O:3/3
IN_MOTOR	Local:3:O.Data.4	_IO_EM_DO_04	O:0/4	O:3/4

P6-7. Conveyor Transfer Station Control. Using a function chart approach, implement the program for the following station that transfers boxes from one conveyor to another.

Figure SP6.7 shows the layout of a station that transfer boxes from one conveyor to another. The inbound conveyor consists of rollers and has boxes placed upon it by other machines. The outbound conveyor is a belt and moves the boxes up to an overhead conveyor system. In summary, each box is individually pushed to the outbound conveyor. Of course, the next box is not moved until the station is ready to move it.



(a)



(b)

Figure SP6.7. Conveyor transfer station: *(a)* top view; *(b)* front view.

Upon initial start-up, the station waits for a box to arrive at the gate (indicated by BOX_PRESENT photoelectric sensor being **on**). The gate ensures that boxes are transferred one at a time. When a box is present, GATE_SOL is activated to hold down the gate so the box can move in front of the ram. When the box has passed the gate, BOX_PRESENT becomes **off** and then GATE_SOL is turned **off**, raising the gate and preventing the next box from entering until the first box is moved to the outbound conveyor. Assume that the box is moved past the gate faster than the box behind moves down the inbound conveyor, so that there is definitely a gap between boxes, if there is more than one box at the gate when it is moved down. After the box has passed the gate, there is a minimum 2 second delay to ensure that the box travels to the end of the conveyor where it is stopped by a fence. After the 2 second delay, a pneumatic ram is extended (using EXTEND_SOL) until LS2 operates (turns **on**). This action by the ram pushes the box onto the outbound conveyor. The ram is then retracted (using RETRACT_SOL) until LS1 operates (turns **on**), signaling that it is fully retracted. One cycle of the station operation is thus completed, and then it waits for the next box on the inbound conveyor.

The gate is moved with a single-action pneumatic cylinder, controlled by the GATE_SOL output. The pneumatic ram is a double-acting pneumatic cylinder controlled by the EXTEND_SOL and RETRACT_SOL outputs.

Both conveyor motor controls, MOTOR1 and MOTOR2 must be **on** whenever the station is running.

Upon initial startup, assume there are no boxes present at the gate. If the stop switch is pressed at any time, the station operation should pause, except when the gate is being held down. The operation **must not** pause when the gate is down (otherwise a box may be "flipped" when the gate solenoid is turned **off** upon reset.) If the stop pushbutton is pressed while the gate is down, the inbound conveyor should continue to run in order to move the box. After the gate is raised, the operation should pause. When the start switch is pressed while the operation is paused, the station should resume the suspended step. When the station is paused, all active outputs are turned **off** and timers paused (or reset). A separate reset switch is provided which retracts the ram and resets any internal steps so that when the start switch is pressed, no boxes are assumed present at the gate. The reset switch should be ignored when the gate is down and when the station is running. The start pushbutton should be ignored during reset (reset switch held down or ram retracting).

Assume the tolerance on all timer values is 0.1 seconds.

Assume the following physical inputs and outputs.

Tag/Var./Symbol	Description
START_PB	Start push button, N. O., **on** when starting.
STOP_PB	Stop push button, N. C., **off** when stopping.
RESET_PB	Reset push button, N. O., **on** when restoring station to initial state.
BOX_PRESENT	Photoelectric sensor, **on** (closed) when box is present at gate.
LS1	Limit switch that closes (**on**) when ram is fully retracted
LS2	Limit switch that closes (**on**) when ram is fully extended
GATE_SOL	Cylinder control to drop gate, **on** to hold down gate. When **off**, gate is held up by a spring.

EXTEND_SOL	Pneumatic ram extension solenoid, **on** to extend ram, **off** has no effect on ram position
RETRACT_SOL	Pneumatic ram retraction solenoid, **on** to retract ram, **off** has no effect on ram position
MOTOR1	Inbound conveyor motor control, **on** to move inbound conveyor
MOTOR2	Outbound conveyor motor control, **on** to move outbound conveyor

The addresses associated with the physical inputs and outputs are:

Tag/Var./Symbol	ControlLogix	Micro800	MLogix	SLC-500
START_PB	Local:2:I.Data.0	_IO_EM_DI_00	I:0/0	I:2/0
STOP_PB	Local:2:I.Data.1	_IO_EM_DI_01	I:0/1	I:2/1
RESET_PB	Local:2:I.Data.2	_IO_EM_DI_02	I:0/2	I:2/2
BOX_PRESENT	Local:2:I.Data.4	_IO_EM_DI_04	I:0/4	I:2/4
LS1	Local:2:I.Data.5	_IO_EM_DI_05	I:0/5	I:2/5
LS2	Local:2:I.Data.6	_IO_EM_DI_06	I:0/6	I:2/6
GATE_SOL	Local:3:O.Data.0	_IO_EM_DO_00	O:0/0	O:3/0
EXTEND_SOL	Local:3:O.Data.1	_IO_EM_DO_01	O:0/1	O:3/1
RETRACT_SOL	Local:3:O.Data.2	_IO_EM_DO_02	O:0/2	O:3/2
MOTOR1	Local:3:O.Data.3	_IO_EM_DO_03	O:0/3	O:3/3
MOTOR2	Local:3:O.Data.4	_IO_EM_DO_04	O:0/4	O:3/4

SP6-8. Hole Drilling Station 1 Control. Using a function chart approach, implement the program for the following station that drills a hole in each part on a conveyor.

Figure SP6.8 shows the layout of a station that drills a hole in each part that comes down the conveyor. The part completely fills the area enclosed in the dashed lines. This station is only one is a series of stations along this conveyor. You are implementing ladder logic for this station only. You have no control over the conveyor, so assume it is always moving. This particular line is asynchronous, that is, each station processes parts at its own speed and does not coordinate its operation with any other station. Because this is an asynchronous line, each station contains a series of two gates that control access to the station and allow parts to queue up before the station.

Upon initial startup, assume that there are no parts waiting at gate 1. When a part is detected at Gate 1 (by PROX1), the following major steps are executed:

Sequence Gate 1 and Gate 2 to allow only one part to move into the drilling position (against Gate 3).
Drill is turned and is extended (moves out) to the correct depth,
Drill is retracted,
Open Gate 3 to allow drilled part to move out.

The operation then repeats. Assume the conveyor is on at all times. The conveyor slides beneath the parts as they are held against a gate or being drilled.

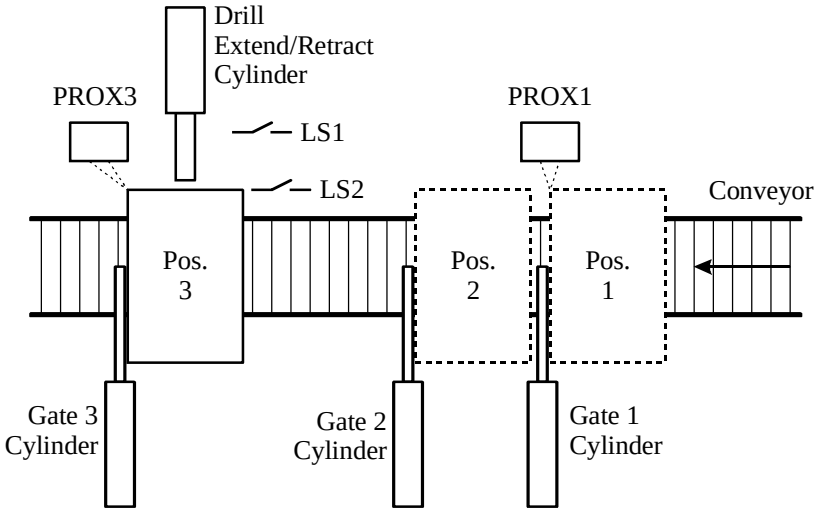


Figure **SP6.8.** Drilling station.

To move only one part into the station, Gates 1 and 2 are sequenced as follows (assume both gates are closed at the start): Gate 1 is opened to allow a part to move into position 2 (a 2 second delay allows this move). Then Gate 1 is closed and Gate 2 is opened. As far as your ladder logic is concerned, assume that Gate 1 is closed and Gate 2 is opened at the same time. Physically, Gate 1 closes much faster than Gate 2 opens, so any part in position 1 is prevented from moving when Gate 2 is open enough to allow the part to move from position 2 to position 3. Gate 2 is closed when the part is in position 3 (sensed by PROX3).

A single-action air cylinder drives each gate. Once a cylinder control is energized, the gate is opened and remains open as long as power is applied (turned **on**). The gate closes when power is removed (turned **off**).

The drill extension/retraction is driven by a double-action air cylinder with two direction controls. Once a direction control is energized, the drill mechanism moves and keeps moving as long as power is applied (turned **on**). The mechanism stops at its current position when power is removed (turned **off**). The mechanism will not move if both opposing directions are energized simultaneously (e.g., extension and retraction).

DRILL_MOTOR must be **on** whenever the drill is being extended or retracted.

Proximity sensor, PROX1, is **on** when a part is in position 1, meaning there is a part to be processed. PROX3 is **on** when a part is in position 3, ready to be drilled. When PROX3 is **off**, the part has passed Gate 3 and moved out of the station.

The drill position is indicated by limit switches. LS1 is **on** when the drill is fully retracted. LS2 is **on** when the drill is extended to the proper hole depth.

The start/stop switches are only for the station. They do not control any other stations or the conveyor. Upon initial startup, assume there are no parts present in any of the positions (1-3). If the stop switch is pressed at any time, the station operation should pause, except during drilling. The operation **must not** pause when the drill is being extended or retracted (otherwise it may jam or ruin the hole). When the start switch is pressed while the operation is paused, the station should resume the suspended step. If paused when not drilling, **do not**

advance to the next step. If you advance to the next step when paused, you will have problems with the gate operations. When the station is paused, the drill motor and the drill extension/retraction solenoids should not be affected. Any open gates must not be closed when paused (or parts may be knocked off).

A separate reset switch is provided that when pressed, a drill not fully retracted is retracted, and the process step is set as if the process is waiting for the next part. Note that the drill motor must remain **on** while a drill is retracting. When the start switch is pressed, no items are assumed present at position 1. To keep the problem simple, do not implement a function chart for reset; just retract the drill (motor **on**) until finished. The reset switch should have no effect unless the operation is already paused.

Assume the tolerance on all timer values is 0.1 seconds.

Assume the following physical inputs and outputs.

Tag/Var./Symbol	Description
START_PB	Start push button, N. O., **on** when starting.
STOP_PB	Stop push button, N. C., **off** when stopping.
RESET_PB	Reset push button, N. O., **on** when restoring station to initial state.
PROX1	Proximity sensor, **on** when part in position 1
PROX3	Proximity sensor, **on** when part in position 3
LS1	Limit switch that closes (**on**) drill is fully retracted
LS2	Limit switch that closes (**on**) when the drill is extended to the proper hole depth.
GATE_1	Gate 1 cylinder control, **on** to open gate 1, **off** closes gate.
GATE_2	Gate 2 cylinder control, **on** to open gate 2, **off** closes gate.
GATE_3	Gate 3 cylinder control, **on** to open gate 3, **off** closes gate.
DRILL_EXTEND	Drill extension cylinder control, **on** to extend drill.
DRILL_RETRACT	Drill retraction cylinder control, **on** to retract drill
DRILL_MOTOR	Drill motor control, **on** to cause drill to rotate.

The addresses associated with the physical inputs and outputs are:

Tag/Var./Symbol	ControlLogix	Micro800	MLogix	SLC-500
START_PB	Local:1:I.Data.0	_IO_EM_DI_00	I:0/0	I:1/0
STOP_PB	Local:1:I.Data.1	_IO_EM_DI_01	I:0/1	I:1/1
RESET_PB	Local:1:I.Data.2	_IO_EM_DI_02	I:0/2	I:1/2
PROX1	Local:1:I.Data.4	_IO_EM_DI_04	I:0/4	I:1/4
PROX3	Local:1:I.Data.5	_IO_EM_DI_05	I:0/5	I:1/5
LS1	Local:1:I.Data.6	_IO_EM_DI_06	I:0/6	I:1/6
LS2	Local:1:I.Data.7	_IO_EM_DI_07	I:0/7	I:1/7
GATE_1	Local:2:O.Data.0	_IO_EM_DO_00	O:0/0	O:2/0
GATE_2	Local:2:O.Data.1	_IO_EM_DO_01	O:0/1	O:2/1
GATE_3	Local:2:O.Data.2	_IO_EM_DO_02	O:0/2	O:2/2
DRILL_EXTEND	Local:2:O.Data.3	_IO_EM_DO_03	O:0/3	O:2/3
DRILL_RETRACT	Local:2:O.Data.4	_IO_EM_DO_04	O:0/4	O:2/4
DRILL_MOTOR	Local:2:O.Data.5	_IO_EM_DO_05	O:0/5	O:2/5

SP6-9. Part Oiler Station Control. Using the function chart approach, implement the program for the following station that squirts oil onto each part that passes on a conveyor.

Figure SP6.9 shows the layout of a station that squirts oil onto every part that comes down the conveyor. The station is only one in a series of stations along this conveyor. You are implementing ladder logic for this station only. You have no control over the conveyor, so assume it is always moving. This particular line is asynchronous, that is, each station processes parts at its own speed and does not coordinate its operation with any other station. Assume that the parts are spaced far enough apart that this operation can complete before the next part comes down the conveyor.

The conveyor consists of two parallel tracks. The part sensing and capturing mechanism is located between the two tracks. The part actually is attached to and rides upon an aluminum platform.

Upon initial startup, assume there are no parts in the station. When a part is detected by the proximity detector, PROX, the following steps are executed:

Capture part by activating ENGAGE_SOL and waiting for 4±0.1 seconds.

The oiler tip is lowered into position.

Open the oiler valve (OIL_VALVE) for 0.5±0.01 seconds, which squirts oil onto a certain place on the part.

The oiler is raised.

The ENGAGE_SOL is released and the part moves out of the station.

The operation then repeats.

The proximity sensor is inductive. PROX senses the platform before the platform reaches the engage position. You must assume that when the part is captured by the engaging mechanism, PROX remains **on**.

The engaging mechanism is driven by a single-action pneumatic cylinder, controlled by ENGAGE_SOL. When ENGAGE_SOL is energized, the "hook" moves up and remains in the "up" position as long as power is applied (turned **on**). The "hook" moves down when power is removed (turned **off**).

The mechanism used to lower and raise the oiler tip is driven by a double-acting pneumatic cylinder. When the OILER_DOWN output is energized, the oiler tip moves down and continues to move down as long as power is applied (turned **on**). When the OILER_UP output is energized, the oiler tip moves up and continues to move up as long as power is applied (turned **on**). The mechanism stops if neither output is **on** or if they are energized simultaneously. LS1 is **on** when the oiler tip is in the fully "up" position. LS2 is **on** when the oiler tip is in the fully "down" position.

When the start switch is pressed (turned **on**) for the first time only, the station assumes there is no part in the station and waits for the first part to arrive. When the stop switch is pressed (turned **off**), the operation should pause **except** when the oiler valve is opened to squirt the oil. If STOP_PB is pressed during the time the oil is being squirted, the step should be completed before pausing. Do not ignore STOP_PB during the step when oil is being squirted. When the operation is paused all outputs (except ENGAGE_SOL and OIL_VALVE) must be turned **off**. Pressing the start switch while the operation of the station is paused causes the station to resume its suspended operation.

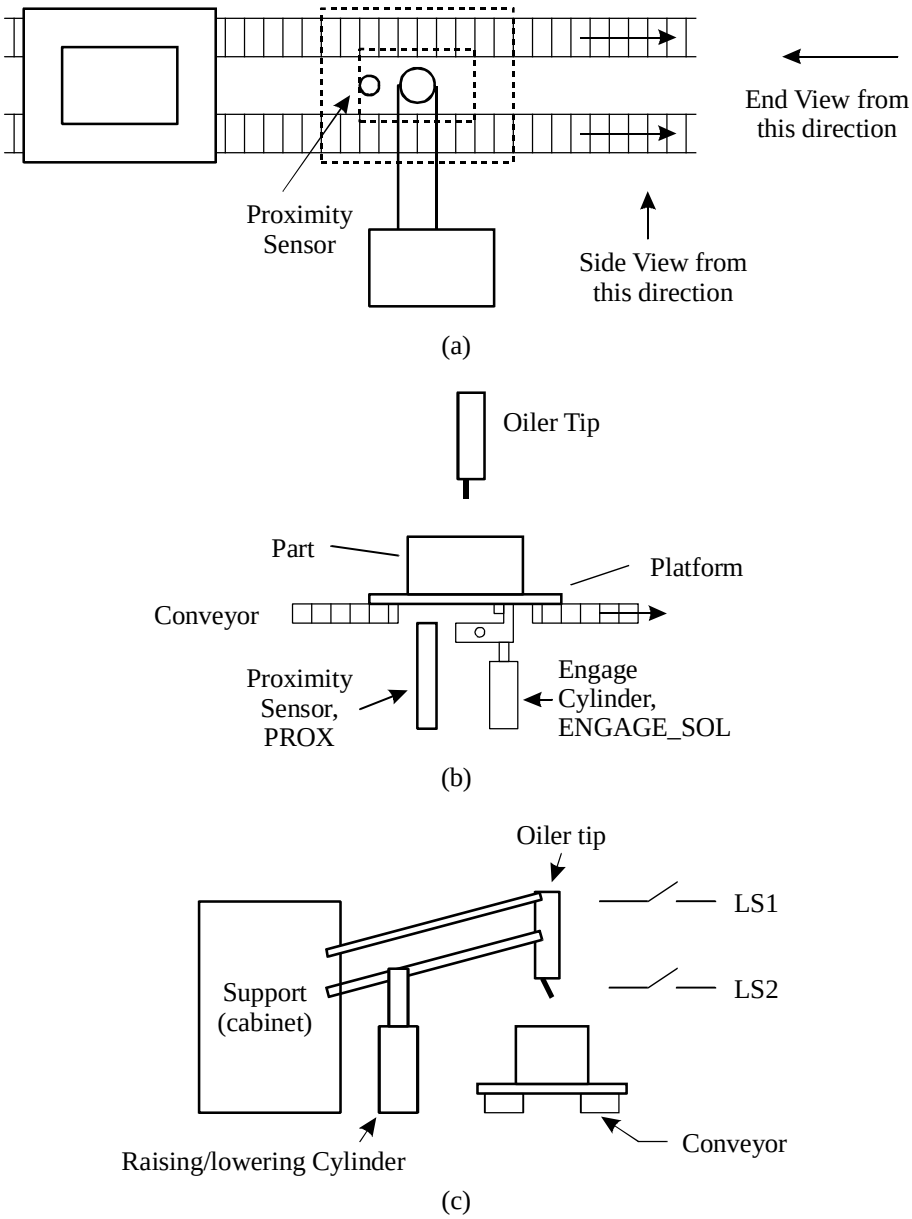


Figure SP6.9. Oiler station: *(a)* top view; *(b)* side view; *(c)* end view.

If a timed step is paused, you need to determine if it is permissible to advance to the next step. It is permissible to advance to the next step if the timing interval has expired and no equipment will be damaged as a result of advancing to the next step.

There is a RESET_PB switch that when **on**, restarts the operation. When RESET_PB is **on**, the oiler is raised (if not in the "up" position), and the internal step is set so that the ladder logic program waits for the next part (but does not actually turn any outputs **on**).

After a reset, START_PB must be pressed to actually restart the station. In other words, after RESET_PB is pressed and then released, the next press of the start switch is treated as the first time the start switch is pressed. However, the reset operation must be completed (oiler raised) before pressing START_PB has any effect. The RESET_PB switch must have no effect unless the operation is already paused.

Assume the following physical inputs and outputs.

Tag/Var./Symbol	Description
START_PB	Start push button, N. O., **on** when starting.
STOP_PB	Stop push button, N. C., **off** when stopping.
RESET_PB	Reset push button, N. O., **on** to restore station to initial state.
PROX	Proximity switch, **on** when platform is in station.
LS1	Limit switch, **on** (closed) when oiler tip is in raised position.
LS2	Limit switch, **on** (closed) when oiler tip is in lowered position.
ENGAGE_SOL	**On** to move up hook to engage platform in station, **off** releases platform to move down conveyor.
OILER_DOWN	**On** to lower oiler tip, **off** has no effect.
OILER_UP	**On** to raise oiler tip, **off** has no effect.
OIL_VALVE	**On** to open valve and squirt oil.

The addresses associated with the physical inputs and outputs are:

Tag/Var./Symbol	ControlLogix	Micro800	MLogix	SLC-500
START_PB	Local:1:I.Data.0	_IO_EM_DI_00	I:0/0	I:1/0
STOP_PB	Local:1:I.Data.1	_IO_EM_DI_01	I:0/1	I:1/1
RESET_PB	Local:1:I.Data.2	_IO_EM_DI_02	I:0/2	I:1/2
PROX	Local:1:I.Data.3	_IO_EM_DI_03	I:0/3	I:1/3
LS1	Local:1:I.Data.4	_IO_EM_DI_04	I:0/4	I:1/4
LS2	Local:1:I.Data.5	_IO_EM_DI_05	I:0/5	I:1/5
ENGAGE_SOL	Local:3:O.Data.8	_IO_EM_DO_00	O:0/8	O:3/8
OILER_DOWN	Local:3:O.Data.9	_IO_EM_DO_01	O:0/9	O:3/9
OILER_UP	Local:3:O.Data.10	_IO_EM_DO_02	O:0/10	O:3/10
OIL_VALVE	Local:3:O.Data.11	_IO_EM_DO_03	O:0/11	O:3/11

P6-10. Pressing Station Control. Using the function chart approach, implement the program for the following station that presses a pattern into each wood piece that passes on a conveyor.

Figure SP6.10 shows the layout of a station that presses a pattern into each wood piece that passes on the conveyor. The station is only one in a series of stations along this conveyor. You are implementing ladder logic for this station only. You have control over the infeed conveyor. However, you have no control over the outfeed conveyor, so assume it is always moving. This particular line is asynchronous; that is, each station processes woods pieces at its own speed and does not coordinate its operation with any other station. Assume

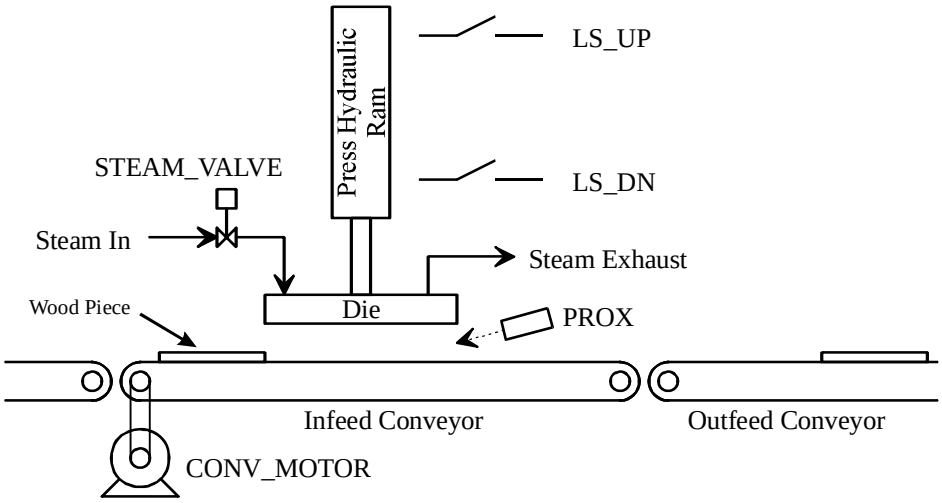


Figure SP6.10. Pressing station (side view).

that the pieces are spaced far enough apart that this operation can complete before the next piece comes down the conveyor.

Upon initial pressing of the START_PB, assume there are no pieces in the station. So the infeed conveyor must be turned on to bring in a new wood piece. When a piece is detected by the proximity detector, PROX, the following happens:

The conveyor is stopped.

The hydraulic press (with the die) is lowered into position.

Press the piece for 30±0.1 seconds. During this time, the steam valve STEAM_VALVE) is opened to provide heat and moisture to the die.

The hydraulic press is raised.

Allow the part to cool for 451 seconds. Cooling cannot be started until the hydraulic press is completely raised.

Activate the conveyor to bring a new piece into the station (and move out the piece just pressed).

The operation then repeats.

The infeed conveyor (controlled by CONV_MTR) must be run in order to bring a piece into the station or move it out of the station. The conveyor must be off during all other steps.

The proximity sensor is a reflective infrared sensor. PROX senses the leading edge of the piece just before the proper position. The time required to stop the conveyor will place the piece into the correct position. You must assume that PROX remains **on** during the pressing and cooling steps.

The mechanism used to lower and raise the press consists of a single action hydraulic solenoid. When the PRESS_DOWN output is energized, the press moves down and continues to move down as long as power is applied (turned **on**). When the PRESS_DOWN output is de-energized, the press moves up and continues to move up as long as the output is de-energized. There is a mechanical stop for the fully up position. Limit switch LS_UP is **on**

when the press is in the fully "up" position. LS_DN is **on** when the press is in the fully "down" (pressing) position.

When the start switch is pressed (turned **on**) for the first time only, the station assumes there is no piece in the station and waits for the first piece to arrive. When the stop switch is pressed (turned **off**), the operation should pause except when the board is being pressed. If STOP_PB is pressed during the time the piece is being pressed, the step should be completed (and advanced to the next step) before pausing. Do not ignore STOP_PB during the pressing step. When the operation is paused all outputs must be turned **off**. Pressing the start switch while the operation of the station is paused causes the station to resume its suspended step.

If a timed step is paused, you need to determine if it is permissible to advance to the next step. While paused, it is permissible to advance to the next step if the timing interval has expired and no equipment will be damaged as a result of advancing to the next step.

Do not add any more timed steps to those explicitly stated in the problem. In other words, do not put a timer in a step unless it is stated that the step duration is a specific time.

There is a RESET_PB switch that when **on**, restarts the operation. When RESET_PB is **on**, the press is raised (if not in the "up" position) and the internal step is set so that the ladder logic program waits for the next piece (but does not actually turn any outputs **on**). Assume there are no pieces in the station when RESET_PB is pressed. After a reset, START_PB must be pressed to actually restart the station. In other words, after RESET_PB is pressed and then released, the next press of the start switch is treated as the first time the start switch is pressed. However, the reset operation must be completed (press raised) before pressing START_PB has any effect. The RESET_PB switch must have no effect unless the operation is already paused.

Assume the following physical inputs and outputs.

Tag/Var./Symbol	Description
START_PB	Start push button, N. O., **on** when starting.
STOP_PB	Stop push button, N. C., **off** when stopping.
RESET_PB	Reset push button, N. O., **on** when restoring station to initial state.
PROX	Proximity switch, **on** when piece is in station.
LS_UP	Limit switch, **on** (closed) when press is in the fully raised position.
LS_DN	Limit switch, **on** (closed) when press is in the fully lowered position.
CONV_MTR	Infeed conveyor motor control, **on** run conveyor motor to move pieces into/out of station.
PRESS_DOWN	Press cylinder control, **on** to lower press, **off** raises press.
STEAM_VALVE	**On** to open valve that directs steam into die.

The addresses associated with the physical inputs and outputs are:

Tag/Var./Symbol	ControlLogix	Micro800	MLogix	SLC-500
START_PB	Local:1:I.Data.0	_IO_EM_DI_00	I:0/0	I:1/0
STOP_PB	Local:1:I.Data.1	_IO_EM_DI_01	I:0/1	I:1/1
RESET_PB	Local:1:I.Data.2	_IO_EM_DI_02	I:0/2	I:1/2

PROX	Local:1:I.Data.3	_IO_EM_DI_03	I:0/3	I:1/3
LS_UP	Local:1:I.Data.4	_IO_EM_DI_04	I:0/4	I:1/4
LS_DN	Local:1:I.Data.5	_IO_EM_DI_05	I:0/5	I:1/5
CONV_MTR	Local:2:O.Data.7	_IO_EM_DO_00	O:0/7	O:2/7
PRESS_DOWN	Local:2:O.Data.8	_IO_EM_DO_01	O:0/8	O:2/8
STEAM_VALVE	Local:2:O.Data.9	_IO_EM_DO_02	O:0/9	O:2/9

SP6-11. Case Erector Control. Using the function chart approach, implement the program for the following case erector station that unfolds a corrugated cardboard carton blank and positions the flaps for the taping head after this station that tapes the bottom of the carton.

Figure SP6.11 shows the layout of a station called a case erector. A case erector unfolds a corrugated cardboard carton blank and positions the flaps for the taping head. The taping head is after this station and tapes the bottom of a carton as it passes. The taping head is purely mechanical and so is not controlled by the PLC. In order to simplify this problem, you will not be concerned about the details how a flat carton is fed off a stack of flat cartons. You only need to activate the appropriate output to feed one in. The station is the first in a series of stations along this conveyor. The other stations fill the carton, put on labels, tape it, etc. You are implementing ladder logic for this station only. You have control over the mechanism that feeds cartons through this station. You have no control over the outfeed conveyor, so assume it is always moving.

There is no START_PB and STOP_PB because there is an overall start and stop for the line. Instead, there is a Run internal coil that is ON when this station is to be running. The station is running as long as the stations downstream of it are using the cartons. As soon as the cartons start backing up, the case erector must be paused until the cartons are being used again.

Upon initial turning on of the Run internal coil, assume there are no cartons in the station. To erect a new carton the following happens:

The infeed rollers are activated to bring in a flat carton (until PROX turns **on**).

Both END_CYL and GATE1 are activated in order to unfold the carton. END_CYL causes a lever arm to rotate and unfold the carton. GATE1 merely prevents the carton from sliding down the station as it is being unfolded.

Activate the LEFT_CYL and RIGHT_CYL simultaneously to push in the shorter side flaps. When LEFT_LS and RIGHT_LS are both **on**, then the flaps are in the proper position.

Activate the BOT_CYL to push up the longer bottom flap. When BOT_LS is **on**, then the flap is in the proper position.

Activate the top chain-like conveyor to move the carton to the taping head. As the carton is moved, the top flap is pushed down and is in position when the carton reaches the taping head.

Deactivate the BOT_CYL to move it out of the way for the next carton. Allow 1 second for this operation.

The operation then repeats.

The infeed rollers (controlled by INFEED_MTR) must be **on** in order to bring a flat carton into the station. The infeed rollers must be **off** during all other steps.

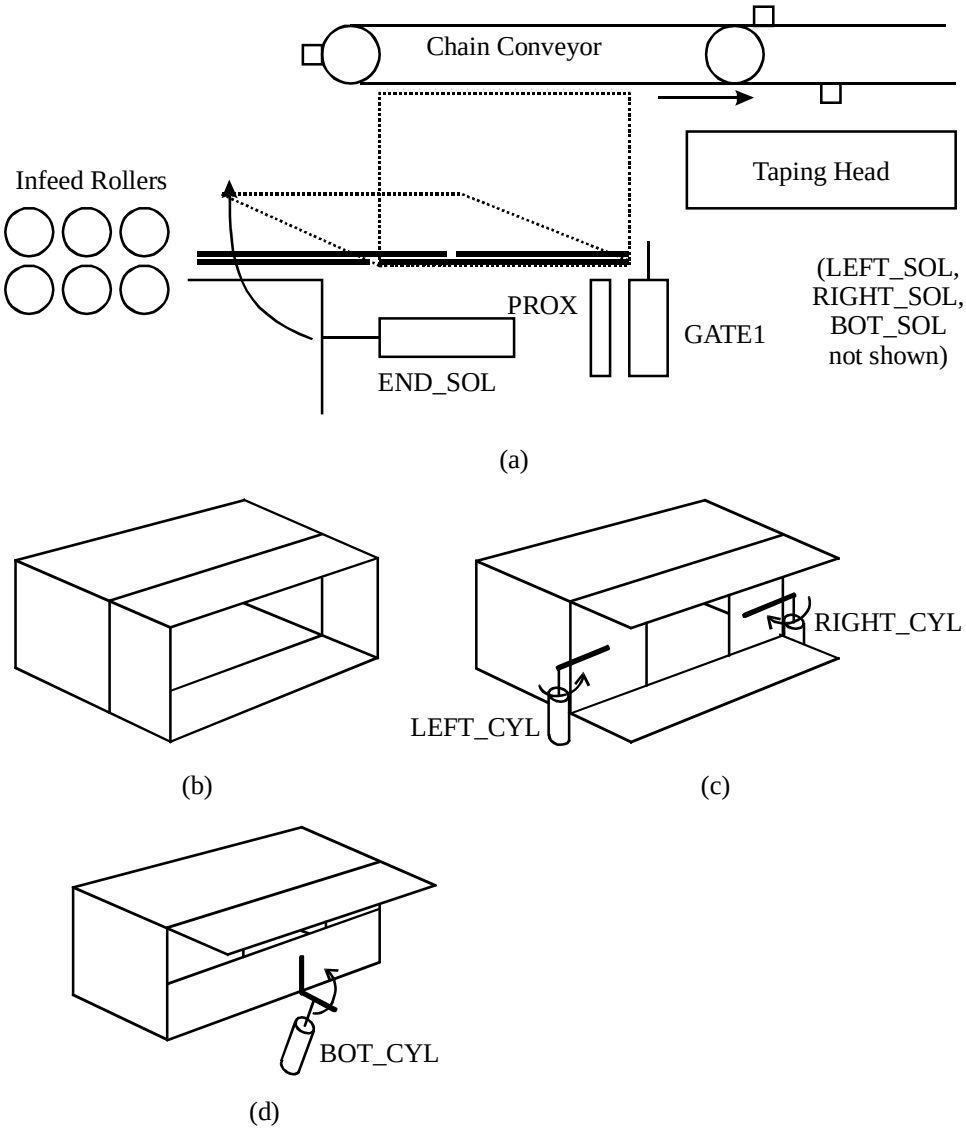


Figure SP6.11. Carton erecting station: *(a)* side view; *(b)* view of erected carton; *(c)* side flaps folded; *(d)* bottom flap folded.

The proximity sensor is a reflective infrared sensor. PROX senses the leading edge of the carton just before the proper position. The time required to stop the infeed rollers will place the piece into the correct position. You must assume that PROX remains **on** as the carton is being erected and the flaps folded in.

The END_CYL is the control for a single action pneumatic cylinder. When the END_CYL output is energized, a lever arm rotates up and unfolds the flat carton by pushing up on the part of the flat carton that will become the left end. When the END_CYL output is de-energized, the lever arm rotates down to a mechanical stop. The limit switch that

determines when END_CYL is fully activated is defective and so this step must be timed. Assume it requires 3.5 seconds to fully activate END_CYL and erect the carton. While END_CYL is **on**, GATE1 must also be **on** to prevent the carton from sliding down the station as it is being unfolded. When activated, GATE1 moves very fast and so is fully extended by the time END_CYL has started to erect the carton.

Both LEFT_CYL and RIGHT_CYL are controls for single action pneumatic cylinders. When the LEFT_CYL output is energized, a lever arm rotates sideways and pushes (folds) the left flap inward. When LEFT_LS is energized, the left flap has been folded properly. When the RIGHT_CYL output is energized, a lever arm rotates sideways and pushes (folds) the right flap inward. When RIGHT_LS is energized, the right flap has been folded properly. Before the bottom flap may be pushed up, both LEFT_LS and RIGHT_LS must be **on**. You can make no assumptions about which flap (left or right) is folded in position last. Do not consider the case when one of these limit switches never turns on.

BOT_CYL is the control for a single action pneumatic cylinder. When the BOT_CYL output is energized, a lever arm rotates upwards and pushes (folds) the bottom flap inward. When BOT_LS is energized, the bottom flap has been folded properly and can now hold the shorter side flaps in position. When the BOT_CYL output is turned **off**, assume it requires 1 second to move out of the way before another flat carton may be moved in.

Both END_CYL and GATE1 must remain **on** as the shorter side flaps are being pushed and must be **off** while the long bottom flap is being pushed, the folded carton is being moved out, and while a new flat carton is being moved in. Both LEFT_CYL and RIGHT_CYL must remain **on** as the longer bottom flap is being pushed, and must be **off** while the folded carton is being moved out, while a new flat carton is being moved in, and while the carton is being erected.

The BOT_CYL must remain **on** while that the folded carton is being moved out, and must be **off** while a new flat carton is being moved in, while the carton is being erected, and while the shorter flaps are being pushed.

To move the carton out of the erector and into the taping head, activate CHAIN_CONV, which drives two overhead chains with a connecting bar between the chains that pushes the carton. The driving wheel on this chain conveyor has an encoder that generates pulses, ENC_PULSE, as it rotates. When ENC_PULSE has transitioned from **off** to **on** 500 times, the conveyor has moved the proper distance, and CHAIN_CONV must be shut **off**.

When the Run internal coil is turned **on** for the first time only, the station assumes there are no cartons in the station and immediately starts to feed in a flat carton. When the Run internal coil is turned **off** the operation should pause at its current step. If the Run is turned **off** while the carton is being erected or flaps being pushed, the step must run to completion, but you may not advance to the next step. When the operation is paused, only INFEED_MTR and CHAIN_CONV must be turned **off**. All other outputs (single action solenoids) must remain **on** if paused in a step where they should be **on**. If Run is turned on while the operation of the station is paused, the station should resume its suspended step.

Do not add any more timed steps to those explicitly stated in the problem. In other words, do not put a timer in a step unless it is stated that the step duration is a specific time.

There is a Reset internal coil that when **on**, restarts the operation. When Reset is **on**, the internal step is set so that the ladder logic program waits to feed in a new carton (but does not actually turn any outputs **on**). After a reset, Run must be turned **on** to actually restart the station. In other words, after Reset is turned **on** and then back **off**, the next time Run turns

on, it is treated as the first time Run turns **on**. The Reset coil must have no effect unless the operation is already paused.

Assume the tolerance on all timer values is ±0.1 seconds.

Assume the following physical inputs, physical outputs, and internal coils.

Tag/Var./Symbol	Description
PROX	Proximity sensor, **on** when flat carton in position to be erected.
LEFT_LS	Limit switch, closes (**on**) when left flap is folded in position.
RIGHT_LS	Limit switch, closes (**on**) when right flap is folded in position.
BOT_LS	Limit switch, closes (**on**) when bottom flap is folded in position.
ENC_PULSE	Pulses to count for position of chain conveyor used to move erected carton out.
INFEED_MTR	Infeed rollers, **on** to move in flat carton.
END_CYL	End cylinder control. When **on**, a lever arm rotates up and unfolds the flat carton by pushing up on the part of the flat carton that will become the left end. When **off**, lever arm rotates down and out of the way.
LEFT_CYL	Left cylinder control. When **on**, a lever arm rotates sideways and pushes (folds) the left flap inward. When **off**, retracts.
RIGHT_CYL	Right cylinder control. When **on**, a lever arm rotates sideways and pushes (folds) the right flap inward. When **off**, retracts.
BOT_CYL	Bottom cylinder control. When **on**, a lever arm rotates upwards and pushes (folds) the bottom flap inward. When **off**, retracts.
GATE1	Gate cylinder control. **On** to prevent the carton from sliding down the station as it is being unfolded. When **off**, retracts allowing carton to move out of station.
CHAIN_CONV	Chain conveyor motor control When **on**, drives chain conveyor to move carton out of station.
Run	When **on**, allow case erector station to run. When **off**, pause (explained above)
Reset	When **on** resets operation of station (explained above).

The addresses associated with the physical inputs and outputs are:

Tag/Var./Symbol	ControlLogix	Micro800	MLogix	SLC-500
PROX	Local:1:I.Data.0	_IO_EM_DI_00	I:0/0	I:1/0
LEFT_LS	Local:1:I.Data.1	_IO_EM_DI_01	I:0/1	I:1/1
RIGHT_LS	Local:1:I.Data.2	_IO_EM_DI_02	I:0/2	I:1/2
BOT_LS	Local:1:I.Data.3	_IO_EM_DI_03	I:0/3	I:1/3
ENC_PULSE	Local:1:I.Data.4	_IO_EM_DI_04	I:0/4	I:1/4
INFEED_MTR	Local:2:O.Data.0	_IO_EM_DO_00	O:0/0	O:2/0
END_CYL	Local:2:O.Data.1	_IO_EM_DO_01	O:0/1	O:2/1
LEFT_CYL	Local:2:O.Data.2	_IO_EM_DO_02	O:0/2	O:2/2

RIGHT_CYL	Local:2:O.Data.3	_IO_EM_DO_03	O:0/3	O:2/3
BOT_CYL	Local:2:O.Data.4	_IO_EM_DO_04	O:0/4	O:2/4
GATE1	Local:2:O.Data.5	_IO_EM_DO_05	O:0/5	O:2/5
CHAIN_CONV	Local:2:O.Data.6	_IO_EM_DO_06	O:0/6	O:2/6

The internal tag/symbol data types or addresses are:

Tag/Var./Symbol	CLogix Data Type	Micro800 Data Type	MLogix/SLC Address
Run	BOOL	BOOL	B35/22
Reset	BOOL	BOOL	B35/25

SP6-12. Batch Control. Using the function chart approach, implement the program for the following batch control system.

The process shown in Figure SP6.12 consists of four tanks with pumps to transfer the liquid contents through the system. Each tank is fitted with sensors to detect empty and full conditions. In addition, tank 2 has an integral heating element with associated temperature sensor. Tank 3 is equipped with a stirrer to mix the two constituent liquids when they are pumped from tanks 1 and 2. The lower tanks, 3 and 4, have twice the capacity of tanks 1 and 2, and will therefore be filled by the contents of tanks 1 and 2 (alkali plus polymer).

The normal operation is described as follows. Tanks 1 and 2 are to be filled simultaneously from supply reservoirs of alkali and polymer respectively, via pumps 1 and 2. Pumps 1 and 2 should turn off as each tank-full sensor operates. The heating element in tank 2 is activated, raising the polymer temperature up to 60°C, when the temperature sensor closes. This action should turn off the heater and turn on pumps 3 and 4 to transfer the liquids into the reaction vessel, tank 3. The stirrer must also run when tank 3 has anything in it. Once tank 3 is full or tank 1 is empty, pump 3 is stopped. In the same manner, pump 4 is stopped once tank 3 is full or tank 2 is empty. After pumps 3 and 4 have stopped, the stirrer remains **on** for an additional 90 seconds. Then pump 5 is turned **on** to transfer the mixture to tank 4 (the product silo) via a filter unit. Pump 5 is stopped once tank 4 is full or tank 3 is empty. Finally, the product is run off into storage using pump 6. The cycle is not repeated and must be restarted by the operator.

The timing of the stirrer must be started as soon as the material transfer into tank 3 commences.

When the start switch is pressed (turned **on**) for the first time only, the process starts its operation as described above. When the stop switch is pressed (turned **off**) the operation should pause. When the operation is paused all outputs except STIRRER and HEATER must be turned **off**. The stirrer and heater must not be allowed to operate while paused. Also, all timer values should be retained. Pressing the start switch while the operation of the station is paused causes the station to resume its suspended step.

There is a reset push button that when **on**, resets the operation. When RESET_PB is **on**, the internal step is set so that the ladder logic program waits for the operator to start a new cycle. Assume that if reset in the middle of a batch operation, the process has manually-operated valves to drain all tanks. After a reset, the operator must press the start push button to actually restart the station. The reset push button must have no effect unless the operation is already paused.

Assume the following physical inputs and outputs.

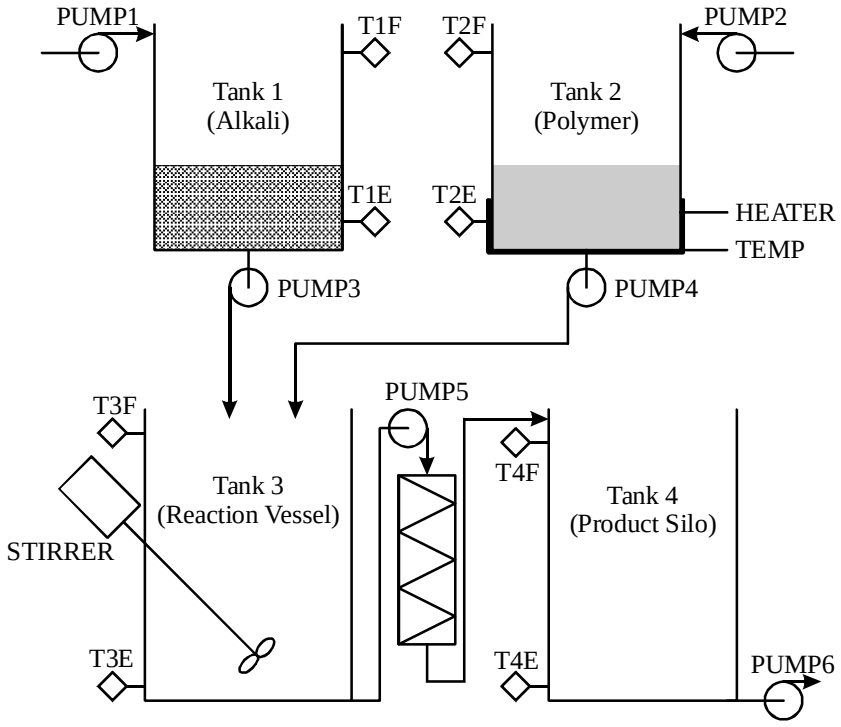


Figure SP6.12. Batch process.

Tag/Var./Symbol	Description
START_PB	Start push button, N. O., **on** when starting.
STOP_PB	Stop push button, N. C., **off** when stopping.
RESET_PB	Reset push button, N. O., **on** when restoring process to initial state.
T1E	Tank 1 empty sensor, **on** when empty
T1F	Tank 1 full sensor, **on** when full
T2E	Tank 2 empty sensor, **on** when empty
T2F	Tank 2 full sensor, **on** when full
T3E	Tank 3 empty sensor, **on** when empty
T3F	Tank 3 full sensor, **on** when full
T4E	Tank 4 empty sensor, **on** when empty
T4F	Tank 4 full sensor, **on** when full
TEMP	Temperature sensor, **on** when temperature of tank 2 has reached 60°C, but turns **off** when tank 2 is half-empty.
PUMP1	Pump 1 motor control, **on** to run pump 1
PUMP2	Pump 2 motor control, **on** to run pump 2
PUMP3	Pump 3 motor control, **on** to run pump 3
PUMP4	Pump 4 motor control, **on** to run pump 4
PUMP5	Pump 5 motor control, **on** to run pump 5

PUMP6	Pump 6 motor control, **on** to run pump 6
HEATER	Tank 2 heater control, **on** to heat tank 2 contents
STIRRER	Tank 3 stirrer motor control, **on** to stir tank 3 contents

The addresses associated with the physical inputs and outputs are:

Tag/Var./Symbol	ControlLogix	Micro800	MLogix	SLC-500
START_PB	Local:1:I.Data.0	_IO_EM_DI_00	I:0/0	I:1/0
STOP_PB	Local:1:I.Data.1	_IO_EM_DI_01	I:0/1	I:1/1
RESET_PB	Local:1:I.Data.2	_IO_EM_DI_02	I:0/2	I:1/2
T1E	Local:1:I.Data.3	_IO_EM_DI_03	I:0/3	I:1/3
T1F	Local:1:I.Data.4	_IO_EM_DI_04	I:0/4	I:1/4
T2E	Local:1:I.Data.5	_IO_EM_DI_05	I:0/5	I:1/5
T2F	Local:1:I.Data.6	_IO_EM_DI_06	I:0/6	I:1/6
T3E	Local:1:I.Data.7	_IO_EM_DI_07	I:0/7	I:1/7
T3F	Local:1:I.Data.8	_IO_EM_DI_08	I:0/8	I:1/8
T4E	Local:1:I.Data.9	_IO_EM_DI_09	I:0/9	I:1/9
T4F	Local:1:I.Data.10	_IO_EM_DI_10	I:0/10	I:1/10
TEMP	Local:1:I.Data.11	_IO_EM_DI_11	I:0/11	I:1/11
PUMP1	Local:2:O.Data.0	_IO_EM_DO_00	O:0/0	O:2/0
PUMP2	Local:2:O.Data.1	_IO_EM_DO_01	O:0/1	O:2/1
PUMP3	Local:2:O.Data.2	_IO_EM_DO_02	O:0/2	O:2/2
PUMP4	Local:2:O.Data.3	_IO_EM_DO_03	O:0/3	O:2/3
PUMP5	Local:2:O.Data.4	_IO_EM_DO_04	O:0/4	O:2/4
PUMP6	Local:2:O.Data.5	_IO_EM_DO_05	O:0/5	O:2/5
HEATER	Local:2:O.Data.6	_IO_EM_DO_06	O:0/6	O:2/6
STIRRER	Local:2:O.Data.7	_IO_EM_DO_07	O:0/7	O:2/7

SP6-13. Erbia Can Tipper/Rotator Control. Using the function chart approach, implement the program for the following process, which tips and rotates cans of erbia (a metal) powder.

The memo below describes the operation of the erbia tipper/rotator. A simplified drawing of the system is shown in Figure SP6.13.

To: A. Doe

From: B. Smith

Subject: Controls for Erbia Line Tipper/Rotator

Following is the proposed controls for the machine to tip and rotate the erbia cans for mixing. Each can is tipped so the axis of rotation is horizontal, thus "tumbling" the powder to provide a uniform mixture. Your comments would be appreciated as well as the detail design and purchase of electronics and controllers. Refer to the drawings of the tipper system for switch numbers:

LS_x is a limit switch,

PS_x is a photocell switch,

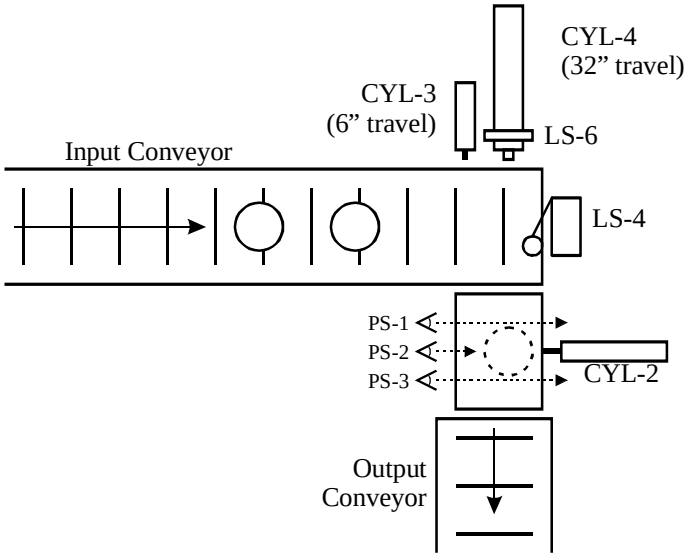
CYL_x is an air cylinder. Please note that the control will be to a solenoid operated 4-way valve. I have purchased these valves.

Sequence:

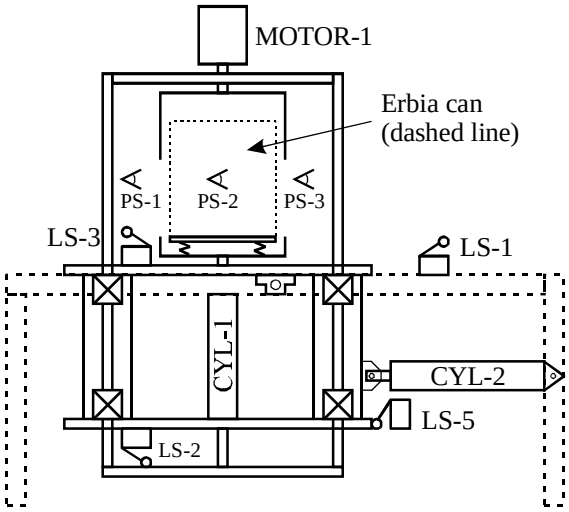
1. A can is pushed along the input conveyor to contact LS_4. If PS_1, PS_2, PS_3 and LS_1 are clear (unactivated) and LS_2 is activated then CYL_3 and CYL_4 are actuated to push the can into the tipper and to stop the next can from actuating LS_4.
2. As the can proceeds into the tipper, PS_1 senses the can, is activated, and then PS_2 is activated. As soon as PS_1 clears (with PS_2 on), CYL-4 is retracted. There is no retract limit switch. Assume 2 seconds are needed to retract CYL-4.
3. When CYL-4 is retracted, PS_1 and PS_3 should be clear and PS_2 activated. Then CYL-1 is extended to hold the can into the rotator.
4. When LS_3 is activated to indicate that the can is clamped, CYL-2 is extended to tip the rotator.
5. When LS_1 is activated to indicate the rotator is horizontal, MOTOR_1 starts and runs for 1 minute to blend the powder. When the time is complete, the motor stops and CYL-2 is retracted.
6. When LS_5 is activated (indicating rotator is vertical), CYL-1 is retracted.
7. When LS_2 is activated (indicating holder is clear of can), CYL-4 is extended until LS_6 is activated (indicating the cylinder is in the fully extended position). Then CYL-3 and CYL-4 are retracted.
8. The system is reset and ready for LS_4 to be reactivated and restart the cycle. Status: CYL_1, 2, 3 and 4 are retracted; LS_1 deactivated; LS_2 activated; LS_3 deactivated; LS_4 deactivated; LS_5 activated; PS_1, 2, and 3 all clear.

Notes to operation memo:

1. The reset mentioned in step 8 of the sequence IS NOT a separate switch.
2. When an output that controls a pneumatic cylinder (ram) is **on**, the ram moves out to its fully extended position and remains there (as long as the AIR CYL output stays **on**). When **off**, the ram moves to the retracted position, if not already there.
3. Assume there is a Start and Stop switch, though not mentioned in the memo. When the start switch is pressed (turned **on**) for the first time only, the station assumes there is no can in the station and waits for the first can to arrive. When the stop switch is pressed (turned **off**) the operation should pause. When the operation is paused all outputs **except** for CYL_1, CYL_2, and CYL_3 controls must be turned **off** and the blend time must be retained. Pressing the start switch while the operation of the station is paused causes the station to resume its suspended step.
4. There is a RESET_PB switch that when **on**, restarts the operation. When RESET_PB is **on**, the tipper is moved to the vertical position (if not in the "vertical" position), the can holder is released (if not already released), and the can is pushed out of the station (if present), and the internal step is set so that the ladder



(a)



(b)

Figure SP6.13. Erbia can tipper/rotator: *(a)* top view; *(b)* side view.

logic program waits for the next can (but does not actually turn any outputs **on**). After a reset, START_PB must be pressed to actually restart the station. In other words, after RESET_PB is pressed and then released, the next press of the start switch is treated as the first time the start switch is pressed. The RESET_PB switch must have no effect unless the operation is already paused.

5. Assume the input and output conveyors are always on when the station is running, even though not shown in your function chart.

Assume the following physical inputs and outputs.

Tag/Var./Symbol	Description
START_PB	Start push button, N. O., **on** when starting.
STOP_PB	Stop push button, N. C., **off** when stopping.
RESET_PB	Reset push button, N. O., **on** when restoring process to initial state.
LS_1	Horizontal position limit switch, **on** (closed) if rotator in horizontal position.
LS_2	Holder clear limit switch, **on** (closed) if holder is clear of can.
LS_3	Holder down limit switch, **on** (closed) if can is held in rotator.
LS_4	Can present on input conveyor limit switch, **on** (closed) when can in position to be pushed into rotator.
LS_5	Vertical position limit switch, **on** (closed) when rotator in vertical position.
LS_6	Cylinder CYL_4 fully extended limit switch, **on** (closed) when cylinder is fully extended, pushing can out of rotator.
PS_1	Left can photoelectric switch, **on** (closed) when can moving into rotator.
PS_2	Middle can photoelectric switch, **on** (closed) when can moving into rotator and when in proper position to be rotated.
PS_3	Right can photoelectric switch, **on** (closed) when can moving out of rotator.
CYL_1	Can holder cylinder control, **on** to clamp can into rotator; **off** unclamps
CYL_2	Tipper cylinder control, **on** to tip rotator horizontal; **off** to restore vertical orientation of can.
CYL_3	Gate cylinder control, **on** to extend and prevent the next can from entering station; **off** to retract.
CYL_4	Can pusher cylinder control, **on** to push can into and out of rotator; **off** to retract.
MOTOR_1	Rotation motor control, **on** to rotate clamped can

The addresses associated with the physical inputs and outputs are:

Tag/Var./Symbol	ControlLogix	Micro800	MLogix	SLC-500
START_PB	Local:1:I.Data.0	_IO_EM_DI_00	I:0/0	I:1/0
STOP_PB	Local:1:I.Data.1	_IO_EM_DI_01	I:0/1	I:1/1
RESET_PB	Local:1:I.Data.2	_IO_EM_DI_02	I:0/2	I:1/2
LS_1	Local:1:I.Data.3	_IO_EM_DI_03	I:0/3	I:1/3
LS_2	Local:1:I.Data.4	_IO_EM_DI_04	I:0/4	I:1/4
LS_3	Local:1:I.Data.5	_IO_EM_DI_05	I:0/5	I:1/5
LS_4	Local:1:I.Data.6	_IO_EM_DI_06	I:0/6	I:1/6
LS_5	Local:1:I.Data.7	_IO_EM_DI_07	I:0/7	I:1/7
LS_6	Local:1:I.Data.8	_IO_EM_DI_08	I:0/8	I:1/8
PS_1	Local:1:I.Data.9	_IO_EM_DI_09	I:0/9	I:1/9

PS_2	Local:1:I.Data.10	_IO_EM_DI_10	I:0/10	I:1/10
PS_3	Local:1:I.Data.11	_IO_EM_DI_11	I:0/11	I:1/11
CYL_1	Local:2:O.Data.0	_IO_EM_DO_00	O:0/0	O:2/0
CYL_2	Local:2:O.Data.1	_IO_EM_DO_01	O:0/1	O:2/1
CYL_3	Local:2:O.Data.2	_IO_EM_DO_02	O:0/2	O:2/2
CYL_4	Local:2:O.Data.3	_IO_EM_DO_03	O:0/3	O:2/3
MOTOR_1	Local:2:O.Data.4	_IO_EM_DO_04	O:0/4	O:2/4

SP6-14. Pick-and-place Machine Control. Using the function chart approach, implement the program for the following pick-and-place machine.

Figure SP6.14 shows the layout of a "pick-and-place" machine, consisting of a horizontal/vertical mechanism which is used to transfer parts from the left-hand table to the right-hand table. The horizontal and vertical movements are each driven by a double-acting air cylinder — a raising/lowering cylinder and a leftward/rightward cylinder. Each cylinder is controlled by two outputs, one for each direction. Once a direction control is energized, the mechanism moves and keeps moving as long as power is applied (turned on). The mechanism stops at its current position when power is removed (turned off). The mechanism will not move if both opposing directions are energized simultaneously (e.g., left and right). The clamp/unclamp action is driven by a single-action air cylinder, which clamps the part into the mechanism when the control is **on** and releases it when the control is **off**. Power must continually be applied to the clamp to hold the part in the mechanism. Limit switches detect the position of the mechanism, and a photoelectric switch, for safety, checks for the presence of parts remaining on the right-hand table.

The basic sequence of operation is as follows: From the home position (signified by the upper and left limit switches both being ON), the machine waits for a part to be moved onto the left table (signified by the part-operation-in-progress switch being OFF). When the part is ready to be moved, the lowering cylinder is operated, moving the mechanism down until the lower limit switch operates (turns ON). This stops the lowering motion and activates the clamp. Wait 2.0 seconds for the clamp to grip the part. The mechanism now moves upward until the upper limit switch operates, causing the horizontal movement to start, taking the mechanism over to the right limit switch and initiating another downward path towards the right-hand table — providing no previous parts remain on the table (tested by the photoelectric switch). If a part remains on the right-hand table, then the mechanism remains in the upper right position until the part is removed. At the right table, the clamp is released (power is removed). Wait 2.4 seconds for the part to be released. The mechanism returns to the home position, where it waits for a new part to be moved off the left table. The circled numbers in the Figure P6.14 indicate the sequence of moves.

When the start switch is pressed (turned **on**) for the first time only, the mechanism starts in the home position and performs the operation cycle continuously. Assume the mechanism is in the home position when the start switch is pressed for the first time. Pressing the start switch when the mechanism is already running must have no effect. When the stop switch is pressed (turned **off**) the mechanism should stop in its current position and in the middle of whatever operation it is doing. It must continue to clamp the part. Pressing the start switch causes the machine to resume its suspended operation.

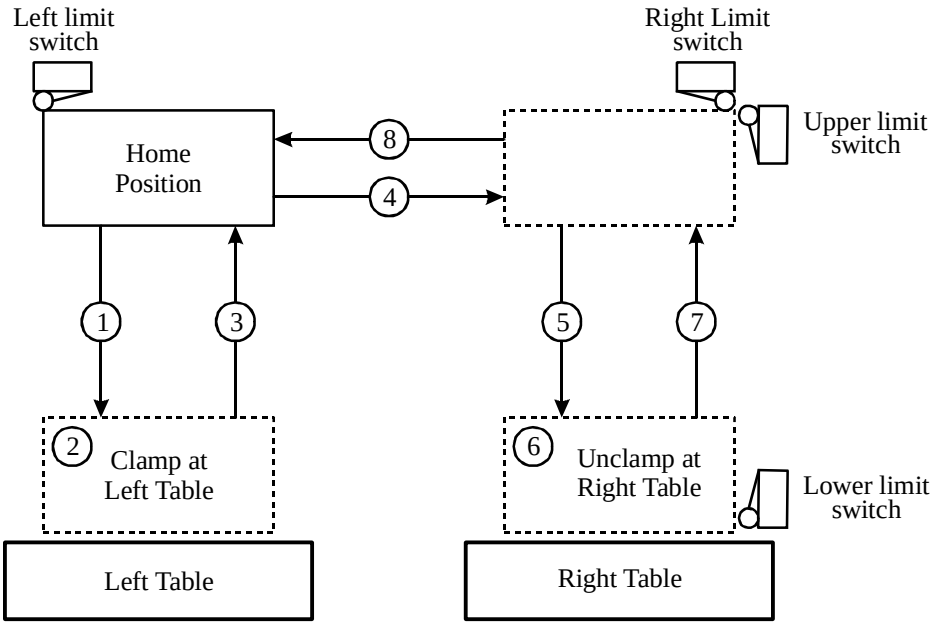


Figure SP6.14. Pick-and-place machine.

When the reset switch is pressed, the mechanism should move to the top and then to the left. Also, the part should be released after reaching home. The reset switch should be ignored if the mechanism is not paused.

Assume the following physical inputs and outputs:

Tag/Var./Symbol	Description
START_PB	Start push button, N. O., **on** when starting.
STOP_PB	Stop push button, N. C., **off** when stopping.
RESET_PB	Reset push button, N. C., **on** to restore process to initial state.
PART_OIP	Part operation-in-progress switch, **on** when not ready to move part off of left table
LEFT_LS	Left limit switch, **on** when mechanism is over left table
RIGHT_LS	Right limit switch, **on** when mechanism is over right table
UPPER_LS	Upper limit switch, **on** when mechanism is raised
LOWER_LS	Lower limit switch, **on** when mechanism is lowered
PART_DETECT	Right table part detection switch, **on** when part is on right table
LEFT_CYL	Left motion cylinder control, **on** to move mechanism to left
RIGHT_CYL	Right motion cylinder control, **on** to move mechanism to right
LOWER_CYL	Lowering cylinder control, **on** to lower mechanism
RAISE_CYL	Raising cylinder control, **on** to raise mechanism
CLAMP_CYL	Clamp cylinder control, **on** to clamp part in mechanism

The addresses associated with the physical inputs and outputs are:

Tag/Var./Symbol	ControlLogix	Micro800	MLogix	SLC-500
START_PB	Local:1:I.Data.0	_IO_EM_DI_00	I:0/0	I:1/0
STOP_PB	Local:1:I.Data.1	_IO_EM_DI_01	I:0/1	I:1/1
RESET_PB	Local:1:I.Data.2	_IO_EM_DI_02	I:0/2	I:1/2
PART_OIP	Local:1:I.Data.3	_IO_EM_DI_03	I:0/3	I:1/3
LEFT_LS	Local:1:I.Data.4	_IO_EM_DI_04	I:0/4	I:1/4
RIGHT_LS	Local:1:I.Data.5	_IO_EM_DI_05	I:0/5	I:1/5
UPPER_LS	Local:1:I.Data.6	_IO_EM_DI_06	I:0/6	I:1/6
LOWER_LS	Local:1:I.Data.7	_IO_EM_DI_07	I:0/7	I:1/7
PART_DETECT	Local:1:I.Data.9	_IO_EM_DI_09	I:0/9	I:1/9
LEFT_CYL	Local:2:O.Data.0	_IO_EM_DO_00	O:0/0	O:2/0
RIGHT_CYL	Local:2:O.Data.1	_IO_EM_DO_01	O:0/1	O:2/1
LOWER_CYL	Local:2:O.Data.2	_IO_EM_DO_02	O:0/2	O:2/2
RAISE_CYL	Local:2:O.Data.3	_IO_EM_DO_03	O:0/3	O:2/3
CLAMP_CYL	Local:2:O.Data.4	_IO_EM_DO_04	O:0/4	O:2/4

SP6-15. Drilling Station Control. Using the function chart approach, implement the program for the following station that drills two holes in each part that comes down the conveyor.

Figure SP6.15 shows the layout of a station that drills two holes in each part that comes down the conveyor. This station is only one in a series of stations along this conveyor. You are implementing ladder logic for this station only. You have no control over the conveyor, so assume it is always moving. This particular line is asynchronous, that is, each station processes parts at its own speed and does not coordinate its operation with any other station. Because this is an asynchronous line, each station contains a series of two gates that control access to the station and allow parts to queue up before the station.

Upon initial startup, assume that there are no parts waiting at gate 1. When a part is detected at Gate 1 (by PROX1), the following major steps are executed:

Sequence Gate 1 and Gate 2 to allow only one part to move into the drilling position (against Gate 3),
Clamp the part into position (and off of the conveyor),
Drill 1 is turned and is extended (moves out) to the correct depth,
Drill 1 is retracted,
Drill 2 is turned and is extended (moves out) to the correct depth,
Drill 2 is retracted,
Unclamp part,
Open Gate 3 to allow drilled part to move out.

The operation then repeats. Assume the conveyor is on at all times. The conveyor slides beneath the parts as they are held against a gate.

To move only one part into the station, Gates 1 and 2 are sequenced as follows (assume both gates are closed at the start): Gate 1 is opened to allow a part to move into position 2 (sensed by PROX2). Then, Gate 1 is closed, and Gate 2 is opened. As far as your ladder logic is concerned, assume that Gate 1 is closed and Gate 2 is opened at the same rate.

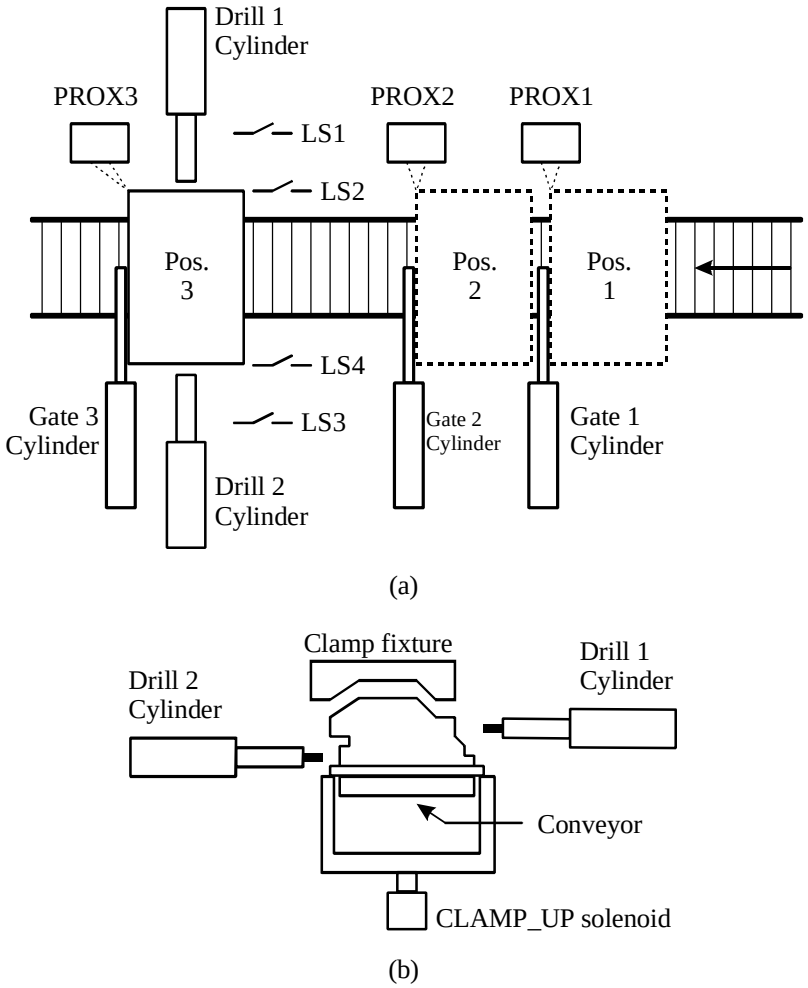


Figure SP6.15. Drilling station: *(a)* top view; *(b)* view from right side.

Physically, Gate 1 closes much faster than Gate 2 opens, so any part in position 1 is prevented from moving when Gate 2 is open enough to allow the part to move from position 2 to position 3. Gate 2 is closed when the part is in position 3 (sensed by PROX3).

The gates are each driven by a single solenoid powered by an air cylinder. Once a solenoid is energized, the gate is opened and remains open as long as power is applied (turned **on**). The gate closes when power is removed (turned **off**).

The clamp is driven by a single-action pneumatic cylinder controlled by a solenoid. Once the CLAMP_UP solenoid is energized, the clamp moves the part up into a fixture to get the part in proper alignment to the drills. Limit switch LS5 indicates that the part is in the proper position. CLAMP_UP must remain **on** to hold the part in the fixture. If CLAMP_UP is turned **off**, the part falls back onto the conveyor. Allow 0.5 seconds for the part to unclamp and fall to the conveyor.

Each drill extension/retraction is driven by a double-action pneumatic cylinder with two controls. Once a direction control is energized, the drill mechanism is moved and keeps moving as long as power is applied (turned **on**). The mechanism stops at its current position when power is removed (turned **off**). The mechanism will not move if both opposing directions are energized simultaneously (e.g., extension and retraction). The drill motor must be **on** whenever the drill is being extended or retracted.

Proximity sensor, PROX1, is **on** when a part is in position 1, meaning there is a part to be processed. PROX2 is **on** when a part is in position 2. PROX3 is **on** when a part is in position 3, ready to be drilled. When PROX3 is **off**, the part has passed Gate 3 and moved out of the station.

The drill position is indicated by limit switches. LS1 is **on** when drill 1 is fully retracted and LS2 is **on** when drill 1 is extended to the proper hole depth. Similarly, LS3 and LS4 indicate the retracted and extended positions, respectively, for drill 2.

The start/stop switches are only for the station. They do not control any other stations, or the conveyor. Upon initial startup, assume there are no parts present in any of the positions (1-3). If the stop switch is pressed at any time, the station operation should **pause**, except when the drill is being extended or retracted. The operation **must not** pause when the drill is being moved (otherwise it may jam or ruin the hole). When the start switch is pressed while the operation is paused, the station should resume the suspended step. When paused, **do not advance** to the next step. If the operation advances to the next step when paused, there will be problems with the gate operations. When the station is paused, the drill motor and the drill cylinder retraction controls should remain **on**. Any open gates must not be closed when paused (or parts may be knocked off) and the CLAMP_UP solenoid must not be turned **off**. The stop push button switch cannot be ignored, even in those steps where the operation cannot be paused.

A separate reset switch is provided that when pressed, any drill not fully retracted is retracted, the part clamp is released, and the process step is set as if the process is waiting for the next part. Note that the part clamp must remain **on** while a drill is retracting. When the start switch is pressed, no items are assumed present at position 1. To keep the problem simple, do not try to implement a function chart for reset, just retract both drills (motor **on**) simultaneously until both limit switches are activated and then release the part. The reset switch should have no effect unless the operation is already paused.

Assume the tolerance on all timer values is ±0.01 seconds.

Assume the following physical inputs and outputs.

Tag/Var./Symbol	Description
START_PB	Start push button, N. O., **on** when starting.
STOP_PB	Stop push button, N. C., **off** when stopping.
RESET_PB	Reset push button, N. O., **on** when restoring process to initial state.
PROX1	Proximity sensor, **on** when part in position 1
PROX2	Proximity sensor, **on** when part in position 2
PROX3	Proximity sensor, **on** when part in position 3
LS1	Limit switch that closes (**on**) when drill 1 is fully retracted
LS2	Limit switch that closes (**on**) when drill 1 is extended to the proper hole depth

LS3	Limit switch that closes (**on**) when drill 2 is fully retracted
LS4	Limit switch that closes (**on**) when drill 2 is extended to the proper hole depth
LS5	Limit switch that closes (**on**) when part clamped in proper position

GATE_1	Gate 1 control, **on** to open gate 1, **off** closes gate.
GATE_2	Gate 2 control, **on** to open gate 2, **off** closes gate.
GATE_3	Gate 3 control, **on** to open gate 3, **off** closes gate.
DRILL1_EXTEND	Drill 1 extension control, **on** to extend drill 1.
DRILL1_RETRACT	Drill 1 retraction control, **on** to retract drill 1.
DRILL1_MOTOR	Drill 1 motor control, **on** to cause drill 1 to rotate.
DRILL2_EXTEND	Drill 2 extension control, **on** to extend drill 2.
DRILL2_RETRACT	Drill 2 retraction control, **on** to retract drill 2.
DRILL2_MOTOR	Drill 2 motor control, **on** to cause drill 2 to rotate.
CLAMP_UP	Clamp control, **on** to move part up into fixture and retain it there.

The addresses associated with the physical inputs and outputs are:

Tag/Var./Symbol	ControlLogix	Micro800	MLogix	SLC-500
START_PB	Local:1:I.Data.0	_IO_EM_DI_00	I:0/0	I:1/0
STOP_PB	Local:1:I.Data.1	_IO_EM_DI_01	I:0/1	I:1/1
RESET_PB	Local:1:I.Data.2	_IO_EM_DI_02	I:0/2	I:1/2
PROX1	Local:1:I.Data.3	_IO_EM_DI_03	I:0/3	I:1/3
PROX2	Local:1:I.Data.4	_IO_EM_DI_04	I:0/4	I:1/4
PROX3	Local:1:I.Data.5	_IO_EM_DI_05	I:0/5	I:1/5
LS1	Local:1:I.Data.6	_IO_EM_DI_06	I:0/6	I:1/6
LS2	Local:1:I.Data.7	_IO_EM_DI_07	I:0/7	I:1/7
LS3	Local:1:I.Data.8	_IO_EM_DI_08	I:0/8	I:1/8
LS4	Local:1:I.Data.9	_IO_EM_DI_09	I:0/9	I:1/9
LS5	Local:1:I.Data.10	_IO_EM_DI_10	I:0/10	I:1/10
GATE_1	Local:2:O.Data.0	_IO_EM_DO_00	O:0/0	O:2/0
GATE_2	Local:2:O.Data.1	_IO_EM_DO_01	O:0/1	O:2/1
GATE_3	Local:2:O.Data.2	_IO_EM_DO_02	O:0/2	O:2/2
DRILL1_EXTEND	Local:2:O.Data.3	_IO_EM_DO_03	O:0/3	O:2/3
DRILL1_RETRACT	Local:2:O.Data.4	_IO_EM_DO_04	O:0/4	O:2/4
DRILL1_MOTOR	Local:2:O.Data.5	_IO_EM_DO_05	O:0/5	O:2/5
DRILL2_EXTEND	Local:2:O.Data.6	_IO_EM_DO_06	O:0/6	O:2/6
DRILL2_RETRACT	Local:2:O.Data.7	_IO_EM_DO_07	O:0/7	O:2/7
DRILL2_MOTOR	Local:2:O.Data.8	_IO_EM_DO_08	O:0/8	O:2/8
CLAMP_UP	Local:2:O.Data.9	_IO_EM_DO_09	O:0/9	O:2/9

SP6-16. Bolt Driving Station Control. Using the function chart approach, implement the program for the following station that drives bolts in gasoline engine assemblies as they come down the conveyor. This is one of the stations on the final assembly line for the vertical shaft engines at a major manufacturer. The problem is simplified since the actual station drives six bolts.

Figure SP6.16 shows the layout of a station that drives two bolts into gasoline engine assemblies riding on a pallet as they come down the conveyor. This station is only one in a series of stations along this conveyor. This solution implements ladder logic for this station only. Another PLC controls the conveyor, so assume it is always moving. This particular line is asynchronous, that is, each station processes assemblies at its own speed and does not coordinate its operation with any other station. Because this is an asynchronous line, each station contains two capturing mechanisms (engaging hooks) that control access to the station and allow assemblies to queue up before the station.

Upon initial startup, assume that there are no pallets waiting at engaging hook, Engage 1. When a pallet is detected at Engage 1 (by PROX21), the following major steps are executed:

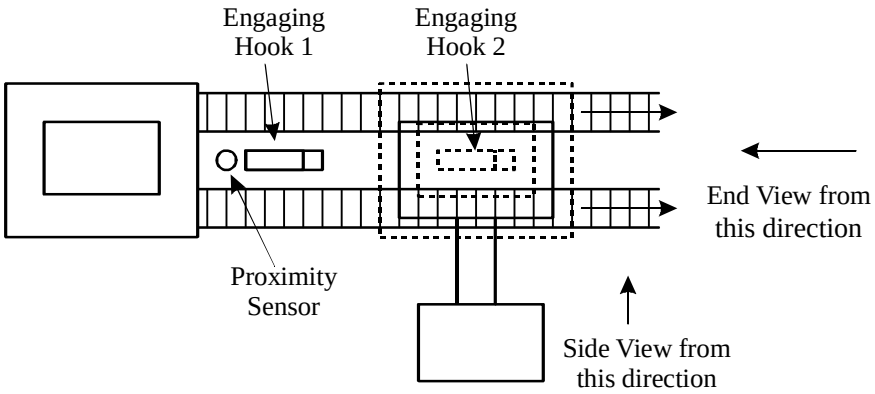
Activate the ENGAGE_21_CYL for 3 seconds to allow only one assembly to move into the station (when hook is raised, the next pallet is caught by the hook),
Raise the pallet off the conveyor,
Lower the bolt driving mechanism to the correct position (LS21_DN closes),
Activate appropriate solenoids and valves to convey the bolts into position (multiple steps),
Run air motors for 4 seconds to drive bolts,
Raise the bolt driving mechanism, until LS21_UP closes,
Lower pallet onto the conveyor,
Activate the ENGAGE_22_CYL for 3 seconds to allow pallet to move out.

The operation then repeats. Assume the conveyor is on at all times. The conveyor slides beneath the pallets as they are held by the engaging hooks or raised off the conveyor.

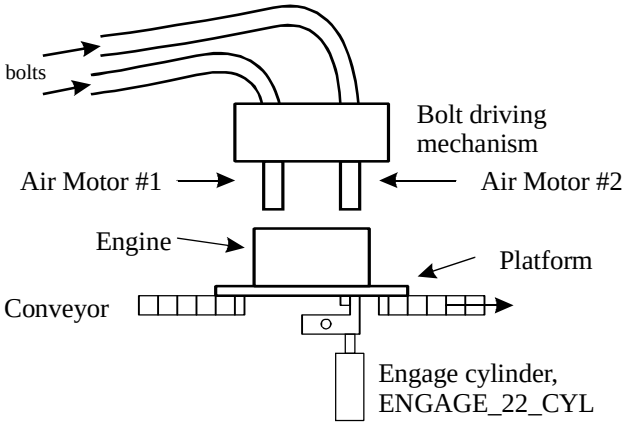
The proximity sensor, PROX21, is inductive and senses the metal assembly pallet. PROX21 senses the pallet before the pallet reaches the engage position. You must assume that when the pallet is captured by the ENGAGE_21_CYL, PROX21 remains **on**.

ENGAGE_21_CYL and ENGAGE_22_CYL are single action solenoids. Once it is energized, the "hook" moves down and remains in the "down" position as long as power is applied (turned **on**). The "hook" moves up when power is removed (turned **off**). The engaging mechanism works this way to be fail-safe, that is, if power is removed because of a failure, no pallets proceed down the conveyor.

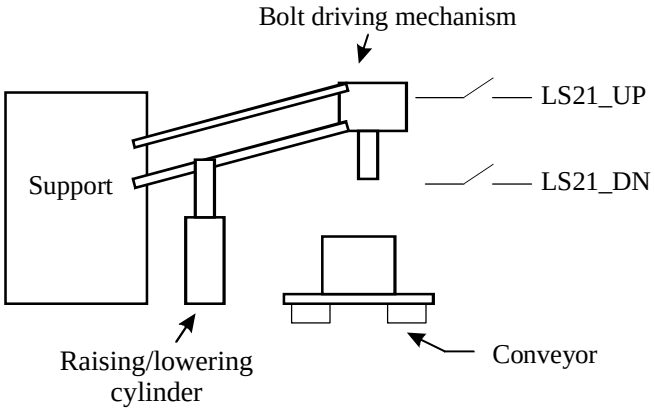
The pallet-raising mechanism is driven by a single-action pneumatic cylinder (solenoid). Once the PALL21_UP output is energized, the clamp moves the pallet (and engine) off the conveyor and into a fixture to get the engine in proper alignment with the gripper clamp. PALL21_UP must remain **on** to hold the assembly in the fixture. If PALL21_UP is turned **off**, the pallet falls back onto the conveyor. There are no limit switches indicating the pallet is in the proper position. Allow 1.5 seconds for the clamping operation to place the pallet into the fixture and 0.5 seconds for the pallet to unclamp and fall to the conveyor.



(a)

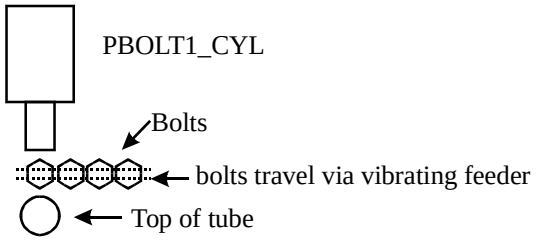


(b)

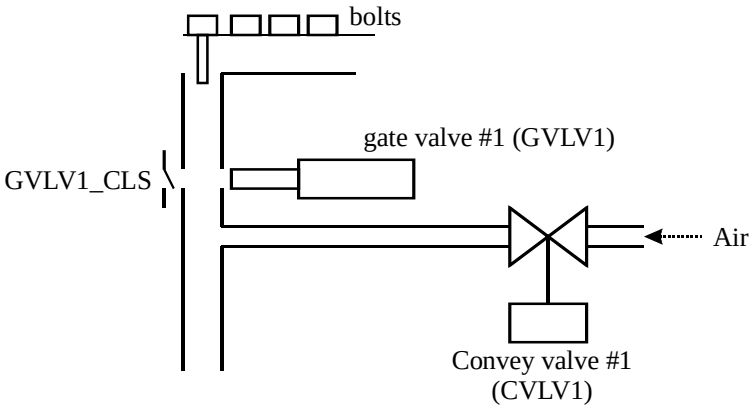


(c)

Figure SP6.16. Bolt driving station: *(a)* top view; *(b)* side view; *(c)* end view; *(d)* top view of bolt conveying mechanism; *(e)* side view of bolt conveying. *(continued)*



(d)



(e)

Figure SP6.16. *(continued)*

The mechanism used to lower and raise the bolt driving mechanism consists of a double-action linear pneumatic cylinder. When the HEAD21_DOWN output is energized, the mechanism moves down and continues to move down as long as power is applied (turned **on**). When the HEAD21_UP output is energized, the mechanism moves up and continues to move up as long as power is applied (turned **on**). The mechanism stops if neither output is **on**, or if they are energized simultaneously. LS21_UP is **on** when the mechanism is in the fully "up" position. LS21_DN is **on** when the mechanism is in the fully "down" position.

The bolts are conveyed into position in the following manner. The bolts are marshaled by a vibrating feeder into a "line" so they are fed one at a time into the mechanism that places the bolt into a hose and is pneumatically conveyed to the bolt driver. Figures P6.19*d* and P6.19*e* show a more detailed view of this mechanism that conveys bolt #1 (the mechanism for bolt #2 is similar). Assume the two vibrating feeders (one for each bolt) are always on and thus are not a part of this problem. In order to convey bolt #1 and bolt #2 to the driving head, the following steps take place:

1. PBOLT1_CYL and PBOLT2_CYL are extended (turned **on**) to push the first bolt in each line into the open pneumatic tubes. These solenoids should only be activated for 1 second. They are single action solenoids and when PBOLT1_CYL

(or PBOLT2_CYL) is turned **off**, the solenoid retracts, allowing the next bolt to become the first in line.

2. GVLV1 and GVLV2 are turned **on** to close gate valve #1 and #2 at the top of the pneumatic tube for bolt #1 and bolt #2. GVLV1_CLS limit switch indicates when gate valve #1 is closed and GVLV2_CLS indicates when gate valve #2 is closed. Both limit switches must be closed before continuing to the next step. Do not consider the case when one of these limit switches never turns **on**. There are no limit switches indicating that the valves are open. These valves are controlled by single action solenoids and so gate valve #1 is opened when GVLV1 is turned **off**.
3. Open convey valve #1 (CVLV1 **on**) and convey valve #2 (CVLV2 **on**) to allow pressurized air to convey the bolts to the driving mechanism. At the head, each bolt enters a hollow tube whose sides are hexagonal and thus serve as the driver for the bolt. There are 2 proximity switches (PROXB1 and PROXB2) that indicate the presence of the bolt in the driving mechanism. Both proximity switches must be **on** before continuing to the next step. Do not consider the case when one of these proximity switches never turns **on**.

Note that gate valve #1 and gate valve #2 must be open for the first step above and closed for the second and third step. These valves are primarily used to seal the end of the pneumatic tube so that air can convey the bolt to the driving head. After the end of step 3, convey valve #1 and convey valve #2 should be closed.

The driving of the bolts is accomplished by two air motors. When AMOTOR1 is turned **on**, the tube that contains bolt #1 rotates and drives the bolt into the engine assembly. AMOTOR2 similarly drives bolt #2.

The start/stop switches are only for the station. They do not control any other stations, or the conveyor. Upon initial startup, assume there are no pallets present in either of the engaging hooks. If the stop switch is pressed at any time, the station operation should **pause**, except when the ENGAGE_21_CYL or ENGAGE_22_CYL engaging hook controls are activated. The operation **must not** pause when ENGAGE_21_CYL is activated (otherwise the station will contain two pallets with no space in between). If the stop switch is pressed when any engaging hook control is activated, the step should complete and the operation should advance to the next step. When the start switch is pressed while the operation is paused, the station should resume the suspended step. When paused, **do not advance** to the next step (exception noted earlier). When the station is paused, the raise/lower and air motor controls should be turned **off**. The pallet up solenoid PALL21_UP must not be turned **off** when paused (or the engine will be dropped onto the conveyor). Also, the engaging hook control, PBOLTx_CYL, and GVLVx **must not** be turned **off** when paused.

A separate reset switch is provided that when pressed, the bolt driving mechanism is raised, PBOLTx_CYL and GVLVx are turned **off**, convey valves closed, and the air motors turned **off**, and the process step is set as if the process is waiting for the next pallet. When the start switch is next pressed, no items are assumed present at the first engage position. To keep the problem simple, do not try to implement a function chart for reset; just activate the proper cylinder controls simultaneously until they are all finished. The reset switch should have no effect unless the operation is already paused.

Assume the tolerance on all timer values is ±0.1 seconds.

Assume the following physical inputs and outputs.

Tag/Var./Symbol	Description
START_PB	Start push button, N. O., **on** when starting.
STOP_PB	Stop push button, N. C., **off** when stopping.
RESET_PB	Reset push button, N. O., **on** when restoring process to initial state.
PROX21	Proximity sensor, **on** when pallet in position 1
LS21_UP	Limit switch that closes (**on**) when bolt driving mechanism is fully up
LS21_DN	Limit switch that closes (**on**) when bolt driving mechanism is fully down
GVLV1_CLS	Limit switch that closes (**on**) when gate valve #1 is fully closed.
GVLV2_CLS	Limit switch that closes (**on**) when gate valve #2 is fully closed.
PROXB1	Proximity switch that in **on** when bolt #1 is in position to be driven.
PROXB2	Proximity switch that in **on** when bolt #2 is in position to be driven.
ENGAGE_21_CYL	Engage hook 1 control, **on** to lower hook, **off** raises hook.
ENGAGE_22_CYL	Engage hook 2 control, **on** to lower hook, **off** raises hook.
HEAD21_UP	Bolt driving mechanism raise control, **on** to raise.
HEAD21_DOWN	Bolt driving mechanism lower control, **on** to lower.
PBOLT1_CYL	Pushes bolt #1 into pneumatic tube, **on** extends, **off** retracts.
PBOLT2_CYL	Pushes bolt #2 into pneumatic tube, **on** extends, **off** retracts.
GVLV1	Closes gate valve #1 to seal pneumatic tube, **on** to close, **off** to open.
GVLV2	Closes gate valve #2 to seal pneumatic tube, **on** to close, **off** to open.
CVLV1	Opens convey valve #1 to convey bolt #1 to driver, **on** to open valve, **off** to close.
CVLV2	Opens convey valve #2 to convey bolt #2 to driver, **on** to open valve, **off** to close.
AMOTOR1	Opens valve to air motor #1 to drive bolt #1, **on** to rotate bolt, **off** to stop rotation.
AMOTOR2	Opens valve to air motor #2 to drive bolt #2, **on** to rotate bolt, **off** to stop rotation.
PALL21_UP	Pallet up control, **on** to move pallet up and off the conveyor and clamp pallet in the proper position.

The addresses associated with the physical inputs and outputs are:

Tag/Var./Symbol	ControlLogix	Micro800	MLogix	SLC-500
START_PB	Local:4:I.Data.0	_IO_EM_DI_00	I:1/0	I:4/0
STOP_PB	Local:4:I.Data.1	_IO_EM_DI_01	I:1/1	I:4/1
RESET_PB	Local:4:I.Data.2	_IO_EM_DI_02	I:1/2	I:4/2
PROX21	Local:4:I.Data.3	_IO_EM_DI_03	I:1/3	I:4/3
LS21_UP	Local:4:I.Data.6	_IO_EM_DI_04	I:1/6	I:4/6
LS21_DN	Local:4:I.Data.7	_IO_EM_DI_05	I:1/7	I:4/7

GVLV1_CLS	Local:4:I.Data.8	_IO_EM_DI_06	I:1/8	I:4/8
GVLV2_CLS	Local:4:I.Data.9	_IO_EM_DI_07	I:1/9	I:4/9
PROXB1	Local:4:I.Data.14	_IO_EM_DI_12	I:1/14	I:4/14
PROXB2	Local:4:I.Data.15	_IO_EM_DI_13	I:1/15	I:4/15
ENGAGE_21_CYL	Local:6:O.Data.0	_IO_EM_DO_00	O:3/0	O:6/0
ENGAGE_22_CYL	Local:6:O.Data.1	_IO_EM_DO_01	O:3/1	O:6/1
HEAD21_UP	Local:6:O.Data.2	_IO_EM_DO_02	O:3/2	O:6/2
HEAD21_DOWN	Local:6:O.Data.3	_IO_EM_DO_03	O:3/3	O:6/3
PBOLT1_CYL	Local:6:O.Data.4	_IO_EM_DO_04	O:3/4	O:6/4
PBOLT2_CYL	Local:6:O.Data.5	_IO_EM_DO_05	O:3/5	O:6/5
GVLV1	Local:6:O.Data.10	_IO_P1_DO_00	O:3/10	O:6/10
GVLV2	Local:6:O.Data.11	_IO_P1_DO_01	O:3/11	O:6/11
CVLV1	Local:7:O.Data.0	_IO_P2_DO_02	O:4/0	O:7/0
CVLV2	Local:7:O.Data.1	_IO_P2_DO_03	O:4/1	O:7/1
AMOTOR1	Local:7:O.Data.6	_IO_P4_DO_00	O:4/6	O:7/6
AMOTOR2	Local:7:O.Data.7	_IO_P4_DO_01	O:4/7	O:7/7
PALL21_UP	Local:7:O.Data.12	_IO_X1_DO_02	O:4/12	O:7/12

SP6-17. Revised Parts Transfer Station Control. Using the function chart approach, implement the program for the following revision to a parts transfer station.

> Problem SP6-3 describes a station that transfers parts from a conveyor to a packaging machine. The parts are now made faster and so the cycle time of the parts transfer station of problem SP6-3 must be decreased. Specifically, the loading of the next 6 parts onto the turntable starts while the ram pushes parts into the packaging machine. Assume the time needed to extend and retract the ram is less than the time to accumulate 6 parts on the turntable.

SP6-18. Revised Drilling Station Control. Using the function chart approach, implement the program for the following revisions to the drilling station.

> Rework the drilling station of problem SP6-15, allowing certain operations to occur in parallel:
>
> 1. The extension/retraction of drill 1 and drill 2 occur simultaneously. However, make no assumption about which drilling operation finishes last.
> 2. Allow a new part to move into position 2 while the previous part is being drilled. However, a new part cannot move from position 2 to position 3 while a drilled part is moving out of position 3.

The assumption about initial startup does not change. Upon initial startup, assume that there are no parts waiting at gate 1 and that there are no parts in the station waiting to be drilled. DO NOT initiate a drilling operation while the first part moves into the station.

SP6-19. Revised Bolt Driving Station Control. Using the function chart approach, implement the program for the following revisions to the bolt driving station of problem SP6-16.

Rework the bolt driving station of SP6-16, allowing certain operations to occur in parallel:

While the bolt driving mechanism is lowering to the correct position, convey all bolts into position. Make no assumption about which operation finishes last.

Also, the station now conveys and drives 6 bolts.

Assume the following additional physical inputs and outputs.

Tag/Var./Symbol	Description
GVLV3_CLS	Limit switch that closes (**on**) when gate valve #3 is fully closed.
GVLV4_CLS	Limit switch that closes (**on**) when gate valve #4 is fully closed.
GVLV5_CLS	Limit switch that closes (**on**) when gate valve #5 is fully closed.
GVLV6_CLS	Limit switch that closes (**on**) when gate valve #6 is fully closed.
PROXB3	Proximity switch that in **on** when bolt #3 is in position to be driven.
PROXB4	Proximity switch that in **on** when bolt #4 is in position to be driven.
PROXB5	Proximity switch that in **on** when bolt #5 is in position to be driven.
PROXB6	Proximity switch that in **on** when bolt #6 is in position to be driven.
PBOLT3_CYL	Pushes bolt #3 into pneumatic tube, **on** extend, **off** retracts.
PBOLT4_CYL	Pushes bolt #4 into pneumatic tube, **on** extend, **off** retracts.
PBOLT5_CYL	Pushes bolt #5 into pneumatic tube, **on** extend, **off** retracts.
PBOLT6_CYL	Pushes bolt #6 into pneumatic tube, **on** extend, **off** retracts.
GVLV3	Closes gate valve #3 to seal tube, **on** to close, **off** to open.
GVLV4	Closes gate valve #4 to seal tube, **on** to close, **off** to open.
GVLV5	Closes gate valve #5 to seal tube, **on** to close, **off** to open.
GVLV6	Closes gate valve #6 to seal tube, **on** to close, **off** to open.
CVLV3	Opens convey valve #3 to convey bolt #3 to driver, **on** to open valve, **off** to close.
CVLV4	Opens convey valve #4 to convey bolt #4 to driver, **on** to open valve, **off** to close.
CVLV5	Opens convey valve #5 to convey bolt #5 to driver, **on** to open valve, **off** to close.
CVLV6	Opens convey valve #6 to convey bolt #6 to driver, **on** to open valve, **off** to close.
AMOTOR3	Opens valve to air motor #3 to drive bolt #3, **on** to rotate bolt, **off** to not rotate.
AMOTOR4	Opens valve to air motor #4 to drive bolt #4, **on** to rotate bolt, **off** to not rotate.

AMOTOR5	Opens valve to air motor #5 to drive bolt #5, **on** to rotate bolt, **off** to not rotate.
AMOTOR6	Opens valve to air motor #6 to drive bolt #6, **on** to rotate bolt, **off** to not rotate.

The addresses associated with the additional physical inputs and outputs are:

Tag/Var./Symbol	ControlLogix	Micro800	MLogix	SLC-500
GVLV3_CLS	Local:4:I.Data.10	_IO_EM_DI_08	I:1/10	I:4/10
GVLV4_CLS	Local:4:I.Data.11	_IO_EM_DI_09	I:1/11	I:4/11
GVLV5_CLS	Local:4:I.Data.12	_IO_EM_DI_10	I:1/12	I:4/12
GVLV6_CLS	Local:4:I.Data.13	_IO_EM_DI_11	I:1/13	I:4/13
PROXB3	Local:5:I.Data.0	_IO_P1_DI_00	I:2/00	I:5/00
PROXB4	Local:5:I.Data.1	_IO_P1_DI_01	I:2/01	I:5/01
PROXB5	Local:5:I.Data.2	_IO_P1_DI_02	I:2/02	I:5/02
PROXB6	Local:5:I.Data.3	_IO_P1_DI_03	I:2/03	I:5/03
PBOLT3_CYL	Local:6:O.Data.6	_IO_EM_DO_06	O:3/6	O:6/6
PBOLT4_CYL	Local:6:O.Data.7	_IO_EM_DO_07	O:3/7	O:6/7
PBOLT5_CYL	Local:6:O.Data.8	_IO_EM_DO_08	O:3/8	O:6/8
PBOLT6_CYL	Local:6:O.Data.9	_IO_EM_DO_09	O:3/9	O:6/9
GVLV3	Local:6:O.Data.12	_IO_P1_DO_02	O:3/12	O:6/12
GVLV4	Local:6:O.Data.13	_IO_P1_DO_03	O:3/13	O:6/13
GVLV5	Local:6:O.Data.14	_IO_P2_DO_00	O:3/14	O:6/14
GVLV6	Local:6:O.Data.15	_IO_P2_DO_01	O:3/15	O:6/15
CVLV3	Local:7:O.Data.2	_IO_P3_DO_03	O:4/2	O:7/2
CVLV4	Local:7:O.Data.3	_IO_P3_DO_01	O:4/3	O:7/3
CVLV5	Local:7:O.Data.4	_IO_P3_DO_02	O:4/4	O:7/4
CVLV6	Local:7:O.Data.5	_IO_P3_DO_03	O:4/5	O:7/5
AMOTOR3	Local:7:O.Data.8	_IO_P4_DO_02	O:4/8	O:7/8
AMOTOR4	Local:7:O.Data.9	_IO_P4_DO_03	O:4/9	O:7/9
AMOTOR5	Local:7:O.Data.10	_IO_X1_DO_00	O:4/10	O:7/10
AMOTOR6	Local:7:O.Data.11	_IO_X1_DO_01	O:4/11	O:7/11