6 Sequential Applications

Chapter Topics:

- Function charts
- Simple ladder logic implementation of function charts
- Parallel operations

OBJECTIVES

Upon completion of this chapter, you will be able to:

- Draw a function chart, given the operational description of a sequential process
- Translate the function chart to ladder logic
- Handle pause and reset of the sequential operation

Scenario: Using a proximity sensor to detect material moving into and out of a station.

Commonly, only one sensor is used to detect the presence of material as it moves into a station, is processed, and moves out of the station. As an example, consider the problem of applying a label to each box as it travels on a conveyor, shown in Figure 6.1. A retro-reflective proximity sensor (PROX) is used to detect the presence of the box. Assume PROX turns **on** when the box is in the proper position to have the label applied. When the labeling station is started, the presence of a box must be detected, the label is applied (involving multiple steps), and then the operation is repeated. A chart of the steps of this process is shown in Figure 6.2. The process starts waiting for a box to be detected. When PROX turns **on**, then the machinery applies the label (multiple steps). When the labeling steps are finished, then the station waits for the next box. However, as depicted in Figure 6.2, the operation **does not work!** After applying the label, PROX remains **on**, (box still in

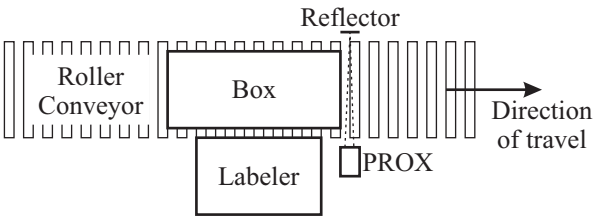

Figure 6.1. Labeling station.

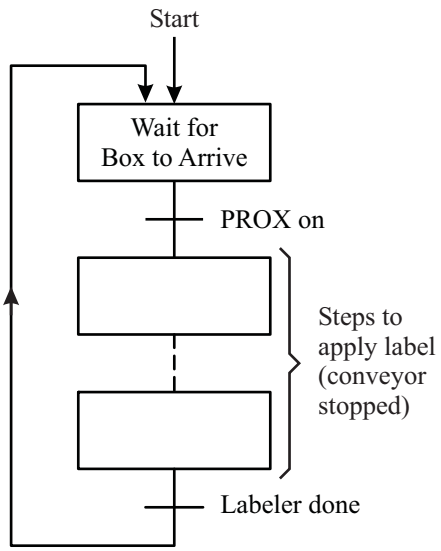

Figure 6.2. Chart of labeling station operation.

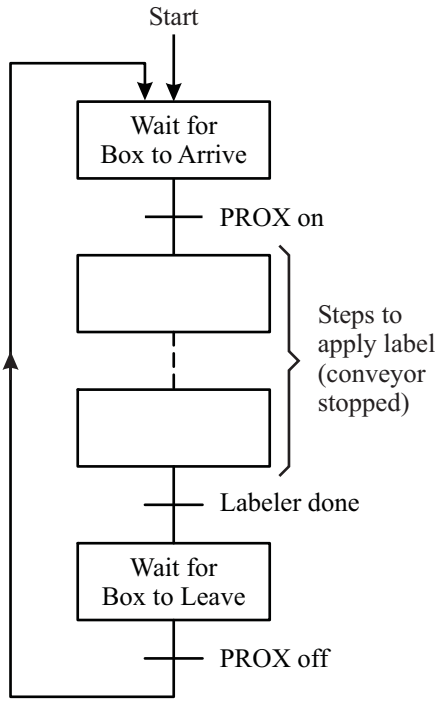

Figure 6.3. Corrected chart of labeling station operation.

station) and so the condition that indicates a new box (PROX **on**) is true. Therefore, a new label is applied to the box before it leaves the station. If left to run unattended, this station will continue to apply multiple labels to the first box that enters the station!

Solution: The station must detect that a labeled box has exited the station before detecting that a new box has entered. Therefore, a step must be added to wait for the box to leave the station, detecting PROX is **off**. The correct chart of this process is shown in Figure 6.3. The moral of this scenario is to remember that one must detect that a proximity sensor is first **off** before it can be detected to be **on**.

Design Tip

When one proximity sensor is used to sense material moving in/out , one must detect that the sensor is **off** before detecting the sensor is **on** (or vice versa).

6.1 INTRODUCTION

With the basic ladder logic contact, timer, and counter instructions, one is able to tackle more significant problems. This chapter introduces ladder logic program design for

sequential applications, a significant contribution of the text. More advanced techniques for sequential control are treated in Chapters 9 and 21.

The sequential design technique is based on describing the operation as a function chart and then translating the function chart to ladder logic code. The ladder logic primarily uses the basic contacts and coils. Timers and counters are used only when explicitly needed by the operation. The ability to pause and reset an operation is added to the basic sequential design. Operations with parallel steps and machine control involving manual and single-step modes are also considered. The design technique uses the set/reset coils or latch/unlatch coils. Some older processors (e.g., Modicon x84 and Siemens TI-5x5) do not have set/reset coils and an alternate technique is presented in Erickson (2011).

6.2 FUNCTION CHART

The basic tool used to design sequential control applications is the function chart. This method of describing sequential operations is described in the IEC 848 standard (IEC, 1988) and incorporated as one of the IEC 61131-3 languages (IEC, 2013). The form of the function chart described in this chapter is a simplified version of the IEC 61131-3 SFC (sequential function chart) language. The full IEC sequential function chart language is described in Chapter 14.

The general form of the function chart is shown in Figure 6.4. The function chart has the following major parts:

Steps of the sequential operation,

Transition conditions to move to next step

Actions of each step

The **initial step** is indicated by the double-line rectangle. The initial step is the initial state of the ladder logic when the PLC is first powered up or when the operator resets the operation. The **steps** of the operation are shown as rectangles on the left side of the diagram. Unless shown by an arrow, the progression of the steps proceeds from top to bottom. Each step rectangle contains a short description of what is happening during the step. To the left of the step rectangle is the variable/symbol/tag name of the step-in-progress coil (or bit) that is **on** when that step is active. The **transition condition** is shown to the right of a horizontal bar between steps. If a step is active and the transition condition below that step becomes true, the step becomes inactive, and the next step becomes active. The stepwise flow continues until the bottom of the diagram. At the bottom, the sequencing may end, as indicated by a filled black circle within another circle, or it may repeat by going back to a previous step. The **actions** associated with a step are shown in the rectangle to the right of the step. The actions are output(s) that are **on** when a step is active. Any outputs not listed are assumed to be **off**. However, the set/reset of outputs may be indicated. Any timer or counter active during a particular step is also listed as an action.

The function chart is prepared from the operational description of the system. Often, the hardest part about formulating the function chart is making a distinction between the transitions and the steps. Also, one must remember that physical outputs are actions associated with a step. In order to help in the recognition of the steps and transitions within an operational description, use the following definitions:

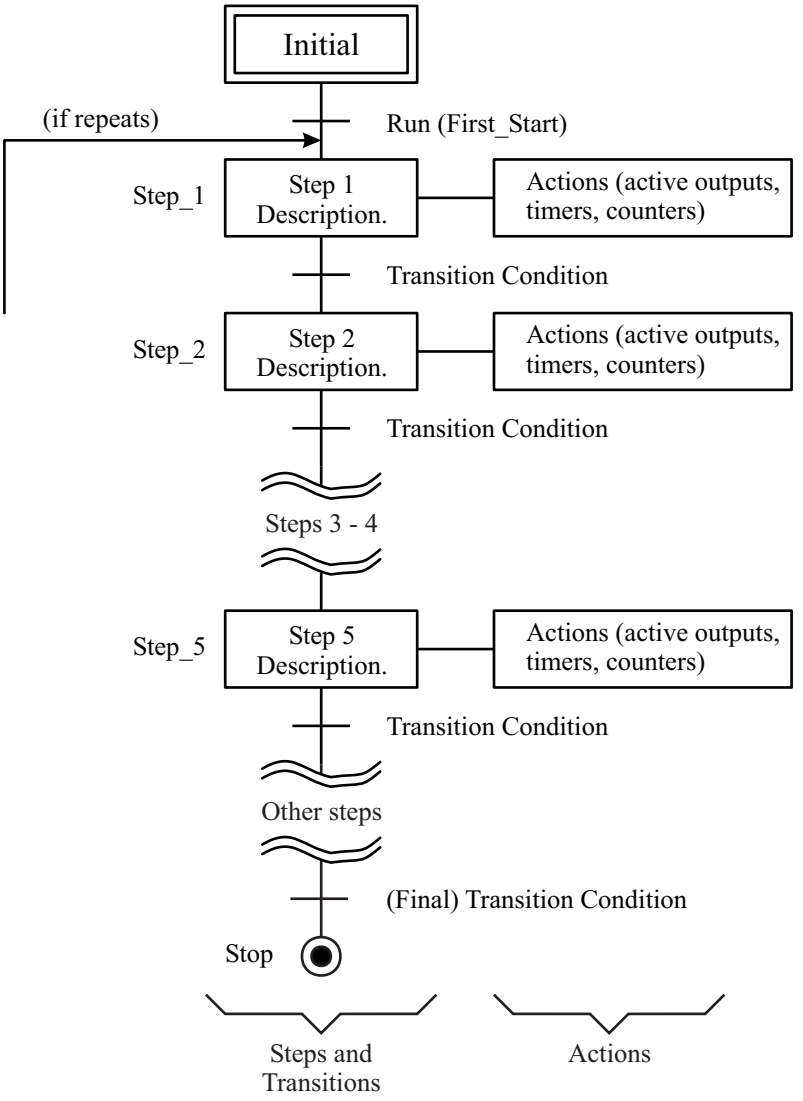

Figure 6.4. General function chart.

Step:

Operation spanning a length of time (however long or short).
The time period may be defined or undefined.

Transition:

Physical input device or internal coil turning **on** or **off**

— or —

Timer or counter **done**

A transition condition is recognized when the narrative describes a physical input device or internal coil turning **on** or **off**. Alternatively, the end of a defined time period also

signals a transition to the next step. If the narrative describes a physical output being turned on/off, that is not a transition. A physical output is considered a step action and the turning on/off of a physical output is handled by a change in the active step. For example, if "Output1" is being turned on as the active step is changed from "Step1" to "Step2," it is accomplished by **not** listing "Output1" as a "Step1" action and by listing it as a "Step2" action. Output1 is **not** a transition condition. The change in Output1 does not cause a change in the active step, but is a consequence of the change in the active step.

Design Tip

When constructing the function chart, remember that physical outputs **never** occur as part of the transition condition. Also, physical inputs are **never** an action.

These concepts are illustrated by the following example.

Example 6.1. Metal Shear Control. Design the function chart of the program to control the metal shear shown in Figure 6.5 and whose operation is described as:

> The shear cuts a continuous length of steel strip. Two conveyors (driven by CONV1_MTR and CONV2_MTR) move the strip into position. Inductive proximity sensor PROX turns **on** to indicate that the strip is in position to be sheared. When the strip is in position, both conveyors should stop. A hydraulic cylinder (controlled by SHEAR_CYL_RET) is then retracted to move the shear down to cut the material. Limit switch DOWN_LS closes (turns **on**) when the shear is fully down. The cylinder is then extended to move the shear blade up. Limit switch UP_LS closes when the shear blade is fully up. Conveyor 2 (controlled by CONV2_MTR) is now turned **on** to move the cut sheet out of the station. The proximity sensor PROX turns **off** when the sheet has been moved out of the station. Both conveyors are now operated to move the strip into position, and the operation repeats.
>
> Your program is not controlling conveyor 3, so assume it is always running.
>
> The shear is controlled by SHEAR_CYL_RET, a single action linear hydraulic cylinder. Once SHEAR_CYL_RET is energized, the shear blade moves down to cut the material until a mechanical stop is reached and remains in the "down" position as long as power is applied (turned **on**). The shear blade moves up when power is removed (turned **off**).
>
> Upon initial startup, no material is in the shear and the conveyors operate to bring the material into the shearing position (PROX turns **on**). The start switch should have no effect if the process is already running. If the stop switch is pressed at any time, the station operation should pause, except when the shear blade is moving. If the stop switch is pressed when the shear blade is moving, the blade movement must complete. When the start switch is pressed while the operation is paused, the station should resume the suspended step. When the station is paused, the conveyor drive motors should be shut off.

Assume the following physical input and physical outputs:

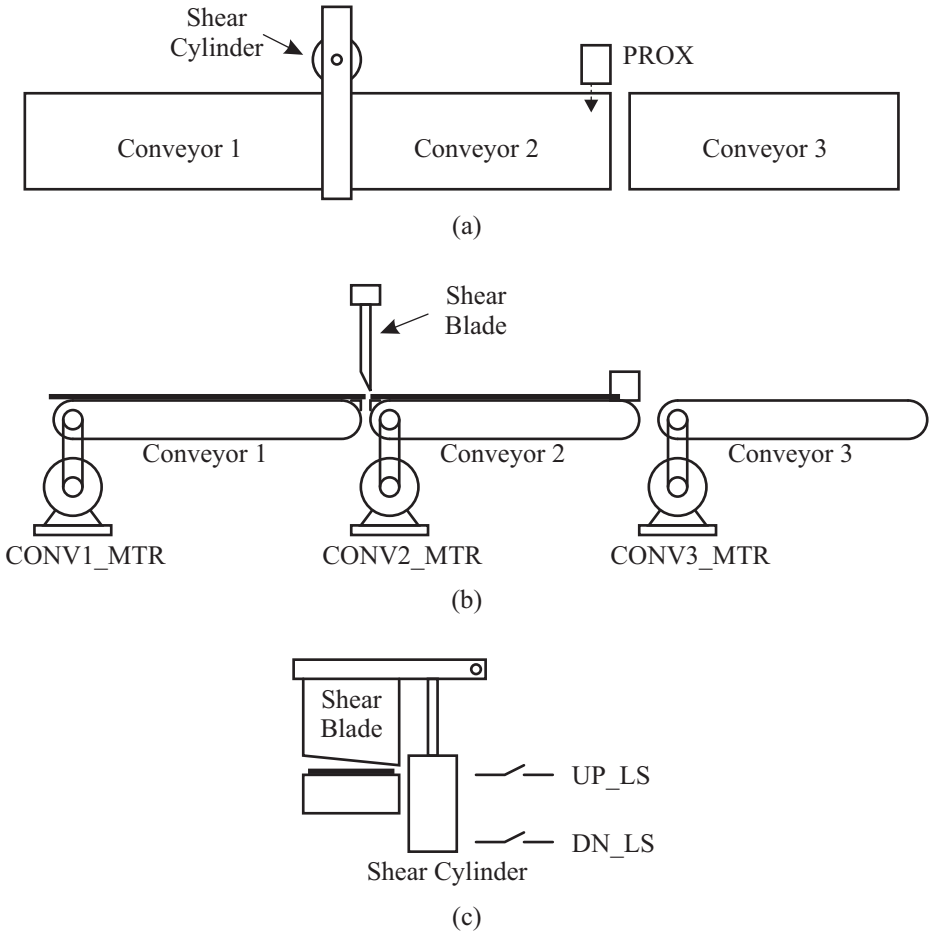

Figure 6.5. Metal shear: *(a)* top view; *(b)* front view; *(c)* side view.

Tag/Var./Symbol	Description
START_PB	Start push button, N. O., **on** when starting
STOP_PB	Stop push button, N. C., **off** when stopping
PROX	Proximity sensor, **on** when strip in shearing position
DOWN_LS	Limit switch, N. O., **on** (closed) when blade fully down
UP_LS	Limit switch, N. O., **on** (closed) when blade fully up
CONV1_MTR	Conveyor 1 control, **on** to move material on conveyor 1
CONV2_MTR	Conveyor 2 control, **on** to move material on conveyor 2
SHEAR_CYL_RET	Shear cylinder control, **on** to retract cylinder and move blade down

Solution. There are two main steps to develop the function chart:

1. Identify the steps and transition conditions.
2. Add step actions.

To identify the steps and transitions, the first paragraph of the process description is repeated, with the steps identified by the underlined phrases and the transition conditions identified by the *italicized phrases*. Often, it is easier to identify the first transition condition (signaled by an input sensor change) and then recognize the step before and the step after the transition condition. Also, many times the steps and transition conditions alternate during the narrative.

> The shear cuts a continuous length of steel strip. Two conveyors (driven by CONV1_MTR and CONV2_MTR) move the strip into position. Inductive proximity sensor *PROX turns **on*** to indicate that the strip is in position to be sheared. When the strip is in position, both conveyors should stop. A hydraulic cylinder (controlled by SHEAR_CYL_RET) is then retracted to move the shear down to cut the material. Limit switch *DOWN_LS closes (turns **on**)* when the shear is fully down. The cylinder is then extended to move the shear blade up. Limit switch *UP_LS closes* when the shear blade is fully up. Conveyor 2 (controlled by CONV2_MTR) is now turned **on** to move the cut sheet out of the station. The proximity sensor *PROX turns **off*** when the sheet has been moved out of the station. Both conveyors are now be operated to move the strip into position, and the operation repeats.

Notice that the phrase "... both conveyors should stop." is not marked as a transition condition. This phrase describes a physical output being turned on/off and that will be handled by a change in the active step.

So, the steps and the transition conditions that indicate the end of each step are:

Step	Transition Condition (out of step)
Move strip into position	PROX on
Move shear down	DOWN_LS on
Move shear up	UP_LS on
Move cut sheet out	PROX off

These steps and the transition conditions between them are shown in Figure 6.6. The "off" state of PROX that signals the end of the fourth step is shown with the "/" in front of the variable name. The variable name of the step-in-progress bit for each step is also shown beside the step box. This particular operation repeats, indicated by the line from the fourth step back to the first step.

The next part of the function chart development is to add the actions to each step. Reading back through the metal shear narrative, the process actions for each step are:

Step	Action
Move strip into position	CONV1_MTR and CONV2_MTR
Move shear down	SHEAR_CYL_RET
Move shear up	
Move cut sheet out	CONV2_MTR

These actions are added to the steps and the transition conditions to form the function chart shown in Figure 6.7.

The part of the narrative that describes the operation pause is handled in the ladder logic code and is considered in the following section.

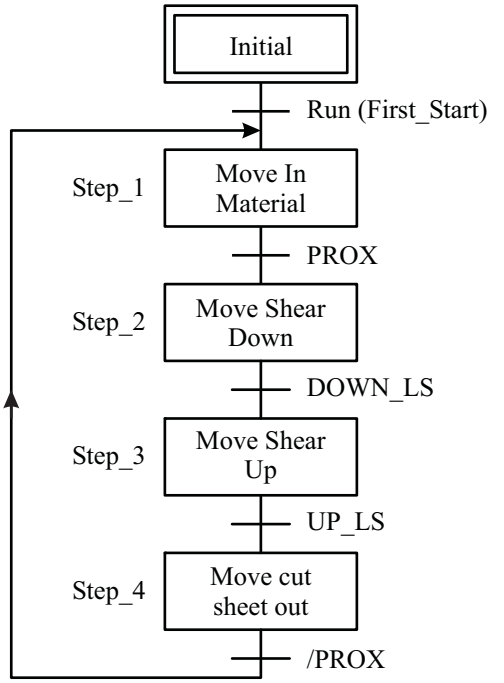

Figure 6.6. Steps and transitions for metal shear.

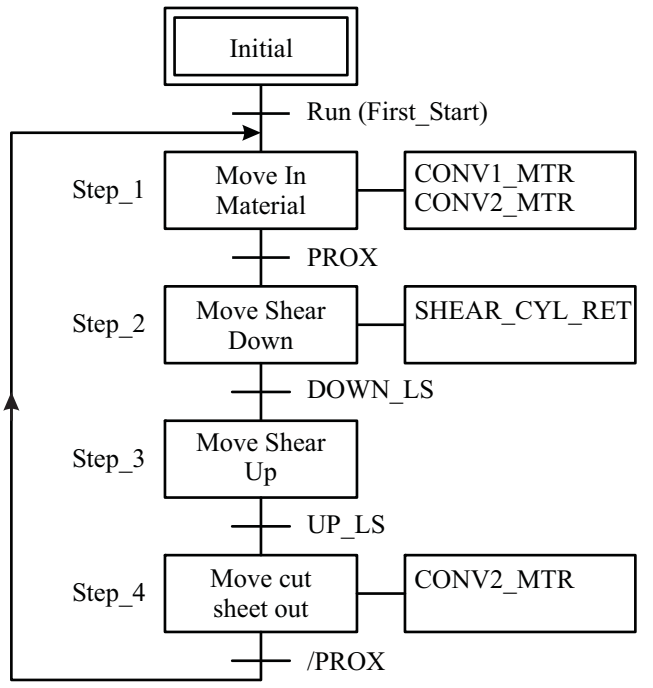

Figure 6.7. Function chart for metal shear.

6.3 IMPLEMENTING FUNCTION CHART IN LADDER LOGIC

Once a function chart has been developed, it needs to be implemented in ladder logic. There are multiple ways to accomplish this task. The design technique described in this chapter utilizes only the basic ladder logic instructions to implement the step and transition logic. Other methods are shown in Chapter 9.

The author calls this method the "cookie cutter" or "template-based" approach because the form of the ladder logic code is the same, regardless of the application. Also, this approach aids in debugging because the logic that handles the transitions and the logic that handles the step actions are distinct. The latter advantage is apparent when comparing this approach to the ad-hoc approach of section 9.5.

The code is broken into the following sections:

Start/stop/pause of overall operation

First start

Transitions between steps

Step actions

Each of these code templates is covered in detail and then applied to the metal shear of Example 6.1.

The start/stop/pause of the overall operation is handled as the rung in Figure 6.8, which is the same general format as the start/stop rung shown in section 2.7. An internal coil (tag/variable/symbol) named Run controls the overall operation of the function chart. It will be used to turn **off** physical outputs that need to be **off** when pausing the operation. The Run may be part of a transition condition. The optional permissive conditions must be satisfied to allow the operation to be started or restarted after an abnormal condition. The optional lockout conditions cause the operation to pause or stop in addition to preventing a restart.

The "first start" transition condition causes the operation to be initiated when no steps are currently active. The ladder logic to set (latch) the first step (Step_1) is shown in Figure 6.9. When the Run internal tag/variable/symbol is turned **on** (start push button pressed) and no steps are active (Step_N is the last step), the Step_1 internal coil is set (latched). START_PB could be used in place of Run in Figure 6.9, but if the run rung has permissive and/or lockout conditions, these conditions should also be repeated on the rung that starts

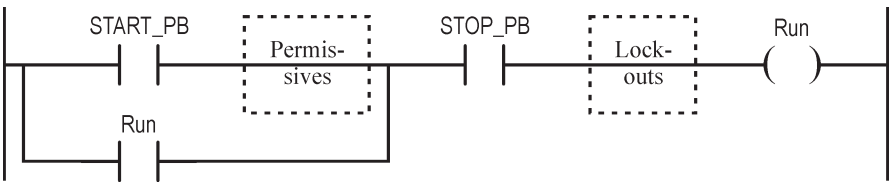

Figure 6.8. General start/stop/pause rung.

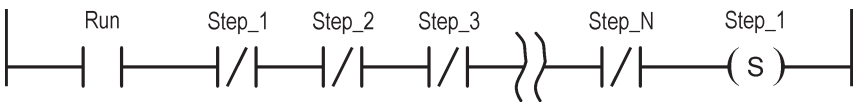

Figure 6.9. General first start rung.

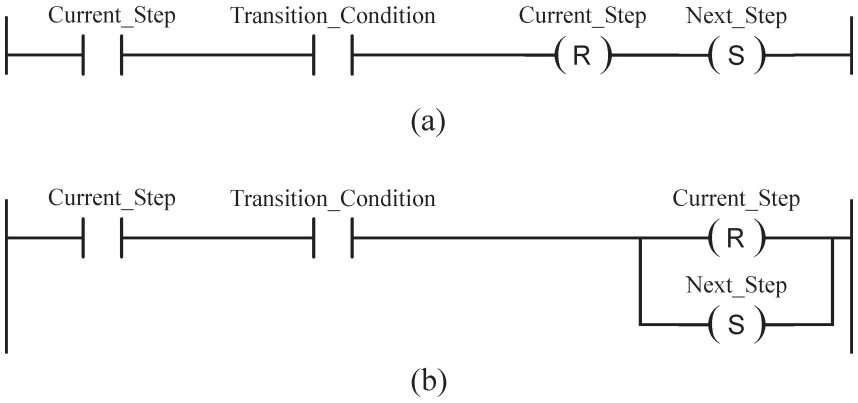

Figure 6.10. General transition between steps: *(a)* series coils; *(b)* parallel coils.

the operation for the first time. As explained in section 2.7, a change to lockouts and permissives should affect only one rung.

Transitions between steps are handled as shown in Figure 6.10. The logic implements the transition condition below the step in the function chart, which is the transition condition **out** of a step. When the current step is active (Current_Step is **on**) and the transition condition is true, then the step-in-progress bit of the current step is reset and the step-in-progress-bit of the next step is set. Thus, the next step becomes active and the current step becomes inactive. For the S7 Portal and Modicon processors, the coils may be in series (as shown in Figure 6.10*a*) or in parallel. ControlLogix unlatch and latch coils may be in series. For the MicroLogix, S7 Classic, and Emerson processors, the coils must be in parallel, as shown in Figures 6.10*b* (except that the MicroLogix uses unlatch and latch coils).

If the PLC does not have set/reset or latch/unlatch coils (e.g., Modicon 984 and Siemens TI-5x5) then another approach is detailed in section 6.8 of Erickson (2011).

The step-in-progress internal coils are used to control the step actions. The appropriate step-in-progress bits turn **on** the outputs and timers that are the step actions. The Run internal coil is also used as part of the condition for those actions that must be **off** when the operation is paused. For example, if the MOTOR_ON output should be **on** in steps 4 and 15 of the function chart (represented by Step_4 and Step_15), then the logic driving MOTOR_ON appears as shown in Figure 6.11. The Run internal coil turns **off** MOTOR_ON if step 4 or step 15 is active and the stop push button is pressed to pause the operation. When the operation is resumed (by pressing the start push button), then MOTOR_ON is reactivated. If the Run internal coil is omitted from the rung in Figure 6.11, then MOTOR_ON will remain **on** when the operation is paused when in step 4 or step 15.

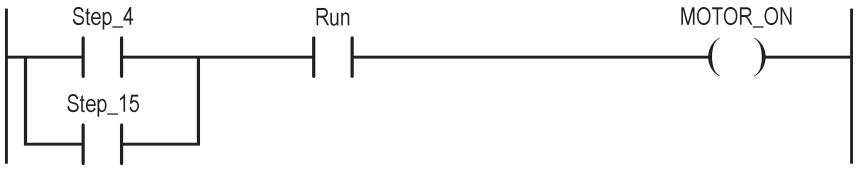

Figure 6.11. Example step action.

Note that in Figure 6.11, the Step_4 and Step_15 step-in-progress bits are in parallel, meaning that MOTOR_ON is an action in steps 4 and 15. The MOTOR_ON output is **off** for any other steps.

If the action associated with a step is a set/reset (latch/unlatch) of an output, then the output coil of Figure 6.11 is replaced by a set/latch or reset/unlatch coil.

Design Tip

Repeating outputs is a common mistake when implementing a function chart where a particular output is the action for more than one step. Consider the output first and then the steps for which it is **on** to avoid repeating output instructions.

Example 6.2. Metal Shear Control. Use ladder logic to implement the metal shear operation described in Example 6.1.

The physical inputs and physical outputs are:

Tag/Var./Symbol	Description
START_PB	Start push button, N. O., **on** when starting
STOP_PB	Stop push button, N. C., **off** when stopping
PROX	Proximity sensor, **on** when strip in shearing position
DOWN_LS	Limit switch, N. O., **on** (closed) when blade fully down
UP_LS	Limit switch, N. O., **on** (closed) when blade fully up
CONV1_MTR	Conveyor 1 control, **on** to move material on conveyor 1
CONV2_MTR	Conveyor 2 control, **on** to move material on conveyor 2
SHEAR_CYL_RET	Shear cylinder control, **on** to retract cylinder and move blade down

The addresses associated with the inputs and outputs:

Tag/Var./Symbol	ControlLogix	MLogix	Siemens	Modicon	Emerson
START_PB	Local:1:I.Data.0	I:0/00	%I0.0	%I0.2.0	%I1
STOP_PB	Local:1:I.Data.1	I:0/01	%I0.1	%I0.2.1	%I2
PROX	Local:1:I.Data.2	I:0/02	%I0.2	%I0.2.2	%I3
DOWN_LS	Local:1:I.Data.3	I:0/03	%I0.3	%I0.2.3	%I4
UP_LS	Local:1:I.Data.4	I:0/04	%I0.4	%I0.2.4	%I5
CONV1_MTR	Local:2:O.Data.0	O:1/00	%Q4.0	%Q0.3.0	%Q1
CONV2_MTR	Local:2:O.Data.1	O:1/01	%Q4.1	%Q0.3.1	%Q2
SHEAR_CYL_RET	Local:2:O.Data.2	O:1/02	%Q4.2	%Q0.3.2	%Q3

Solution. The function chart for the shear operation is shown in Figure 6.7. Before developing the ladder logic code, the internal tags/variables/symbols should be identified:

Tag/Var./Symbol	Description
Run	Indicates operation running
Step_1 to Step_4	Step-in-progress bits for steps

The addresses or data types associated with the tags/variables/symbols:

Tag/Var./Symbol	CLogix Data Type	MLogix Addr.	Siemens Addr.	Type	Modicon Data Type	Emerson Addr.
Run	BOOL	B3/0	%M0.0	BOOL	EBOOL	%M1
Step_1 to	BOOL	B20/1	%M50.1	BOOL	EBOOL	%M51
Step_4	BOOL	B20/4	%M50.4	BOOL	EBOOL	%M54

The ladder logic code is broken into the following sections:

Start/stop/pause of overall operation

First start

Transitions between steps

Step actions

The IEC 61131-3 code for the metal shear, shown in Figure 6.12, is developed using the code templates shown in Figures 6.8 - 6.11. A rung comment is shown within a rectangle above the rung. The function of each rung is as follows:

1. Start/stop/pause of overall operation
2. Transition from initial step (starting the operation for the very first time)
3. Transition from step 1 to step 2
4. Transition from step 2 to step 3
5. Transition from step 3 to step 4
6. Transition from step 4 to step 1
7. Control of conveyor 1 (an action for step 1)
8. Control of conveyor 2 (an action for steps 1 and 4)
9. Control of shear cylinder (an action for step 2)

The initial start of the operation is handled like Figure 6.9. Note the use of the Run in rungs 7 and 8 to turn **off** the conveyors when the station is paused. Since the shear cylinder operation should not stop if the stop push button is pressed while it is moving, Run is not used as a condition in rung 9.

The CONV2_MTR output is an action for 2 steps, as shown in rung 8 of Figure 6.12. Note that if a particular output is an action for multiple steps, then the step-in-progress bits of each step are placed in parallel. When a particular output is the action for more than one step, novice programmers often repeat the outputs. If one did not consider the output first and then steps for which it is **on** then the ladder logic driving the physical outputs for the shear may appear as in Figure 6.13. Rungs 8 and 10 both drive the CONV2_MTR output. What is the result? Since rung 10 is scanned after rung 8, the logic of rung 10 will override the logic of rung 8. Consequently, CONV2_MTR is never **on** in step 1, causing the material to jam as it is conveyed into position.

Depending on the particular PLC used to implement this example, the ladder logic will appear different from the ladder logic shown in Figure 6.12. The ControlLogix and CompactLogix use latch/unlatch coils and the MicroLogix/SLC-500 processors parallel the latch and unlatch coils.

Example 6.2 does not have all of the features of a real application, but serves to illustrate the basic approach to implementing a function chart in ladder logic. The next example adds timers, counters, and reset to an application.

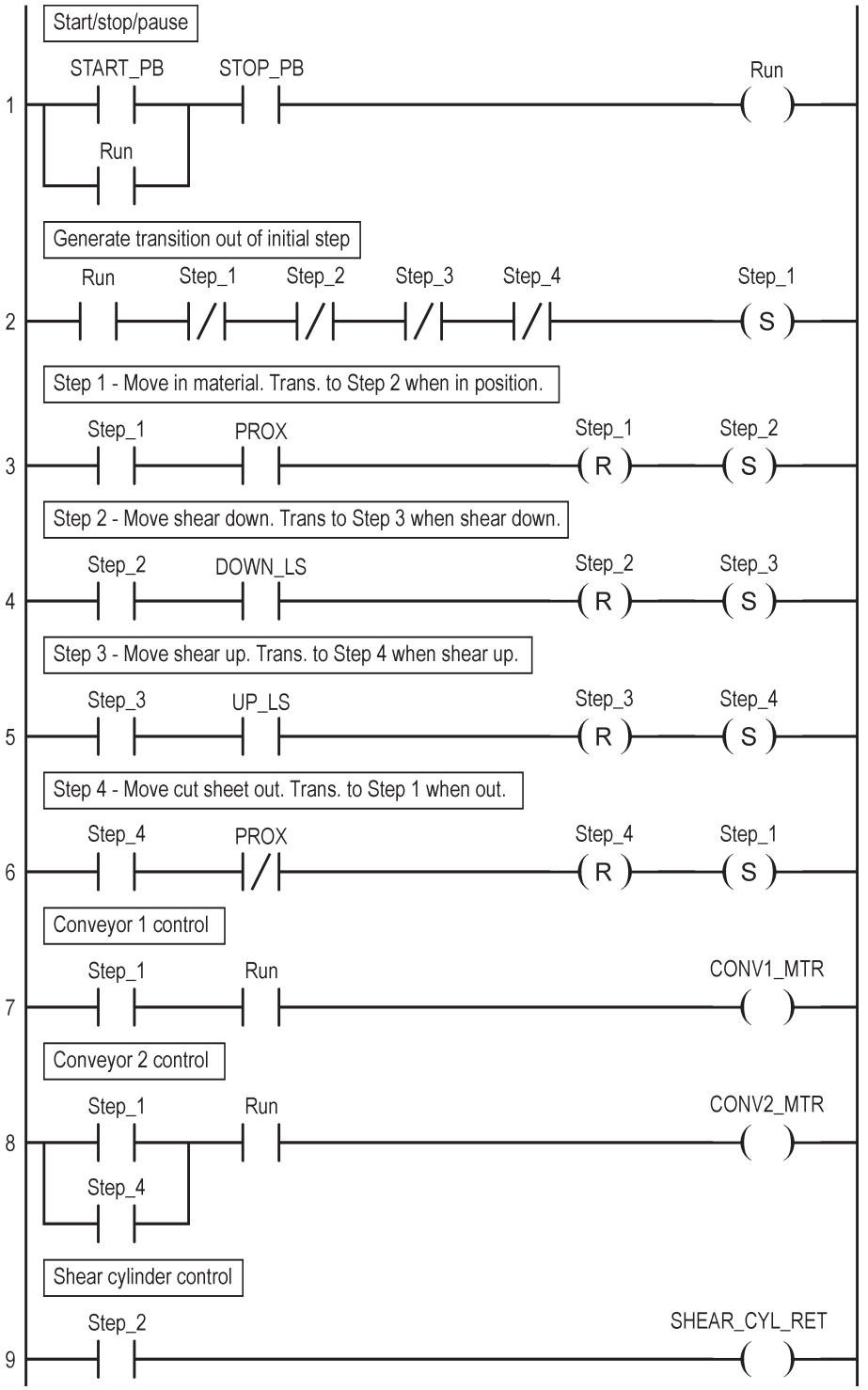

Figure 6.12. IEC ladder logic for metal shear.

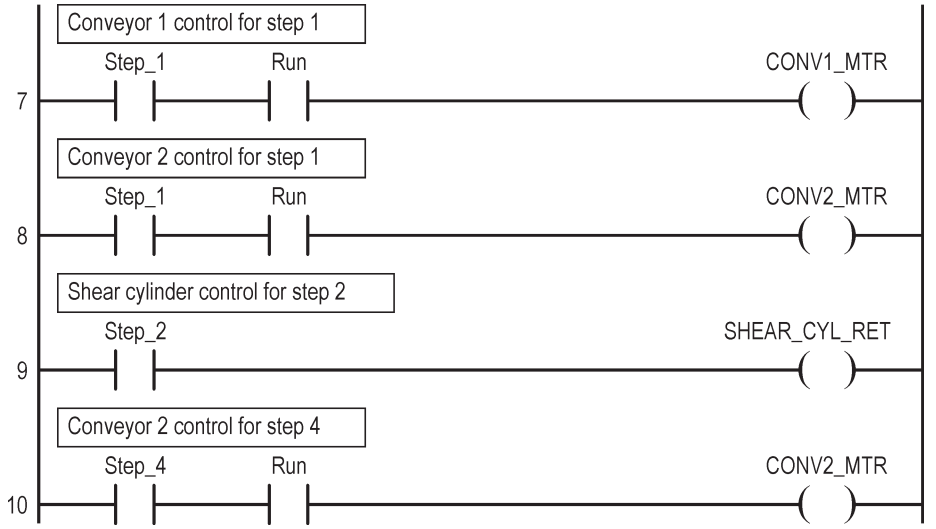

Figure 6.13. Incorrect output logic for metal shear.

Example 6.3. Tub Loader Control. Design the function chart of the program to control the tub loader described below. Also, implement the control with ladder logic.

Figure 6.14 shows the layout of a parts tub loader machine. Parts are placed on the belt conveyor by a milling machine. The parts move down the conveyor and drop into the parts tub. Parts on the belt conveyor are detected by a photoelectric sensor, PE272, which is **off** as a part interrupts the beam. Assume PE272 detects the part as it falls into the tub. After 100 parts are deposited in the tub, the tub is moved out and a new, empty tub moves into position. To change the tub, the following operation must take place:

Open Gate 1 (GATE1_OPLS senses when open).

Hold Gate 1 open and wait for TUB_PROX to be **off** for 3 seconds to allow the full tub to be moved out of the loading station. Run the tub roller conveyor to move out the full tub.

Gate 1 is closed (GATE1_CLLS senses when closed).

Gate 2 is opened (GATE2_OPLS senses when open).

Hold Gate 2 open to allow an empty tub to move down a slight incline into the loading station. When the tub contacts the tub roller conveyor, the tub roller conveyor moves the tub into position. When TUB_PROX is **on** for 5 seconds, the tub is in position (front resting on Gate 1).

Gate 2 is closed (GATE2_CLLS senses when closed).

The TUB_PROX proximity sensor is **on** when the tub is present, though not necessarily in position. Hence, the delays ensure the empty tub has moved in and the full one has moved out.

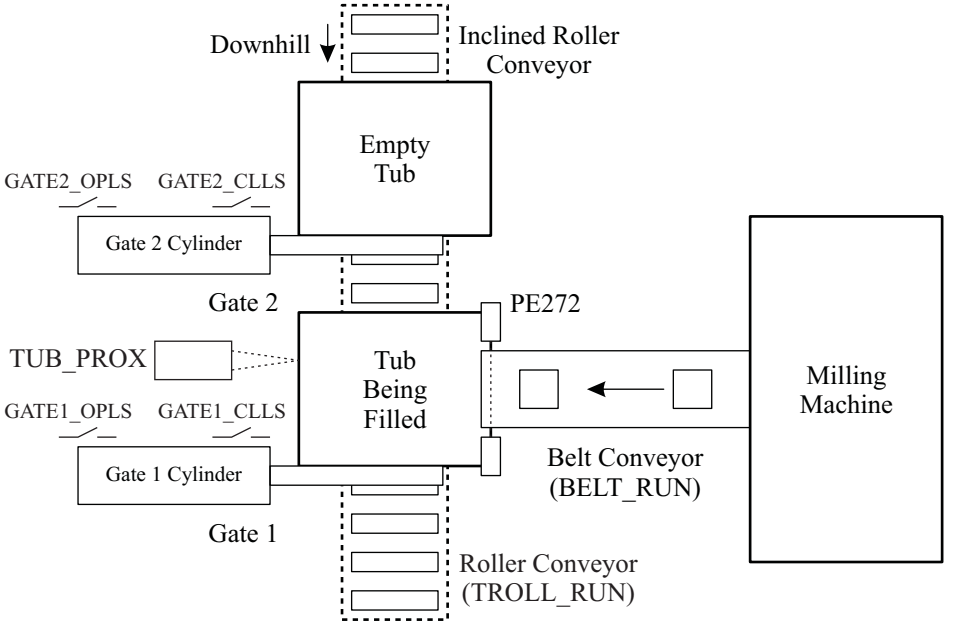

Figure 6.14. Parts tub loading station.

While the tub is being changed, the belt conveyor motor must be stopped (BELT_RUN **off**) and an internal coil, Tub_Permissive, must be turned **off**. After a new tub is in position, BELT_RUN is turned **on**, the Tub_Permissive coil is turned **on**, and the counting of parts is resumed. The Tub_Permissive is used by the milling machine ladder logic. When Tub_Permissive is **on**, the machine produces parts.

The roller conveyor for the tubs has two sections. The section between the two gates and extending out of the station is powered and controlled by the TROLL_RUN output. The roller conveyor section before Gate 1 is unpowered and inclined to allow new tubs to move into the station. In order to completely move the empty tub into the station, the powered section must be running.

Single-action pneumatic cylinders control Gate 1 and Gate 2. Once GATE1_RET is energized, gate 1 opens and remains in the open position as long as power is applied (turned **on**). The gate closes when power is removed (turned **off**). Limit switches GATE1_OPLS and GATE1_CLLS sense the open and closed positions, respectively. Similarly, GATE2_RET controls Gate 2. The GATE2_OPLS and GATE2_CLLS limit switches sense the position of Gate 2.

Single-speed motors drive the two conveyors. When BELT_RUN is **on**, the conveyor moves parts from the milling machine to the tub. When BELT_RUN is **off**, the conveyor is stopped. When TROLL_RUN is **on**, the powered section of the roller conveyor moves. When TROLL_RUN is **off**, the powered section of the roller conveyor is stopped.

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**,

the tub loading operation should be paused at the current step. When paused, do not advance to the next step. When the Run coil turns **on** while the operation is paused, the tub loader should resume the suspended step. When paused, both conveyors must be stopped, all counter and timer accumulator values must be retained, and the ladder logic program must remain in the step in which the Run coil changed from **on** to **off**. If the Run coil turns **off** when changing tubs, the pneumatic cylinder controls must continue to be activated, holding the gate open (otherwise, a tub may be damaged).

There is another internal coil, Reset, that when **on**, restarts the operation. The Reset 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 an empty tub is in position. The Reset internal coil must be ignored while Run is **on**.

Assume the following physical input, physical output, and internal coil assignments:

Tag/Var./Symbol	Description
PE272	Photoelectric sensor, **off** when part passes.
TUB_PROX	Proximity sensor, **on** (closed) when tub is present, though not necessarily in position to receive parts.
GATE1_OPLS	Limit switch, **on** (closed) when Gate 1 is open.
GATE1_CLLS	Limit switch, **on** (closed) when Gate 1 is closed.
GATE2_OPLS	Limit switch, **on** (closed) when Gate 2 is open.
GATE2_CLLS	Limit switch, **on** (closed) when Gate 2 is closed.
BELT_RUN	Belt conveyor control, **on** to run conveyor to move parts from milling machine to the parts tub.
TROLL_RUN	Powered roller conveyor control, **on** to run conveyor to move parts tub.
GATE1_RET	Gate 1 cylinder control, **on** to retract cylinder and open gate; **off** closes gate.
GATE2_RET	Gate 2 cylinder control, **on** to retract cylinder and open gate; **off** closes gate.
Run	Internal coil, **on** when loading enabled to operate (controlled by another part of the ladder logic).
Reset	Internal coil, **on** to reset tub loader operation (controlled by another part of the ladder logic).
Tub_Permissive	Internal coil, **on** when milling machine is permitted to run (controlled by this part of the ladder logic).

The addresses associated with the physical inputs and outputs are:

Tag/Var./Symbol	ControlLogix	MLogix	Siemens	Modicon	Emerson
PE272	Local:1:I.Data.2	I:1/02	%I0.2	%I0.2.2	%I3
TUB_PROX	Local:1:I.Data.3	I:1/03	%I0.3	%I0.2.3	%I4
GATE1_OPLS	Local:1:I.Data.4	I:1/04	%I0.4	%I0.2.4	%I5
GATE1_CLLS	Local:1:I.Data.5	I:1/05	%I0.5	%I0.2.5	%I6
GATE2_OPLS	Local:1:I.Data.6	I:1/06	%I0.6	%I0.2.6	%I7
GATE2_CLLS	Local:1:I.Data.7	I:1/07	%I0.7	%I0.2.7	%I8
BELT_RUN	Local:2:O.Data.0	O:2/00	%Q4.0	%Q0.3.0	%Q1

TROLL_RUN	Local:2:O.Data.1	O:2/01	%Q4.1	%Q0.3.1	%Q2
GATE1_RET	Local:2:O.Data.2	O:2/02	%Q4.2	%Q0.3.2	%Q3
GATE2_RET	Local:2:O.Data.3	O:2/03	%Q4.3	%Q0.3.3	%Q4

The addresses/data types associated with the internal tags/variables/symbols are:

	CLogix	PLC-5	Siemens		Modicon	Emerson
Tag/Variable	Data Type	Addr.	Addr.	Type	Data Type	Addr.
Run	BOOL	B3/100	%M62.0	BOOL	BOOL	%M100
Reset	BOOL	B3/101	%M62.1	BOOL	BOOL	%M101
Tub_Permissive	BOOL	B3/102	%M62.2	BOOL	BOOL	%M102

Solution. This example introduces the following aspects of sequential problems:

Using timers

Using counters

Using Run as part of the transition condition

Reset of the operation

As illustrated in Example 6.1, there are two main steps to develop the function chart:

1. Identify the steps and transition conditions.
2. Add step actions.

To identify the steps and transitions, the first paragraph of the process description is repeated, with the steps identified by the underlined phrases and the transition conditions identified by the *italicized phrases*. As in example 6.1, many times it is easier to identify the first transition condition (signaled by an input sensor change) and then recognize the step before and the step after the transition condition. Often, the steps and transition conditions alternate during the narrative.

Figure 6.14 shows the layout of a parts tub loader machine. Parts are placed on the belt conveyor by a milling machine. The parts move down the conveyor and drop into the parts tub. Parts on the belt conveyor are detected by a photoelectric sensor, PE272, which is **off** as a part interrupts the beam. Assume PE272 detects the part as it falls into the tub. After *100 parts are deposited* in the tub, the tub is moved out and a new, empty tub moves into position. To change the tub, the following operation must take place:

Open Gate 1 (*GATE1_OPLS senses* when open).

Hold Gate 1 open and wait for *TUB_PROX to be **off** for 3 seconds* to allow the full tub to be moved out of the loading station. Run the tub roller conveyor to move out the full tub.

Gate 1 is closed (*GATE1_CLLS senses* when closed).

Gate 2 is opened (*GATE2_OPLS senses* when open).

Hold Gate 2 open to allow an empty tub to move down a slight incline into the loading station. When the tub contacts the tub roller conveyor, the tub roller conveyor moves the tub into position. When *TUB_PROX is **on** for 5 seconds*, the tub is in position (front resting on Gate 1).

Gate 2 is closed (*GATE2_CLLS senses* when closed).

Since the timer accumulator values must be retained when paused, retentive timers must be used for the time delays. Also, the Run coil must be one of the conditions that controls the timer.

The sentence, "When paused, do not advance to the next step." normally means that the internal Run coil is part of the transition condition. However, since retentive timers are used for the transition out of the steps holding the gates open, the Run coil is not needed for these steps. One could argue that the Run coil is not needed for the transitions out of the other steps since the conveyors are stopped when paused, but for the purposes of the example, the Run coil is used.

So, the steps and the transition conditions that indicate the end of each step are:

Step	Transition Condition (out of step)
Parts into tub	Part_Ctr done (100 parts detected) and Run
Open Gate 1	GATE1_OPLS on and Run
Hold Gate 1 open	G1_Hold_Tmr done (TUB_PROX off for 3 sec.)
Close Gate 1	GATE1_CLLS on and Run
Open Gate 2	GATE2_OPLS on and Run
Hold Gate 2 open	G2_Hold_Tmr done (TUB_PROX on for 5 sec.)
Close Gate 2	GATE2_CLLS on and Run

The next part of the function chart development is to add the actions to each step. Reading back through the tub loader narrative, the process actions for each step are:

Step	Actions
Parts into tub	BELT_RUN and Tub_Permissive and Part_Ctr (counts 100 parts with /PE272)
Open Gate 1	GATE1_RET
Hold Gate 1 open	GATE1_RET and TROLL_RUN and G1_Hold_Tmr (3 sec.)
Close Gate 1	
Open Gate 2	GATE2_RET
Hold Gate 2 open	GATE2_RET and TROLL_RUN and G2_Hold_Tmr (5 sec.)
Close Gate 2	

The function chart for the tub loader is shown in Figure 6.15. This particular operation repeats, indicated by a line from the last step back to the first step. Before developing the ladder logic code, the internal tags/variables/symbols should be identified:

The addresses or data types associated with the tags/variables/symbols:

	CLogix	MLogix	Siemens		Modicon	Emerson
Tag/Var./Symbol	Data Type	Addr.	Addr.	Type	Data Type	Addr.
Step_1 to	BOOL	B20/1	%M50.1	BOOL	EBOOL	%M51
Step_7	BOOL	B20/7	%M50.7	BOOL	EBOOL	%M57
Ctr_Done	n/a	n/a	n/a	n/a	n/a	%M59
Part_Ctr	COUNTER	C5:1	%DB2	C*	CTU	%R101
G1_Hold_Tmr	TIMER	T4:1	%DB3	T*	TON	%R104
G2_Hold_Tmr	TIMER	T4:2	%DB4	T*	TON	%R107
G1Tic_Tmr	n/a	n/a	n/a		TON	n/a
G2Tic_Tmr	n/a	n/a	n/a		TON	n/a

The ladder logic code is broken into the following sections:

Start/stop/pause of overall operation

First start

Transitions between steps

Step actions

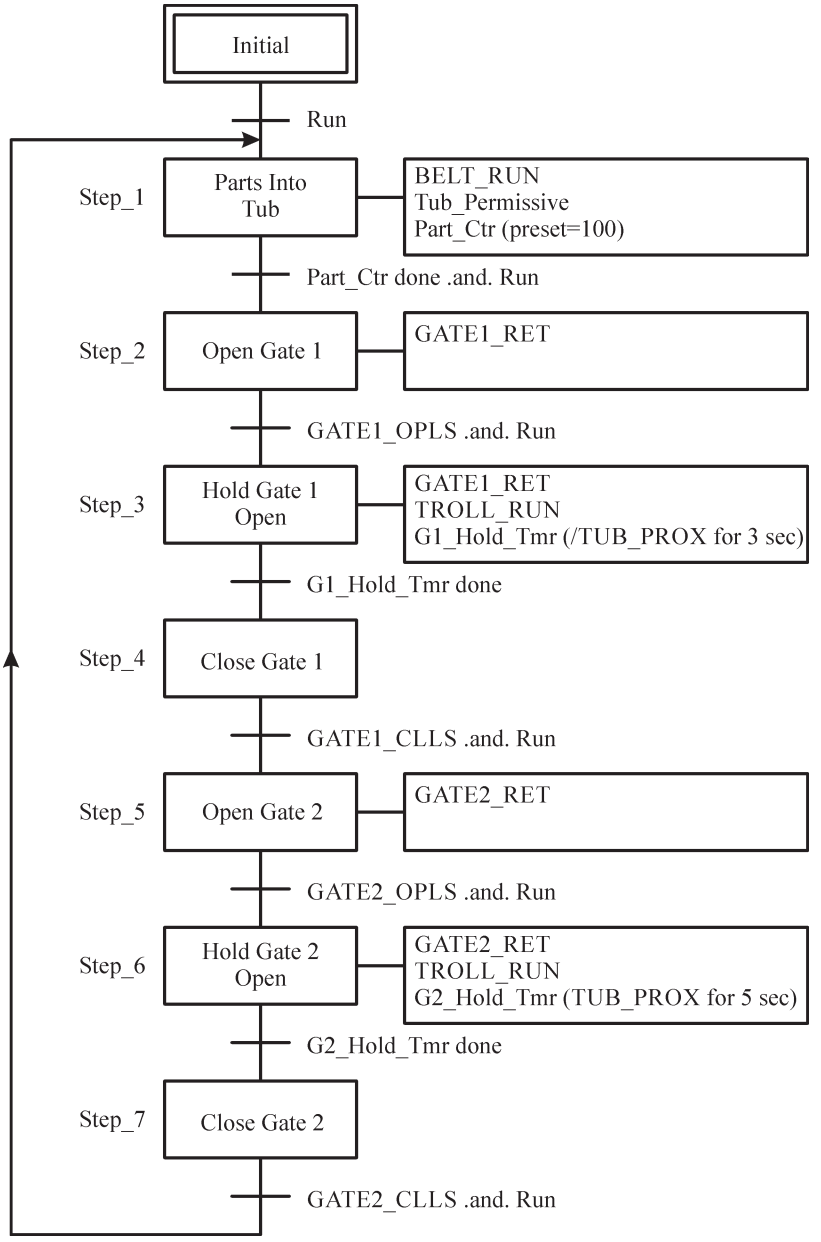

Figure 6.15. Function chart for parts tub loader.

Since the timers and counters are shown as actions, they may be placed with the rungs that drive the physical outputs. However, since they are also part of the transitions, they may also be placed with the rungs handling the transitions. The author favors the latter approach since the transition condition is more likely to be changed.

The ControlLogix ladder logic for the tub loader, shown in Figure 6.16, is developed using the code templates shown earlier in this chapter (using latch/unlatch coils instead of set/reset coils). A rung comment is shown within a rectangle above the rung. The function of each rung is as follows:

1. Transition from initial step to step 1 (starting operation for the first time)
2. Transition from step 1 to step 2 and counting parts
3. Transition from step 2 to step 3
4. Transition from step 3 to step 4 and delay tub prox. off
5. Transition from step 4 to step 5
6. Transition from step 5 to step 6
7. Transition from step 6 to step 7 and delay tub prox. on
8. Transition from step 7 to step 1
9. Reset of steps
10. Control of belt conveyor (an action for step 1)
11. Control of roller conveyor (an action for steps 3 and 6)
12. Control of gate 1 cylinder (an action for steps 2 and 3)
13. Control of gate 2 cylinder (an action for steps 5 and 6)
14. Control of Tub_Permissive (an action for step 1)

Logic is connected to the right of the RTO timer blocks in rungs 4 and 7 to accomplish the transition to the next step, but a contact for the .DN Boolean must still be present, as the output of a timer block is the .EN Boolean, not the .DN Boolean. The Run internal coil is part of the input condition for each retentive timer, thus pausing the timer when the station operation is paused. The transition logic for a step can be connected to the right of a CTU block. However, this approach is not good practice and does not work for Step_1 in this application due to the Run contact that is also a part of the transition condition. If the logic that is parallel to the CTU in rung 2 is placed in series with the CU output of the CTU, and the operation is paused when Part_Ctr.DN is on, PE272 must remain off while the operation is paused. In general, this cannot be guaranteed, as the belt conveyor's inertia may carry the part past PE272 while it is stopping, and PE272 turn back on, blocking the transition when the operation is resumed.

The reset condition for the counter and the retentive timers must also be defined. Two situations must be considered: normal operation and operator-initiated reset. Both of these situations are handled if the counter/timer is reset when the operation is not in the step where the counter/timer is used. For example, the counter used to count parts in step 1 is reset when the operation is not in step 1, the step where parts are counted (Figure 6.16, rung 2). The retentive timers are reset when the operation is not in the step using the timer (Figure 6.16, rungs 4 and 7).

When the reset push button is pressed while the station is paused, all step-in-progress coils are reset. This action effectively places the station operation in the initial state.

The MicroLogix/SLC-500 ladder logic is nearly identical to the ControlLogix ladder logic in Figure 6.16. The basic differences are:

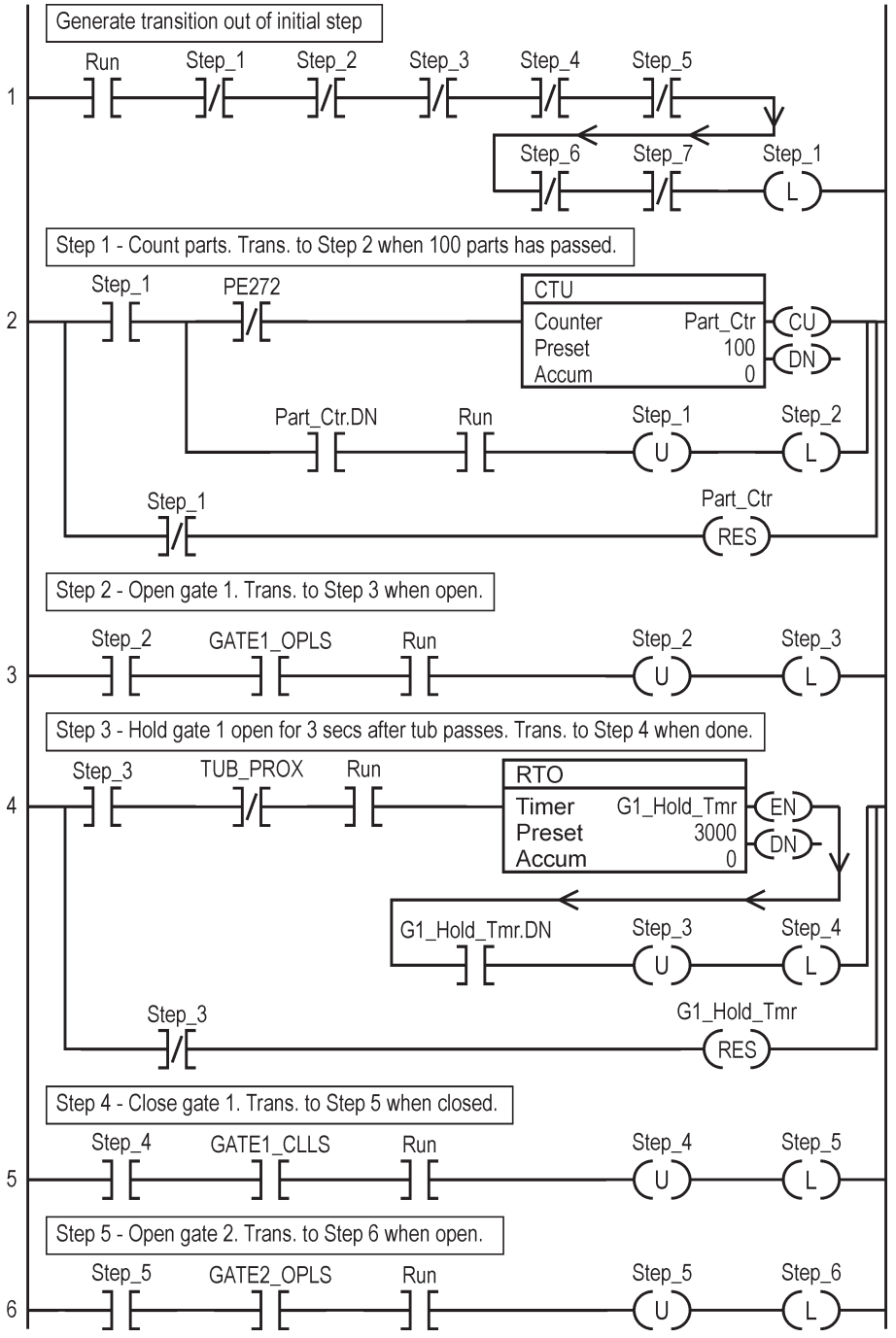

Figure 6.16. ControlLogix ladder logic for tub loader. *(continued)*

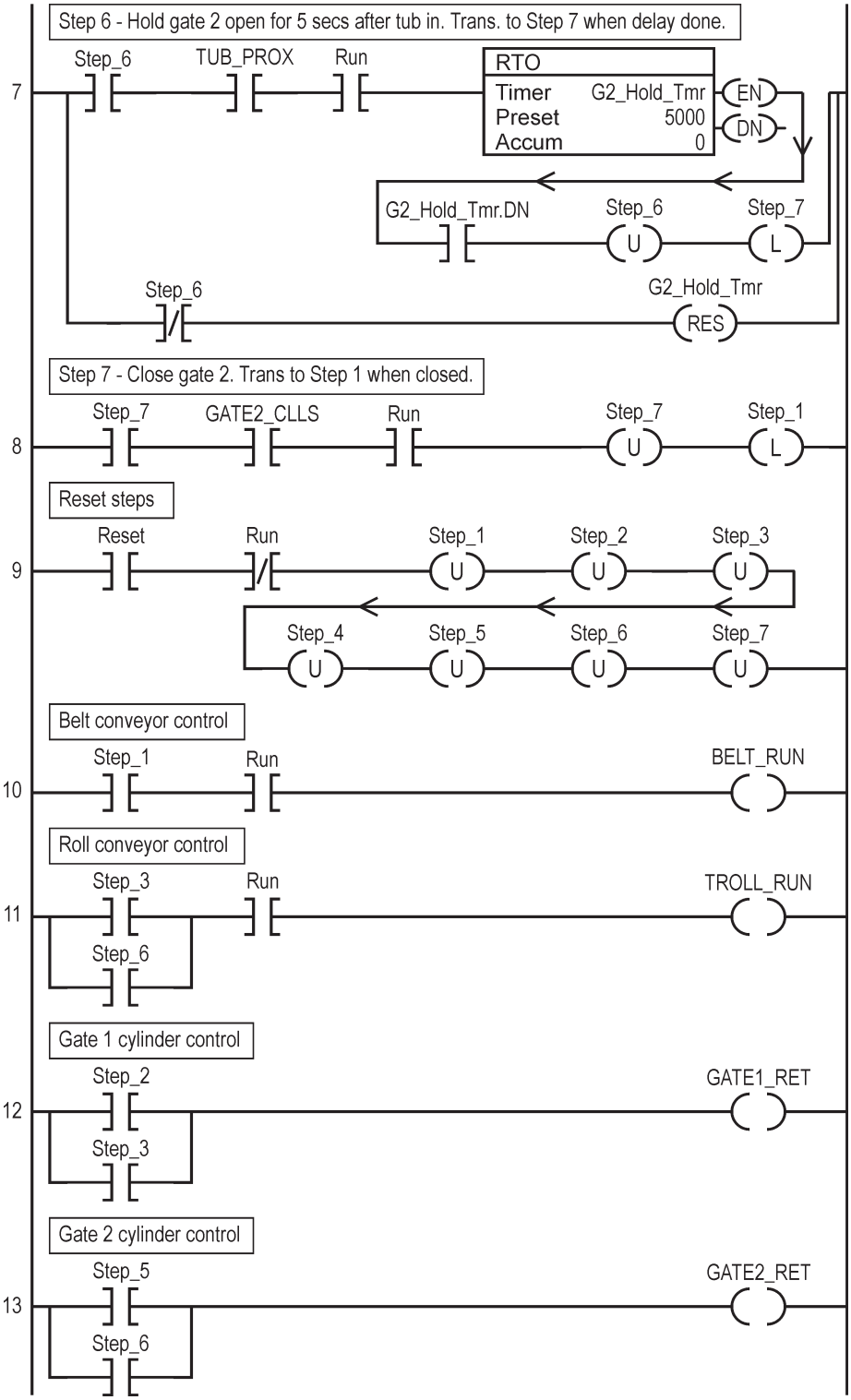

Figure 6.16. *(continued)*

Figure 6.16. *(continued)*

1. Logic cannot be placed to the right of the counter and timer blocks and so the transition logic is placed in parallel with the counter or timer.
2. The "Part_Ctr" symbol appears above the CTU block in rung 2, and the address "C5:1" appears in the Counter field.
3. For the timers (rungs 4 and 7), a "Time Base" field is present, set to 0.01 and the Preset value is divided by 10. Also, the timer address appears in the Timer field of the RTO block, and the timer symbol is above the MicroLogix/SLC-500 RTO block.
4. Multiple coils on the right side of a rung are placed in parallel instead of in series.

The Siemens S7-1200/1500 ladder logic code is shown in Figure 6.17. If implemented for the S7-300/400, the TONR retentive timer is replaced by the TON/CTU combination decribed in section 5.5.3 and multiple coils on the right side of a network are placed in parallel rather than in series.

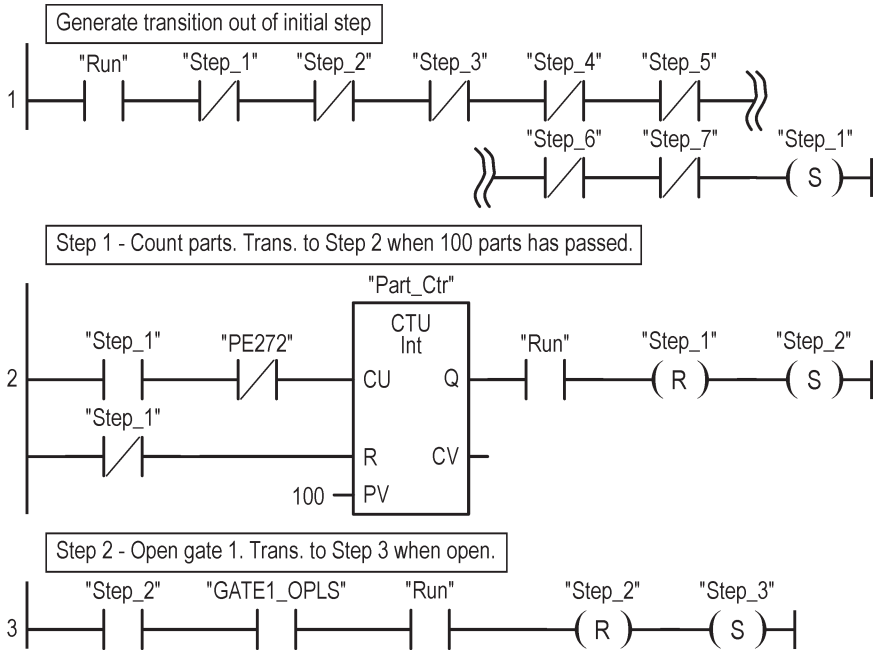

Figure 6.17. S7-1200/1500 ladder logic for tub loader. *(continued)*

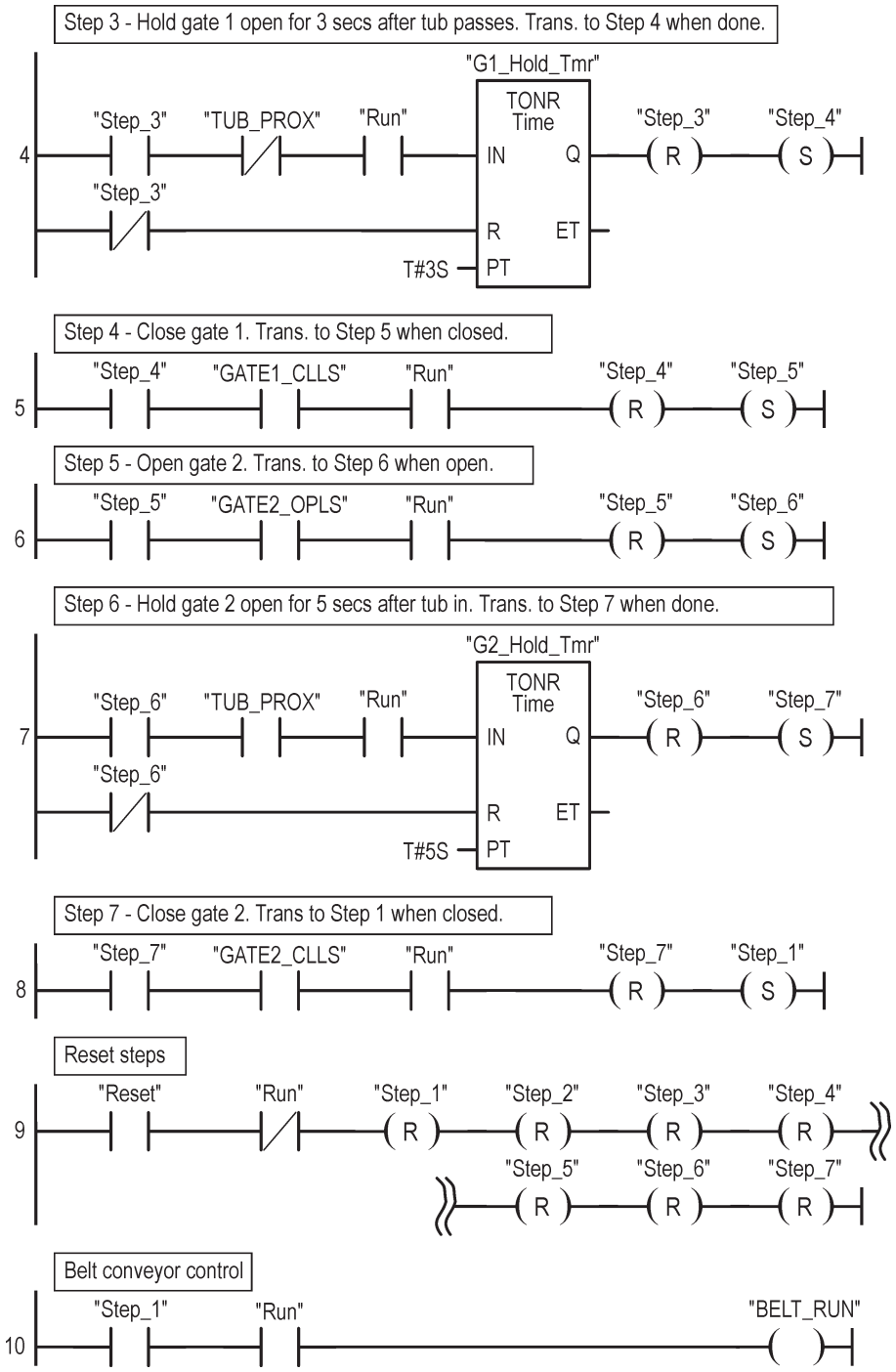

Figure 6.17. *(continued)*

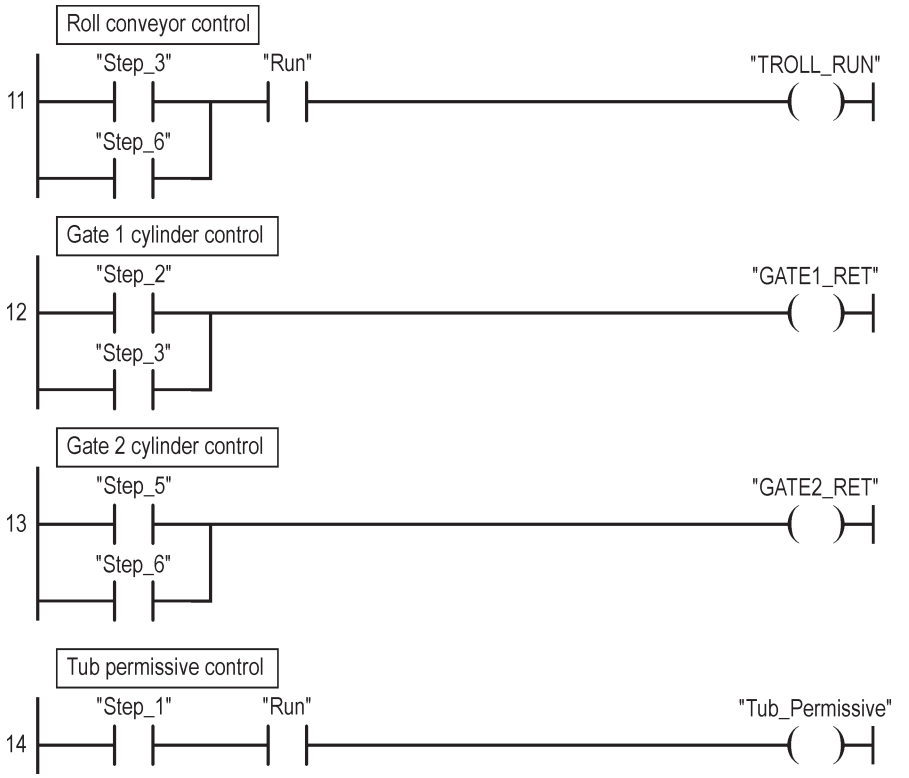

Figure 6.17. *(continued)*

The Modicon ladder logic is shown in Figure 6.18. Since Modicon does not define a retentive on-delay timer, one must be constructed as outlined in section 5.6.3. On rungs 4 and 7 a non-retentive TON timer generates a "tick" every 0.1 seconds which is counted. The counter provides the retentive function. The Run internal coil is part of the input condition for each retentive timer, thus pausing the timer when the station operation is paused.

The Emerson ladder logic is shown in Figure 6.19 and is similar to the S7 ladder logic except that coils cannot be placed in series. This implementation is for all processors. For 90-30 processors, the output of the counter in rung 2 cannot connect to a contact, and so an

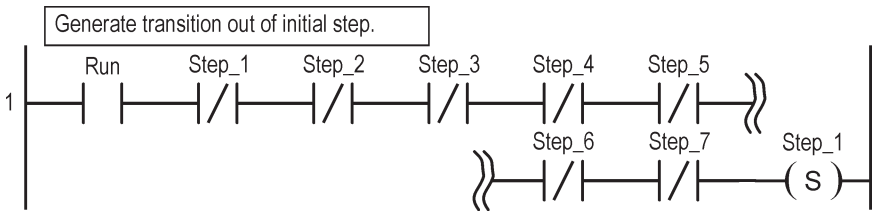

Figure 6.18. Modicon ladder logic for tub loader. *(continued)*

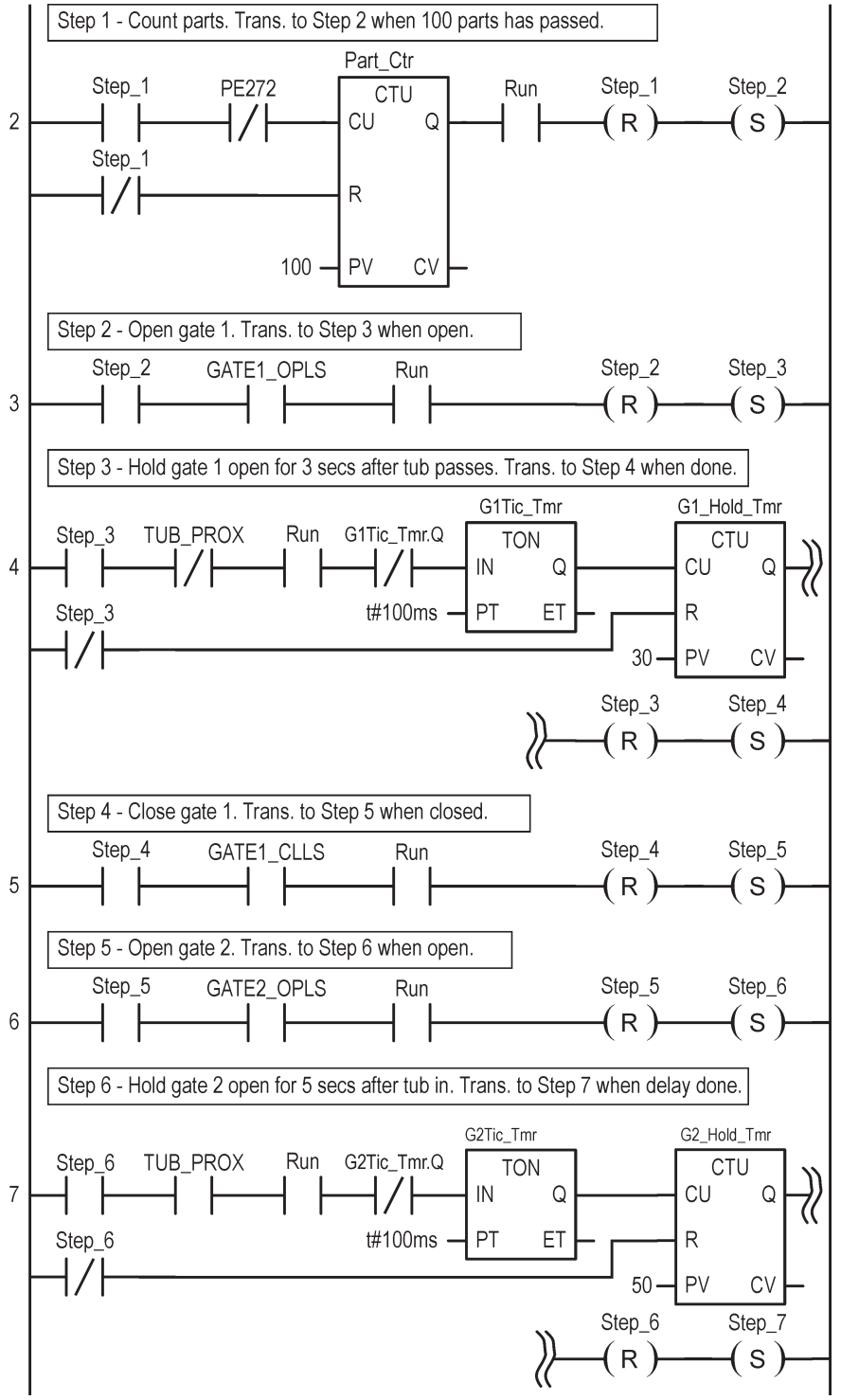

Figure 6.18. *(continued)*

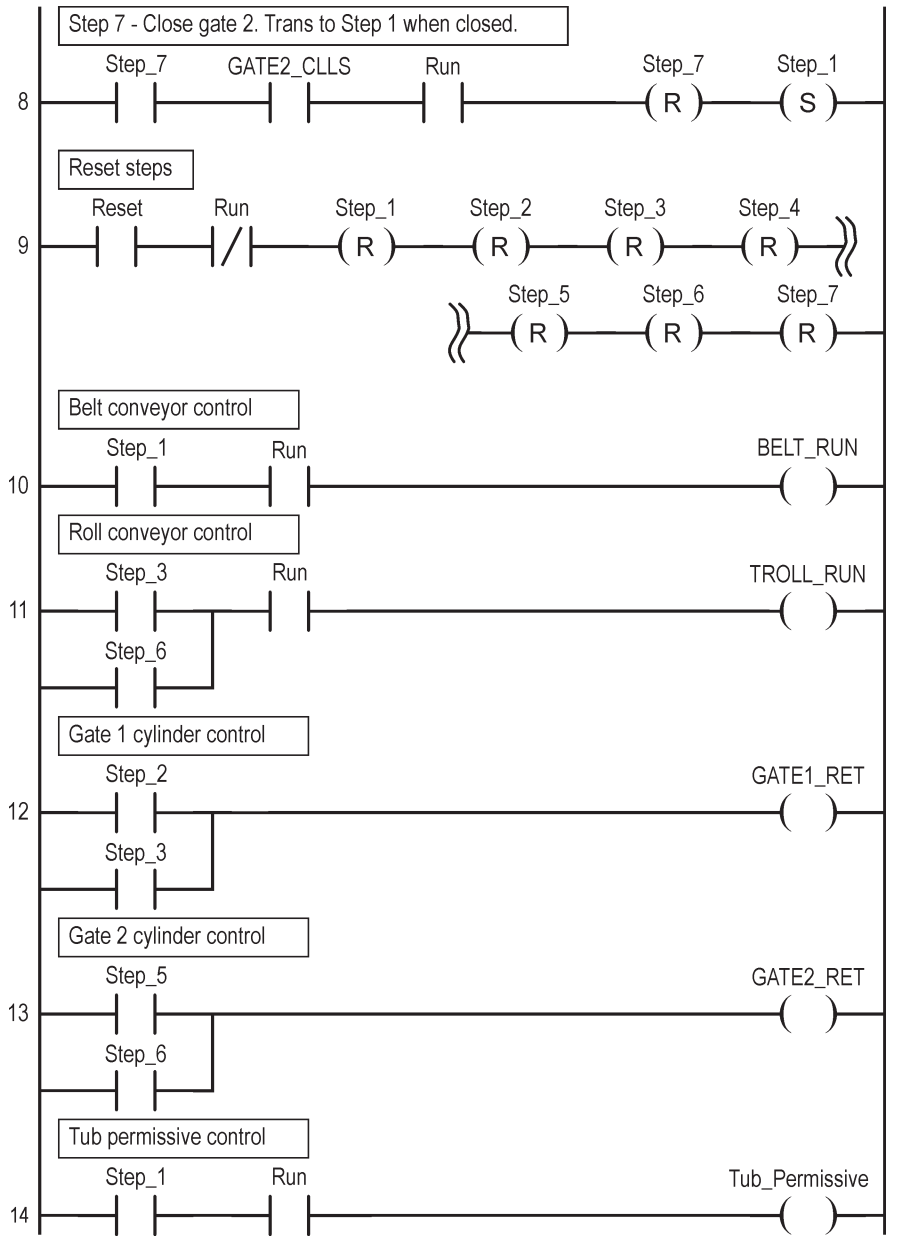

Figure 6.18. *(continued)*

extra internal coil and rung is added to accommodate the specification that the operation cannot advance to the next step when paused. Rungs 2 and 3 can be combined on other Emerson processors.

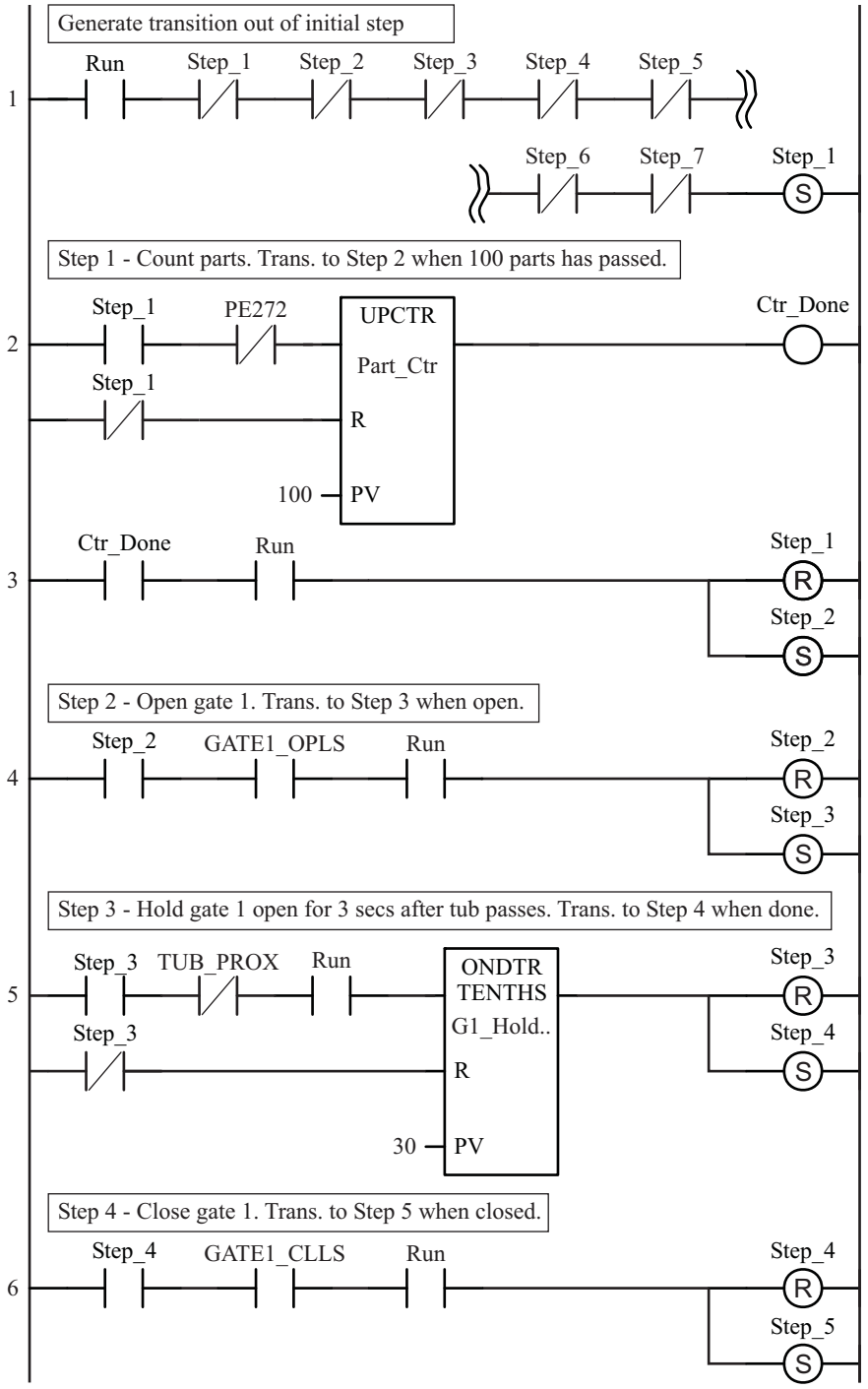

Figure 6.19. Emerson ladder logic for tub loader. *(continued)*

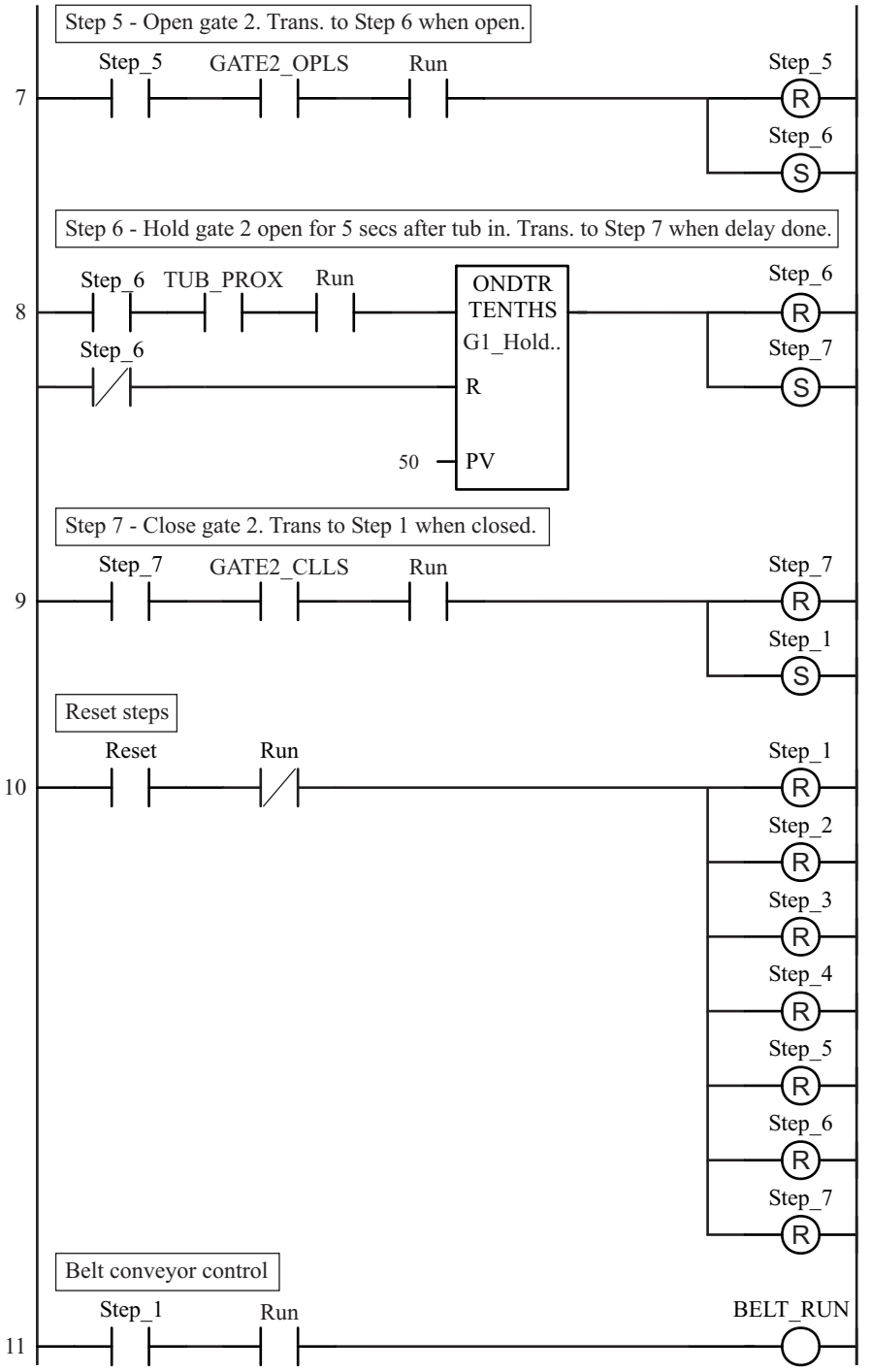

Figure 6.19. *(continued)*

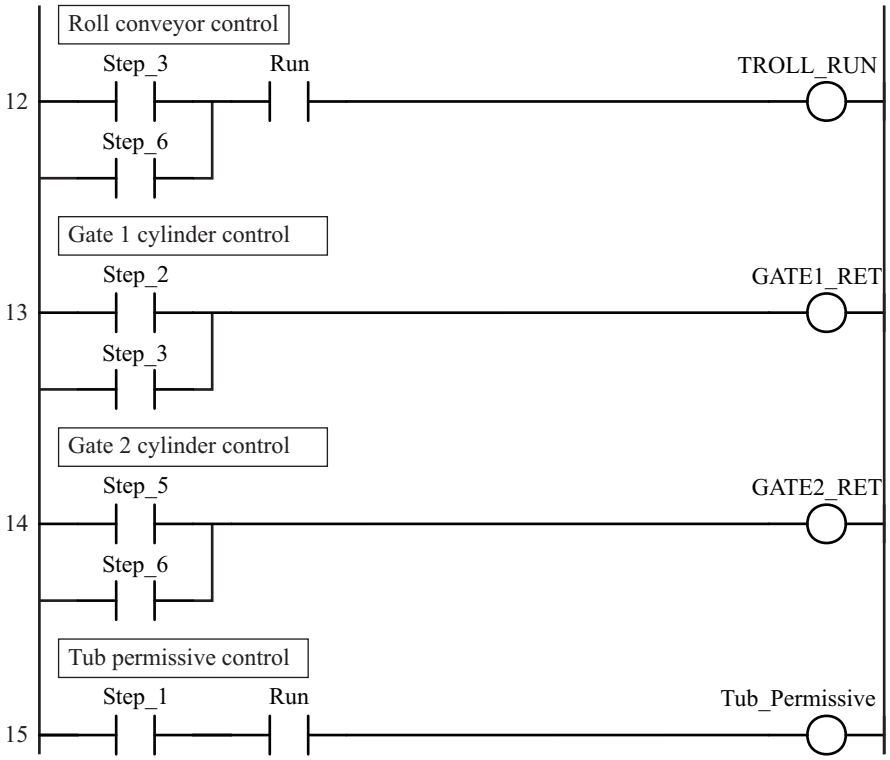

Figure 6.19. *(continued)*

6.4 COMPLICATED RESET OPERATION

Example 6.3 has most of the features of a real application. The next example illustrates a problem in which the reset operation is more complicated than in the previous example.

Example 6.4. Engine Inverter Station Control. Design the function chart of the program to control the following station that inverts (turns over) gasoline engine assemblies and implement the control with ladder logic.

Figure 6.20 shows the layout of a station that inverts 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. Implement ladder logic for this station only. The conveyor is controlled by another PLC, 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, the station contains two capturing mechanisms (engaging hooks) that control access to the station and allow pallets to queue up before the station.

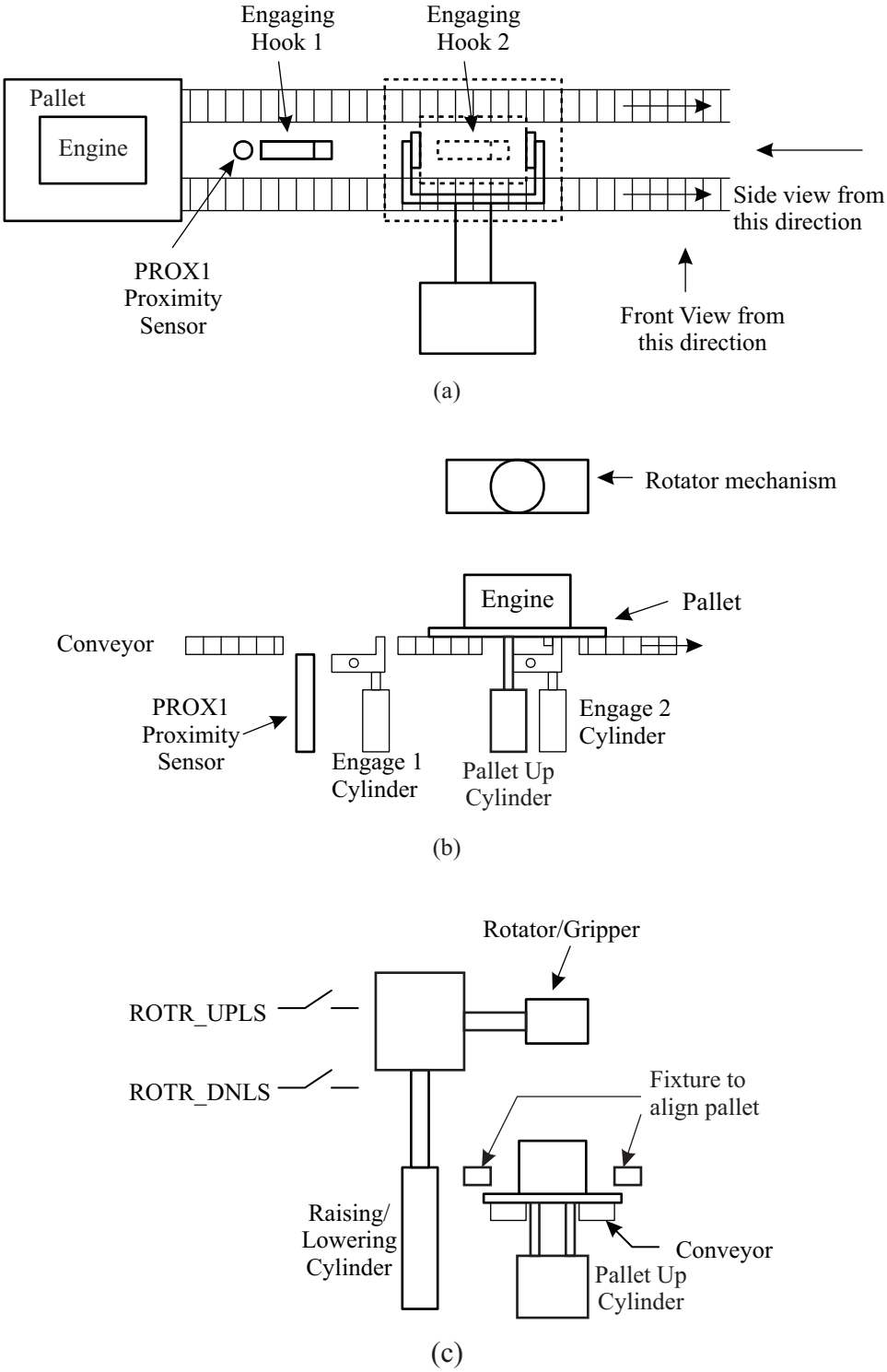

Figure 6.20. Engine inverter station: *(a)* top view; *(b)* front view; *(c)* side view.

Upon initial startup, assume that there are no pallets waiting at engaging hook Engage 1. When a pallet is detected at Engage 1 (by PROX1), the following major steps are executed:

Lower the Engage 1 hook (by activating ENG1_RET) for 2 seconds to allow only one assembly to move into the station and be caught by the Engage 2 hook. When the Engage 1 hook is raised, it catches the next pallet.

Raise the pallet off the conveyor.

Lower the rotator mechanism to the correct position (ROTR_DNLS closed).

Clamp the engine.

Raise the rotator mechanism (until ROTR_UPLS closes).

Rotate the engine one-half turn clockwise (until ROTR_CWLS closes).

Lower the rotating mechanism to the correct position (ROTR_DNLS closed).

Unclamp the engine.

Raise the rotator mechanism (until ROTR_UPLS closes).

Rotate the clamp one-half turn counterclockwise (until ROTR_CCWLS closes).

Activate the ENG2_RET for 3 seconds to allow pallet to move out.

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

The proximity sensor, PROX1, is inductive and senses the metal assembly pallet. PROX1 senses the pallet before the pallet reaches the engage position. You must assume that when Engage 1, the first engaging hook, captures the pallet, PROX1 remains **on**.

ENG1_RET and ENG2_RET are controls for single action pneumatic cylinders that move the engaging hooks. Once ENG1_RET is energized, the Engage 1 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 in this manner to be fail-safe, that is, if electrical power or air pressure is interrupted because of a failure, no pallets proceed down the conveyor. ENG2_RET controls the second engaging hook, Engage 2, in a similar manner.

The pallet-raising mechanism is driven by a single-action pneumatic cylinder controlled by PALL_UPCTL. Once the PALL_UPCTL output is energized, the clamp moves the pallet (and engine) off the conveyor and into a fixture to properly align the engine to the gripper clamp. PALL_UPCTL must remain **on** to hold the assembly in the fixture. If PALL_UPCTL is turned **off**, the pallet falls back onto the conveyor. The PALL_UPLS is **on** when the pallet is in the proper position.

The mechanism used to lower and raise the rotating mechanism consists of a double-action linear hydraulic cylinder. When the ROTR_DOWN output is energized (turned **on**), the rotator moves down and continues to move down as long as it is energized and a mechanical stop is not reached. When the ROTR_UP