

# **ControlLogix Rev 31+ Code Design Guidelines**

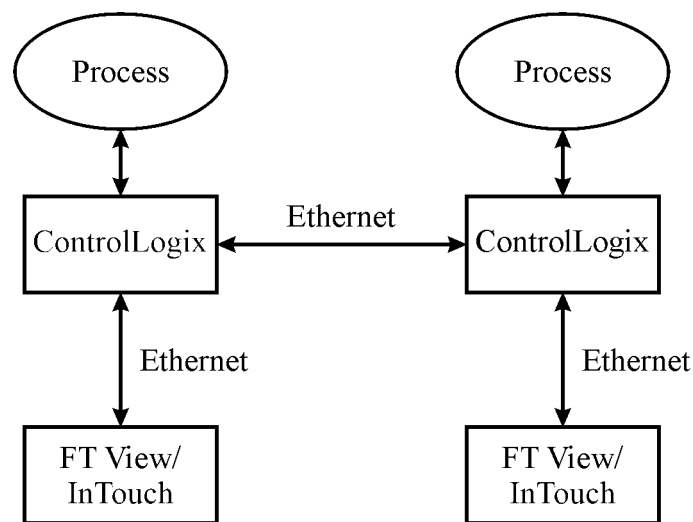
## Table of Contents

<b>1. CONTROL SYSTEM FUNCTIONAL LAYOUT .....</b>	<b>3</b>
<b>2. AREA CONTROL AND SECURITY .....</b>	<b>3</b>
<b>3. PROGRAM STRUCTURE .....</b>	<b>4</b>
<b>4. PLC/PLC AND OI/PLC COMMUNICATIONS .....</b>	<b>6</b>
<b>5. FIELD EQUIPMENT CONTROL .....</b>	<b>9</b>
5.1 STANDARD MOTOR DEVICE CONTROL .....	9
5.2 CONVEYOR MOTOR DEVICE CONTROL .....	13
5.3 DISCRETE VALVE DEVICE CONTROL .....	17
5.4 SLIDE GATE CONTROL .....	21
5.5 FLOP GATE CONTROL .....	26
<b>6. SEQUENCED CONTROL .....</b>	<b>31</b>
6.1 SEQUENCE CODE .....	31
6.2 BRANCHES .....	38
6.3 TRANSFERS WITH MULTIPLE SOURCES/DESTINATIONS .....	40
6.4 OPERATOR RESPONSE .....	44
6.5 COORDINATION WITH ANOTHER UNIT .....	45
6.6 DISPLAY REMAINING TIME FOR TIMED STEPS .....	46
<b>7. CONTINUOUS CONTROL .....</b>	<b>48</b>
<b>8. ANALOG INPUTS AND OUTPUTS .....</b>	<b>52</b>
<b>9. ALARMS .....</b>	<b>52</b>
<b>10. TAG CONVENTIONS .....</b>	<b>55</b>
<b>11. SIMULATION LOGIC .....</b>	<b>58</b>
<b>12. REVISION HISTORY .....</b>	<b>58</b>

## 1. Control System Functional Layout

The control system is centered on Allen-Bradley ControlLogix/CompactLogix processors which provide the discrete device control, loop device control, sequence control, and alarm evaluation. Rockwell FactoryTalk View or Wonderware Intouch provides the operator interface (OI). The ControlLogix ladder logic is described here. The operator interface standards are described in a separate document.

Figure 1 details a typical system's communications layout. Communication between ControlLogix/CompactLogix processors uses Ethernet. The OI communicates to the ControlLogix/CompactLogix processors over Ethernet.



**Figure 1.** Overall view of control system.

## 2. Area Control and Security

The OI operator handles both control of the sequential operations and the individual pieces of equipment. In most systems, there are three levels of privilege a particular operator has for an area (unit in this project):

- View – operator can view critical information. The operator cannot operate individual pieces of equipment or initiate unit sequencing with this privilege.
- Maintenance – operator can view critical information and operate individual pieces of equipment. Sequencing cannot be initiated with this privilege level.

- Operate – operator can view critical information, affect analog loop control setpoints and variable speed drive outputs, and initiate area sequencing in this privilege.

There is also a local/remote mode which defines the location of the operator that initiates the sequences. In local mode, the operator is assumed to be close to the unit, at a local touch panel (LTP). In remote mode, the operator is at a distant control room, or possibly a supervisory controller that coordinates the operation of multiple units. Maintenance privilege can only be granted to a local operator.

For this project, only two levels of privilege will be considered: maintenance and operate. The operator screen for a unit will also have a local/remote button that can toggle between these two modes. The maintenance privilege can only be granted in the local control mode if the unit is in Hold, Shutdown, or E-Shutdown.

### 3. Program Structure

The ControlLogix program is organized into programs. Each unit in the controller has its own program, even when the processor controls multiple units. The MainProgram handles communications with the other units. The Simulation program contains the simulation logic. In the unit program (the name of the program is the shortened name of the unit), "MainRoutine" consists of JSR's to the other routines in the program. Device control is grouped routines according the type of device (motor, valve, slide gate, or flop gate). All of the motor device control is combined into the "Motors" routine. All of the valve device control is combined into the "Valves" routine. All of the pneumatic gate device control is combined into the "Gates\_Flop" routine. Slide gates that are controlled by motors are in the "Gates\_Slide" routine. The code for each sequence diagram has its own routine. The maintenance/operate privilege control and the abnormal conditions are each handled in a separate routine. The structure of the main task is:

```

MainTask
  MainProgram
    Parameters and Local Tags
    Communications (the main routine)
  Simulation (program)
    Parameters and Local Tags
    Simulation (the main routine)
  Unit1 (program)
    Parameters and Local Tags
    MainRoutine
    Gates
    Motors
    PID_Loops
    Unit1_000Startup    [Note: starting Unit1.Msg number part of name]
    Unit1_040Operate
    Unit1_060Hold
    Unit1_080Shutdown
    Unit1_100EShutdown
    Unit1_Abnormal

```

## Unit1\_Misc Valves

Most tags, especially those associated with a unit should be defined as controller tags. If not, they may not be accessible to the OI. Timers associated with the communications program and simulation program should be defined as program tags, as they will not be accessed by an external device (another controller or OI).

The following add-on instructions handle device control:

Motor_Std	Standard motor control
Motor_Conv	Conveyor motor control (motor + speed switch)
Valve_Disc	On/off valve
Gate_Slide	Slide gate controlled by reversing motor
Gate_Flop	Gate controlled by pneumatic cylinder

In addition, the following user-defined types (UDT's) are defined:

Unit_Type	Unit-related operator control and indication
Motor_Std_Type	Motor sequence control and operator indications
Motor_Conv_Type	Conveyor sequence control and operator indications
Valve_Disc_Type	Discrete valve sequence control and operator indications
Gate_Slide_Type	Slide gate sequence control and operator indications
Gate_Flop_Type	Flop gate sequence control and operator indications
Seq_Type	Sequence data

The "unit name" tag is of Unit\_Type whose fields are as follows:

<b><u>Name</u></b>	<b><u>Data Type</u></b>	<b><u>Description</u></b>
Alm_Reset	BOOL	Alarm reset
Local	BOOL	Local/remote control indication
Maint	BOOL	Maintenance privilege for device control
Man_StartOpen	BOOL	Manual start/open from operator
Man_StopClose	BOOL	Manual stop/close from operator
Man_DevNum	DINT	Number of device started/stopped at OI
Msg	DINT	Message number for OI
Time_Remaining	DINT	Remaining time (sec) in timed steps

The information contained in the device-related types is described in section 5 with the device control. The information contained in the sequence-related types is described in section 6.

## 4. PLC/PLC and OI/PLC Communications

Exchanging information between units can be done in two ways: (1) MSG blocks, or (2) Producer/Consumer. While the Producer/Consumer method of exchanging data is automatic (no MSG blocks are required), any time a processor is tested, the other processors that are producers for data must be implemented. With MSG blocks, the processor that produces the data does not need to be implemented. The MSG block that reads the data from the other processor executes with an error (does not find the source) and thus does not write over the local copy of the data. This data can be manipulated by the tester to simulate the presence of the other processor.

In those cases where information is exchanged between a ControlLogix processor and a PLC-5/SLC-500 processor, the ControlLogix processor does both read and write MSG blocks since many of the PLC-5/SLC-500 processors cannot communicate with a ControlLogix processor.

If MSG blocks are used to exchange information, reads are used because multiple PLCs can read the same information from a particular PLC, whereas only one can write. Also, it is difficult to troubleshoot writes to a processor. The information to be exchanged is described in a separate document.

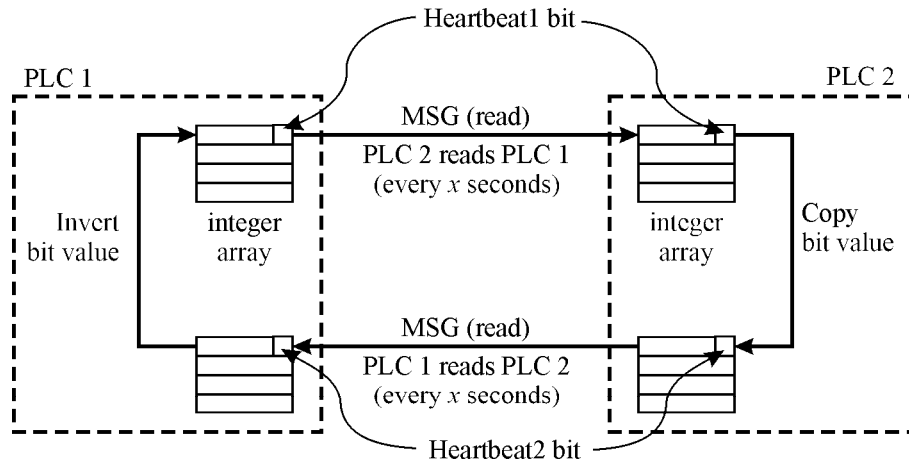
Communication between the PLCs and between the PLC and OI are often vital, hence, programming, which is called the Heartbeat logic, is used to verify the communications bridge. The Heartbeat programming is located in both the PLCs and the OI. If either the PLC or PC fails to set their given bit within the 10 seconds, an alarm will occur and be indicated at the PLC and the OI panels. When communications cease for more than 10 seconds, certain unit operations may be initiated.

The general structure of PLC-to-PLC heartbeat logic with MSG communication is shown in Figure 2a. Both PLCs are assumed to be periodically reading an integer block of data from the other PLC. In each integer block, a bit is identified as the “heartbeat” bit. Heartbeat1 is a bit in the integer array in PLC 1 and read by PLC 2 and Heartbeat2 is a bit in the integer array in PLC 2 and read by PLC 1. PLC 2 copies Heartbeat1 to Heartbeat2 and PLC 1 copies the invert of Heartbeat2 to Heartbeat1. As a result of these operations, both Heartbeat1 and Heartbeat2 oscillate between **on** and **off**. The on and off period is somewhere between one to two times the MSG read period. For example, if each MSG in Figure 2a is executed every 1 second, the on and off period of Heartbeat1 and Heartbeat2 is between 1 and 2 seconds, depending on the synchronization between the timers in the two PLCs. If Heartbeat1 (or Heartbeat2) is **on** for 10 seconds or **off** for 10 seconds, communication has ceased and a “heartbeat” alarm should be generated. If either PLC is disconnected from the network the heartbeat alarm for both PLCs should turn on. Also, if either PLC is switched to program mode, the heartbeat alarm on the other PLC should turn on. When both PLC's are restored to normal operation (connected to network and in run mode) both heartbeat alarms should be off.

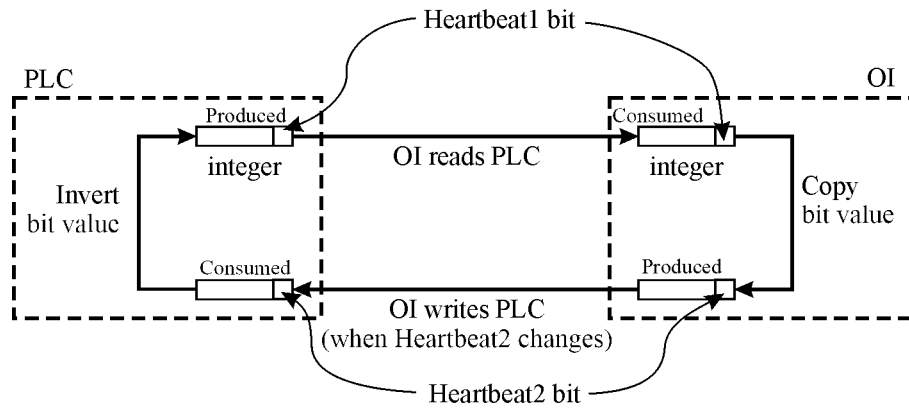
The general structure of PLC-to-PLC heartbeat logic with producer/consumer communication is shown in Figure 2b and is similar to a PLC-to-PLC heartbeat. Each PLC has a produced integer that is consumed by the other PLC. Heartbeat1 is one bit of an integer produced by PLC 1 and

consumed by PLC 2. Heartbeat2 is one bit of an integer produced by PLC 2 and consumed by PLC 1.

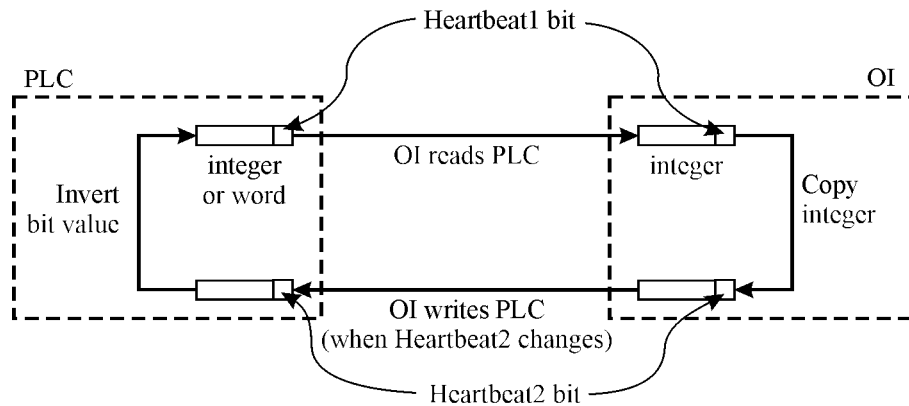
The general structure of PLC-to-OI heartbeat logic is shown in Figure 2c and is similar to a PLC-to-PLC heartbeat. Since it generally not possible for a PLC to read/write to the OI, the OI controls the communication. A script in the OI periodically runs to copy the integer read from the PLC to another integer that is written to the PLC. Depending on the OI, it may not be possible for the OI to do this operation.



(a)



(b)



(c)

**Figure 2.** Heartbeat concept: (a) PLC-to-PLC with MSG; (b) PLC-to-PLC with producer/consumer; (c) PLC-to-OI.



## 5. Field Equipment Control

The PLC views equipment control with one of two privileges: maintenance and operate. Maintenance privilege means no sequence currently has control of the unit containing that device and the unit is in local control. Otherwise, the equipment is being controlled with the operate privilege. The operate privilege calls for PLC interlocking, maintenance privilege does not. The maintenance privilege is usually granted for a grouping of equipment.

### 5.1 Standard Motor Device Control

A motor device tag (associated with a pump) is shown in Figure 3. Example 21.1 in the text contains more information about this device. The physical connections to the PLC are:

Physical Inputs:

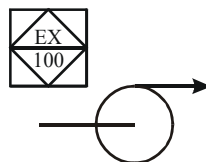
EX100\_Aux Auxiliary contact (**on** when motor running)  
 EX100\_OL Overload trip (**on** when motor overloaded)  
 EX100\_HOA HOA switch auto contact (**on** when auto contact is closed)

Physical Outputs:

EX100\_Start Starter (**on** to start/run motor)

The device tag (for example, “EX100”) is of Motor\_Std\_Type whose fields are as follows:

<u>Name</u>	<u>Data Type</u>	<u>Description</u>
Run_Status	BOOL	Run status
Any_Fail	BOOL	Failure alarm
Aux_Fail	BOOL	Auxiliary fail alarm
OL_Fail	BOOL	Overload alarm
HOA_Fail	BOOL	HOA-switch-not-in-auto alarm
Seq_Start	BOOL	Device start command from sequence
Seq_Stop	BOOL	Device stop command from sequence



**Figure 3.** P&ID symbol for motor device control associated with pump.

The operator commands and indications are:

Operator Commands:

Unit.Man\_StartOpen Manual Start for unit  
 Unit.Man\_StopClose Manual Stop for unit  
 Unit.Man\_DevNum Number of device started/stopped at OI.  
 Unit.Maint Maintenance/Operate control privilege  
 Unit.Alm\_Reset Reset alarm

Operator Indications:

EX100.Run_Status	Run status
EX100.Any_Fail	Failure alarm
EX100.Aux_Fail	Auxiliary fail alarm
EX100.OL_Fail	Overload alarm
EX100.HOA_Fail	HOA-switch-not-in-auto alarm

The device is started/stopped by an automatic sequence with:

EX100.Seq_Start	Start
EX100.Seq_Stop	Stop

The motor control uses the Motor\_Std add-on instruction as shown in Figure 4. The ladder logic that implements this function block is shown in Figure 5. The use of this block is shown in Figure 4 to control a motor with tag EX100. The fields of the Motor\_Std block are specified as the appropriate variables.

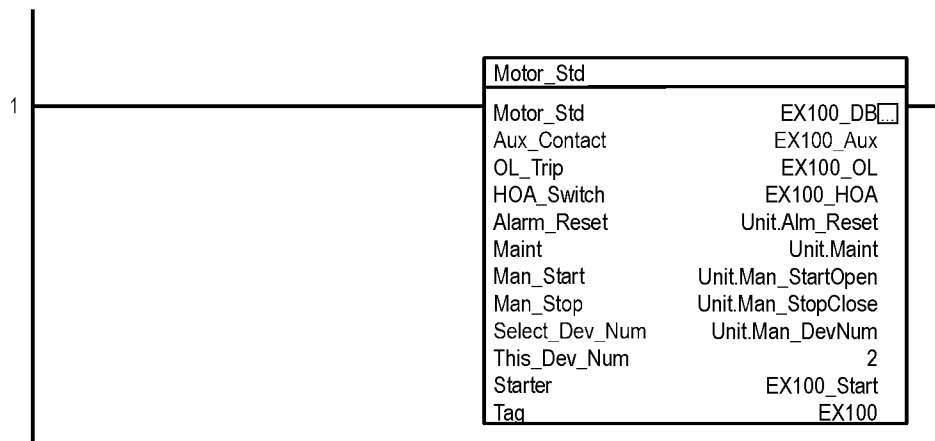
The parameters (fields) of the Motor\_Std add-on instruction are as follows:

<b><u>Field</u></b>	<b><u>Usage</u></b>	<b><u>Type</u></b>	<b><u>Description</u></b>
Motor_Std	In	Motor_Std	Data for add-on
Aux_Contact	In	BOOL	Auxiliary contact
OL_Trip	In	BOOL	Overload trip
HOA_Switch	In	BOOL	HOA switch auto contact
Alarm_Reset	In	BOOL	Resets alarm
Maint	In	BOOL	Maintenance privilege
Man_Start	In	BOOL	Manual start button
Man_Stop	In	BOOL	Manual stop button
Select_Dev_Num	In	DINT	Device selected to start/stop
This_Dev_Num	In	DINT	Number for this device
Starter	Out	BOOL	Motor starter contact
Tag	InOut	Motor_Std_Type	Device tag

The local tags of the Motor\_Std add-on instruction are as follows:

<b><u>Field</u></b>	<b><u>Type</u></b>	<b><u>Description</u></b>
Fail_Tmr	TIMER	Timer for aux fail alarm
Start_Req	BOOL	Start request
Stop_Req	BOOL	Stop request

When in maintenance mode (Maint input **on**), the operator is at a local touch-screen panel that has a common start and stop button. The operator sets the appropriate motor to be controlled by setting the "Select\_Dev\_Num" variable and then presses the start (Man\_Start) or stop (Man\_Stop) button to control the motor. The "This\_Dev\_Num" constant is the number associated with this device. For example to start the EX100 pump in Figure 4, the operator will select the pump, which will write the number 2 to data word, Unit.Man\_DevNum. After selecting the pump, the operator will press the start button turning on the input bit, Unit.Man\_StartOpen.

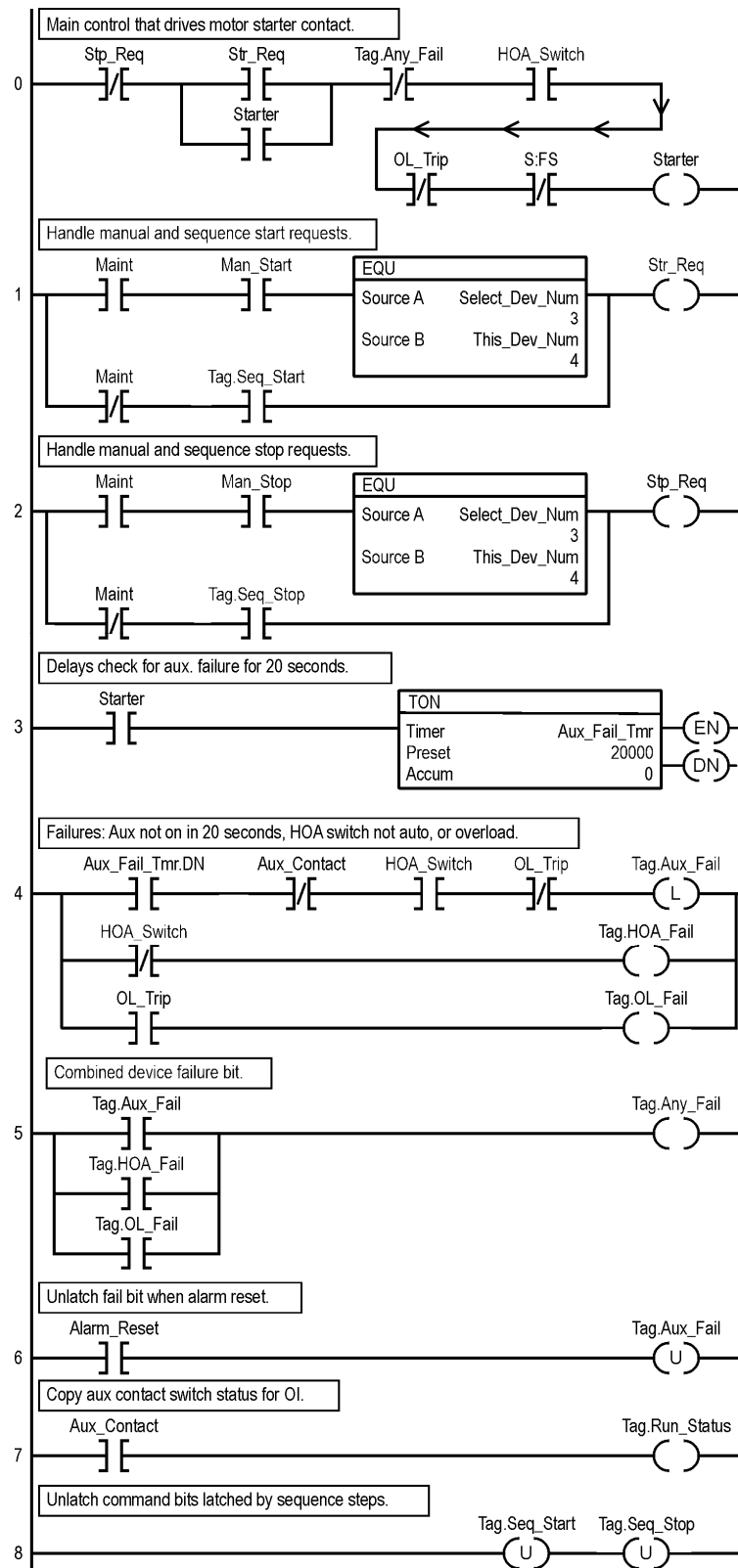


**Figure 4.** Motor (pump) device control with Motor\_Std add-on instruction.

This motor control is implemented in ladder logic for a ControlLogix PLC as in Figure 5. The zeroth rung controls the physical output that drives the motor contactor (or motor starter). Note that the motor is turned **off** on the first scan of the ladder (S:FS Boolean tag), when there is an overload, or when any failure occurs. Though an overload also causes the Tag.Any\_Fail to turn **on**, it is placed on the first rung so that an overload immediately turns **off** the motor, rather than waiting one scan for the Tag.Any\_Fail. The start and stop tags to drive the motor contactor are determined by the second and third rungs.

In rungs 1 and 2, the operator generates the start and stop commands when the control is in the maintenance state. The operator is at a local touch-screen panel that has a common start and stop button. The operator sets the appropriate motor to be controlled by setting the "Select\_Dev\_Num" input parameter and then presses the start or stop button to control the motor. If the "Select\_Dev\_Num" input parameter matches the "This\_Dev\_Num" input parameter, the device is started or stopped. When the control is in the operate privilege (not maintenance privilege), the motor is started and stopped by steps in the various sequences (function charts). The appropriate step latches the Tag.Seq\_Start or Tag.Seq\_Stop tag to control the motor. These two tags are always unlatched by this ladder logic. This method of sequence-based control allows one to change the steps in which a particular device is controlled by changing the ladder logic in the sequences logic and leaving the device logic untouched.

The third rung delays checking for the auxiliary fail alarm until 20 seconds after the motor is started. The first branch of the fourth rung generates the auxiliary fail alarm (Tag.Aux\_Fail). This alarm must be latched since this failure will cause the output to the starter to be turned off, thus disabling the conditions for this alarm. The second and third branches of the fourth rung generate the overload fail alarm and the indication that the HOA switch is not in the auto position. The fifth rung generates one summary failure alarm indication. All of these alarms are generally logged by the operator interface and appear on an alarm summary screen. The seventh rung resets the auxiliary failure alarm so that another start attempt is allowed. The last rung unlatches the sequential control commands.



**Figure 5.** Motor\_Std add-on instruction implementation.

## 5.2 Conveyor Motor Device Control

A conveyor motor device tag is shown in Figure 6. The physical connections to the PLC are:

**Physical Inputs:**

C1503_Aux	Auxiliary contact ( <b>on</b> when motor running)
C1503_OL	Overload trip ( <b>on</b> when motor overloaded)
C1503_HOA	HOA switch auto contact ( <b>on</b> when auto contact is closed)
SS1503	Speed switch ( <b>on</b> when conveyor running)

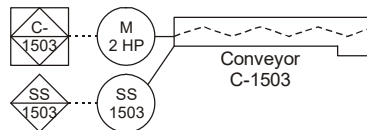
**Physical Outputs:**

C1503_Start	Starter ( <b>on</b> to start/run motor)
-------------	---

The device tag (for example, “C1503”) is of Motor\_Conv\_Type whose fields are as follows:

<u>Name</u>	<u>Data Type</u>	<u>Description</u>
Run_Status	BOOL	Run status of conveyor (not just motor)
Any_Fail	BOOL	Failure alarm
Aux_Fail	BOOL	Auxiliary fail alarm
SS_Fail	BOOL	Speed switch fail alarm
OL_Fail	BOOL	Overload alarm
HOA_Fail	BOOL	HOA-switch-not-in-auto alarm
Seq_Start	BOOL	Device start command from sequence
Seq_Stop	BOOL	Device stop command from sequence

which is the same as standard motor control with the addition of the speed switch.



**Figure 6.** P&ID symbol for motor device control associated with conveyor.

The operator commands and indications are:

**Operator Commands:**

Unit.Man_StartOpen	Manual Start for unit
Unit.Man_StopClose	Manual Stop for unit
Unit.Man_DevNum	Number of device started/stopped at OI.
Unit.Maint	Maintenance/Operate control privilege
Unit.Alm_Reset	Reset alarm

**Operator Indications:**

C1503.Run_Status	Run status
C1503.Any_Fail	Failure alarm
C1503.Aux_Fail	Auxiliary fail alarm
C1503.SS_Fail	Speed switch fail alarm
C1503.OL_Fail	Overload alarm
C1503.HOA_Fail	HOA-switch-not-in-auto alarm

The device is started/stopped by an automatic sequence with:

C1503.Seq_Start	Start
C1503.Seq_Stop	Stop

The motor control uses the Motor\_Conv add-on instruction (block) as shown in Figure 7. The ladder logic that implements this AOI block is shown in Figure 8. The use of this block is shown in Figure 7 to control a conveyor with tag C1503. The fields of the Motor\_Conv block are specified as the appropriate variables.

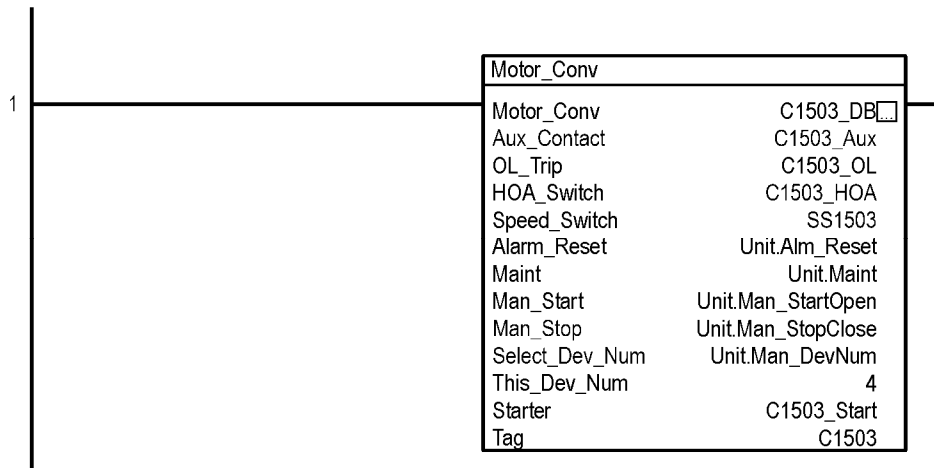
The parameters (fields) of the Motor\_Conv add-on instruction are as follows:

<b><u>Field</u></b>	<b><u>Usage</u></b>	<b><u>Type</u></b>	<b><u>Description</u></b>
Motor_Conv	In	Motor_Conv	Data for add-on
Aux_Contact	In	BOOL	Auxiliary contact
OL_Trip	In	BOOL	Overload trip
HOA_Switch	In	BOOL	HOA switch auto contact
Speed_Switch	In	BOOL	Speed switch contact
Alarm_Reset	In	BOOL	Resets alarm
Maint	In	BOOL	Maintenance privilege
Man_Start	In	BOOL	Manual start button
Man_Stop	In	BOOL	Manual stop button
Select_Dev_Num	In	DINT	Device selected to start/stop
This_Dev_Num	In	DINT	Number for this device
Starter	Out	BOOL	Motor starter contact
Tag	InOut	Motor_Conv_Type	Device tag

The local tags of the Motor\_Conv add-on instruction are as follows:

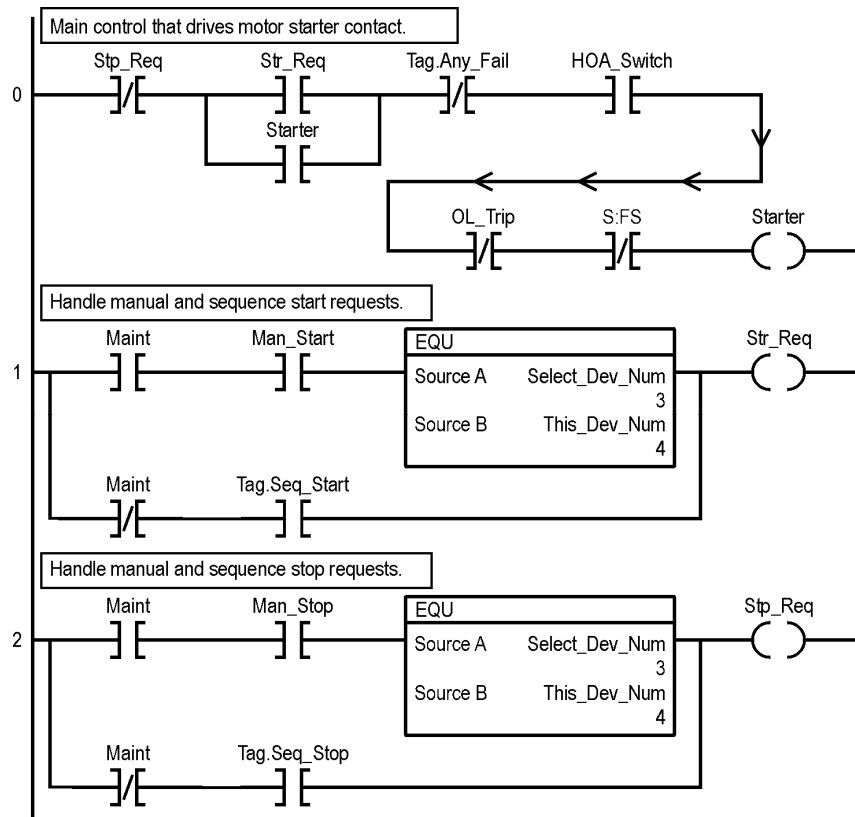
<b><u>Field</u></b>	<b><u>Type</u></b>	<b><u>Description</u></b>
Fail_Tmr	TIMER	Timer for aux and speed switch fail alarms
Start_Req	BOOL	Start request
Stop_Req	BOOL	Stop request

When in maintenance mode (Maint input **on**), the operator is at a local touch-screen panel that has a common start and stop button. This method of manual control is described in the section on standard motor control.



**Figure 7.** Conveyor device control with Motor\_Conv add-on instruction.

The implementation for the conveyor device control (Figure 8) differs from the standard motor control of Figure 5 only in the generation of the speed switch failure alarm. The remainder of the add-on block ladder logic is the same.



**Figure 8.** Motor\_Conv add-on instruction implementation. *(continued)*

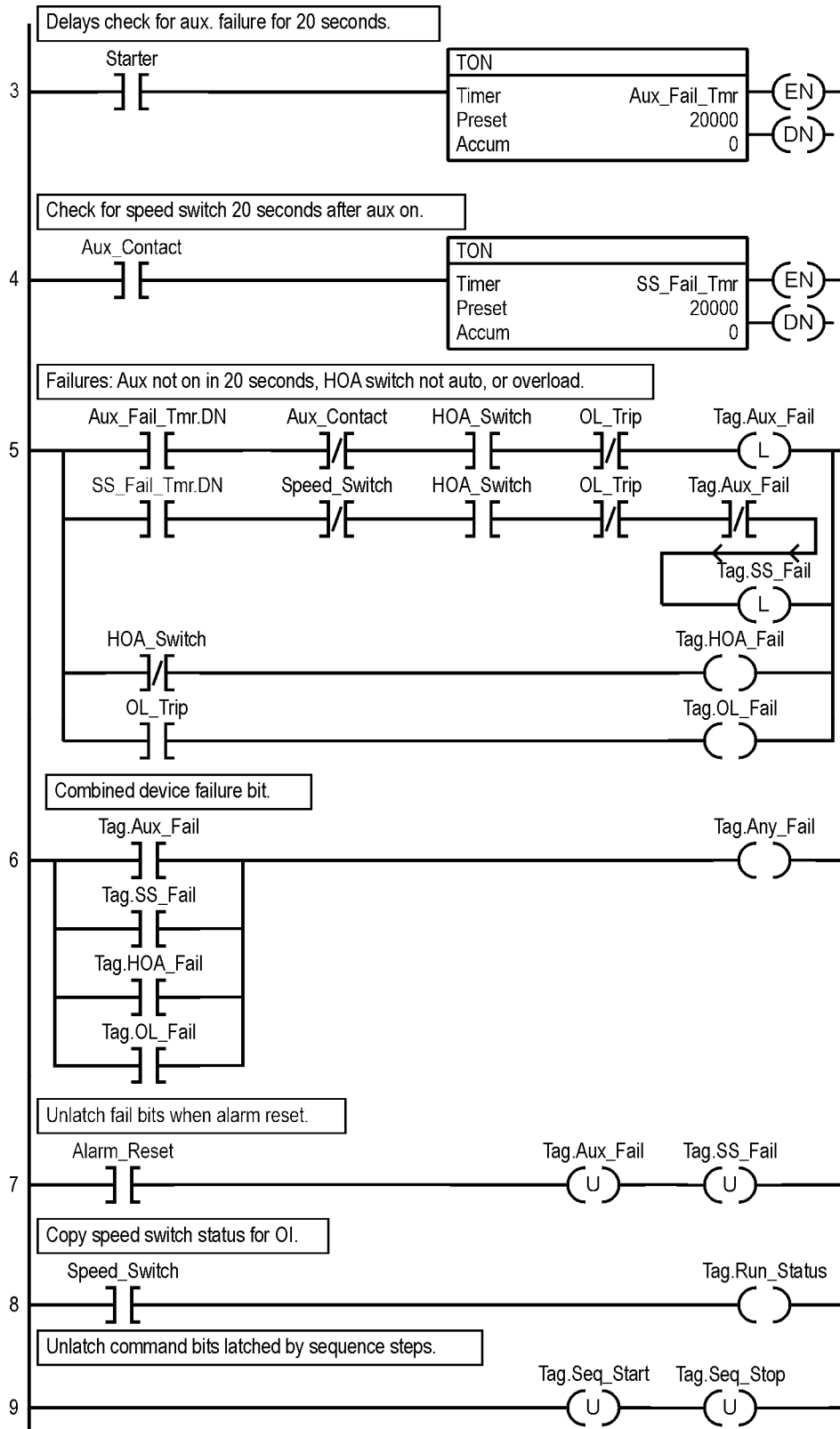


Figure 8. (continued)



### 5.3 Discrete Valve Device Control

A discrete valve device tag is shown in Figure 9. Example 21.3 in the text contains more information about this device. The physical connections to the PLC are:

Physical Inputs:

XV102\_OpenLS      Valve-open limit switch (**on** when valve fully open)  
 XV102\_CloseLS      Valve-closed limit switch (**on** when valve fully closed)

Physical Output:

XV102\_Vlv\_Sol      Valve solenoid (**on** to open the valve)

The device tag (for example, “XV102”) is of Valve\_Disc\_Type whose fields are as follows:

<u>Name</u>	<u>Data Type</u>	<u>Description</u>
Open_Status	BOOL	Open status of valve
Close_Status	BOOL	Closed status of valve
Any_Fail	BOOL	Failure alarm
FTO_Fail	BOOL	Fail-to-open fail alarm
FTC_Fail	BOOL	Fail-to-close fail alarm
Seq_Open	BOOL	Device open command from sequence
Seq_Close	BOOL	Device close command from sequence

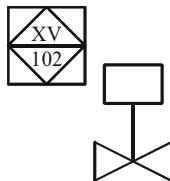
The operator commands and indications are:

Operator Commands:

Unit.Man\_StartOpen      Manual Open/Start for unit  
 Unit.Man\_StopClose      Manual Close/Stop for unit  
 Unit.Man\_DevNum      Number of device started/stopped at OI.  
 Unit.Maint      Maintenance/Operate control privilege  
 Unit.Alm\_Reset      Reset alarm

Operator Indications:

XV102.Open\_Status      Open status of valve  
 XV102.Close\_Status      Close status of valve  
 XV102.Any\_Fail      Failure alarm  
 XV102.FTO\_Fail      Fail-to-open alarm  
 XV102.FTC\_Fail      Fail-to-close alarm



**Figure 9.** P&ID symbol for discrete valve device.

The device is opened/closed by an automatic sequence with:

```
XV102.Seq_Open    Open
XV102.Seq_Close   Close
```

The discrete valve control uses the Valve\_Disc add-on instruction (block) as shown in Figure 10. The ladder logic that implements this function block is shown in Figure 11. The use of this block is shown in Figure 10 to control a discrete valve with tag XV102. The fields of the Valve\_Disc block are specified as the appropriate variables.

The parameters (fields) of the Valve\_Disc add-on instruction are as follows:

<b><u>Field</u></b>	<b><u>Usage</u></b>	<b><u>Type</u></b>	<b><u>Description</u></b>
Valve_Discrete	In	Valve_Disc	Data for add-on
Open_LS	In	BOOL	Valve open limit switch
Close_LS	In	BOOL	Valve closed limit switch
Alarm_Reset	In	BOOL	Resets alarm
Maint	In	BOOL	Maintenance privilege
Man_Open	In	BOOL	Manual open button
Man_Close	In	BOOL	Manual close button
Select_Dev_Num	In	DINT	Device selected to start/stop
This_Dev_Num	In	DINT	Number for this device
Vlv_Sol	Out	BOOL	Valve solenoid
Tag	InOut	Valve_Disc_Type	Device tag

The local tags of the Valve\_Disc add-on instruction are as follows:

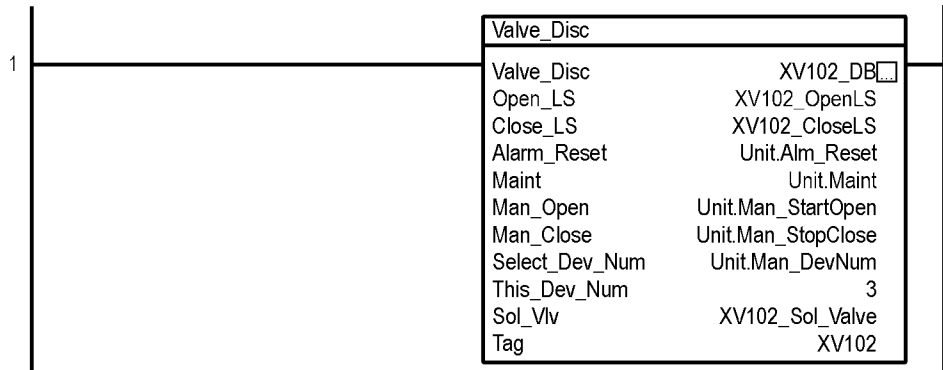
<b><u>Field</u></b>	<b><u>Type</u></b>	<b><u>Description</u></b>
Fail_Tmr	TIMER	Timer for failure alarms

When in maintenance mode (Maint input **on**), the operator is at a local touch-screen panel that has a common start and stop button. This method of manual control is described in the section on standard motor control.

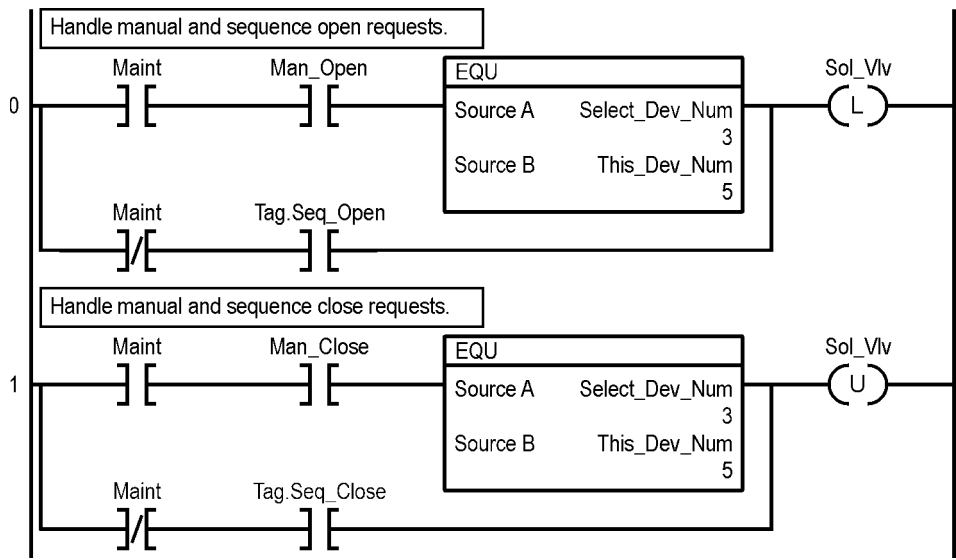
Valve control is implemented in ladder logic for the ControlLogix in Figure 11. The first two rungs control the physical output that drives the valve solenoid coil. Latching outputs are used, in contrast to the motor control in the previous sections, because failures do not automatically close the valve. For a failure, the valve solenoid control should not change. A frequent source of valve failure is a "sticky" stem. Maintenance personnel will often "bang on the valve" to release the stem, and then the valve will move to the desired position. The open/close commands function in the same manner as the start/stop commands for the on/off motor control in Figure 5. As for the motor control examples, the maintenance state shown here is for a grouping of equipment. Rung 2 times the failure conditions so the alarms are generated after a condition persists for 20 seconds. This delay allows time for the valve to change state when neither limit switch will be closed. A failure is assumed when one of four conditions persists for 20 seconds:

- Both limit switches closed
- Both limit switches open
- Valve commanded to open and open limit switch not closed
- Valve commanded to close and closed limit switch not closed

Rung 3 generates the actual alarms. Rung 4 generates one summary failure indication that would appear on an alarm summary screen and rung 5 resets the failure alarms. The limit switch indications are passed to the OI on rung 6 and the last rung unlatches the sequential control commands.



**Figure 10.** Discrete valve device control with Valve\_Disc add-on instruction.



**Figure 11.** Valve\_Disc add-on instruction implementation. *(continued)*

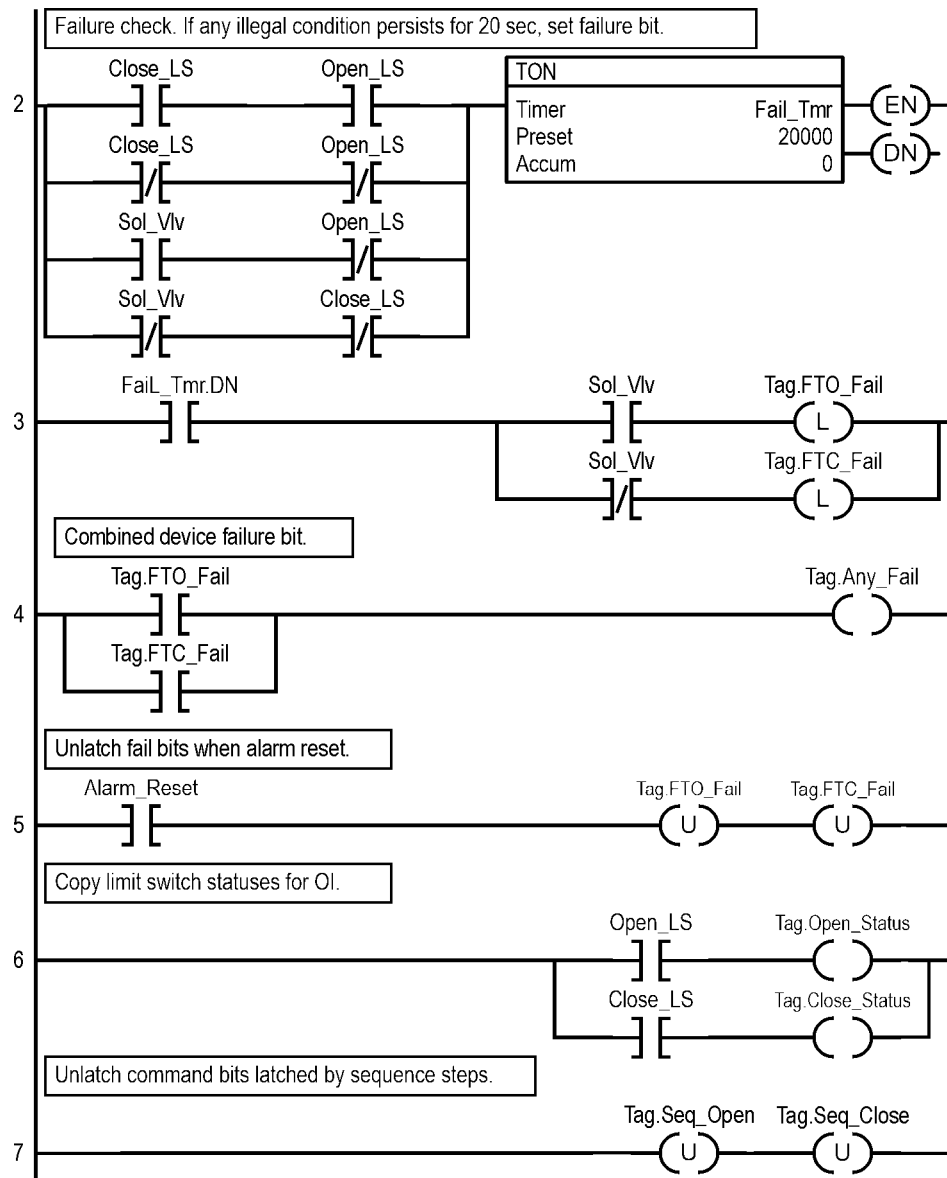


Figure 11. (continued)

## 5.4 Slide Gate Control

A slide gate device tag is shown in Figure 12. The gate is driven by a reversing motor that uses a rack and pinion to drive the gate. When the motor runs in the “open” direction, the pinion gear drives the rack mounted on the gate, opening the gate. When the motor runs in the “close” direction, the gate closes. The physical connections to the PLC are:

### Physical Inputs:

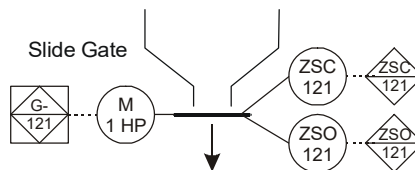
G121_Open_Aux	Open auxiliary contact ( <b>on</b> when motor running to open)
G121_Close_Aux	Close auxiliary contact ( <b>on</b> when motor running to close)
G121_OL	Overload trip ( <b>on</b> when motor overloaded)
G121_HOA	HOA switch auto contact ( <b>on</b> when auto contact is closed)
ZSO121	Gate-open limit switch ( <b>on</b> when gate open)
ZSC121	Gate-closed limit switch ( <b>on</b> when gate closed)

### Physical Outputs:

G121_Start_Open	Open starter ( <b>on</b> to start/run motor to open)
G121_Start_Close	Close starter ( <b>on</b> to start/run motor to close)

The device tag (for example, “G121”) is of Gate\_Slide\_Type whose fields are as follows:

<u>Name</u>	<u>Data Type</u>	<u>Description</u>
Run_Status	BOOL	Motor running status
Open_Status	BOOL	Open status of gate
Close_Status	BOOL	Closed status of gate
Any_Fail	BOOL	Failure alarm
Aux_Fail	BOOL	Auxiliary fail alarm
OL_Fail	BOOL	Overload alarm
HOA_Fail	BOOL	HOA-switch-not-in-auto alarm
Seq_Open	BOOL	Device open command from sequence
Seq_Close	BOOL	Device close command from sequence



**Figure 12.** P&ID symbol for slide gate device.

The operator commands and indications are:

### Operator Commands:

Unit.Man_StartOpen	Manual Open/Start for unit
Unit.Man_StopClose	Manual Close/Stop for unit
Unit.Man_DevNum	Number of device started/stopped at OI.
Unit.Maint	Maintenance/Operate control privilege

Unit.Alm_Reset	Reset alarm
Operator Indications:	
G121.Run_Status	Running status of motor
G121.Open_Status	Open status of gate
G121.Close_Status	Close status of gate
G121.Any_Fail	Failure alarm
G121.Aux_Fail	Auxiliary fail alarm
G121.OL_Fail	Overload alarm
G121.HOA_Fail	HOA-switch-not-in-auto alarm

The device is opened/closed by an automatic sequence with:

G121.Seq_Open	Open
G121.Seq_Close	Close

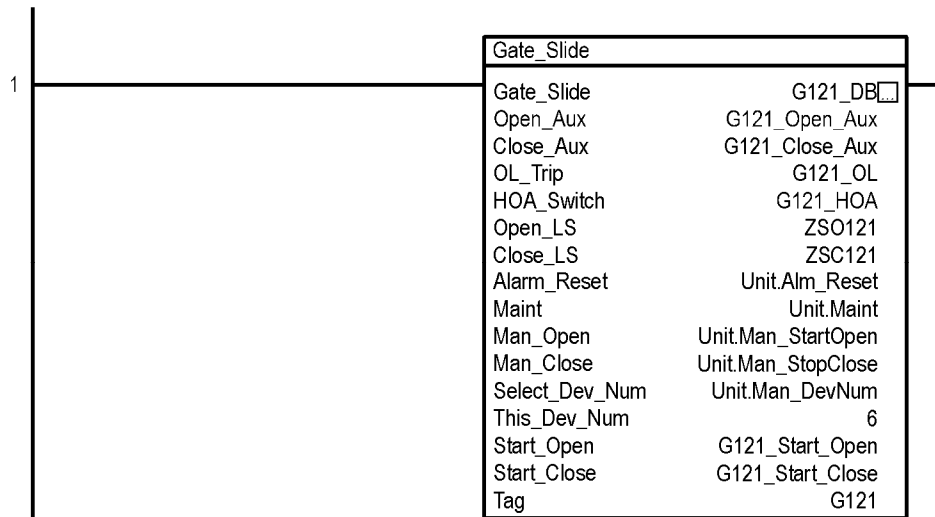
The slide gate control uses the Gate\_Slide add-on instruction (block) as shown in Figure 13. The ladder logic that implements this function block is shown in Figure 14. The use of this block is shown in Figure 13 to control a slide gate with tag G121. The fields of the Gate\_Slide block are specified as the appropriate variables.

The parameters (fields) of the Gate\_Slide add-on instruction are as follows:

<b><u>Field</u></b>	<b><u>Usage</u></b>	<b><u>Type</u></b>	<b><u>Description</u></b>
Gate_Slide	In	Gate_Slide	Data for add-on
Open_Aux	In	BOOL	Open auxiliary contact
Close_Aux	In	BOOL	Close auxiliary contact
OL_Trip	In	BOOL	Overload trip
HOA_Switch	In	BOOL	HOA switch auto contact
Open_LS	In	BOOL	Gate open limit switch
Close_LS	In	BOOL	Gate closed limit switch
Alarm_Reset	In	BOOL	Resets alarm
Maint	In	BOOL	Maintenance privilege
Man_Open	In	BOOL	Manual open button
Man_Close	In	BOOL	Manual close button
Select_Dev_Num	In	DINT	Device selected to start/stop
This_Dev_Num	In	DINT	Number for this device
Start_Open	Out	BOOL	Open starter contact
Start_Close	Out	BOOL	Close starter contact
Tag	InOut	Gate_Slide_Type	Device tag

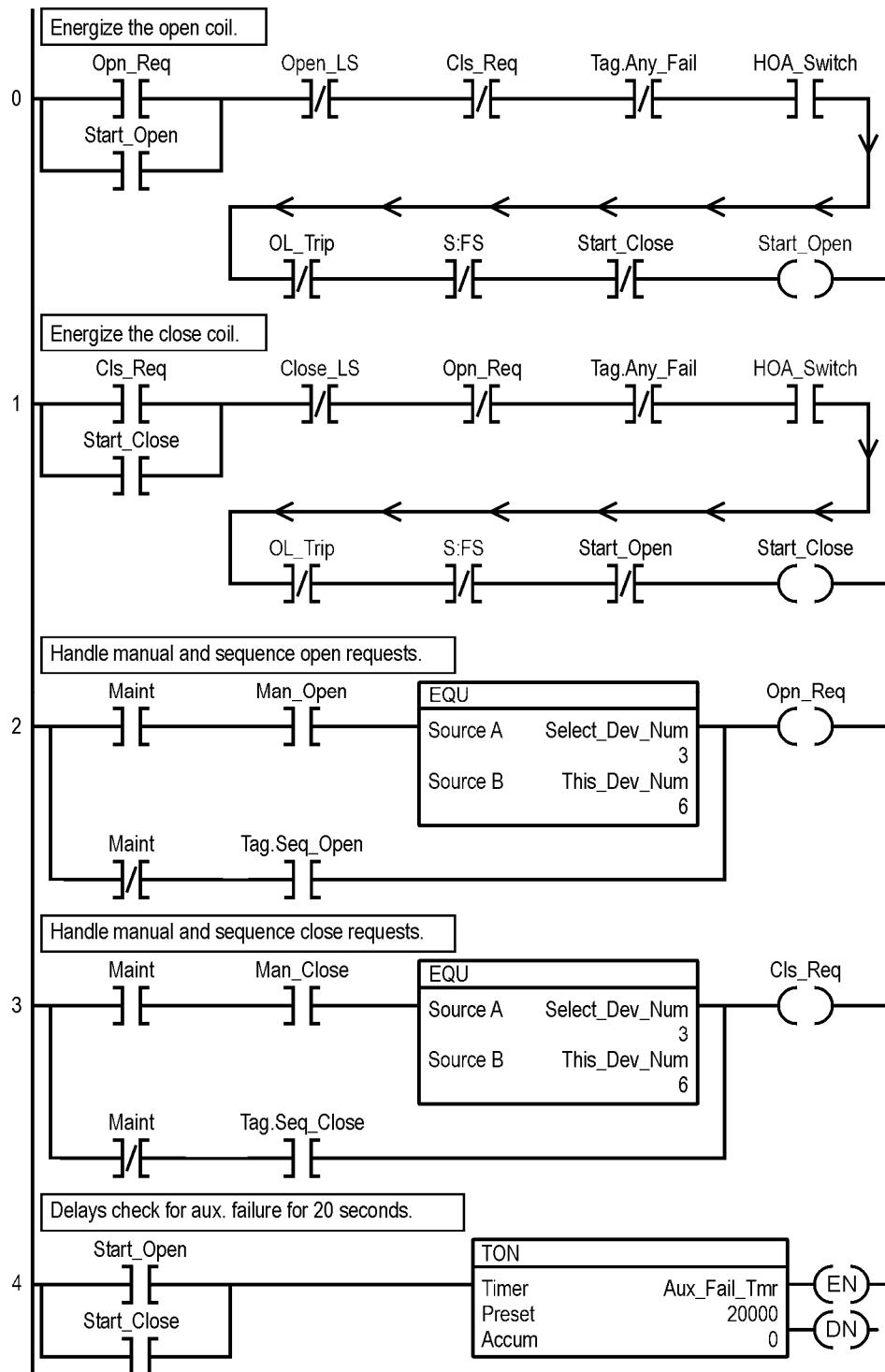
The local tags of the Gate\_Slide add-on instruction are as follows:

<b><u>Field</u></b>	<b><u>Type</u></b>	<b><u>Description</u></b>
Aux_Fail_Tmr	TIMER	Timer for aux fail alarm
Open_Req	BOOL	Open request
Close_Req	BOOL	Close request



**Figure 13.** Slide gate device control with Gate\_Slide add-on instruction.

The control for the slide gate device differs from the standard motor control of Figure 5 in that the motor is reversing and has two starter coils. Also, the motor runs until the gate has moved to its new position and then stops. Figure 14 shows the ladder logic for this device control. The control for the open starter coil is in rung 0. Note that it is disabled when a closed command is received. The control for the close starter coil is shown in rung 1 and is similar to the open starter coil. The fail timer runs for either starter coil, and one auxiliary failure is reported for either starter coil.



**Figure 14.** Gate\_Slide add-on instruction implementation. *(continued)*



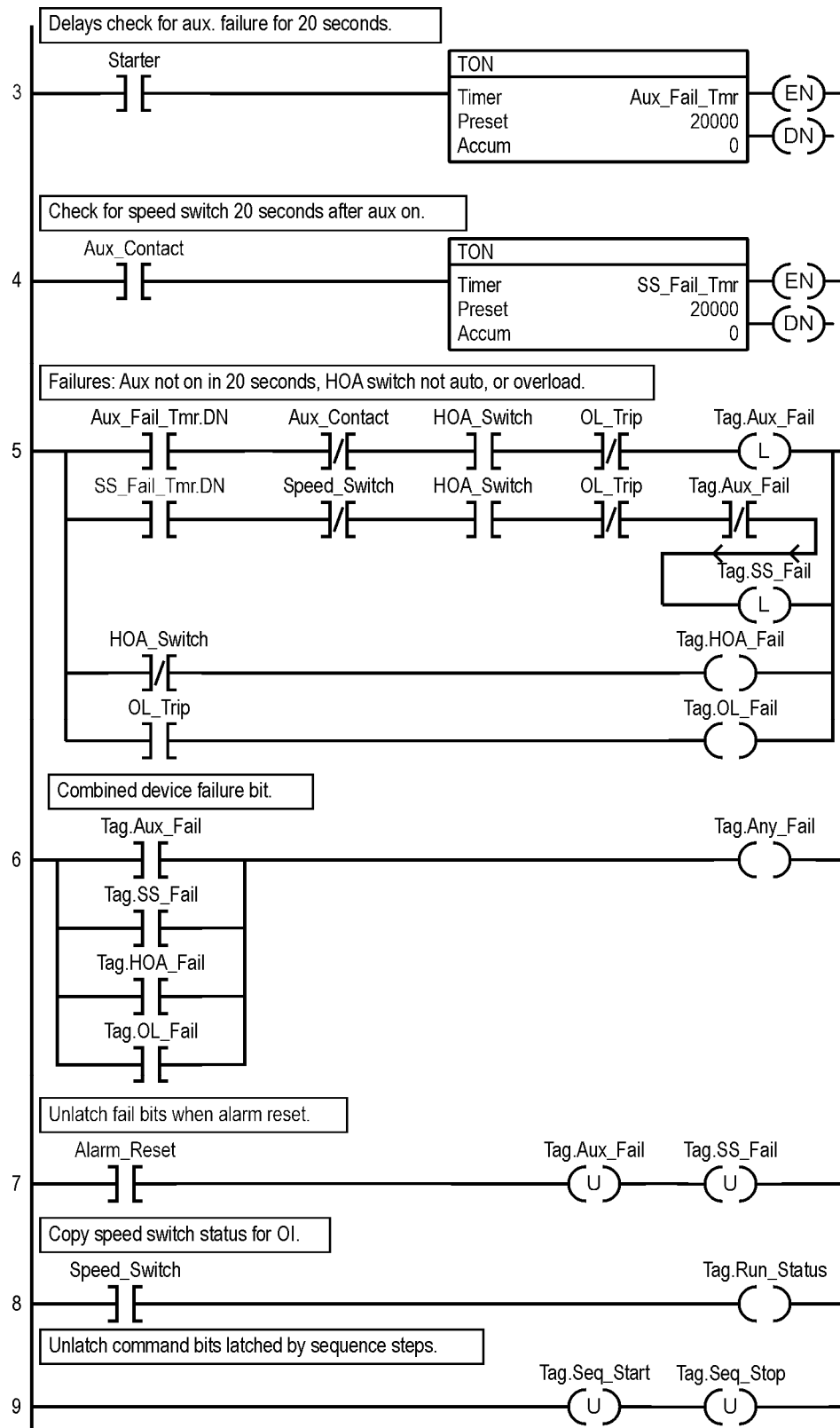


Figure 14. (continued)

## 5.5 Flop Gate Control

A flop gate device tag is shown in Figure 15. The gate is driven by a double-acting pneumatic cylinder connected to a hinged flap that diverts a gravity-fed material to one of two places, called “left” and “right.” When the “left” direction control is on, the cylinder moves the gate to the right, diverting the material to the left path. When the “right” direction control is on, the cylinder moves the gate to the left, diverting the material to the right path. The physical connections to the PLC are:

**Physical Inputs:**

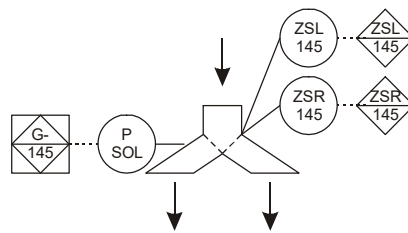
ZSL145	Left-path limit switch ( <b>on</b> when diverted left)
ZSR145	Right-path limit switch ( <b>on</b> when diverted right)
G145_HOA	HOA switch auto contact ( <b>on</b> when auto contact is closed)

**Physical Outputs:**

G145_Sol_Left	Left path solenoid ( <b>on</b> to move gate to divert left)
G145_Sol_Right	Right path solenoid ( <b>on</b> to move gate to divert right)

The device tag (for example, “G145”) is of Gate\_Flop\_Type whose fields are as follows:

<u>Name</u>	<u>Data Type</u>	<u>Description</u>
Run_Status	BOOL	Solenoid moving status
Left_Status	BOOL	Left status of gate
Right_Status	BOOL	Right status of gate
Any_Fail	BOOL	Failure alarm
HOA_Fail	BOOL	HOA-switch-not-in-auto alarm
FTL_Fail	BOOL	Fail-to-divert-left alarm
FTR_Fail	BOOL	Fail-to-divert-right alarm
Seq_Left	BOOL	Device left command from sequence
Seq_Right	BOOL	Device right command from sequence



**Figure 15.** P&ID symbol for flop gate device.

The operator commands and indications are:

**Operator Commands:**

Unit.Man_StartOpen	Manual Open/Start for unit (gate diverts left)
Unit.Man_StopClose	Manual Close/Stop for unit (gate diverts right)
Unit.Man_DevNum	Number of device started/stopped at OI.

Unit.Maint	Maintenance/Operate control privilege
Unit.Alm_Reset	Reset alarm
Operator Indications:	
G145.Run_Status	Moving status of solenoid
G145.Left_Status	Left limit switch status of gate
G145.Right_Status	Right limit switch status of gate
G145.Any_Fail	Failure alarm
G145.HOA_Fail	HOA-switch-not-in-auto alarm
G145.FTL_Fail	Fail-to-divert-left alarm
G145.FTR_Fail	Fail-to-divert-right alarm

The device is opened/closed by an automatic sequence with:

G145.Seq_Left	Left position
G145.Seq_Right	Right position

The flop gate control uses the Gate\_Flop add-on instruction (block) as shown in Figure 16. The ladder logic that implements this function block is shown in Figure 17. The use of this block is shown in Figure 16 to control a slide gate with tag G145. The fields of the Gate\_Flop block are specified as the appropriate variables.

The parameters (fields) of the Gate\_Flop add-on instruction are as follows:

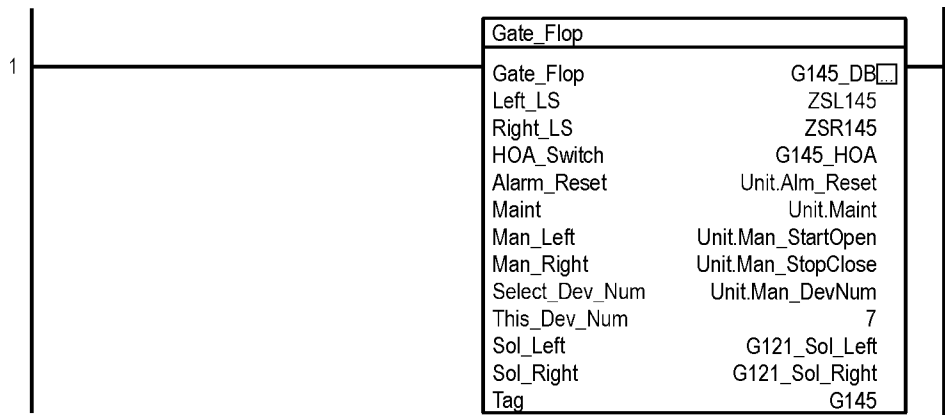
<b><u>Field</u></b>	<b><u>Usage</u></b>	<b><u>Type</u></b>	<b><u>Description</u></b>
Gate_Flop	In	Gate_Flop	Data for add-on
Left_LS	In	BOOL	Left position limit switch
Right_LS	In	BOOL	Right position limit switch
HOA_Switch	In	BOOL	HOA switch auto contact
Alarm_Reset	In	BOOL	Resets alarm
Maint	In	BOOL	Maintenance privilege
Man_Left	In	BOOL	Manual left button
Man_Right	In	BOOL	Manual right button
Select_Dev_Num	In	DINT	Device selected to left/right
This_Dev_Num	In	DINT	Number for this device
Sol_Left	Out	BOOL	Left position solenoid valve
Sol_Right	Out	BOOL	Right position solenoid valve
Tag	InOut	Gate_Flop_Type	Device tag

The local tags of the Gate\_Slide add-on instruction are as follows:

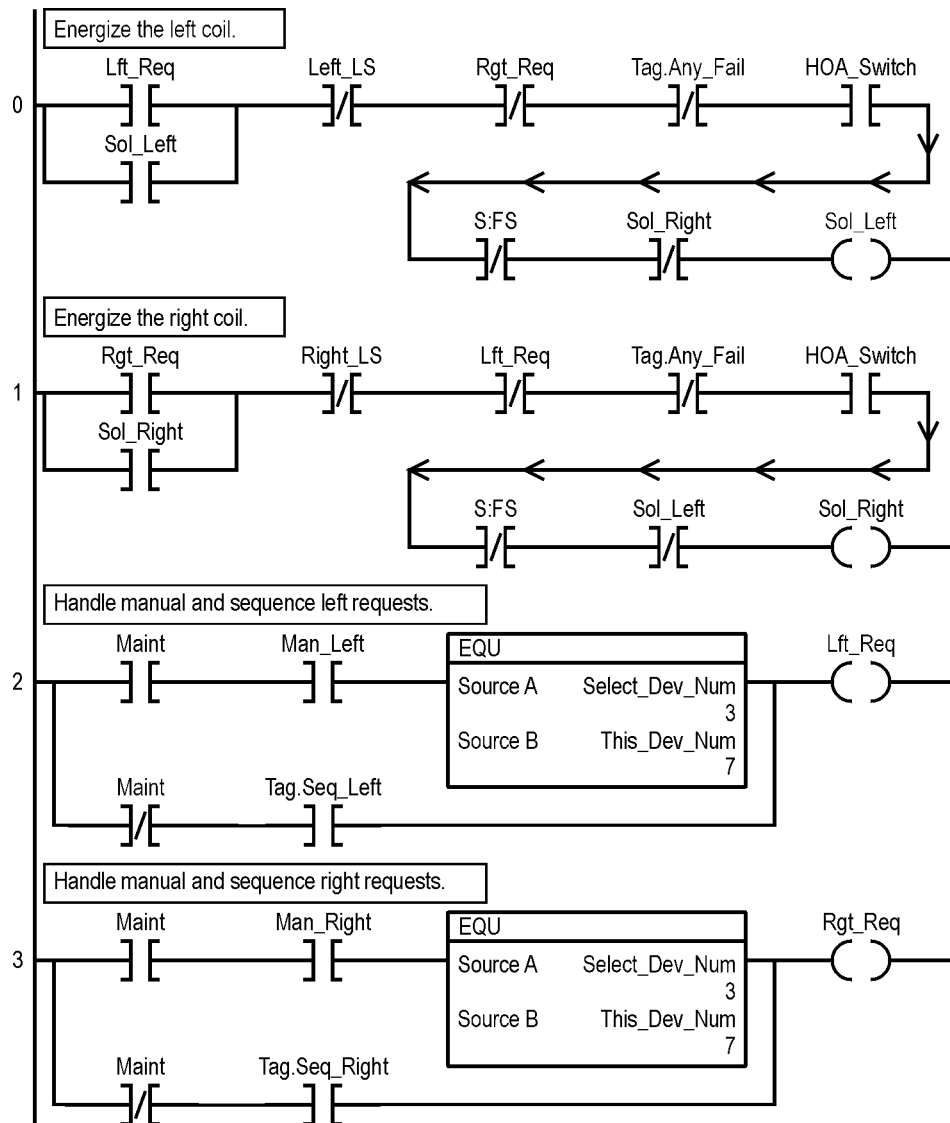
<b><u>Field</u></b>	<b><u>Type</u></b>	<b><u>Description</u></b>
Fail_Tmr	TIMER	Timer for fail alarms
Left_Req	BOOL	Left request
Right_Req	BOOL	Right request

The control for the flop gate device is similar to the slide control, except that there is no overload trip contact. Figure 17 shows the ladder logic for this device control. The control for the left valve control is in rung 0. Note that it is disabled when a right command is received. The control for the right valve control is shown in rung 1 and is similar to the left valve control. The fourth

rung times the failure conditions in a similar manner to a discrete valve. The fifth rung generates the actual alarms similarly to the discrete valve.



**Figure 16.** Flop gate device control with Gate\_Flop add-on instruction.



**Figure 17.** Gate\_Flop add-on instruction implementation. *(continued)*

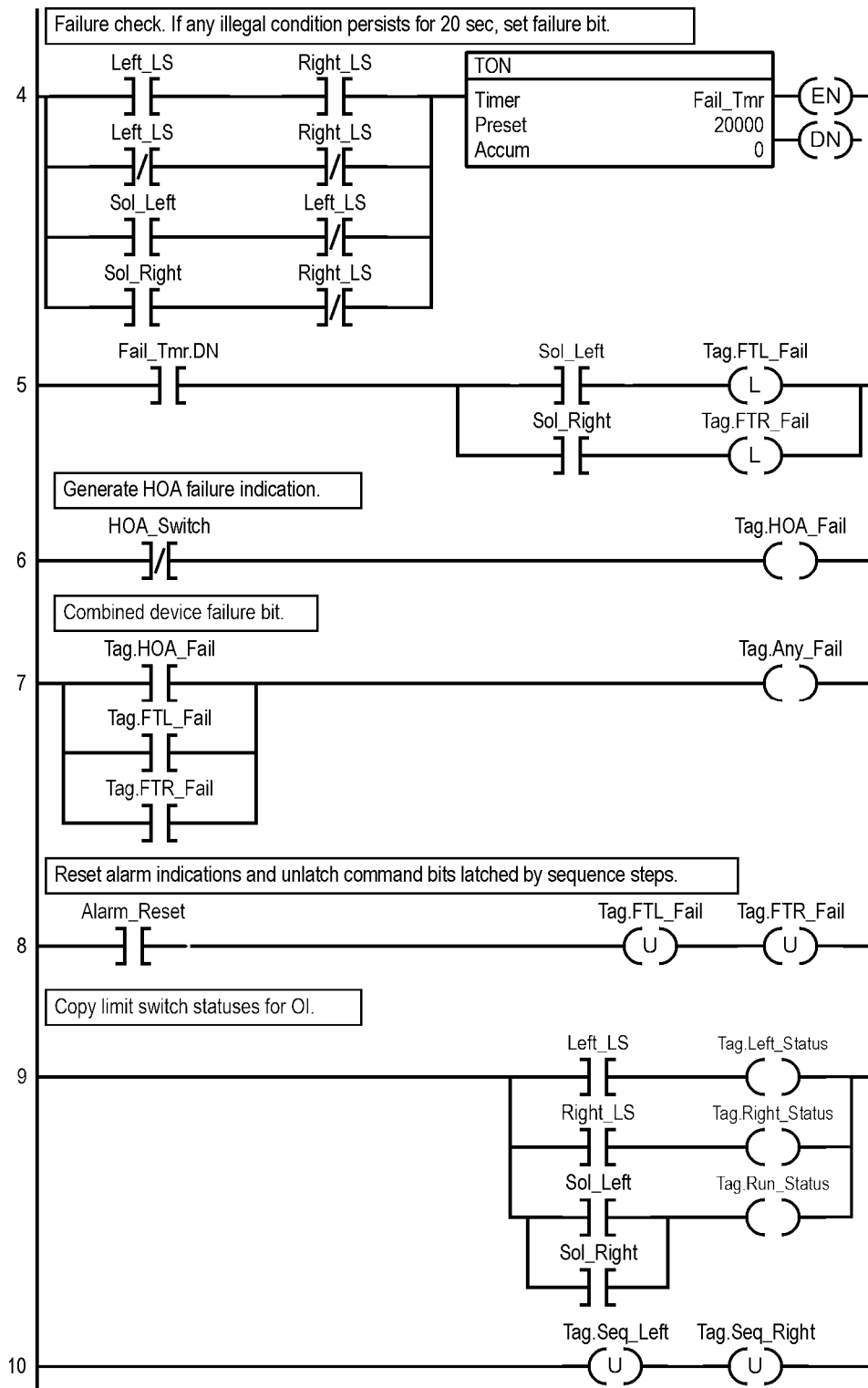


Figure 17. (continued)

## 6. Sequenced Control

A sequence is initiated in one of two ways: operator request through the OI or a request from this or another PLC due to process conditions. The local/remote mode defines the location of the operator that initiates the sequences. In local mode, the operator is assumed to be close to the unit, at a local touch panel (LTP). In remote mode, the operator is at a distant control room, or possibly a supervisory controller that coordinates the operation of multiple units.

All sequenced control logic is programmed in the PLC. In addition, the PLC performs all interlocking. Sequenced loop control is programmed in the PLC. Sequence requests for device control are latched during a sequence. In order to keep the operator abreast of sequence activities, message numbers are passed to the OI via integers. Corresponding text messages may be displayed on an OI. In similar fashion, the PLC conveys the time remaining in a timed sequence activity for display on an OI.

The Seq\_Type user-defined type (UDT) holds the data associated with a sequence. The Seq\_Type fields are as follows:

<u>Name</u>	<u>Data Type</u>	<u>Description</u>
Step_Num	DINT	Current step number
Running	BOOL	Running indication
PC_Start	BOOL	Sequence start request from PC
LTP_Start	BOOL	Sequence start request from LTP
Ons	BOOL	One-shot storage to start
Auto_Start	BOOL	Auto-start request
Auto_Ons	BOOL	Auto-start one-shot storage
Req_Tmr	TIMER	Auto-start hold timer

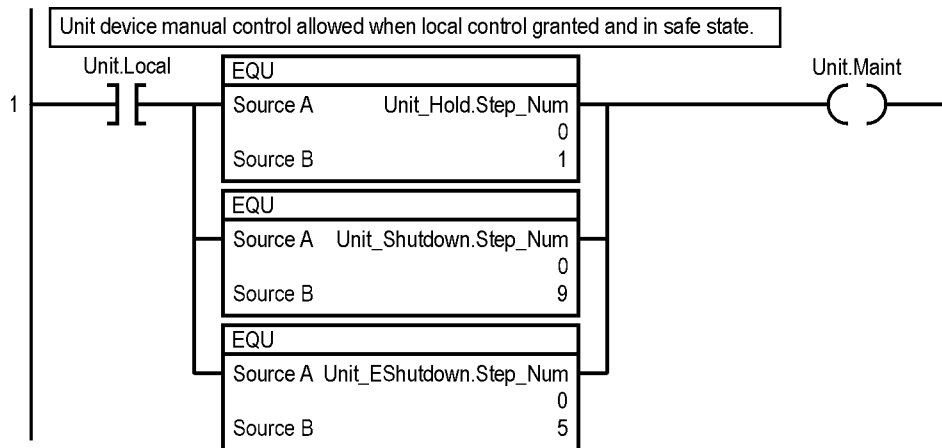
### 6.1 Sequence Code

There are five sections in a PLC sequence and these parts are standard for the different sequences implemented in the PLC:

1. Grant maintenance privilege for unit device control.
2. Initiate sequence.
3. Automatic start for sequence.
4. Handle step-in-progress bits, including increment step counter
5. Transitions between steps and device control from sequence

Note that since part 1 is for all sequences for a unit, it is contained in one routine (for example, Unit\_Misc). All of the other parts to a sequence are contained in a routine. The abnormal conditions that are frequently used for automatic starts are contained in a separate routine (for example, Unit\_Abnormal).

Figure 18 shows the first section of the sequence. This rung handles the granting of maintenance privilege to the unit. In maintenance privilege, the devices are started/stopped manually by the operator rather than automatically by a sequence. Maintenance privilege is granted when the Local indication is set and the “Hold”, “Shutdown”, or “E-Shutdown” sequences are in their last steps. Maintenance privilege may be granted in other sequences, but these are nearly universal.



**Figure 18.** Granting maintenance privilege for unit devices.

The second part of the sequence (Figure 19) initiates the sequence, generates the message number, and sets the indication that the sequence is in progress. The sequence is initiated one of three ways:

1. By the operator from the PC (when not in local mode), or
2. By the operator from the LTP (when in local mode), or
3. By an internal PLC start request ("auto-request"), which is the fifth part of a sequence.

An "auto-request" is a programmatic start of a sequence. For example, abnormal conditions may "auto-start" the shutdown sequence. Note the use of the GEQ to seal around the start requests. The reason for this method of sealing is explained later.

The start request is blocked if any other sequence in the group is auto-starting. When the sequence is initiated, the Unit\_Startup.Step\_Num is set to 1 and the step numbers for the other sequences in the group are set to zero. The output logic also adds a constant to the step number to calculate the message number and also generates an indication that the sequence is in progress. The message number and sequence-running indication is for the operator interface. Each sequence in a unit adds a different constant to calculate the message number such that each step in each unit sequence has a unique number.

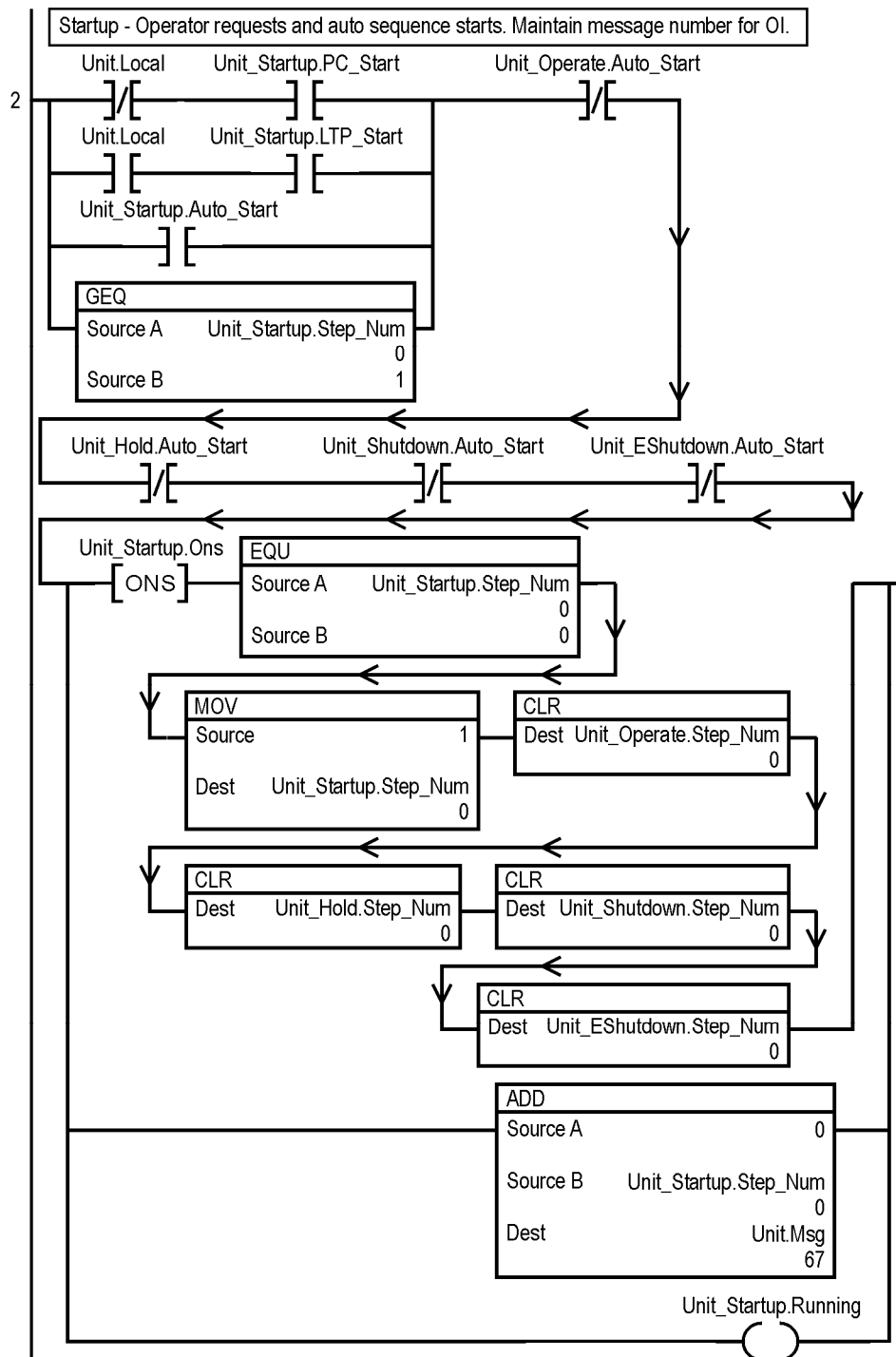
Note that a ">=" comparison is used to seal the sequence start request. Using the ".Running" bit for a sequence does not work in this case. A sequence is turned off when its step number is zeroed due to another sequence starting. If the ".Running" bit is used for the seal, the seal is not broken if this sequence's step number is zeroed due to an operator request or auto-start request to start another sequence. The use of the greater-than-or-equal to seal the sequence start request also handles the case where one set of steps is used for multiple sequences. One common case where this is done is in material transfer from one source tank to one of multiple destination tanks. As far as the operator is concerned, each transfer is a separate sequence. Each of the transfer sequences has an individual ".Running" bit but all share the same set of steps. Since the only difference among these sequences is which valve is opened/closed in a few steps, it is reasonable to combine all of these transfers in one sequence and use the individual ".Running"



bits to open/close the appropriate valves in the steps. This is one of the two methods to handle multiple transfers explained in section 6.3.

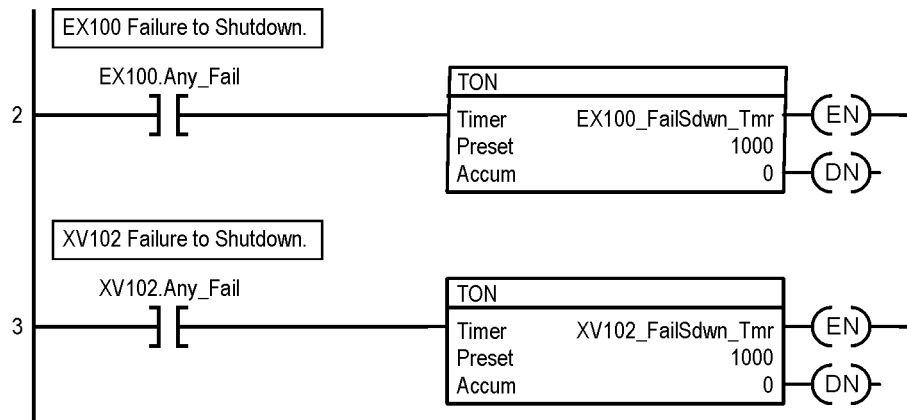
In like manner, **never use the ".Running" bit of any other sequence in the logic of this rung.** Check the appropriate sequence counter accumulator to verify if another sequence is running.

The use of the EQU after the ONS handles the case where one wants to start at another step other than step 1. This happens when the hold sequence functions as a “pause” and does not reset this sequence’s counter. When the operator exits the hold sequence, this sequence automatically resumes the suspended step. The step counter is not zero and there does not need to be a transition into the first step or any counter resets. This method of pausing also requires comparison check that “Hold.Ctr.ACC $\neq$ 0” in series with the auto-start contacts on this rung. This method of pausing is not normally used in this project.



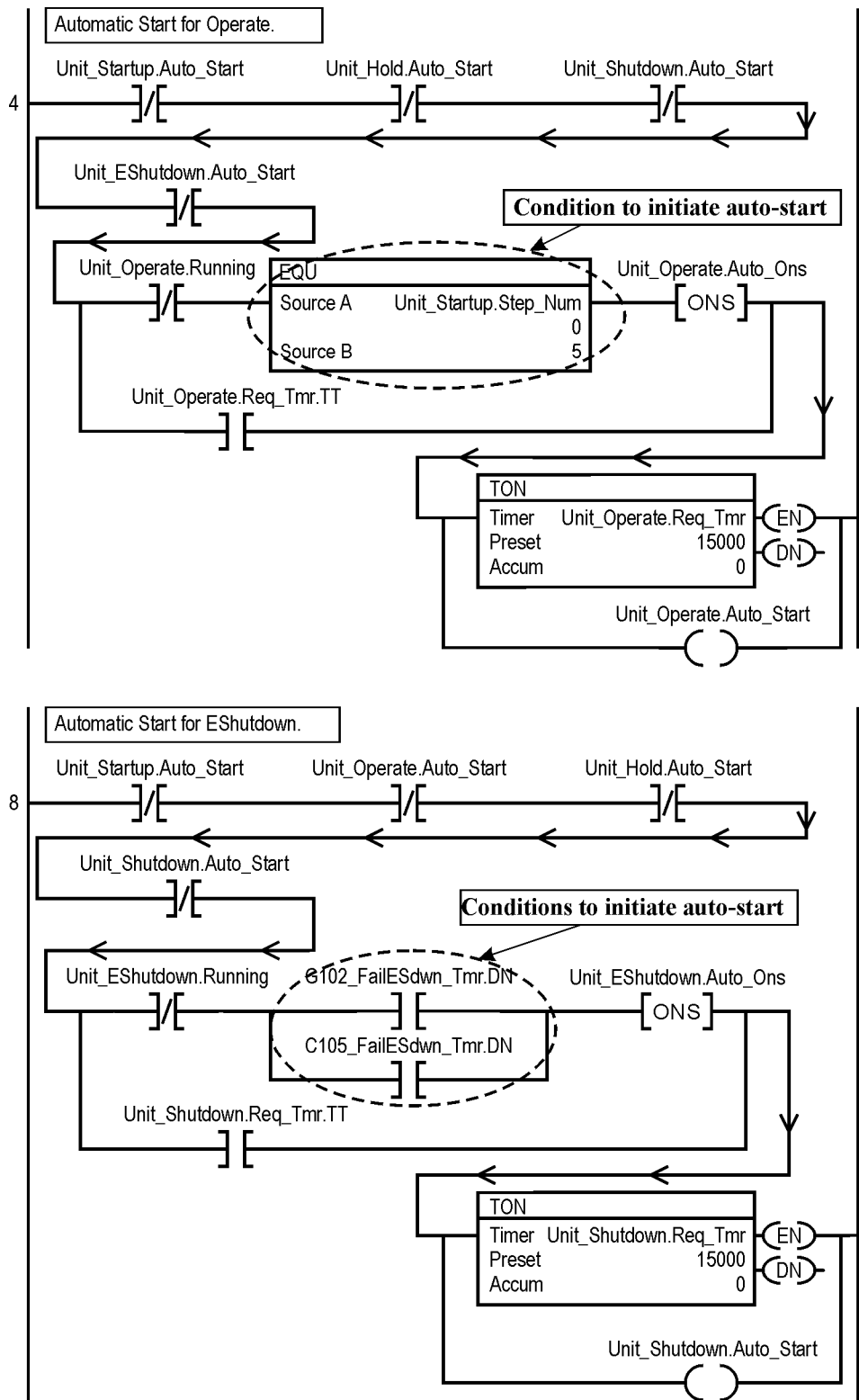
**Figure 19.** Initiate sequence.

The third section is the logic that handles all of the “automatic start” start sequence requests. Associated with the “automatic start” logic are the rungs that delay various failures (Figure 20) that cause a certain sequence, commonly Hold, Shutdown, or E-Shutdown to be started. These are contained in a separate routine.



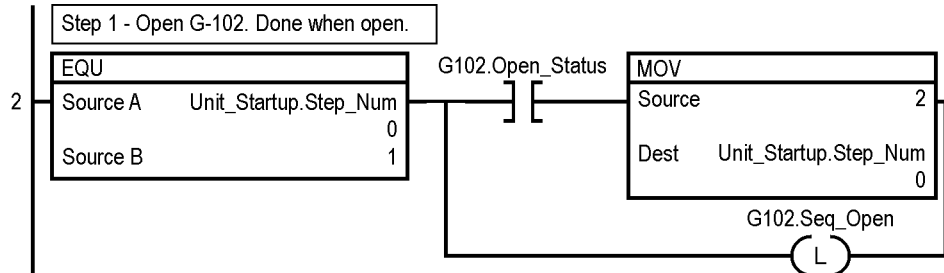
**Figure 20.** Failure conditions that invoke a sequence.

Rung 4 in Figure 21 is the automatic start for the Unit\_Operate sequence. This sequence is automatically started when the Unit\_Startup is in the last step (step 6). The series conditions at the beginning of the rung prevent the sequence from starting if another sequence in the unit is already auto-starting. There is also a condition that prevents an auto restart of the Unit\_Operate if it is already running. The start condition (circled) uses a one-shot or transitional contact to turn on the auto start bit, and a timer is used to hold the auto start request bit **on** for 15 seconds (to allow time for other logic to be resolved). Rung 8 is a sample automatic start for the Unit\_EShutdown sequence. The conditions that start the sequence are the parallel combination of the done bits from the device failure delay timers.



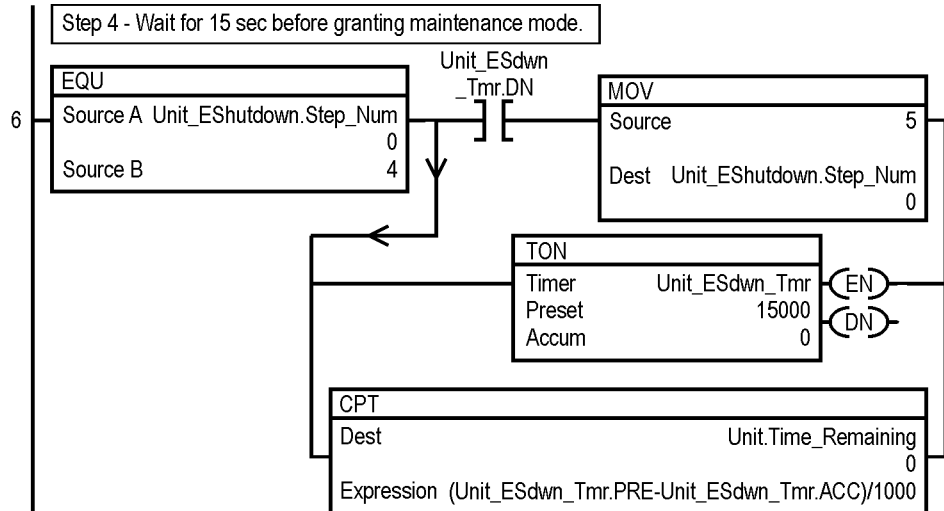
**Figure 21.** Sequence automatic starts.

The fourth section (Figure 22) is the “meat” of the Startup sequence. Each rung evaluates the transition condition. If the transition condition is satisfied, then the step number is changed to the next step. This figure shows only one transition. Rung 2 demonstrates a step that commands G-102 to open. In general, if a step commands a valve to open/close, a motor to start/stop, then the appropriate bit is latched. This bit will be unlatched by the valve/motor/etc. device code.



**Figure 22.** Step transition.

The next-to-last step in the E-Shutdown sequence (Figure 23) delays 15 seconds before transitioning to the last step. This delay allows equipment to finish closing, stopping, etc. before maintenance privilege, and therefore local control of the devices, is granted. In the E-Shutdown sequence, the steps command the devices to go to a safe state without waiting for the device to get to the safe state. Note that the time remaining for this step is also displayed, as shown in section 6.6.

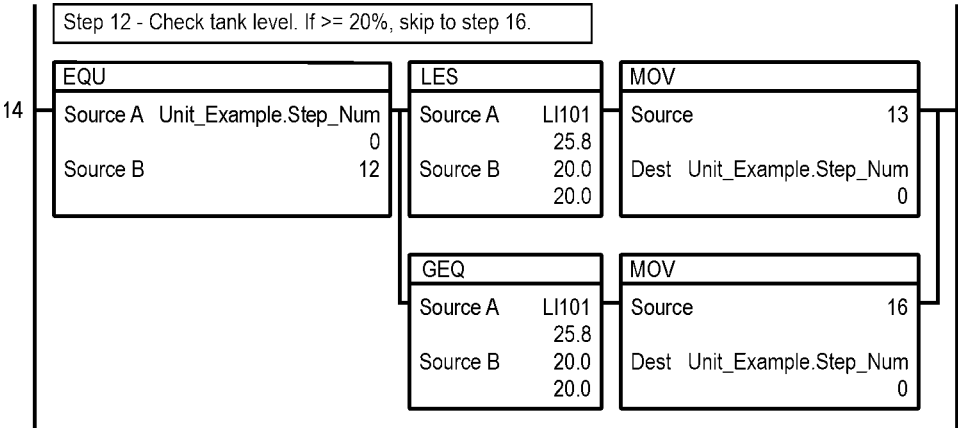


**Figure 23.** E-Shutdown next-to-last step transition.

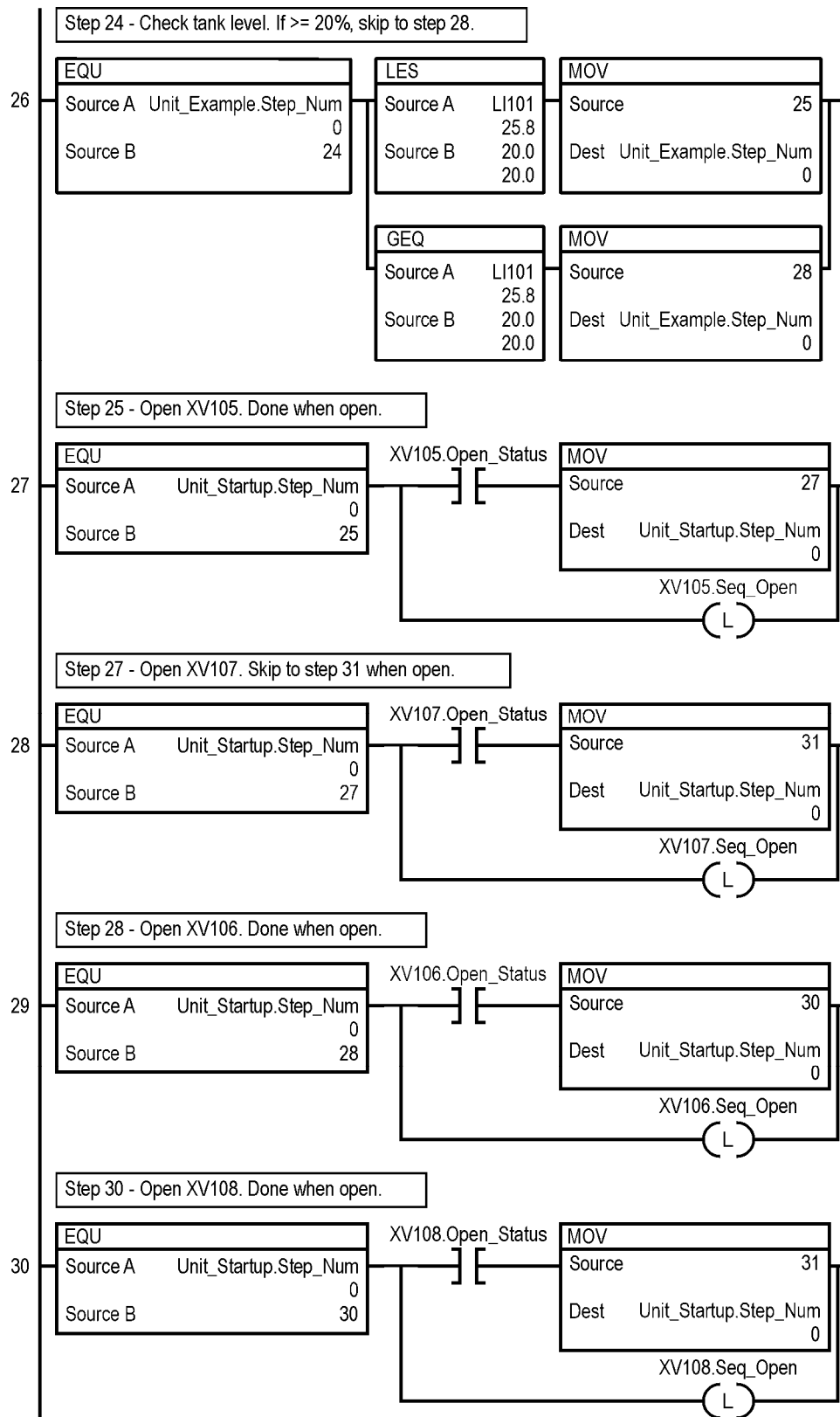
### 6.2 Branches

Branches are handled by MOVing the new step value (branch destination) into the sequence counter accumulator. The skip of Figure 3 of the “Sequence Diagram Guidelines” is implemented in the transition rungs as shown in Figure 24. If the level is < 20, then the next step is the normal one, step 13. If the level  $\geq$  20, then the step number is set to 16.

The more general branching of Figure 4 of the "Sequence Diagram Guidelines" is implemented in the transition rungs as shown in Figure 25. If the level is < 20, then the normal change to the next step, step 25 happens. If the level  $\geq$  20, then the step number is set to 28. Note that the transition out of step 27 (rung 28) sets the step number to 31, skipping over steps 28 - 30, which implement the right-hand branch. The transition out of step 30 (rung 30) is handled normally.



**Figure 24.** Code for sequence skip.

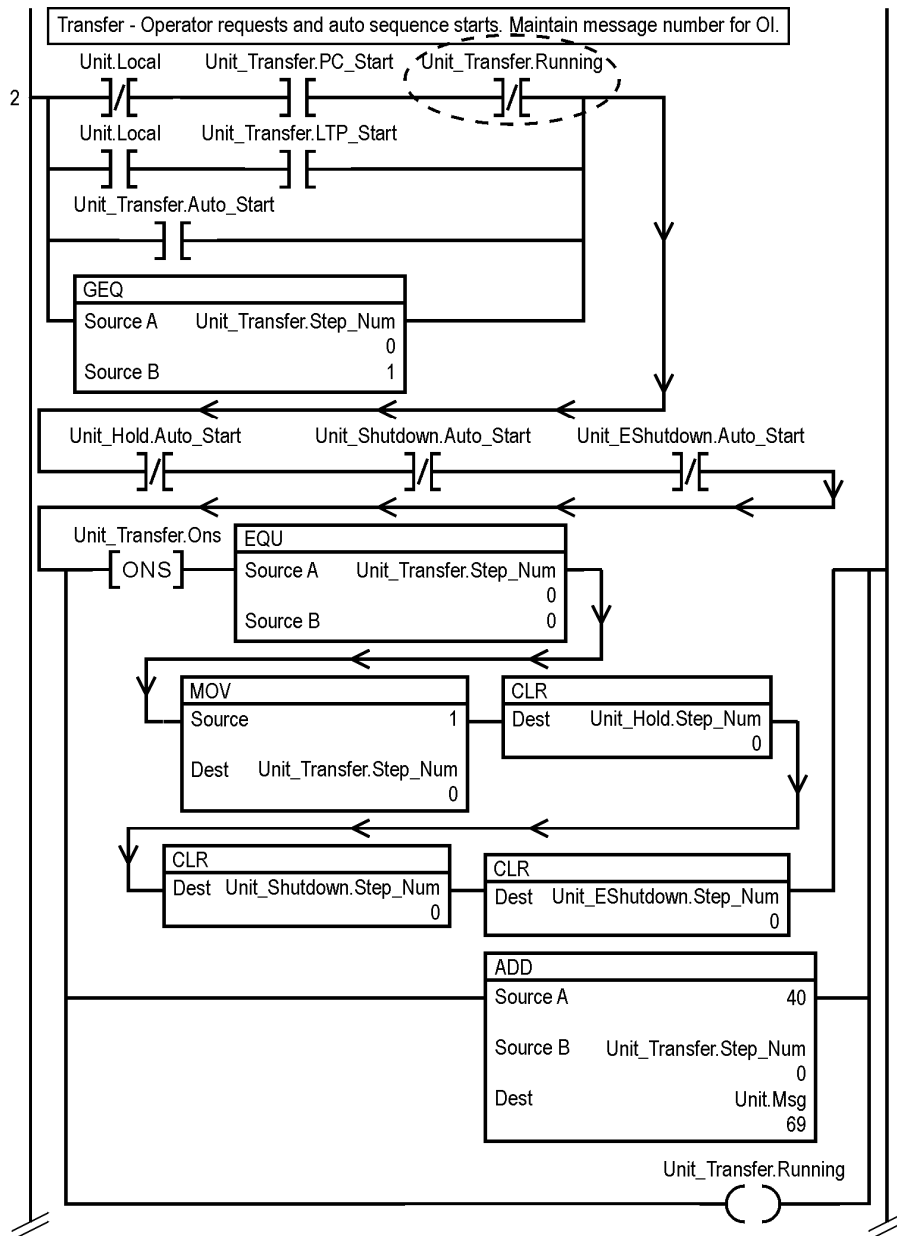


**Figure 25.** Code for sequence branching.

### 6.3 Transfers with Multiple Sources/Destinations

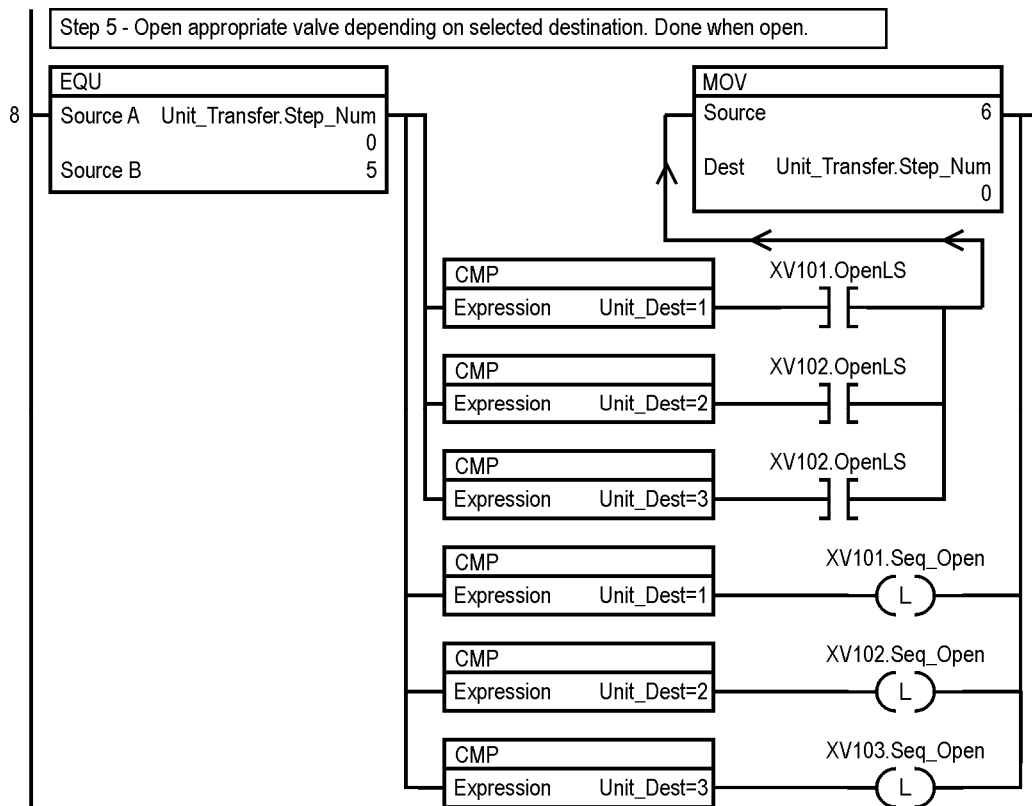
There are two ways to handle these sequences: (1) one start button for the transfer, with the source/destination selected with a separate operator response, or (2) separate start buttons, one for each source/destination. The second method is used when there is only one source or only one destination. If there are multiple sources and destinations, the first method is used.

For the first method, Figure 26 shows the rung that starts the transfer sequence of Figure 5 of the “Sequence Diagram Guidelines.” Note that if a transfer is running, another one is not permitted to start. The fifth step, shown in Figure 27 opens the appropriate valve, depending on the actual transfer. The Unit\_Dest integer value has already been set by the operator.



**Figure 26.** Start rung for common transfer sequence with one start.

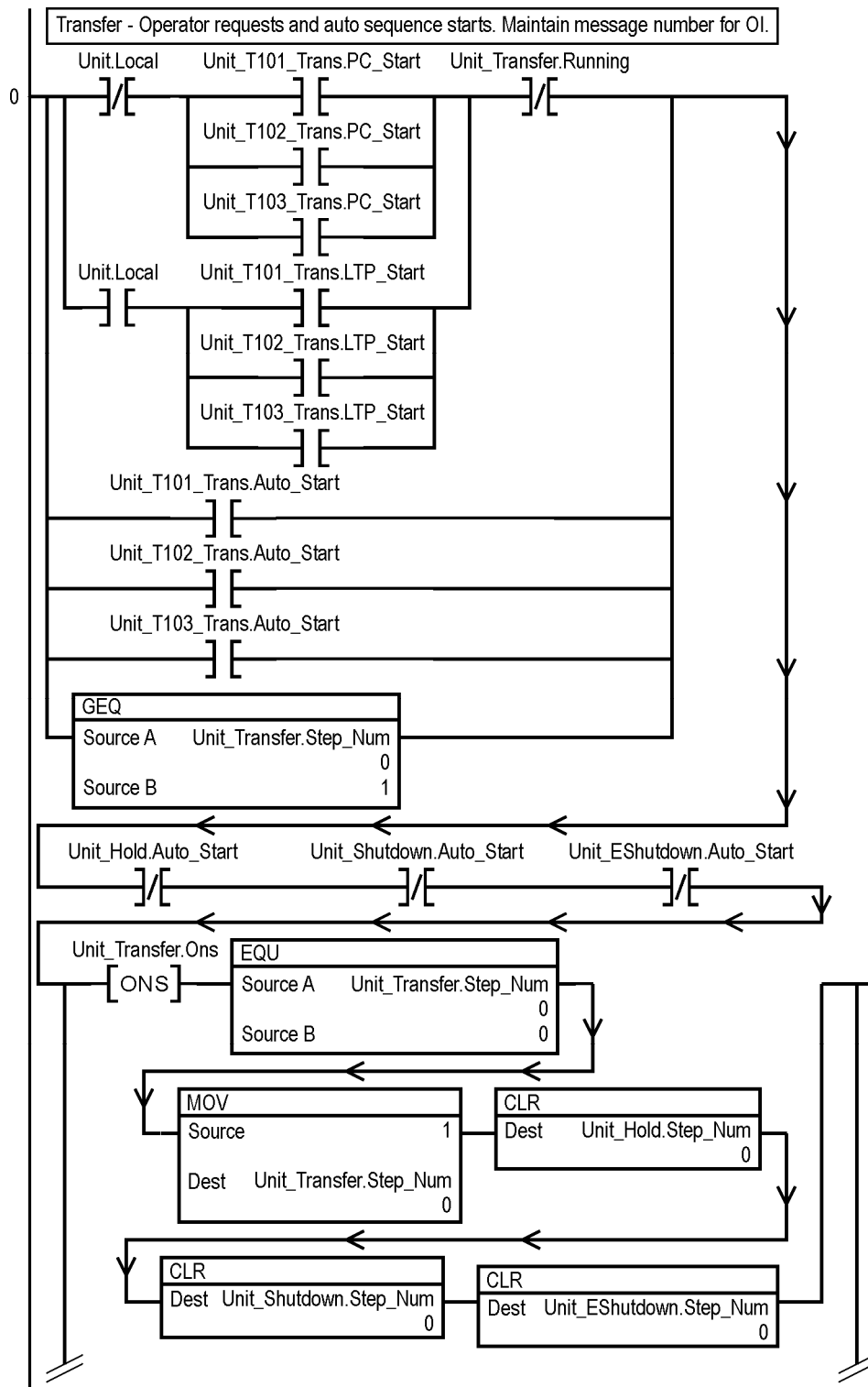




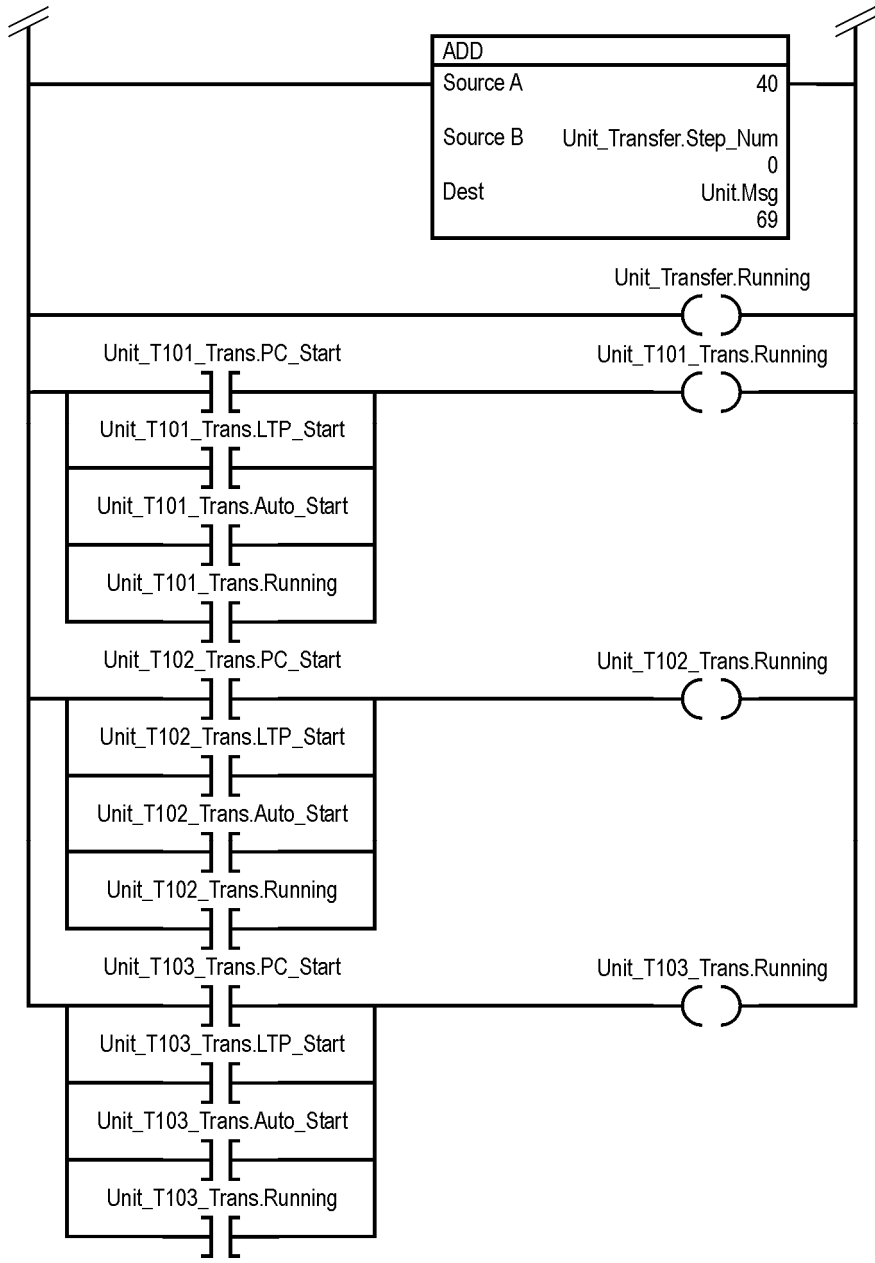
**Figure 27.** Conditional valve open for step 5 of transfer sequence with one start.

For the second method, Figure 28 shows the rung that starts the transfer sequence of Figure 6 of the “Sequence Diagram Guidelines.” Note that if a transfer is running, another one is not permitted to start. The fifth step, shown in Figure 29 opens the appropriate valve, depending on the actual transfer.

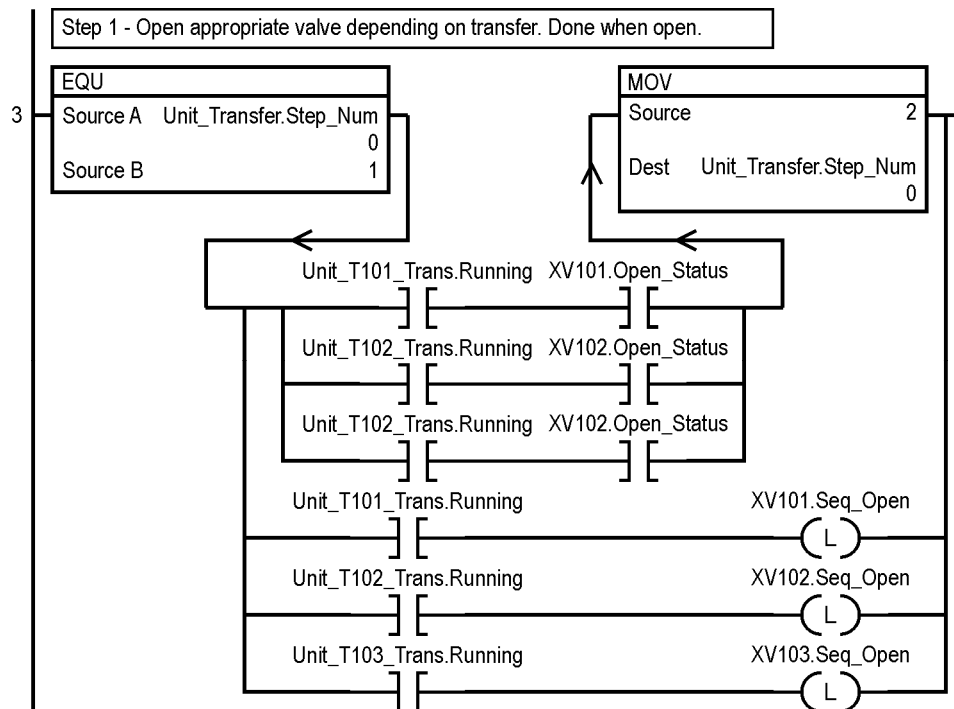
Note that for this method, the Unit\_T101\_Trans, Unit\_T102\_Trans, and Unit\_T103\_Trans tags are of type Seqxx\_Type, and must be created even though only the running and auto-starts are used. Also, logic to generate the auto-starts needs to be added.



**Figure 28.** Start rung for common transfer sequence with multiple starts.



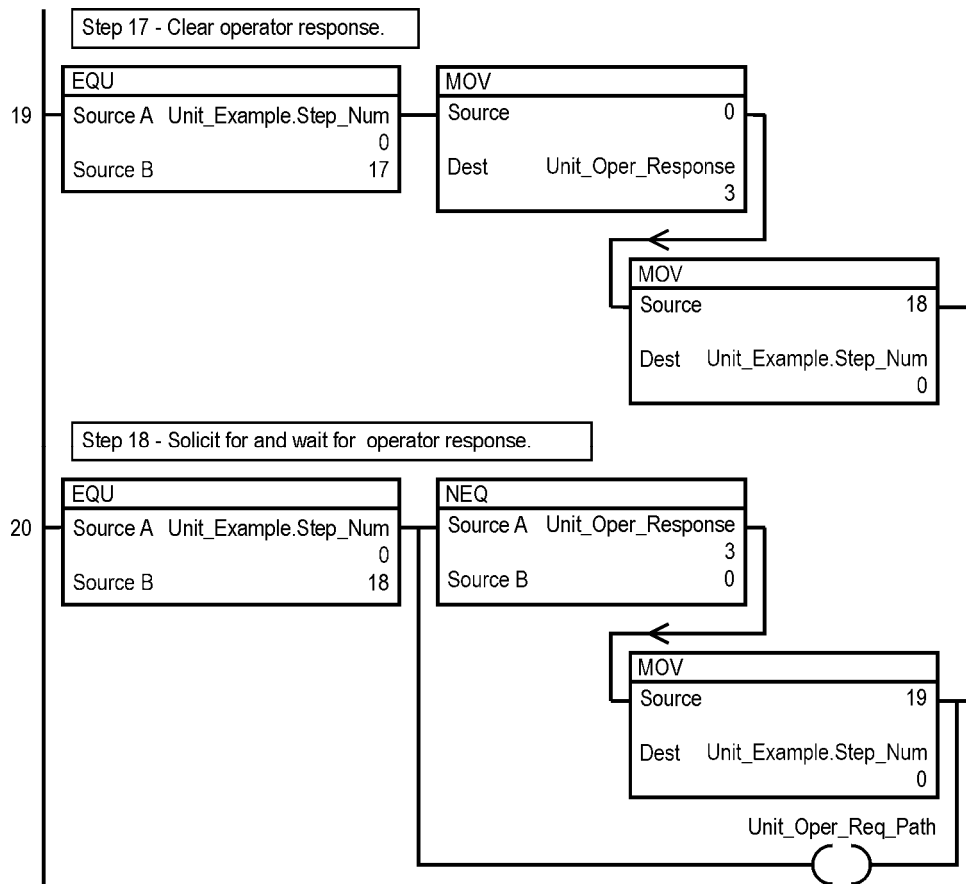
**Figure 28.** *(continued)*



**Figure 29.** Conditional valve open for step 5 of transfer sequence with multiple starts.

## 6.4 Operator Response

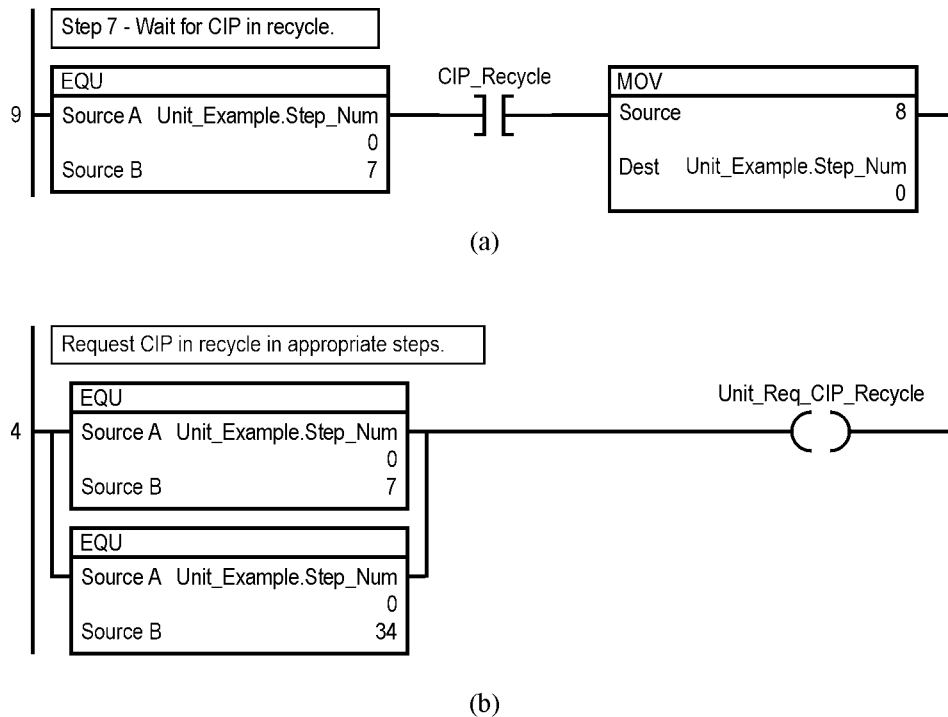
The ladder logic that implements the sequence diagram of Figure 7 of the “Sequence Diagram Guidelines” is shown in Figure 30. The sequence clears the operator-entered value and then turns on a Boolean (Unit\_Oper\_Req\_Path) that is the signal to the OI to display the prompt and the value field. Note that Unit\_Oper\_Req\_Path is not latched. When the operator enters a non-zero value, the sequence transitions to the next step.



**Figure 30.** Code for soliciting and receiving operator response.

## 6.5 Coordination With Another Unit

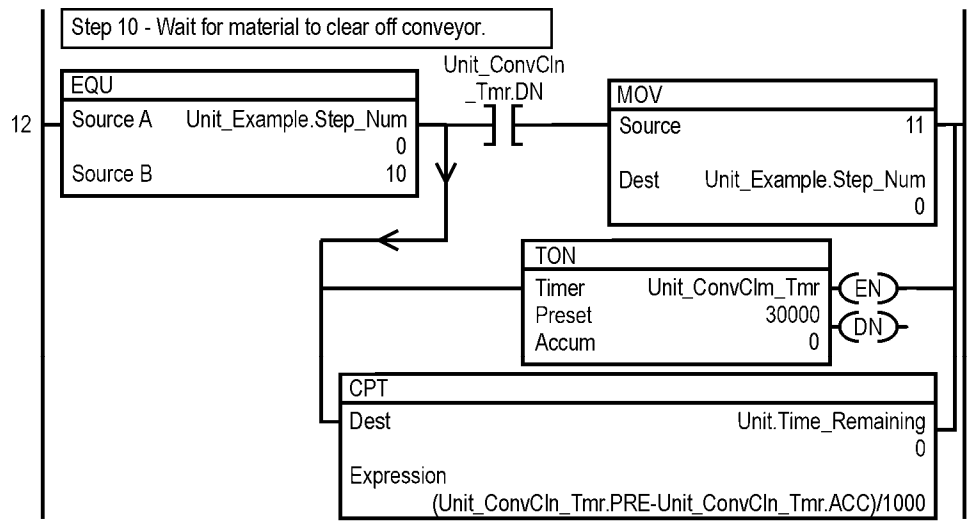
Figure 8 of the “Sequence Diagram Guidelines” is implemented as shown in Figure 31. The appropriate sequence step-in-progress bit waits for the CIP\_Recycle indication from the other unit (Figure 31a). In the Communications routine, the appropriate step numbers turn on the Boolean Unit\_Req\_CIP\_Recycle command that is communicated to the other unit (Figure 31b). The command to the other unit is not included in the transition logic so that this command can be issued by more than one step (and avoid a repeated output). Note that Unit\_Req\_CIP\_Recycle is not latched.



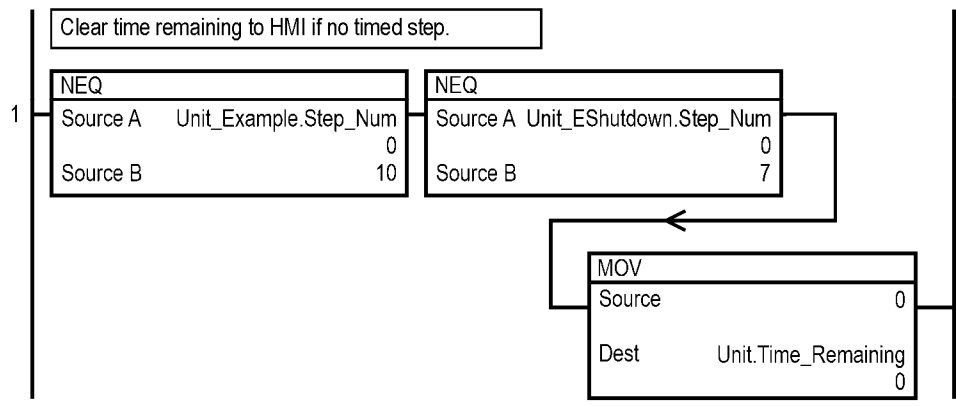
**Figure 31.** Code for sending request to another unit: (a) in sequence code; (b) in Communications routine.

## 6.6 Display Remaining Time for Timed Steps

For steps that are timed, the remaining time in the step should be calculated so that it can be displayed on the HMI as shown in Figure 32a. The destination for the calculation is the same for all sequences in the unit. Depending on the length of the time, either seconds or minutes should be calculated, but it should be consistent for the unit. The code shown here calculates time in seconds. When not in any timed steps, the remaining time should be cleared, as shown in Figure 32b. The rung that clears the time should be on the last rung in the “Unit\_Misc” routine.



(a)



(b)

**Figure 32.** Display of remaining time: (a) calculation in step transition logic; (b) clearing time.

## 7. Continuous Control

A simple PID loop, FIC101, is shown in Figure 33. FT101 is the tag for the analog input sensor (flow in this example) and FY101 is the tag for the analog output channel connected to the final control element (valve). The PID loop is executed every second. The sequences interact with the loop via the following locations:

FIC101.SWM	Latch to set mode to manual; unlatch to set mode to auto
FIC101.SO	Copy (MOV) the desired output value (0-100) to this location when the loop is in the manual mode (when FIC101.SWM latched)
FIC101.SP	Copy (MOV) the desired setpoint to this location.

The sequence commands of Figure 10 of the “Sequence Diagram Guidelines” are implemented as shown in Figure 34.

The presence of an operator manual station (Figure 10.65) is assumed, though it does not have to actually be present. The operator manual station sets the FIC101.MO bit and the FIC101\_Tiebk value. The FIC101\_Tiebk value is copied to the controller output (CV) when FIC101.MO is on. The FIC101.MO bit overrides the FIC101.SWM bit. If FIC101.MO is on, the controller is in the manual mode, regardless of the state of FIC101.SWM. In order for the sequences to control the controller mode, the FIC101.MO bit must be off.

The analog input channel configuration is set so that the scaled minimum and maximum values correspond to the range of the sensor in its units. For example, if the flow sensor range is 0 - 2000 gpm, then the scaled minimum should be "0" and the scaled maximum should be "2000." The analog output channel configuration is set so that the scaled minimum and maximum values correspond to the range of the final control element, commonly 0 - 100. For example, if the final control element is a valve, its range is 0 - 100% and the scaled minimum should be "0" and the scaled maximum should be "100."

When setting up the PID scaling parameters, make sure the Manual mode is not checked. Set the PV "Unscaled Max" and "Unscaled Min" to the same values used for the analog input channel. The "Engineering Unit Max" should be the same as the "PV Unscaled Max" and the "Engineering Unit Min" should be the same as the "PV Unscaled Min." Set the CV "Max (at 100%)" and "Min (at 0%)" to the same scaling values used for the analog output channel.

When setting up the PID configuration parameters, set the "Loop Update Time" to match the preset time of the timer used to control the PID block execution. Normally, the "CV High Limit" is 100 and the "CV Low Limit" is 0.



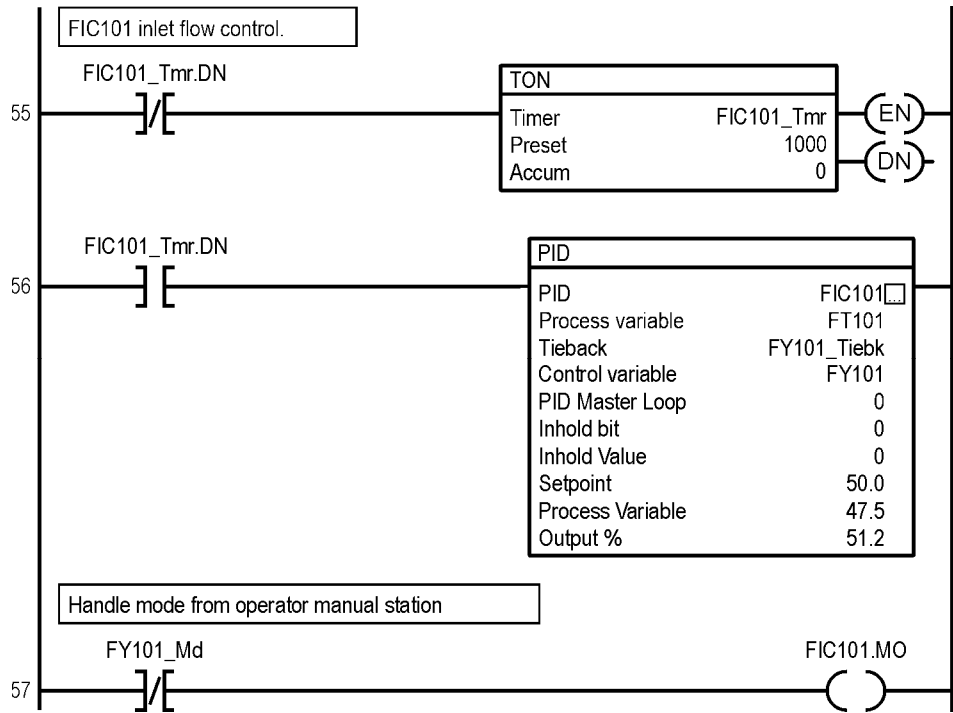
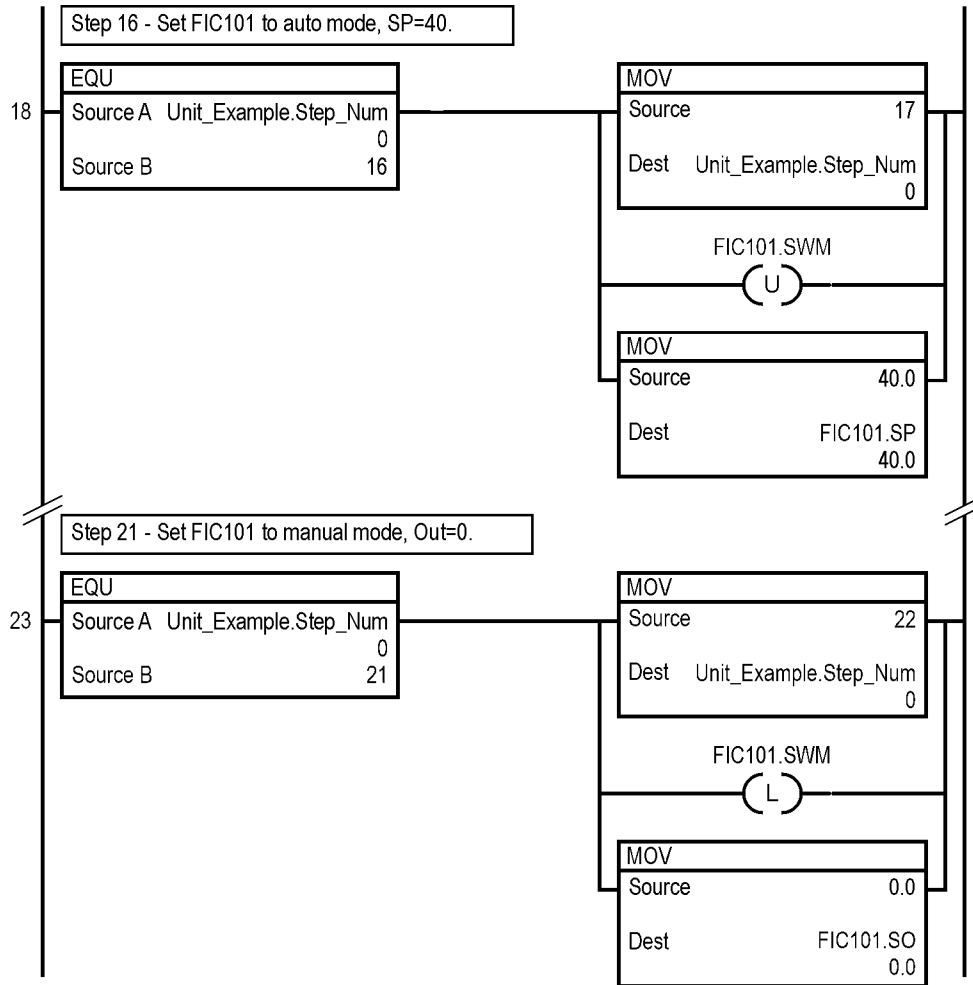
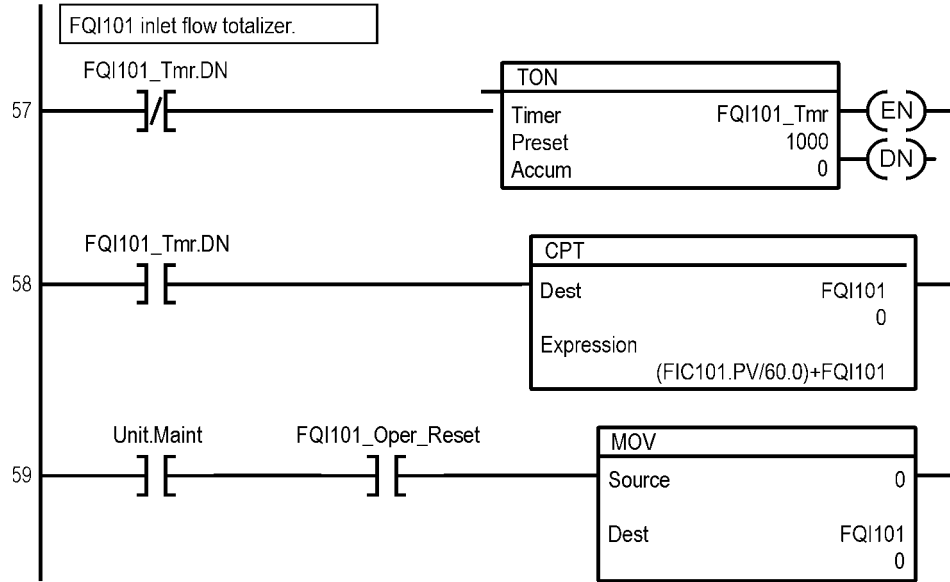


Figure 33. Simple PID loop.



**Figure 34.** PID sequence commands.

Flow totalization is used when one needs to know how much material is being transferred. The totalization can be done on a flow measurement, or may augment a PID controller. The flow measurement is mathematically integrated in order to obtain the totalization. Ladder logic for a totalizer, FQI101, is shown in Figure 35. In this case, the flow from the PID loop of Figure 33 (FIC101.PV) is being totalized. The totalization is performed with rectangular integration, executed every second. The divide by 60 converts the flow in  $x$  per second to  $x$  per minute for the proper units of the FQI value. Sequences reset the totalizer by moving a zero into the totalizer (for example, FQI101=0.0). The totalizer may be reset by the operator through the FQI101\_Oper\_Reset variable, but only when the unit is in the maintenance mode.



**Figure 35.** Flow totalization.

## 8. Analog Inputs and Outputs

The analog input channel configuration is usually set so that the module does the scaling into the engineering units of the sensor. The “Comm Format” is set to “Float Data” and the low and high engineering units are set to the limits of the sensor values. For example, if a flow sensor is connected to a 4-20 mA analog input and the flow sensor is calibrated to measure flow in the range of 0.0 - 45 gpm, then the channel configuration parameters would be set to:

Input Range:	0 to 20 mA		
High Signal:	20	High Engineering:	45.0
Low Signal:	4	Low Engineering:	0.0

In a similar manner, the analog output channel configuration is usually set so that the module does the de-scaling from the engineering units of the actuator. The “Comm Format” is set to “Float Data” and the low and high engineering units are set to the limits of the actuator values. For example, if a variable frequency drive control signal is connected to a 4-20 mA analog output and the drive signal specifies the frequency in the range of 0.0 - 60 Hz, then the channel configuration parameters would be set to:

Output Range:	0 to 20 mA		
High Signal:	20	High Engineering:	60.0
Low Signal:	4	Low Engineering:	0.0

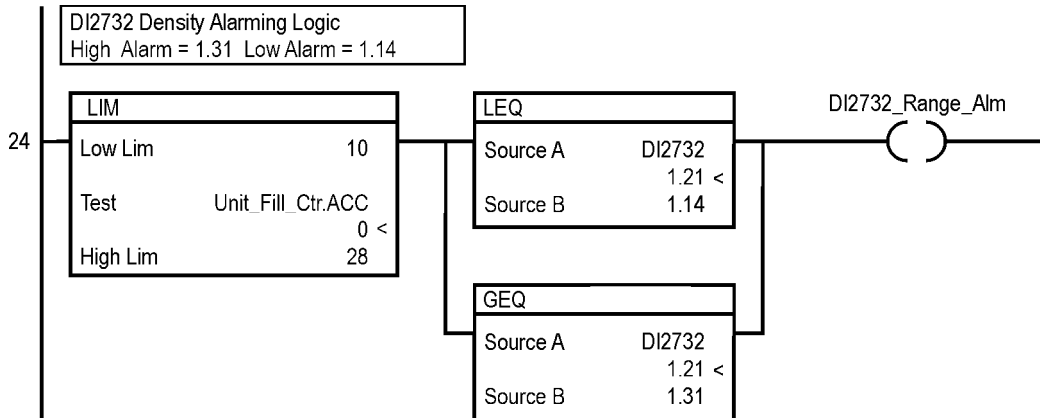
## 9. Alarms

This section deals with generation of alarms by the PLC. Alarm display and management is handled by the HMI package. The PLC evaluates and masks all alarms based upon current process conditions. The HMI monitors the resulting alarms and generates the alarm displays. Alarms may also be displayed on the process graphic displays. Certain alarms may cause an alarm horn to sound. In this case, the operator must press a “horn silence” or an “acknowledge” button to silence the horn. In this case, a PLC is responsible for generating the signal driving the horn and for handling the button to silence the horn.

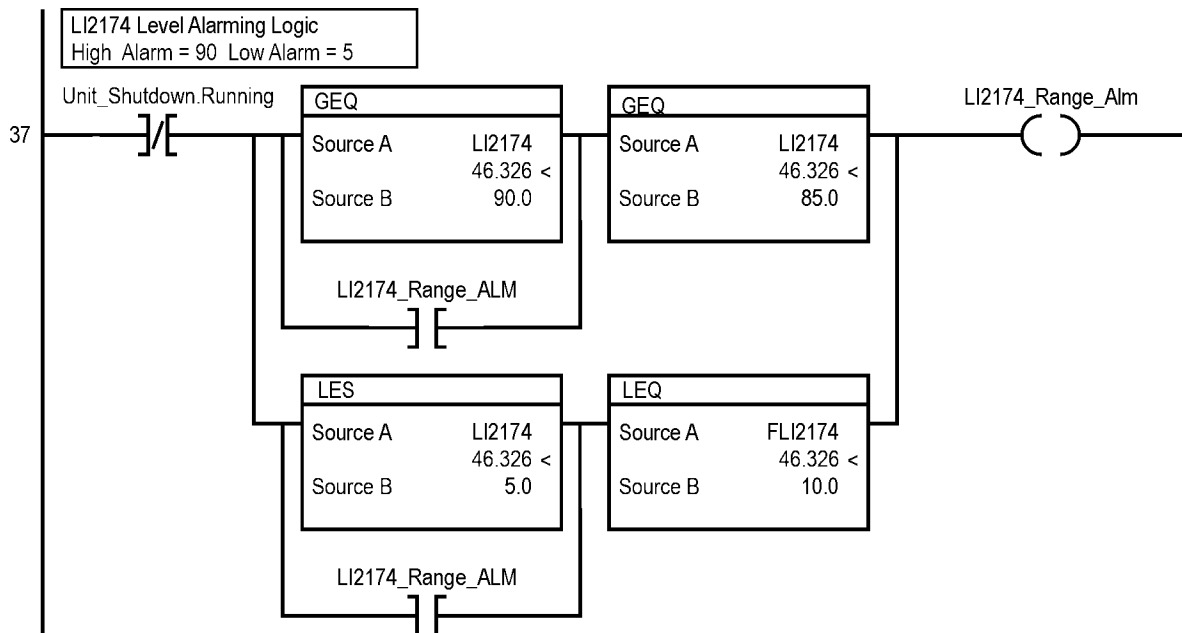
Alarms due to a motor or discrete valve failure are described in section 5 of this document. Other types of alarms are described in this section. In general, an alarm should only be generated if it is an actual alarm. For example, a pressure sensor on the discharge side of a pump will have a low pressure alarm. But the alarm is not generated unless the pump has been running long enough to pressurize the discharge piping.

The density alarm shown in Figure 36 is generated only when the tank fill sequence is in a certain range of step numbers. The density alarm has an upper and lower alarm limit. In certain cases, it may be desirable to have separate alarms for the density above an alarm limit and for the density below an alarm limit. Figure 37 shows alarm logic that is more complicated than the density alarm logic. The level alarm is only active when the tank is not in the shutdown

sequence. The logic also includes deadbands for the upper and lower alarm limits to prevent alarm “jitter,” where the alarm transitions rapidly between the on and off states as the level bounces around the alarm point (levels are rarely constant). The level alarm in Figure 37 will be on when the level reaches 90% and will stay in alarm until the level falls below 85%. Also, the level alarm will engage when the level falls below 5% and will not reset until the level rises above 10%.

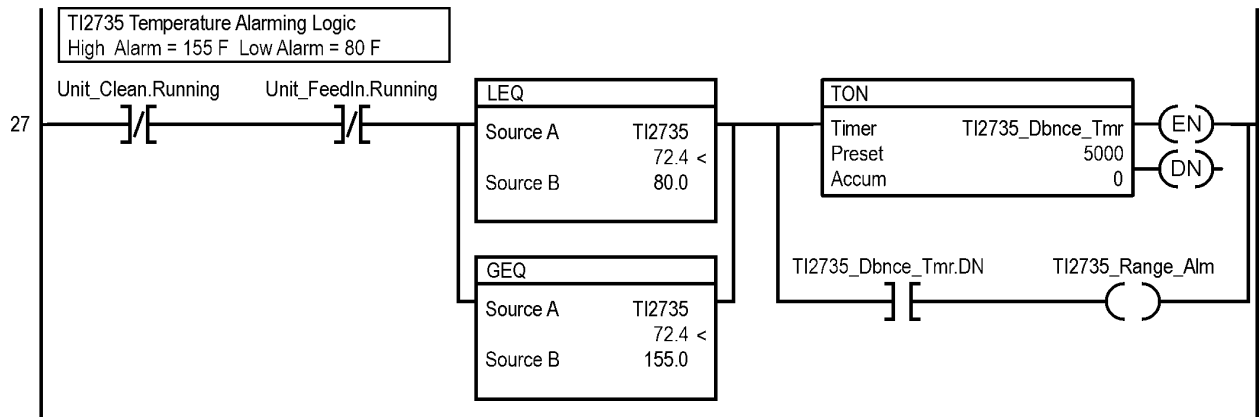


**Figure 36.** Density range alarm logic.



**Figure 37.** Level high and low alarm logic with deadbands.

The temperature alarm in Figure 38 checks both the upper and lower limits when the device is not in the clean and feed in sequences. The alarm logic uses a delay timer to prevent alarm jitter as the temperature crosses the alarm limits. After the delay has timed out, the alarm will turn on.



**Figure 38.** Temperature alarm logic with debounce timer.

## 10. Tag Conventions

Supporting installed projects becomes easier with increased software structure and convention. In addition to the structure and convention discussed above, adoption of naming conventions eases project maintenance.

PLC tag names shall describe the functionality of the device and have the device tag number included. However, note that if the equipment designation has a hyphen in it, the hyphen must be omitted in the PLC tag. The PLC descriptors have five description fields. To aid in searching by tag number in the PLC program, the first description field shall be the device tag number and the other four description fields shall be the tag name. For example, a pump, with a tag number P-2700, used in a cooling tower operation in the reactor area shall be tagged:

Description 1	P-2700
Description 2	Reactor
Description 3	Cooling Tower
Description 4	Pump A
Description 5	

Other tag naming conventions:

### Sequences

Unit_Tag + "_" + Seq_Name + ".Step_Num"	Sequence step number
Unit_Tag + "_" + Seq_Name + ".Auto_Start"	Sequence auto start bit
Unit_Tag + "_" + Seq_Name + ".Auto_Ons"	Sequence auto-start one-shot storage bit
Unit_Tag + "_" + Seq_Name + ".Req_Tmr"	Sequence auto-start hold timer
Unit_Tag + "_" + Seq_Name + ".Running"	Sequence running indication
Unit_Tag + "_" + Seq_Name + ".Ons"	Sequence one-shot storage to start
Unit_Tag + "_" + Seq_Name + ".PC_Start"	Sequence start request from PC (remote)
Unit_Tag + "_" + Seq_Name + ".LTP_Start"	Sequence start request from LTP (local)
Unit_Tag + ".Msg"	For a group of sequences, holds the message number
Unit_Tag + ".Local"	For a group of sequences, <b>on</b> when controlled from LTP; <b>off</b> when controlled from remote PC.
Unit_Tag + ".Maint"	For a group of sequences, <b>on</b> when in maintenance privilege: devices started/stopped at OI.
Unit_Tag + ".Man_DevNum"	For equipment group, when in maintenance privilege: number of device started/stopped at OI.

## Motors

### Commands:

Equip_Tag + ".Seq_Start"	Start motor with sequence step
Equip_Tag + ".Seq_Stop"	Stop motor with sequence step
Unit_Tag + ".Man_StartOpen"	Start motor by command from operator screen
Unit_Tag + ".Man_StopClose"	Stop motor by command from operator screen

### Indications:

Equip_Tag + ".Run_Status"	Motor running indication
---------------------------	--------------------------

### Physical Inputs:

Equip_Tag + "_Aux"	Auxiliary contact ( <b>on</b> when motor running)
Equip_Tag + "_OL"	Overload trip ( <b>on</b> when motor overloaded)
Equip_Tag + "_HOA"	HOA switch auto contact ( <b>on</b> when in auto)

### Physical Outputs:

Equip_Tag + "_Start"	Starter ( <b>on</b> to start/run motor)
----------------------	---

## Conveyors

### Commands:

Equip_Tag + ".Seq_Start"	Start conveyor with sequence step
Equip_Tag + ".Seq_Stop"	Stop conveyor with sequence step
Unit_Tag + ".Man_StartOpen"	Start conveyor by command from operator screen
Unit_Tag + ".Man_StopClose"	Stop conveyor by command from operator screen

### Indications:

Equip_Tag + ".Run_Status"	Conveyor running indication
---------------------------	-----------------------------

### Physical Inputs:

Equip_Tag + "_Aux"	Auxiliary contact ( <b>on</b> when motor running)
Equip_Tag + "_OL"	Overload trip ( <b>on</b> when motor overloaded)
Equip_Tag + "_HOA"	HOA switch auto contact ( <b>on</b> when in auto)
SS_Equip_Tag	Speed switch ( <b>on</b> when conveyor running)

### Physical Outputs:

Equip_Tag + "_Start"	Starter ( <b>on</b> to start/run motor)
----------------------	---

## Valves

### Commands:

Equip_Tag + ".Seq_Open"	Open valve with sequence step
Equip_Tag + ".Seq_Close"	Close valve with sequence step
Unit_Tag + ".Man_StartOpen"	Open valve by command from operator screen
Unit_Tag + ".Man_StopClose"	Close valve by command from operator screen

### Indications:

Equip_Tag + ".Open_Status"	Open indication for valve
Equip_Tag + ".Close_Status"	Closed indication for valve

### Physical Inputs:

Equip_Tag + "_OpenLS"	Valve-open limit switch ( <b>on</b> when valve fully open)
Equip_Tag + "_CloseLS"	Valve-closed limit switch ( <b>on</b> when valve fully closed)

### Physical Output:

Equip_Tag + "_Vlv_Sol"	Valve solenoid ( <b>on</b> to open the valve)
------------------------	---

## Slide Gates



**Commands:**

Equip_Tag + ".Seq_Open"	Open slide gate with sequence step
Equip_Tag + ".Seq_Clos"	Close slide gate with sequence step
Unit_Tag + ".Man_StartOpen"	Open gate by command from operator screen
Unit_Tag + ".Man_StopClose"	Close gate by command from operator screen

**Indications:**

Equip_Tag + ".Open_Status"	Open indication for gate
Equip_Tag + ".Close_Status"	Closed indication for gate

**Physical Inputs:**

Equip_Tag + "_Open_Aux"	Open auxiliary contact ( <b>on</b> when motor runs to open)
Equip_Tag + "_Close_Aux"	Close auxiliary contact ( <b>on</b> when motor runs to close)
Equip_Tag + "_OL"	Overload trip ( <b>on</b> when motor overloaded)
Equip_Tag + "_HOA"	HOA switch auto contact ( <b>on</b> when in auto)
ZSO_Equip_Tag	Gate-open limit switch ( <b>on</b> when gate open)
ZSC_Equip_Tag	Gate-closed limit switch ( <b>on</b> when gate closed)

**Physical Outputs:**

Equip_Tag + "_Start_Open"	Open starter ( <b>on</b> to start/run motor to open)
Equip_Tag + "_Start_Close"	Close starter ( <b>on</b> to start/run motor to close)

**Flop Gates****Commands:**

Equip_Tag + ".Seq_Left"	Move flop gate to divert left with sequence step
Equip_Tag + ".Seq_Right"	Move flop gate to divert right with sequence step
Unit_Tag + ".Man_StartOpen"	Move gate left by command from operator screen
Unit_Tag + ".Man_StopClose"	Move gate right by command from operator screen

**Indications:**

Equip_Tag + ".Left_Status"	Left indication for gate
Equip_Tag + ".Right_Status"	Right indication for gate

**Physical Inputs:**

ZSL_Equip_Tag	Left-path limit switch ( <b>on</b> when diverted left)
ZSR_Equip_Tag	Right-path limit switch ( <b>on</b> when diverted right)
Equip_Tag + "_HOA"	HOA switch auto contact ( <b>on</b> when in auto)

**Physical Outputs:**

Equip_Tag + "_Sol_Left"	Left path solenoid ( <b>on</b> to move gate to divert left)
Equip_Tag + "_Sol_Right"	Right path solenoid ( <b>on</b> to move gate to divert right)

**PID Loops****Commands:**

Loop_Tag + ".SWM"	Latch: loop set to manual mode
	Unlatch: loop set to auto mode
Loop_Tag + ".SO"	Set loop output in manual mode
Loop_Tag + ".SP"	Set loop setpoint

**Indications:****Totalizers**

**Indications:**

Tag

Totalization

**Notes:**

Equipment tags are all upper case and cannot have hyphens (for example P-1000 becomes P1000)

Use indicator tags for sequencer code (LI2000 instead of LT2000)

**11. Simulation Logic**

Any simulation logic should simulate the physical inputs of the following items, based on the device:

Motor – Aux contact, overload, HOA switch

Conveyor – Aux contact, overload, HOA switch, speed switch

Discrete valve – Open LS, close LS

Slide gate – Open aux contact, close aux contact, overload, HOA switch, open LS, close LS

Flop gate – Left LS, right LS, HOA switch

PID loop – Copy output to measurement. If the operator manual station is absent, the .MO Boolean must be off.

The simulation logic must be in a separate program named “Simulation”. Tag names for any timers or other tags needed for simulation must be simulation program tags.

**12. Revision History**

Rev 6.0 2023Aug19 Rev 5.7 revised to use move-based sequencer.