

S7-1500 Step7 TIA Portal Code Design Guidelines

Table of Contents

1. CONTROL SYSTEM FUNCTIONAL LAYOUT.....	3
2. AREA CONTROL AND SECURITY.....	3
3. PROGRAM STRUCTURE.....	4
4. PLC/PLC AND OI/PLC COMMUNICATIONS.....	7
5. FIELD EQUIPMENT CONTROL	9
5.1 MOTOR DEVICE CONTROL.....	9
5.2 CONVEYOR MOTOR DEVICE CONTROL.....	15
5.3 DISCRETE VALVE DEVICE CONTROL.....	19
5.4 SLIDE GATE CONTROL	23
5.5 FLOP GATE CONTROL	27
6. SEQUENCED CONTROL.....	32
6.1 SEQUENCE CODE	32
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
7. CONTINUOUS CONTROL	47
8. ANALOG INPUTS AND OUTPUTS.....	51
9. ALARMS.....	52
10. VARIABLE NAME CONVENTIONS.....	53
11. REVISION HISTORY	56

1. Control System Functional Layout

The control system is centered on Siemens' S7-1500 PLCs programmed with Step7 TIA Portal which provide the discrete device control, loop device control, sequence control, and alarm evaluation. InTouch or WinCC running on PCs provides the operator interface (OI). The 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 PLC's uses Ethernet. The OI communicates over Ethernet.

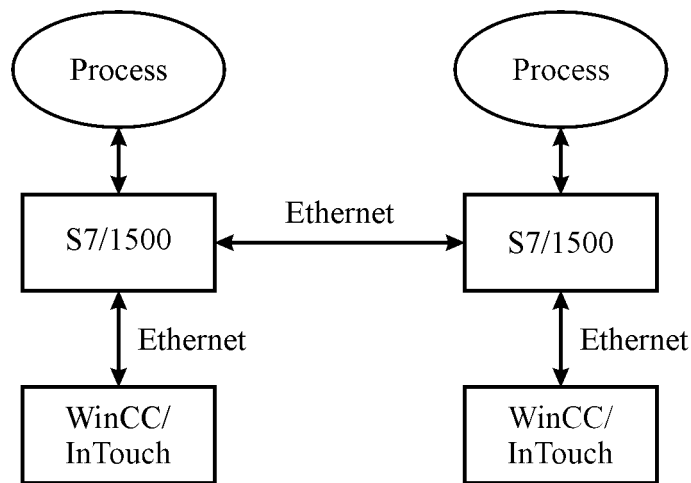


Figure 1. Overall view of control system.

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 program is organized into function blocks. The blocks associated with a unit are organized into programs groups.

Program Blocks

- Main [OB1]
- Startup [OB100]
- Duplicate_Ins [FC40]
- Comms [FB30] (DB30 is instance data block)
- Simulate [FB40] (DB40 is instance data block)
- Comms_DB [DB30]
- Simulate_DB [DB40]
- Unit1_Export_Data [DB1] (export info to other units)
- Unit1_Import_Data [DB2] (import info from other units)
- Unit1 (program group)
 - Unit1_000Main [FB100]
 - Unit1_000Startup [FB101] (Note: starting Unit1.Msg number part of name)
 - Unit1_040Operate [FB102]
 - Unit1_060Hold [FB112]
 - Unit1_080Shutdown [FB113]
 - Unit1_100EShutdown [FB114]
 - Unit1_990Misc [FB115]
 - Unit1_991Abnormal [FB116]
 - Unit1_Analog_Scaling [FB120]
 - Unit1_Gates [FB121]
 - Unit1_Motors [FB122]
 - Unit1_PIDLoops [FB123]
 - Unit1_Valves [FB124]
 - Unit1 [DB100] (of Unit_Type, holds unit-level tags)
 - Unit1_DB [DB101] (instance data block for Unit1_000Main)
 - Unit1G [DB103] (contains data for Unit1's flop gate devices)
 - Unit1M [DB104] (contains data for Unit1's motors and slide gates)
 - Unit1P [DB105] (contains data for Unit1's PID loops)
 - Unit1S [DB102] (contains data for Unit1's sequences)
 - Unit1V [DB106] (contains data for Unit1's discrete valves)
- Unit2 (program group)

```

Unit2_000Main [FB200]
Unit2_000Startup [FB201] (Note: starting Unit2.Msg number part of name)
Unit2_040Operate [FB202]
... (continue for Unit2)
Unit3 (program group)
Unit3_000Main [FB300]
Unit3_000Startup [FB301] (Note: starting Unit2.Msg number part of name)
Unit3_040Operate [FB302]
... (continue for Unit3)
zzDevice_Control (program group)
Gate_Flop [FB1099]
Gate_Slide [FB1098]
Motor_Conv [FB1096]
Motor_Std [FB1095]
Valve_Disc [FB1097]

```

Note that the blocks associated with a unit are organized into a program group and each unit has a block of 100 block numbers. Unit1 has FB's and DB's numbered from 100 to 199. Unit2 has FB's and DB's numbered from 200 to 299. In the FB and DB names above, "Unit1", "Unit2" or "Unit3" is replaced by the shortened name of the unit, for example, "TWet_Unl", "Blend", "React_1".

The routines before the first program group handle global functions. The Main OB1 basically calls the following FB/FC's:

```

Comms
Duplicate_Ins
Simulate
Unit1_000Main
Unit2_000Main
Unit3_000Main

```

The "Comms" FB contains the communication-related function blocks. The Simulate FB handles the process simulation and the Duplicate_Ins FC transfers the real I/O to the simulated input tags, as shown in text section 21.5.5.

For a particular unit, the "Unitx_000Main" calls the blocks associated with a unit. For example, for a unit named "Unit1", the FB's are

```

Unit1_000Startup
Unit1_040Operate
Unit1_060Hold
Unit1_080Shutdown
Unit1_100EShutdown
Unit1_990Misc
Unit1_991Abnormal
Unit1_Analog_Scaling
Unit1_Gates
Unit1_Motors

```

Unit1_PIDLoops

Unit1_Valves

Scaling for analog ins/outs is contained in the "Unit1_Analog_Scaling" FB and PID loops are contained in the "Unit1_PIDLoops" FB. All of the control for the motor devices is combined into "Unit1_Motors " FB. All of the valve device control is combined into the "Unit1_Valves " FB. All of the slide and pneumatic gate device control is combined into the "Unit1_Gates" FB. The code for each sequence associated with a unit is contained in one function block.

The following FBs handle device control:

FB1095	Motor_Std	Standard motor control
FB1096	Motor_Conv	Conveyor motor control (motor + speed switch)
FB1097	Valve_Disc	On/off valve
FB1098	Gate_Slide	Slide gate controlled by reversing motor
FB1099	Gate_Flop	Gate controlled by pneumatic cylinder

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

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

The "unit name" DB tag is of Unit_Type whose fields are as follows:

<u>Name</u>	<u>Type</u>	<u>Description</u>
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 device data (not the instance DB's for the device FB's) is organized by the unit and contained in the following data blocks:

<u>DB #</u>	<u>Name</u>	<u>Description</u>
x05	UnitxP	PID_Loops
x04	UnitxM	Motors
x06	UnitxV	Valves
x03	UnitxG	Gates

Where "x" is the block of 100 allocated to the unit and "Unitx" is replaced by the shortened name of the unit, for example, "TWet_Unl", "Blend", "React_1".

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.

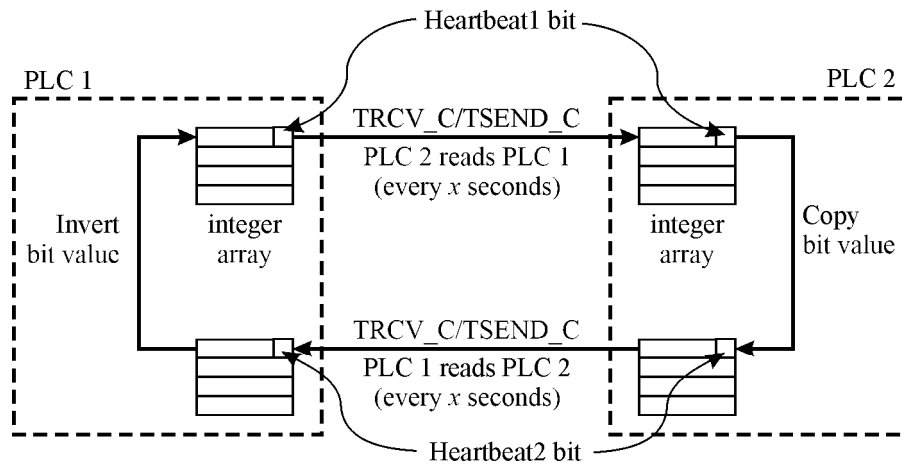
4. PLC/PLC and OI/PLC Communications

Exchanging information between processors is done with TSEND_C/TRCV blocks for S7-1200/1500 processors and AG_RECV/AG_SEND blocks for S7-300/400 processors. In general, a processor that needs information from another processor reads the information 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.

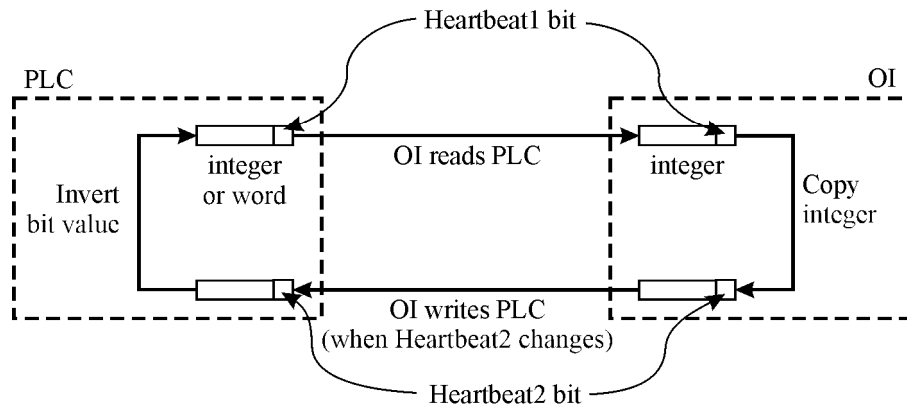
Communication between the PLCs and between the PLC and OI are often vital, hence, programming, which is called the Heartbeat logic, is used to verify the communications bridge. The Heartbeat programming is located in both the PLCs and the OI. If either the PLC or PC fails to set their given bit within the 30 seconds, an alarm will occur and be indicated at the PLC and the OI panels. When communications cease for more than 30 seconds, certain unit operations may be initiated.

The general structure of PLC-to-PLC heartbeat logic 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 READ_REG read period. For example, if each TRCV 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 30 seconds or **off** for 30 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-OI heartbeat logic is shown in Figure 2b 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)

Figure 2. Heartbeat concept: (a) PLC-to-PLC; (b) 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 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

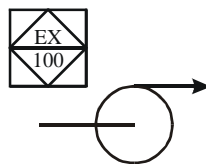


Figure 3. P&ID symbol for motor device control associated with pump.

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 function block 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 inputs and outputs of the Motor_Std block are connected to the appropriate variables. The appropriate sequence step latches the EX100.Seq_Start or EX100.Seq_Stop variable to control the motor. These two variables are always reset at the last part of the block code.

The parameters of the Motor_Std FB are as follows:

<u>Name</u>	<u>Data type</u>	<u>Default value</u>	<u>Description</u>
Input			
Aux_Contact	Bool	false	Auxiliary contact
OL_Trip	Bool	false	Overload trip
HOA_Switch	Bool	true	HOA switch auto contact
Alarm_Reset	Bool	false	Resets alarm
Maint	Bool	true	Maintenance privilege
Man_Start	Bool	false	Manual start
Man_Stop	Bool	false	Manual stop
Select_Dev_Num	Int	0	Device selected to start/stop
This_Dev_Num	Int	0	Number for this device
Output			
Starter	Bool		Motor starter contact
InOut			
Tag	Motor_Type		Device tag
Static			
Start_Req	Bool	false	Start request
Stop_Req	Bool	false	Stop request
Mtr_Strtr	Bool	false	Internal copy of output
Alw_On	Bool	true	Always on for last network
Fail_Tmr	TON		Timer for aux fail alarm

The motor is in the manual control state when the unit is in the maintenance state. 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_StartOpen) or stop (Man_StopClose) 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.

This motor control FB is implemented in ladder logic as in Figure 5. The first network controls an internal FB tag that is copied to the physical output. Note that the motor is turned **off** on the first scan of the ladder, 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 network so that an overload immediately turns **off** the motor, rather than waiting one scan for the Tag.Any_Fail. The start and stop internal coils to drive the motor contactor are determined by the second and third networks.

In the second and third networks, the operator generates the start and stop commands when the control is in the maintenance state (Maint FB input). The operator is at a local touch-screen panel that has a common start and stop button. 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 sets the Tag.Seq_Start or Tag.Seq_Stop internal coil to control the motor. These two variables are always reset at the last part of the block code. 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 sequence logic and leaving the device logic untouched.

The fourth network copies the internal output to the physical output that drives the motor contactor (or motor starter). An internal FB tag must be used as the output of the first network in order to use this tag as a contact and avoid a warning message.

The fifth network delays checking for the auxiliary fail alarm until 20 seconds after the motor is started. Note that the timer is part of the static data. The auxiliary fail alarm (Tag.Aux_Fail) in the sixth network 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 seventh and eighth networks generate the overload fail alarm and the indication that the HOA switch is not in the auto position. The ninth network generates one summary failure alarm indication. All of these alarms are generally logged by the operator interface and appear on an alarm summary screen. Network 10 resets the auxiliary failure alarm so that another start attempt is allowed. As for the maintenance/operate control privilege, the reset may be for a group of equipment, rather than specific to each device. The eleventh network generates the running indication. Network 12 resets the sequential control commands. The last network sets the ENO output.

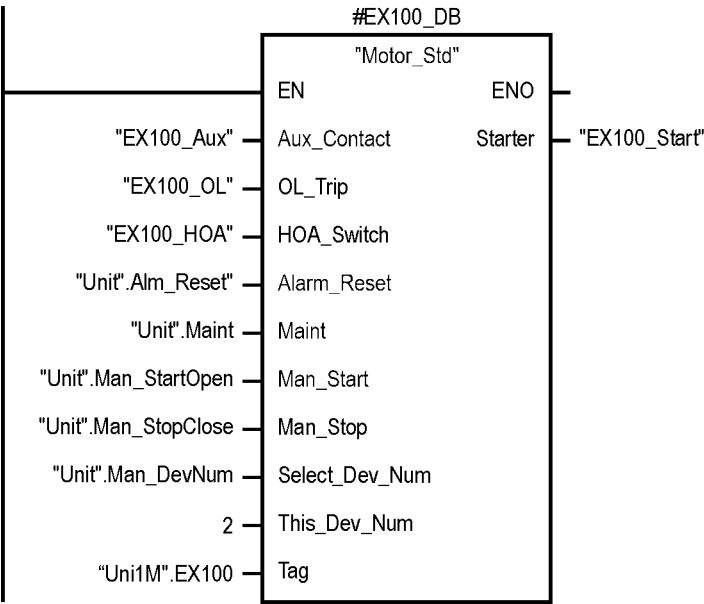


Figure 4. Motor (pump) device control with Motor_Std FB.

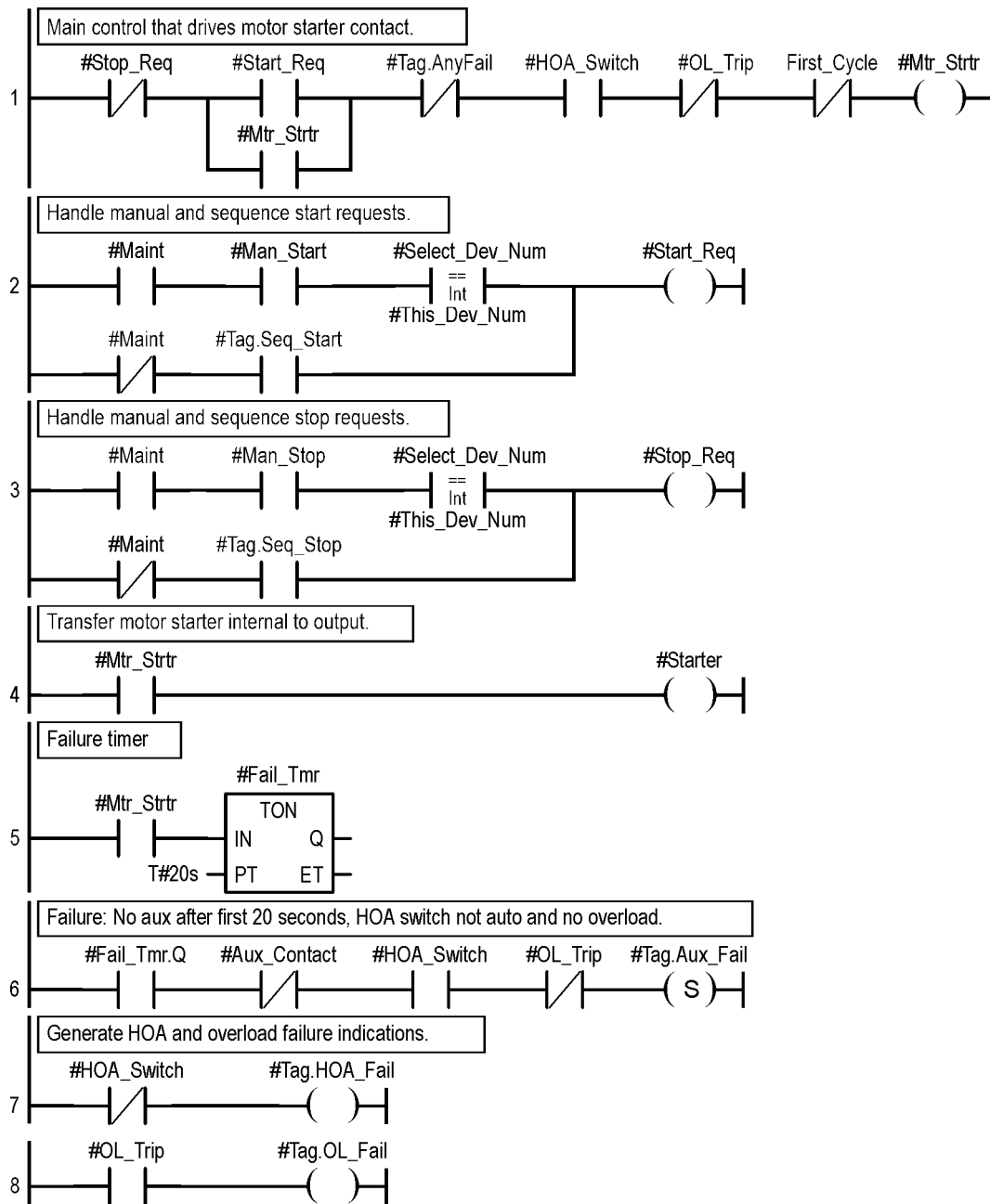


Figure 5. Motor_Std FB implementation. *(continued)*

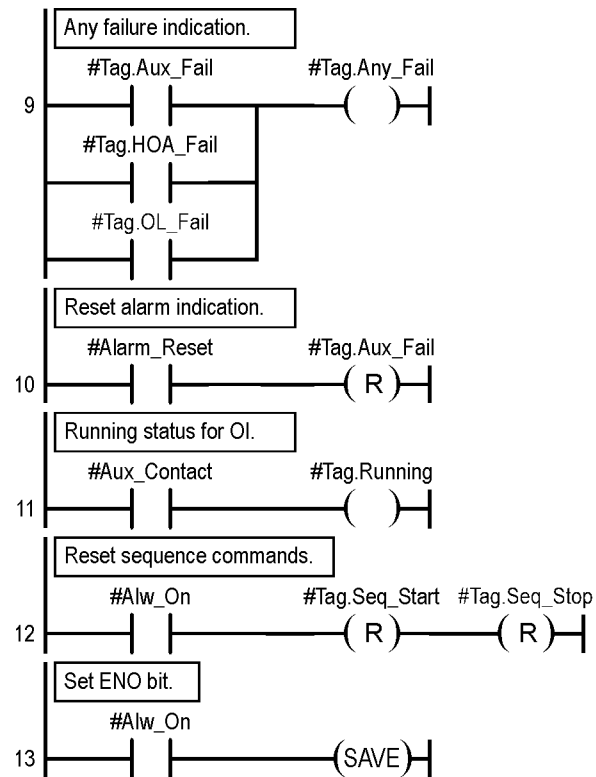


Figure 5. (continued)

The “first scan” bit (`First_Cycle` in Figure 5, Network 1) is generated by OB100, which is the first OB to be executed after a restart. This OB latches the `First_Cycle` bit and the last network of OB1 resets it. Thus, this Boolean is on for only one scan of OB1. The `#Alw_On` bit is initialized to true by the block interface.

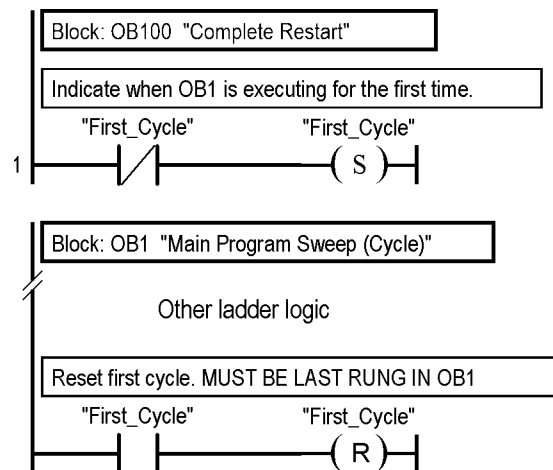


Figure 6. First_Cycle bit from processor status.

5.2 Conveyor Motor Device Control

A conveyor motor device tag is shown in Figure 7. The physical connections to the PLC are:

Physical Inputs:

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

Physical Outputs:

C1503_Start	Starter (on 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.

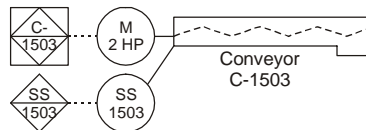


Figure 7. P&ID symbol for motor device control associated with conveyor.

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 FB as shown in Figure 8. The ladder logic that implements this function block is shown in Figure 9. The use of this block is shown in Figure 8 to control a conveyor with tag C1503. The fields of the Motor_Conv block are specified as the appropriate variables.

The parameters of the Motor_Conv FB are as follows:

<u>Name</u>	<u>Data type</u>	<u>Default value</u>	<u>Description</u>
Input			
Aux_Contact	Bool	false	Auxiliary contact
OL_Trip	Bool	false	Overload trip
HOA_Switch	Bool	true	HOA switch auto contact
Speed_Switch	Bool	False	Speed switch contact
Alarm_Reset	Bool	false	Resets alarm
Maint	Bool	true	Maintenance privilege
Man_Start	Bool	false	Manual start
Man_Stop	Bool	false	Manual stop
Select_Dev_Num	Int	0	Device selected to start/stop
This_Dev_Num	Int	0	Number for this device
Output			
Starter	Bool		Motor starter contact
InOut			
Tag	Motor_Type		Device tag
Static			
Start_Req	Bool	false	Start request
Stop_Req	Bool	false	Stop request
Mtr_Strtr	Bool	false	Internal copy of output
Alw_On	Bool	true	Always on for last network
Fail_Tmr	TON		Timer for aux fail alarm

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.

The implementation for the conveyor device control (Figure 9) differs from the standard motor control of Figure 5 only in the generation of the speed switch failure alarm. The remainder of the FB block ladder logic is the same.

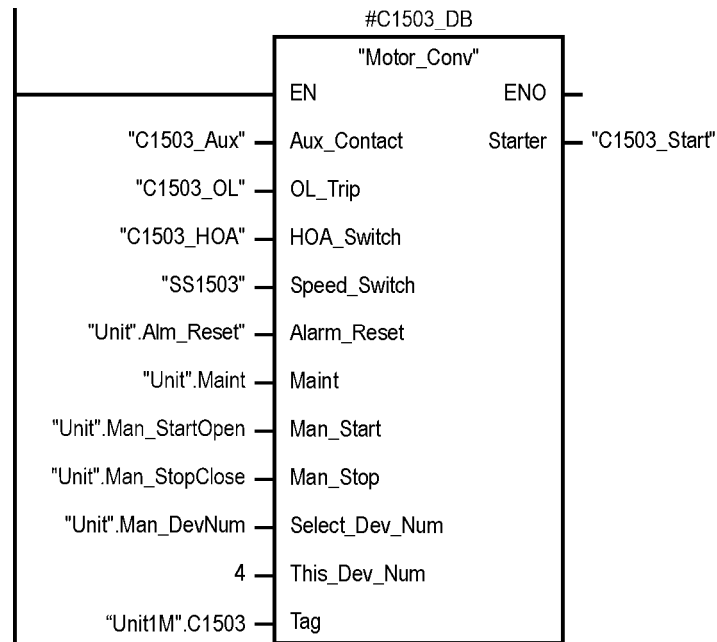


Figure 8. Conveyor device control with Motor_Conv FB.

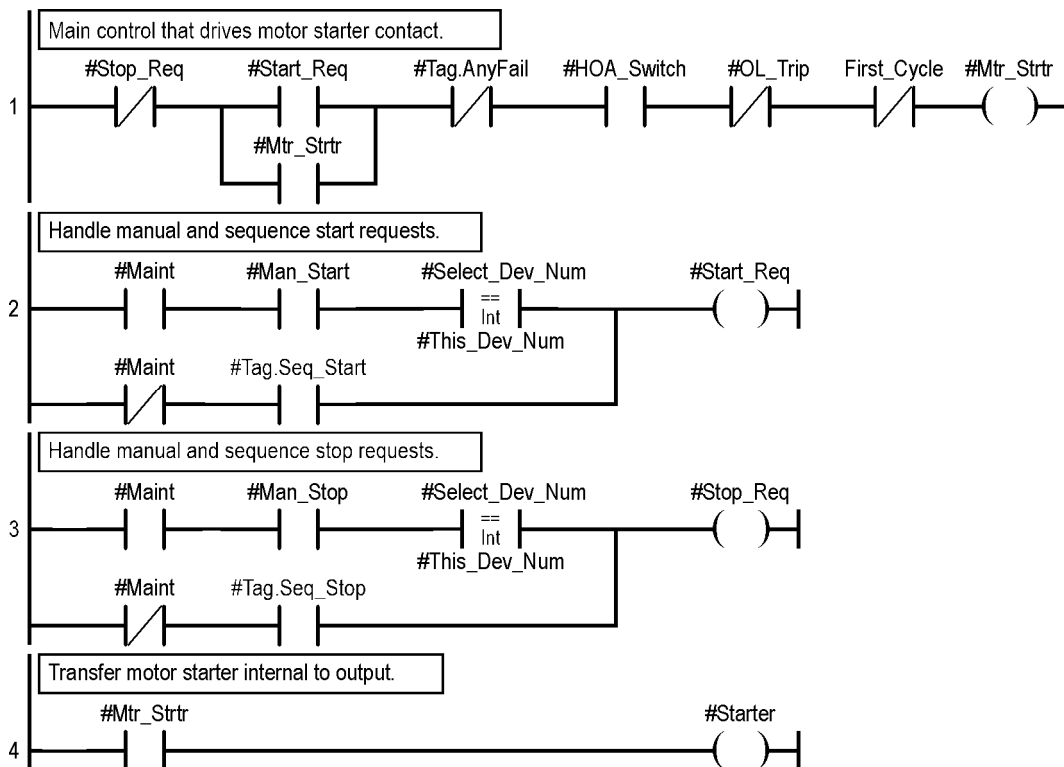


Figure 9. Motor_Conv FB implementation. *(continued)*

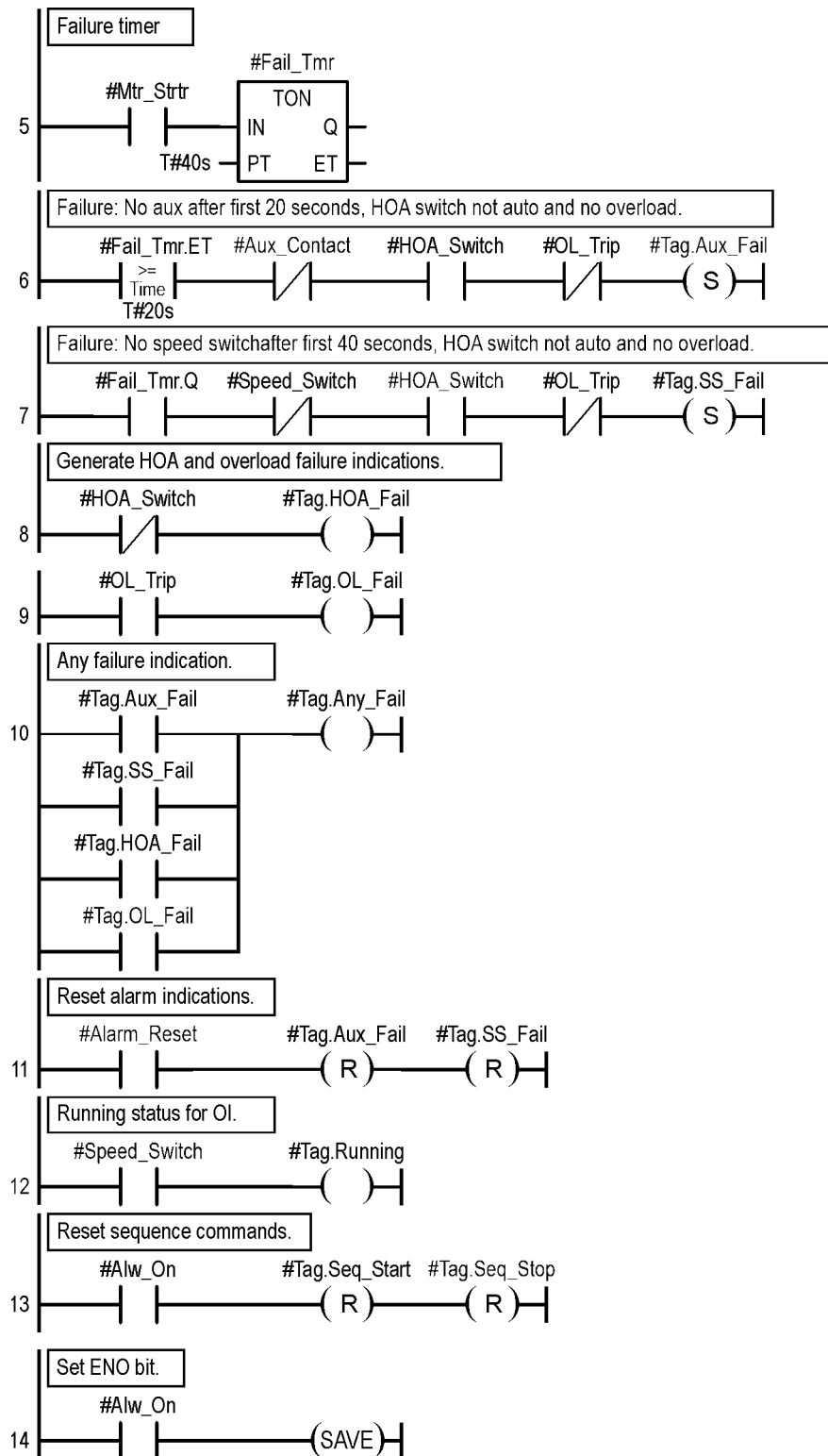


Figure 9. (continued)

5.3 Discrete Valve Device Control

A discrete valve device tag is shown in Figure 10. 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_Sol_Vlv	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

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

XV102.Seq_Open	Open
XV102.Seq_Close	Close

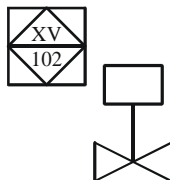


Figure 10. P&ID symbol for discrete valve device.

The Valve_Disc function block that implements this valve control is shown in Figure 11. This motor control FB is implemented in ladder logic as in Figure 12. The use of this block is shown in Figure 11 to control the valve with tag XV102. The inputs and outputs of the Valve_Disc block are connected to the appropriate variables.

The parameters of the Valve_Disc FB are as follows:

<u>Name</u>	<u>Data type</u>	<u>Default value</u>	<u>Description</u>
Input			
Open_LS	Bool	False	Valve open limit switch
Close_LS	Bool	True	Valve closed limit switch
Alarm_Reset	Bool	False	Resets alarm
Maint	Bool	True	Maintenance privilege
Man_Open	Bool	False	Manual open
Man_Close	Bool	False	Manual close
Select_Dev_Num	Int	0	Device selected to open/close
This_Dev_Num	Int	0	Number for this device
Output			
Vlv_Sol	Bool		Valve solenoid
InOut			
Tag	Valve_Disc_Type		Device tag
Static			
Fail_Tmr	TON		Timer for fail alarms
Valve	Bool	false	Internal copy of output
Alw_On	Bool	true	Always on for last network

The first two networks in Figure 12 control the internal coil that ultimately drives the valve. 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. The third network copies the internal output to the physical output that drives the valve solenoid coil. An internal FB tag must be used as the output of the first two networks in order to use this tag as a contact and avoid a warning message.

The fourth network 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

The fifth network generates the actual alarms. The sixth network generates one summary failure indication that would appear on an alarm summary screen. The seventh network resets the failure alarms. The open and close status for the OI are determined and the two sequence command variables are always reset at the end of block execution. The last network always turns on the ENO output.

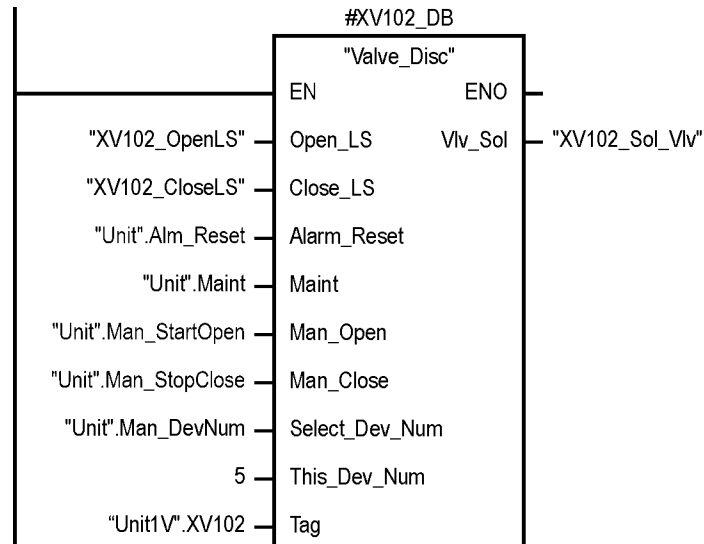


Figure 11. Discrete valve device control with Valve_Disc FB.

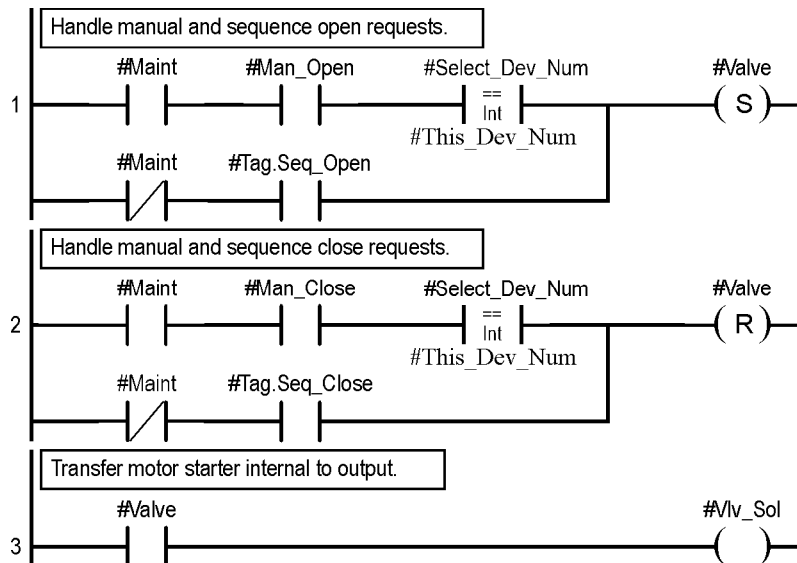


Figure 12. Valve_Disc FB implementation. *(continued)*

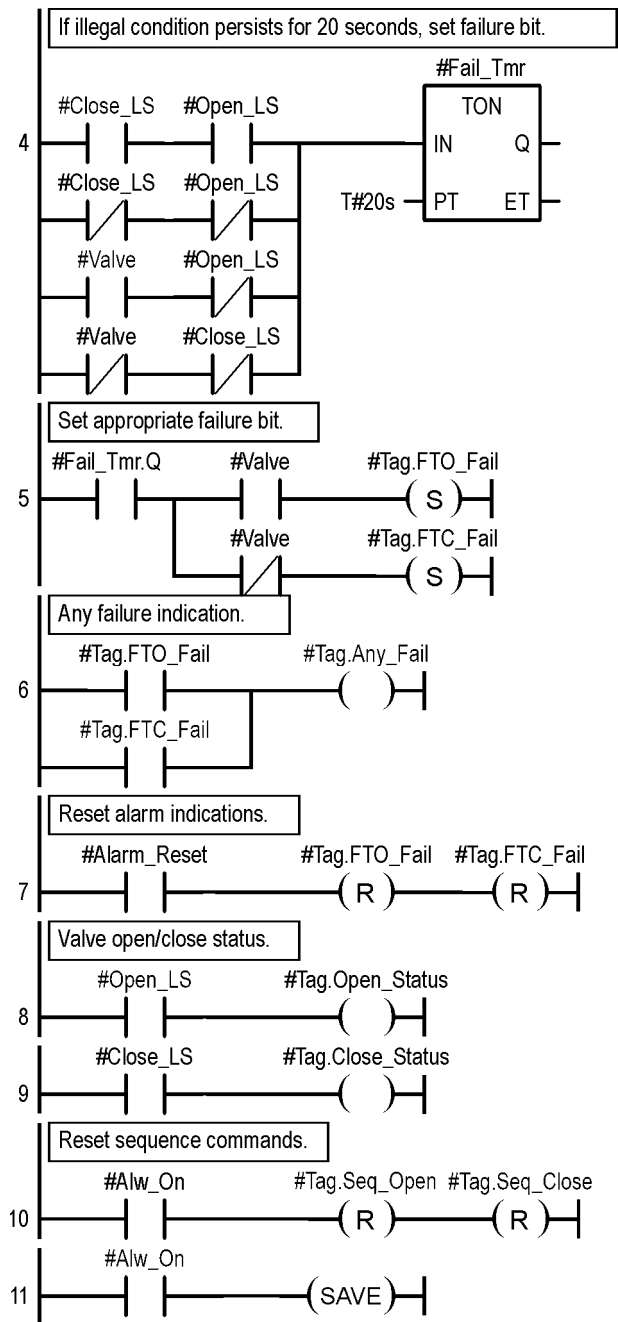


Figure 12. (continued)

5.4 Slide Gate Control

A slide gate device tag is shown in Figure 13. 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 (on when motor running to open)
G121_Close_Aux	Close auxiliary contact (on when motor running to close)
G121_OL	Overload trip (on when motor overloaded)
G121_HOA	HOA switch auto contact (on when auto contact is closed)
ZSO121	Gate-open limit switch (on when gate open)
ZSC121	Gate-closed limit switch (on when gate closed)

Physical Outputs:

G121_Start_Open	Open starter (on to start/run motor to open)
G121_Start_Clos	Close starter (on 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

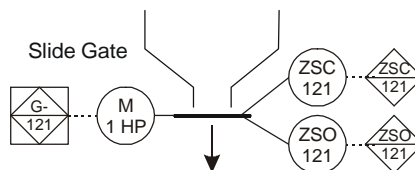


Figure 13. P&ID symbol for slide gate device.

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 FB function block that implements this valve control is shown in Figure 14 to control a slide gate with tag G-121. The slide gate control FB is implemented in ladder logic as in Figure 15. The inputs and outputs of the Gate_Slide block are connected to the appropriate variables. The appropriate sequence step latches the G121.Seq_Open or G121.Seq_Close variable to control the gate.

The parameters of the Gate_Slide FB are as follows:

<u>Name</u>	<u>Data type</u>	<u>Default value</u>	<u>Description</u>
Input			
Open_Aux	Bool	false	Open auxiliary contact
Close_Aux	Bool	false	Close auxiliary contact
OL_Trip	Bool	true	Overload trip
HOA_Switch	Bool	false	HOA switch auto contact
Open_LS	Bool	false	Gate open limit switch
Close_LS	Bool	false	Gate close limit switch
Alarm_Reset	Bool	false	Resets alarm
Maint	Bool	true	Maintenance privilege
Man_Open	Bool	false	Manual open
Man_Close	Bool	false	Manual close
Select_Dev_Num	Int	0	Device selected to start/stop
This_Dev_Num	Int	0	Number for this device
Output			
Start_Open	Bool		Open starter contact
Start_Close	Bool		Close starter contact
InOut			
Tag	Gate_Slide_Type		Device tag
Static			
Open_Req	Bool	false	Open request
Close_Req	Bool	false	Close request
Strtr_Open	Bool	false	Internal copy of output
Strtr_Close	Bool	false	Internal copy of output
Alw_On	Bool	true	Always on for last network
Fail_Tmr	TON		Timer for aux fail alarm

The control for the slide gate device (Figure 15) 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. The control for the open starter coil is in the first network. Note that it is disabled when a closed command is received. The control for the close starter coil is shown in the second network 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.

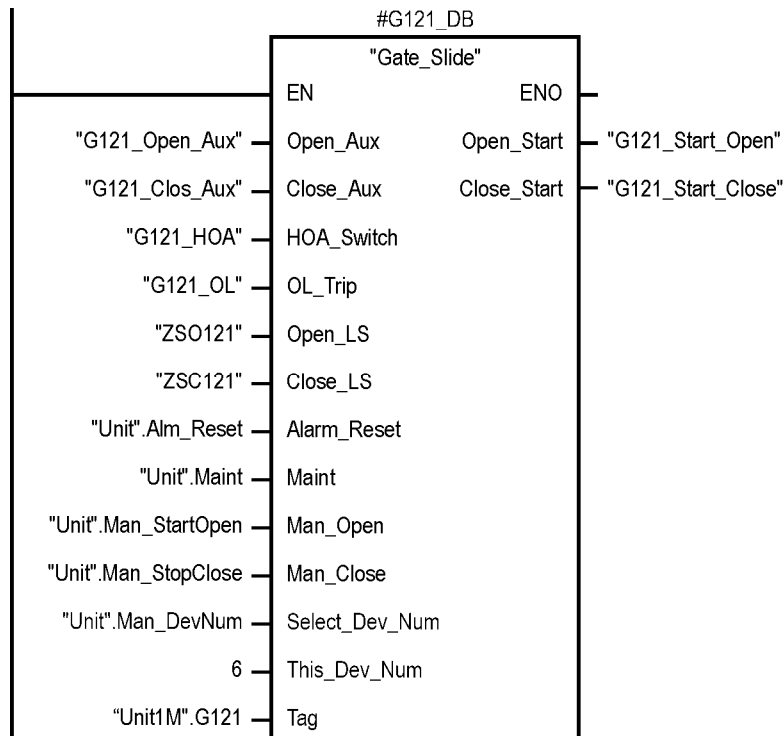


Figure 14. Slide gate device control with Gate_Slide FB.

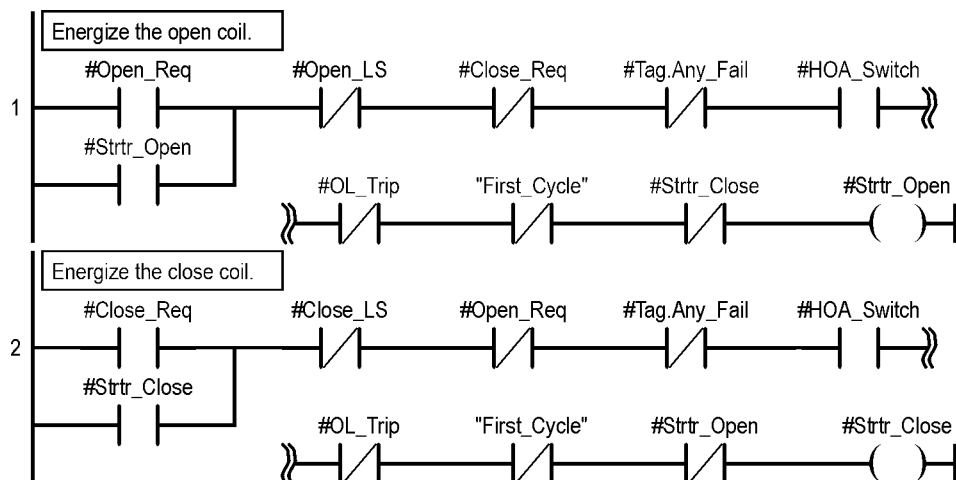


Figure 15. Gate_Slide FB implementation. *(continued)*

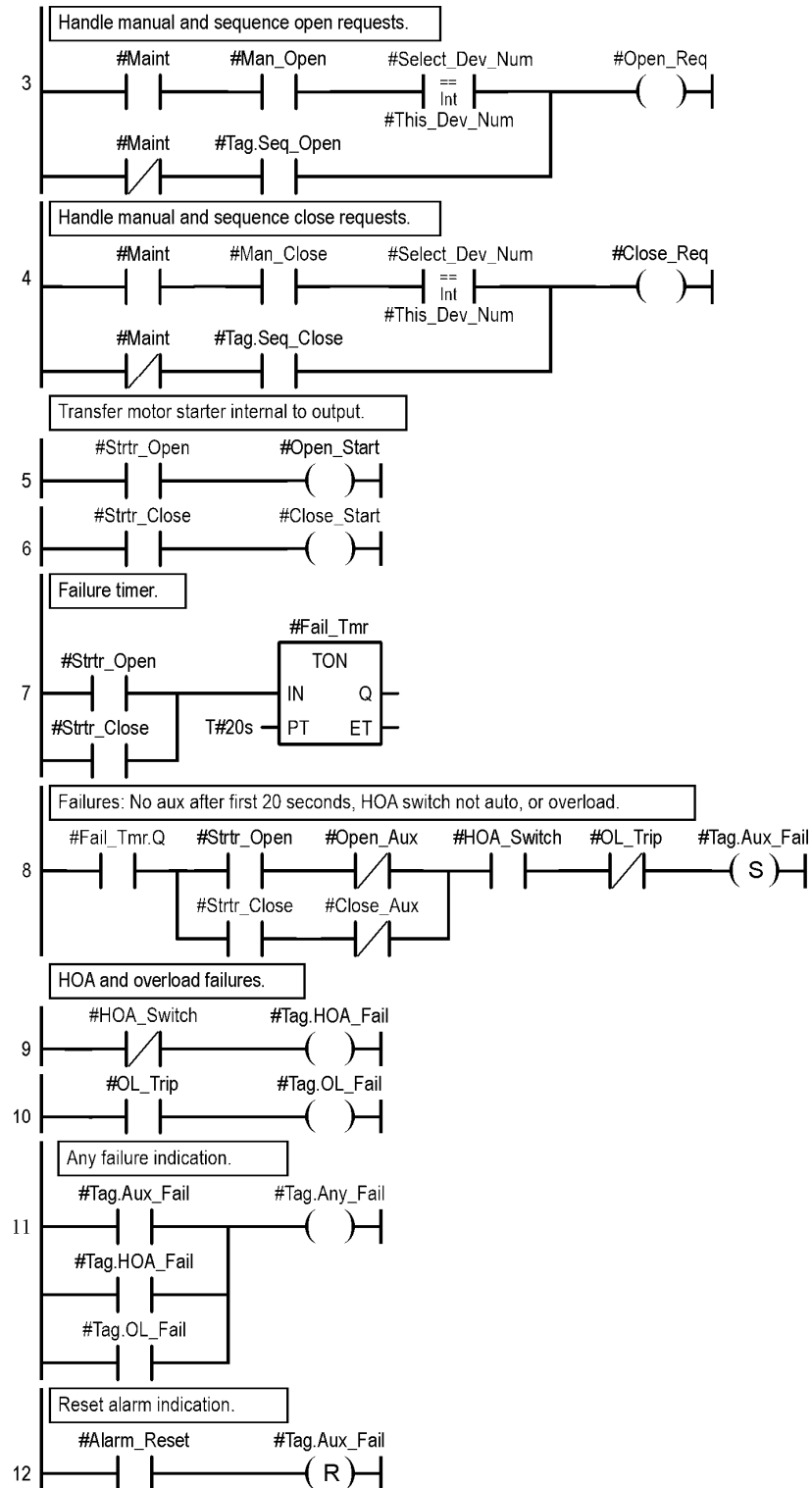


Figure 15. (continued)

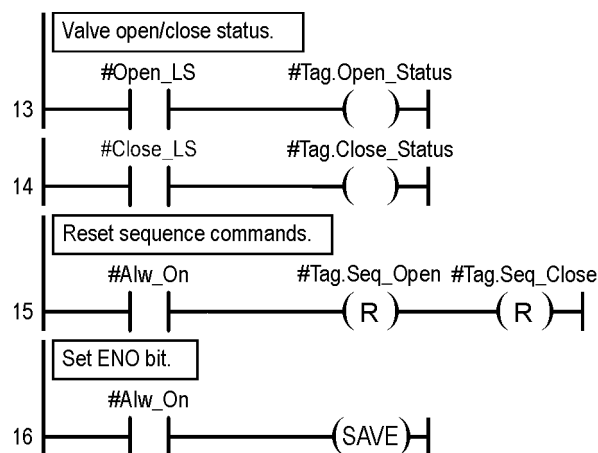


Figure 15. *(continued)*

5.5 Flop Gate Control

A flop gate device tag is shown in Figure 16. 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 (on when diverted left)
ZSR145	Right-path limit switch (on when diverted right)
G145_HOA	HOA switch auto contact (on when auto contact is closed)

Physical Outputs:

G145_Sol_Left	Left path solenoid (on to move gate to divert left)
G145_Sol_Right	Right path solenoid (on 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

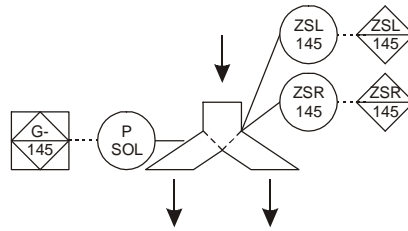


Figure 16. P&ID symbol for flop gate device.

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 function block that implements this valve control is shown in Figure 17 to control a flop gate with tag G-145. The flop gate control FB is implemented in ladder logic as in Figure 18. The inputs and outputs of the Gate_Flop block are connected to the appropriate variables. The appropriate sequence step latches the G145.Seq_Left or G145.Seq_Right variable to control the gate.

The parameters of the Gate_Flop FB are as follows:

<u>Name</u>	<u>Data type</u>	<u>Default value</u>	<u>Description</u>
Input			
Left_LS	Bool	false	Left position limit switch
Right_LS	Bool	false	Right position limit switch
HOA_Switch	Bool	false	HOA switch auto contact
Alarm_Reset	Bool	false	Resets alarm
Maint	Bool	true	Maintenance privilege
Man_Left	Bool	false	Manual left command
Man_Right	Bool	false	Manual right command

Select_Dev_Num	Int	0	Device selected to start/stop
This_Dev_Num	Int	0	Number for this device
Output			
Sol_Left	Bool		Left position solenoid valve
Sol_Right	Bool		Right position solenoid valve
InOut			
Tag	Gate_Flop_Type		Device tag
Static			
Left_Req	Bool	false	Left request
Right_Req	Bool	false	Right request
Valve_Left	Bool	false	Internal copy of output
Valve_Right	Bool	false	Internal copy of output
Alw_On	Bool	true	Always on for last network
Fail_Tmr	TON		Timer for fail alarm

The control for the flop gate device is similar to the slide control, except that there is no overload trip contact. Figure 18 shows the ladder logic for this device control. The control for the left valve control is in network 1. Note that it is disabled when a right command is received. The control for the right valve control is shown in network 2 and is similar to the left valve control. The failure conditions are handled similarly to a discrete valve.

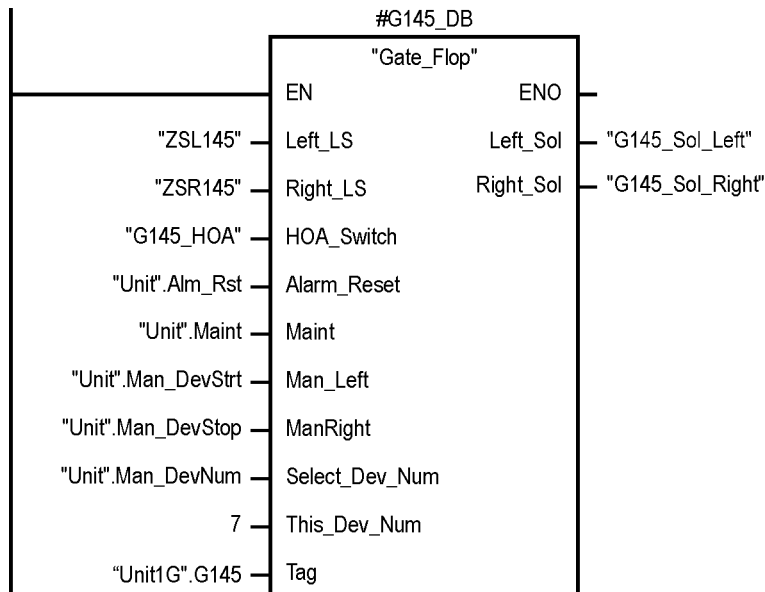


Figure 17. Flop gate device control with Gate_Flop FB.

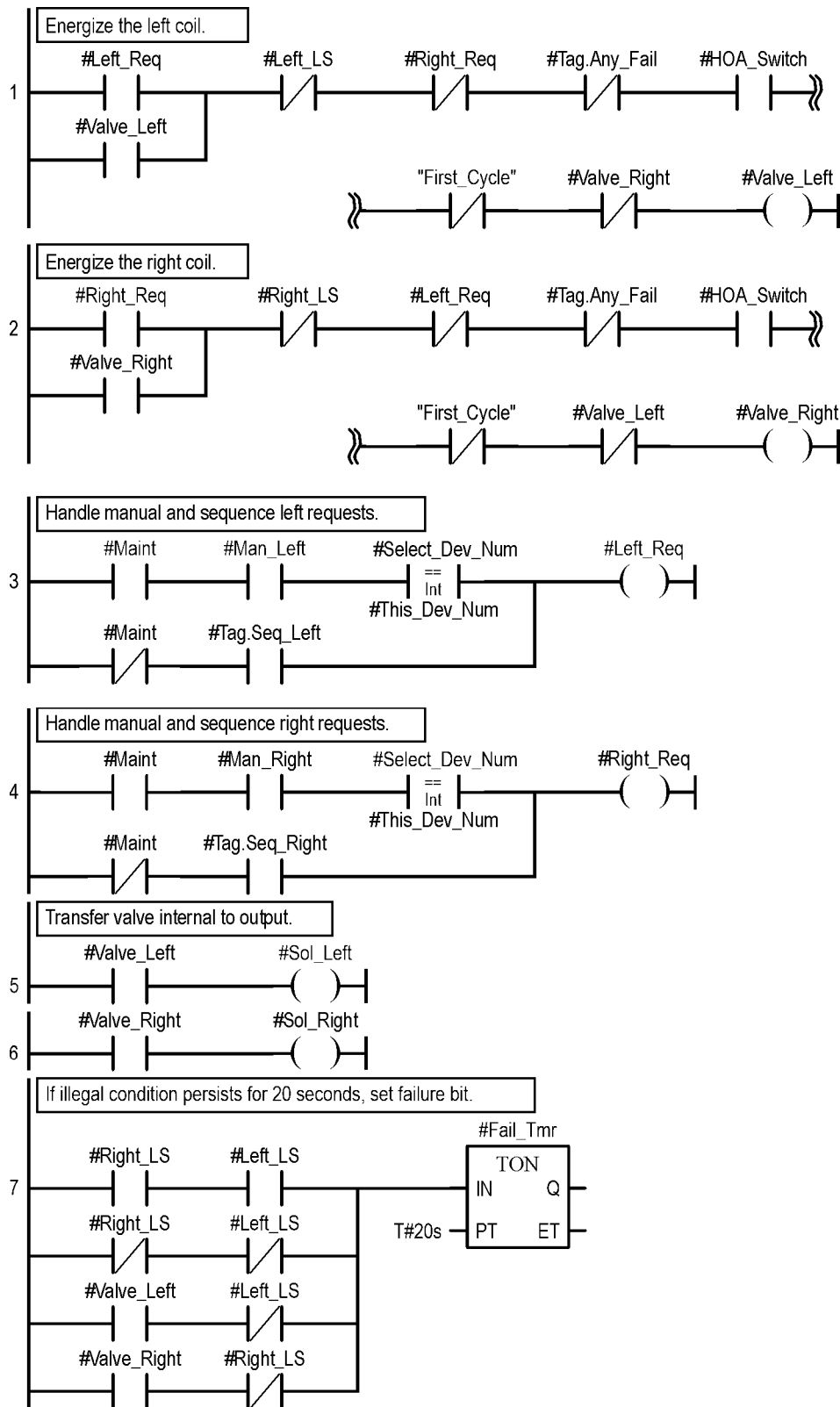
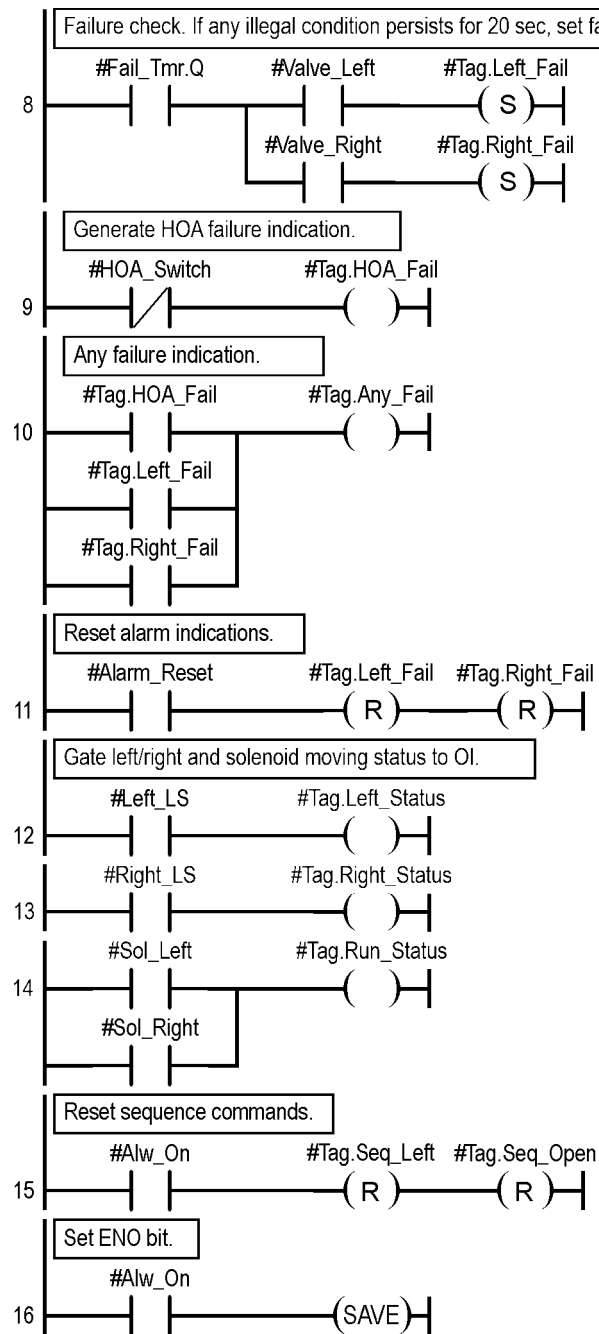


Figure 18. Gate_Flop FB implementation. (*continued*)

**Figure 18. (continued)**

6. Sequenced Control

A sequence is initiated in one of two ways: operator request via OI sequence control 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 conditional text messages are displayed in the lower portion of each major graphics display. In similar fashion, the PLC conveys the time remaining in a timed sequence activity for display in the lower left corner of a graphic screen.

The Seq_Type user-defined data 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

The UDTs for the sequences in a unit must be contained in a data block. For example, for a unit named "Unit1," that contains "Startup," "Operate," "Hold," "Shutdown," and "EShutdown" sequences, the data for the sequences is contained in a data block, "Unit1S," could be defined as

<u>Name</u>	<u>Data Type</u>	<u>Description</u>
Static		
Startup	Seq_Type	Startup sequence data
Operate	Seq_Type	Operate sequence data
Hold	Seq_Type	Hold sequence data
Shutdown	Seq_Type	Shutdown sequence data
EShutdown	Seq_Type	E-Shutdown sequence data

6.1 Sequence Code

There are four 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. Transitions between steps and device control from sequence

Parts 2 to four of a sequence are contained in one function block.

Figure 19 shows the first section of the sequence. This network handles the granting of maintenance privilege to the unit. In maintenance privilege, the motors and valves 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. This network is in the Unit1_990Misc function block.

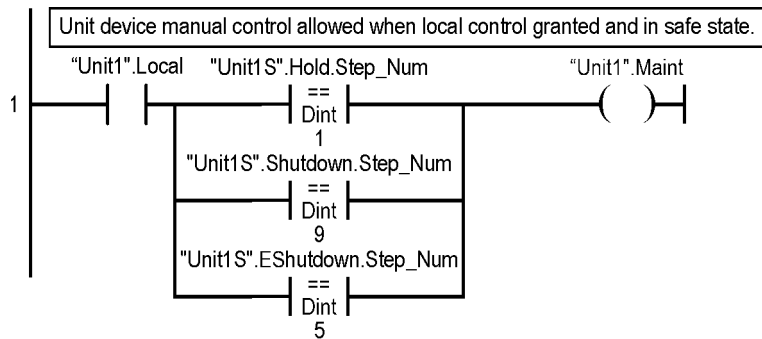


Figure 19. Granting maintenance privilege for unit devices.

The second part of the sequence (Figure 20) 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 third 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 “>=” comparison contact 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, its step number is set to one and the step numbers for the other unit sequences are set to zero. The output logic also adds a constant to the sequence 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 a request to start another sequence. The use of the greater-than-equal to seal the sequence start request also handles the case where one set of steps is used for multiple sequences. A common case of this situation is for 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.

The use of the == compare after the P_TRIG 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 number is not zero and there does not need to be a transition into the first step or zeroing of steps in the other unit sequences. This method of pausing also requires a "Hold.Running" NC contact in series with the auto-start contacts on this rung. This method of pausing is not normally used in this project.

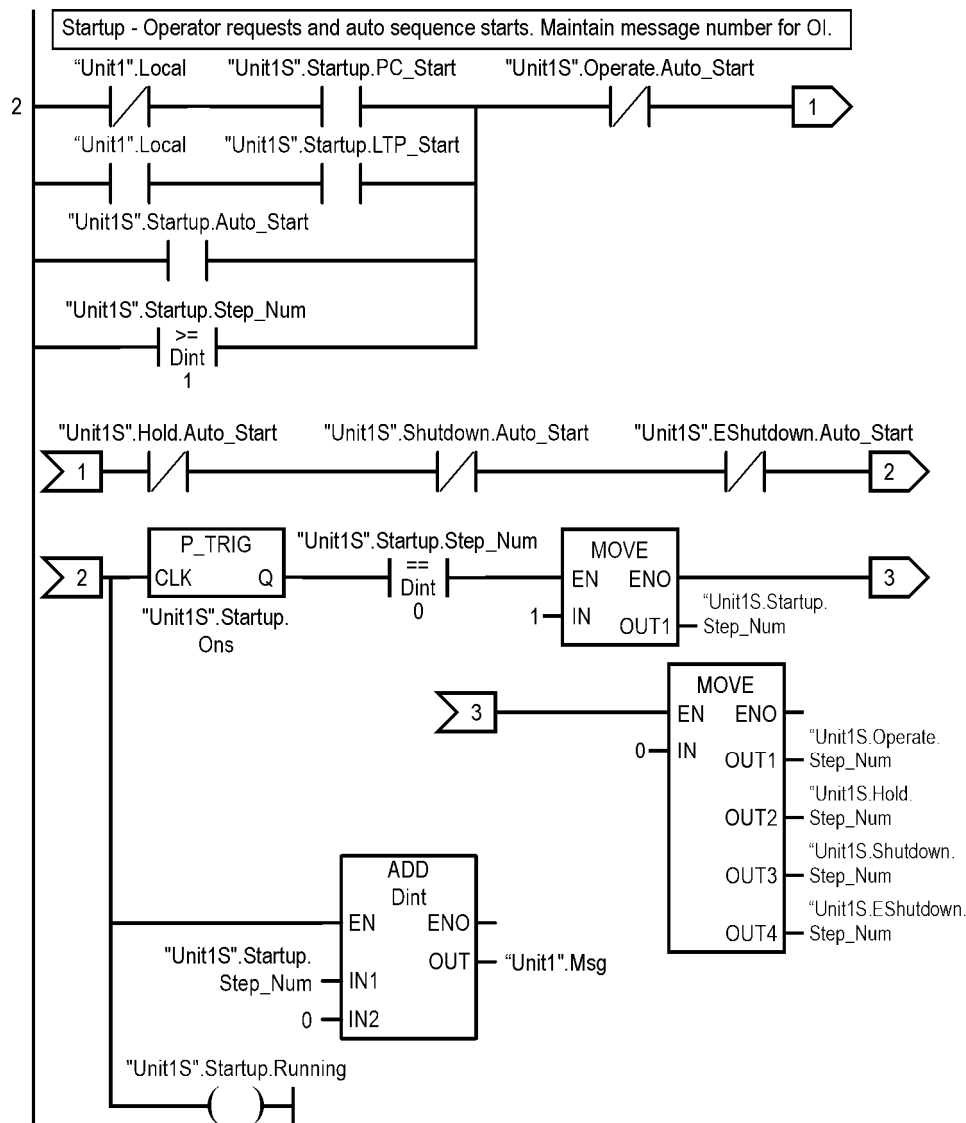


Figure 20. Initiate sequence.

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 21) that cause a certain sequence, commonly Hold, Shutdown, or E-Shutdown to be started. These are contained in a separate function block, called "Unit1_Abnormal".

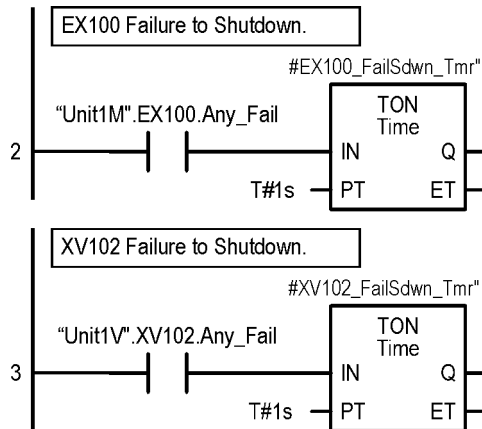


Figure 21. Failure conditions that invoke a sequence.

Figure 22a is the automatic start for the Unit_Operate sequence. This sequence is automatically started when the Unit_Startup is in the last step (step 7). The series conditions at the beginning of the network 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. A transitional contact for the start condition turns on the auto start bit, and a timer is used to hold the auto start request bit **on** for 5 seconds (to allow time for other logic to be resolved). Figure 21b 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. Note that the failure delay timers are static data in the instance data block for the "Unit1_991Abnormal" FB.

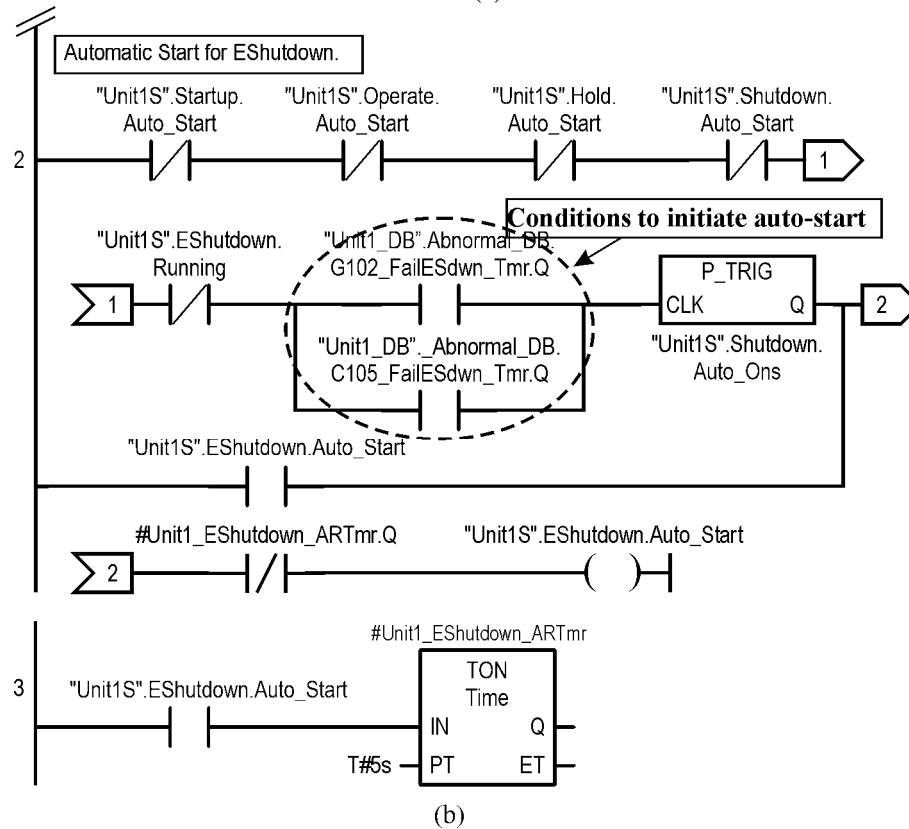
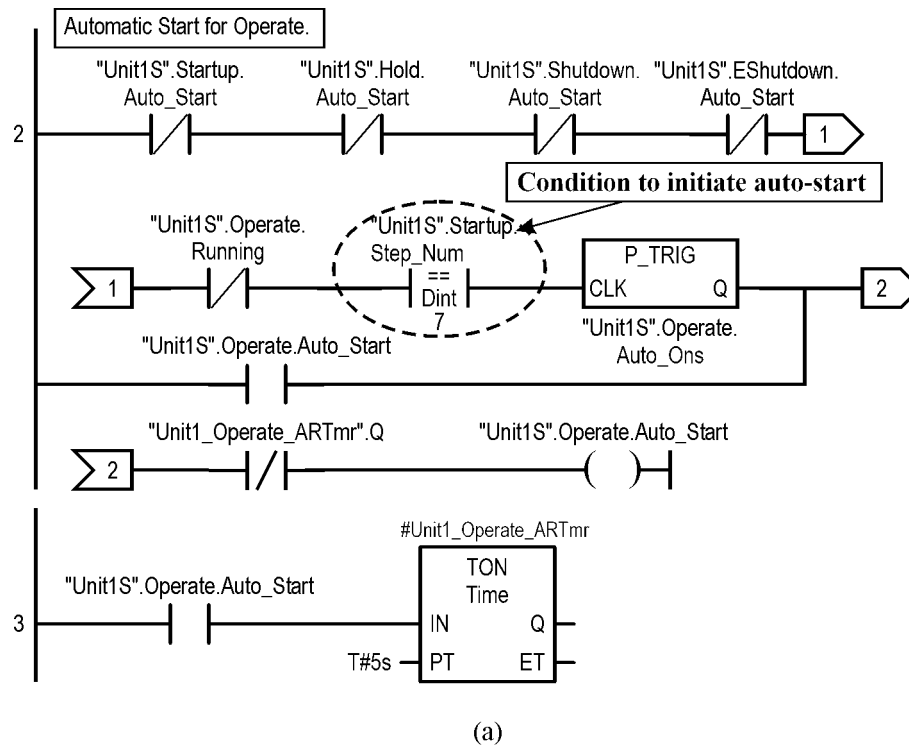


Figure 22. Sequence automatic starts: (a) Unit_Operate sequence; (b) Unit_EShutdown sequence.

The fourth section (Figure 23) is the “meat” of the Startup sequence. Each network 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. Network 3 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 set. This bit will be cleared by the valve/motor/etc. device code. The command must be on a branch parallel to the transition condition and move block, or the command will not be executed.

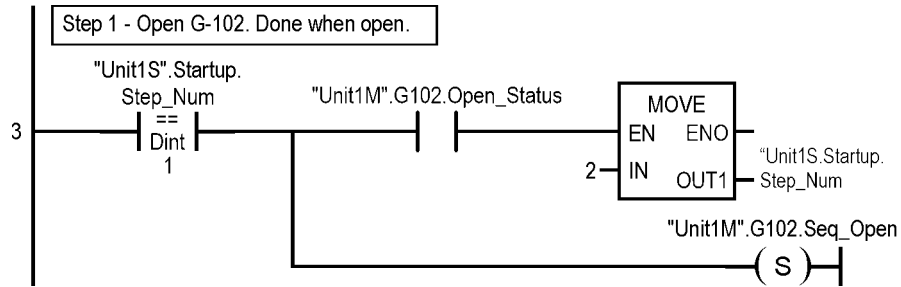


Figure 23. Step transition.

The next-to-last step in the E-Shutdown sequence (Figure 24) 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.

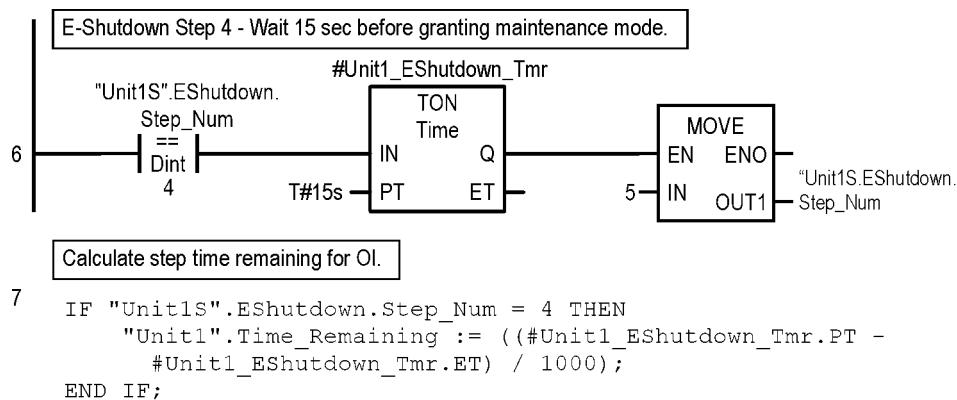


Figure 24. E-Shutdown next-to-last step transition.

6.2 Branches

Sequence diagram branches are handled by a parallel ladder branch with a MOVE block to set the step number as needed. The skip of Figure 3 of the “Sequence Diagram Guidelines” is implemented in the transition networks as shown in Figure 25. If the level is < 20 , then the step number is set to 13. If the level ≥ 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 networks as shown in Figure 26. If the level is < 20 , then a MOVE block sets the step number to 25. If the level ≥ 20 , then the next step is set to 28. Note that the transition out of step 27 (network 29) changes the next step to 31 and thus implement the right-hand branch. The transition out of step 30 (network 32) is handled normally.

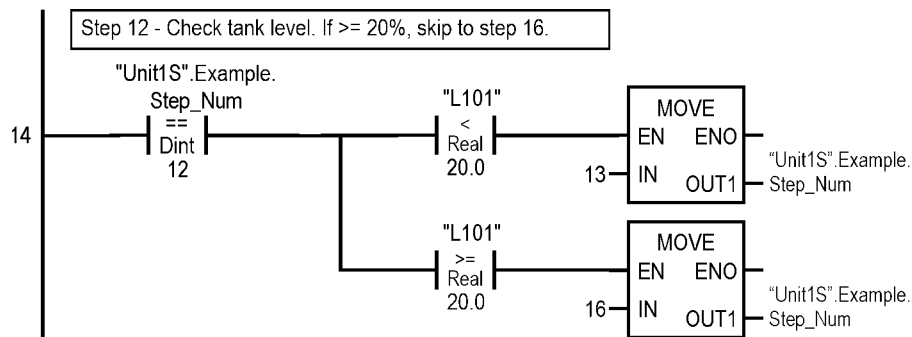


Figure 25. Code for sequence skip.

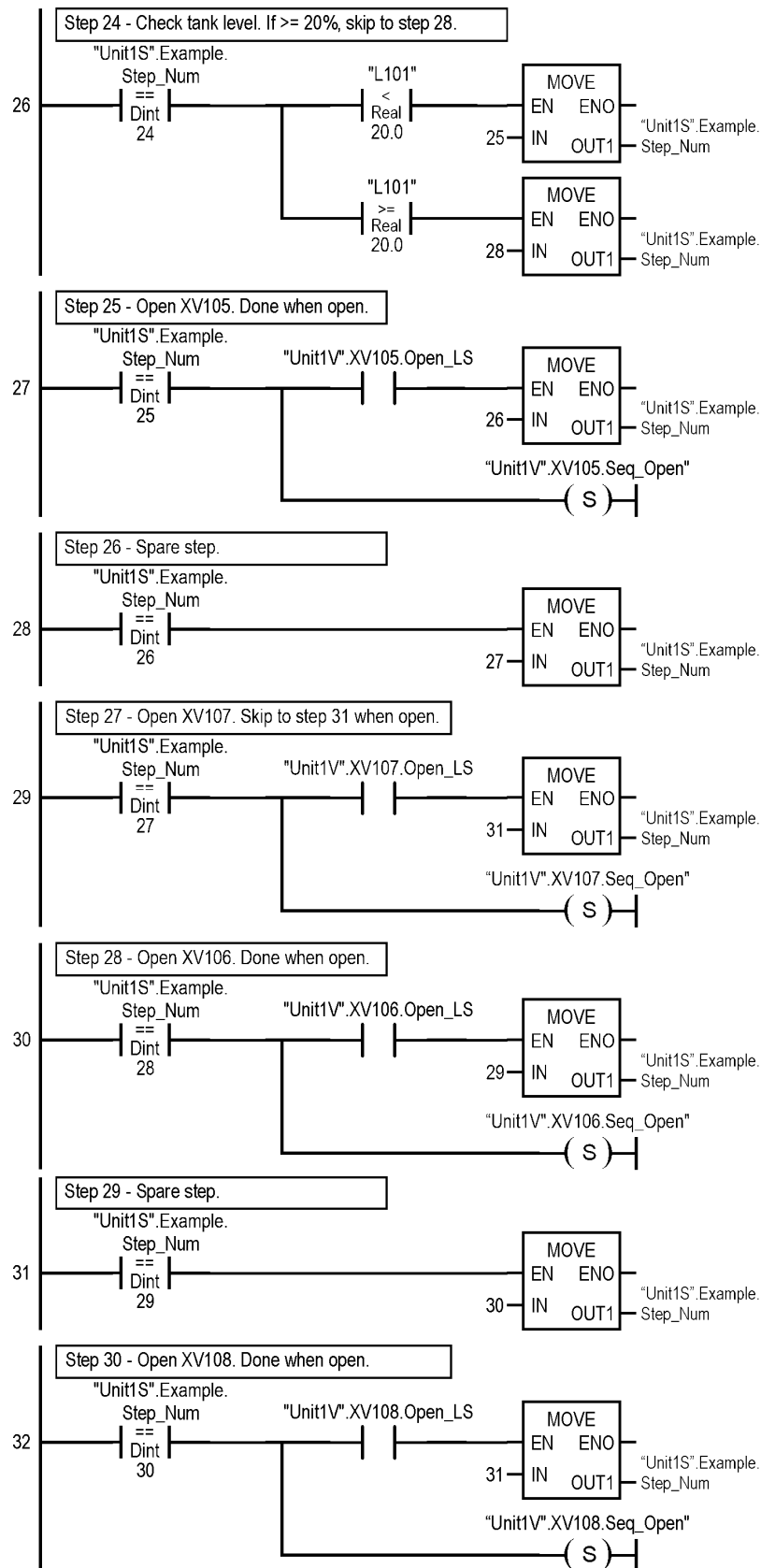


Figure 26. Code for sequence branching.

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 27 shows the network 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 28 opens the appropriate valve, depending on the actual transfer. The appropriate Unit_T10x_Dest has already been set by the operator.

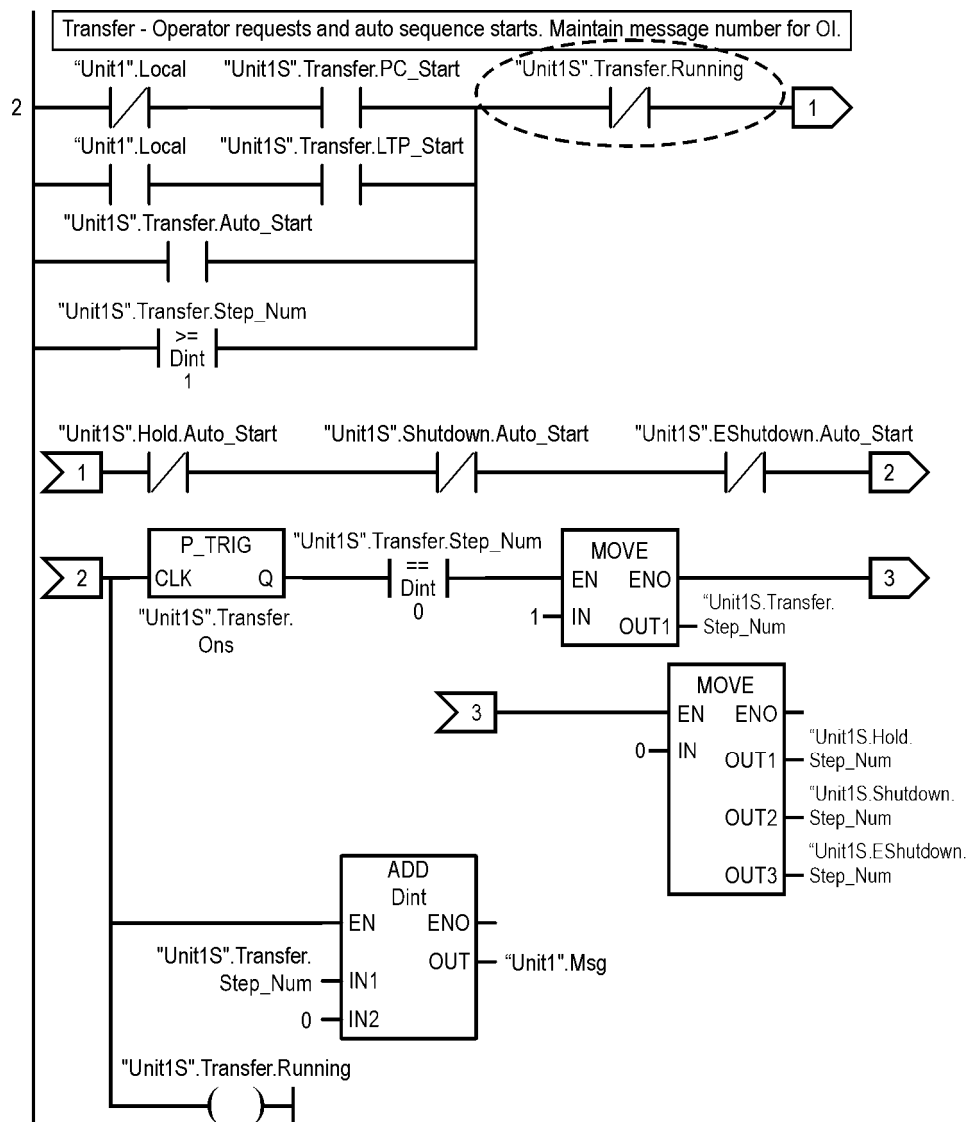


Figure 27. Start rung for common transfer sequence with one start.

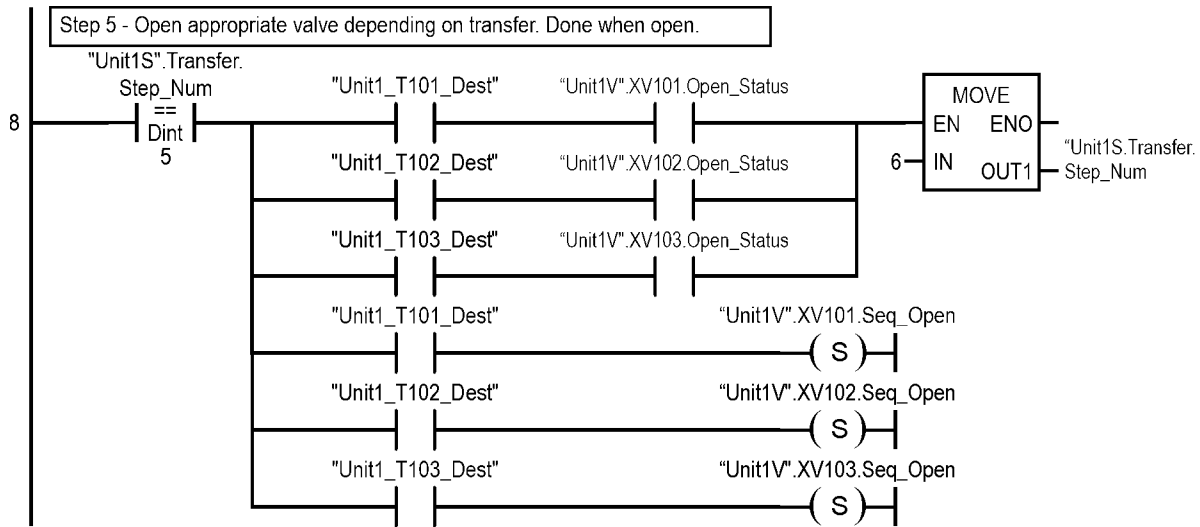


Figure 28. Conditional valve open for step 5 of transfer sequence with one start.

For the second method, Figure 29 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 30 opens the appropriate valve, depending on the actual transfer.

Note that for this method, the "Unit1S".T101_Trans, "Unit1S".T102_Trans, and "Unit1S".T103_Trans variables are of type Seq_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.

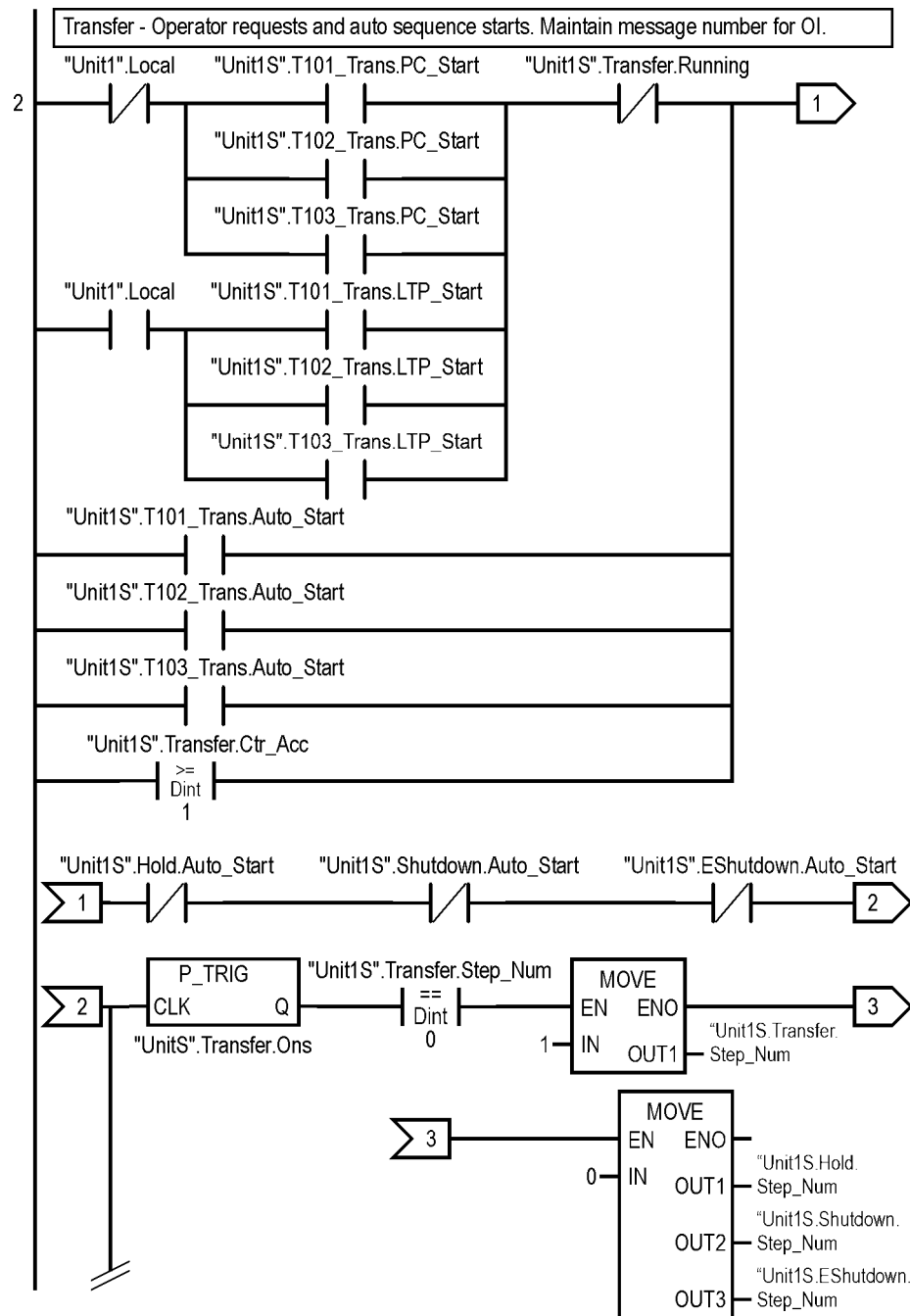


Figure 29. Start rung for common transfer sequence with multiple starts.

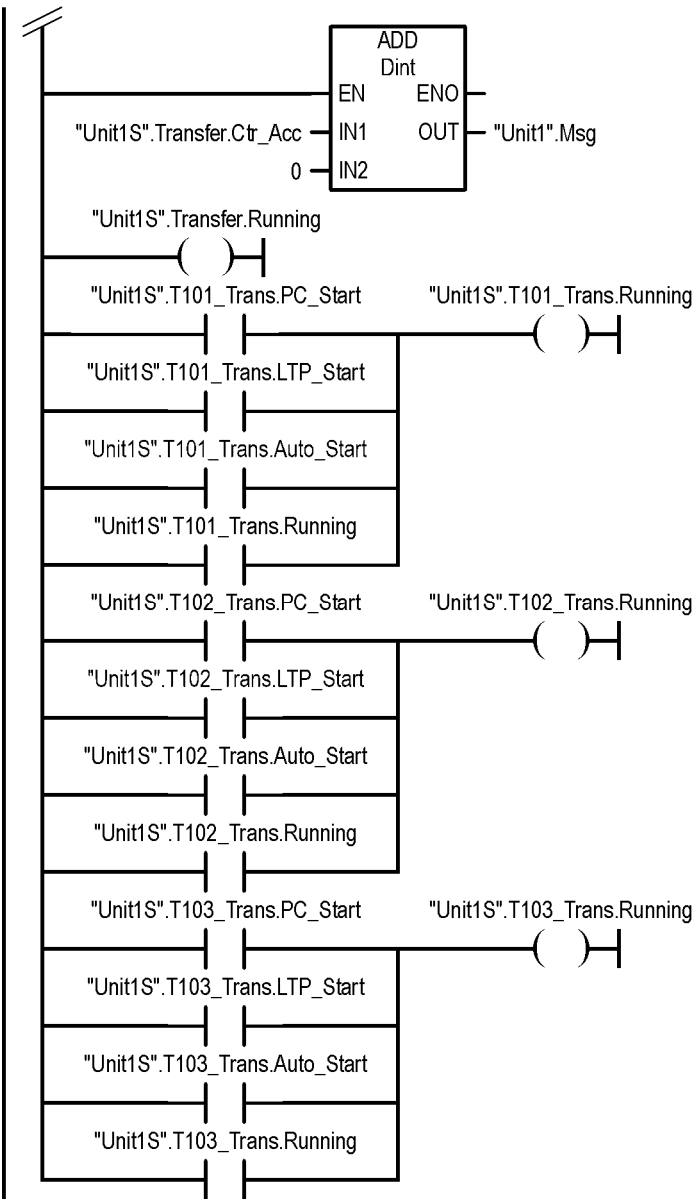


Figure 29. (continued)

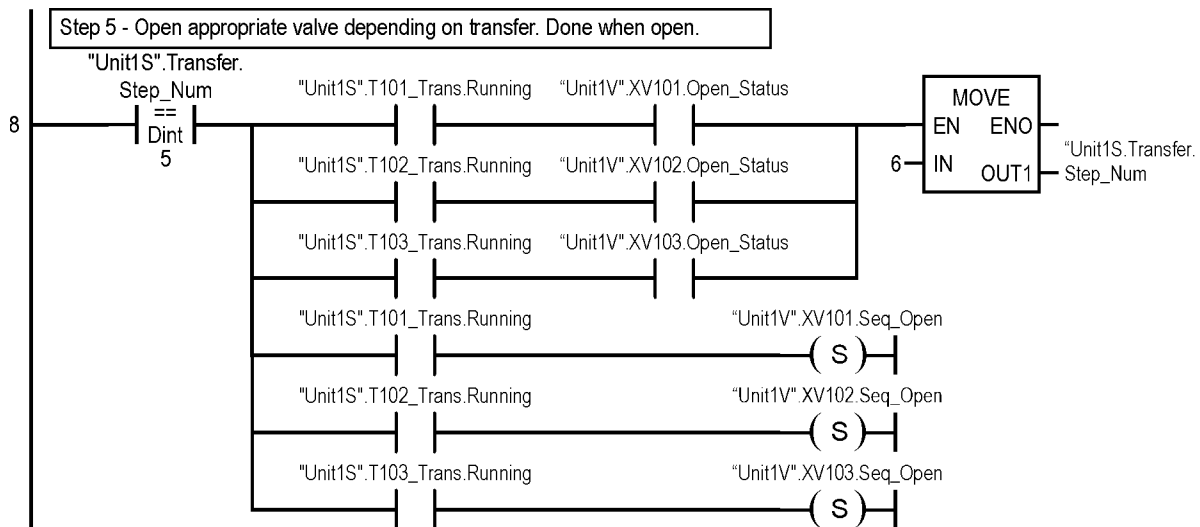


Figure 30. Conditional valve open for step 5 of transfer sequence with multiple starts.

6.4 Operator Response

The ladder logic that implements the sequence diagram of Figure 6 of the “Sequence Diagram Guidelines” is shown in Figure 31. 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.

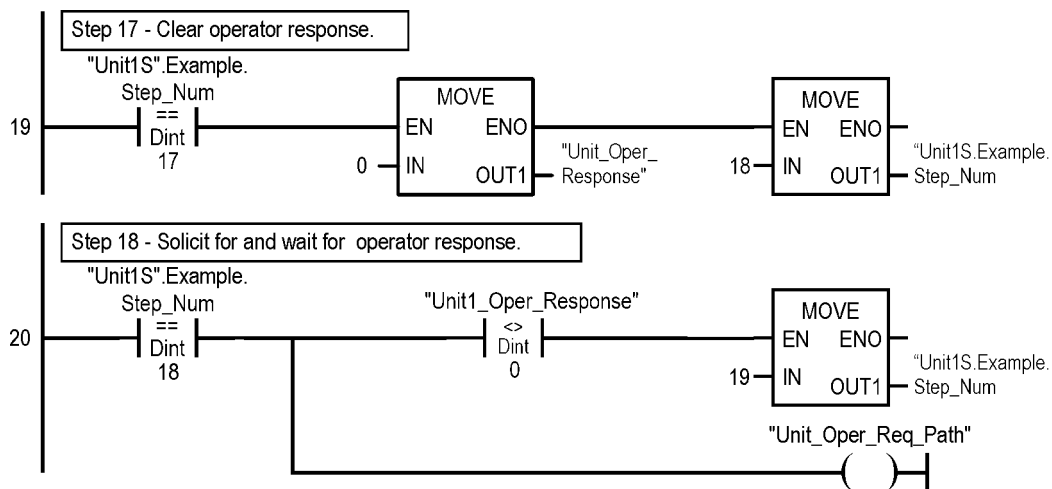


Figure 31. Code for soliciting and receiving operator response.

6.5 Coordination With Another Unit

Figure 8 of the “Sequence Diagram Guidelines” is implemented as shown in Figure 32. The appropriate sequence step-in-progress bit waits for the CIP_Recycle indication from the other unit (Figure 32a). In the Comms block, the appropriate step-in-progress bits turn on the Boolean Unit_Req_CIP_Recycle command that is communicated to the other unit (Figure 32b). 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.

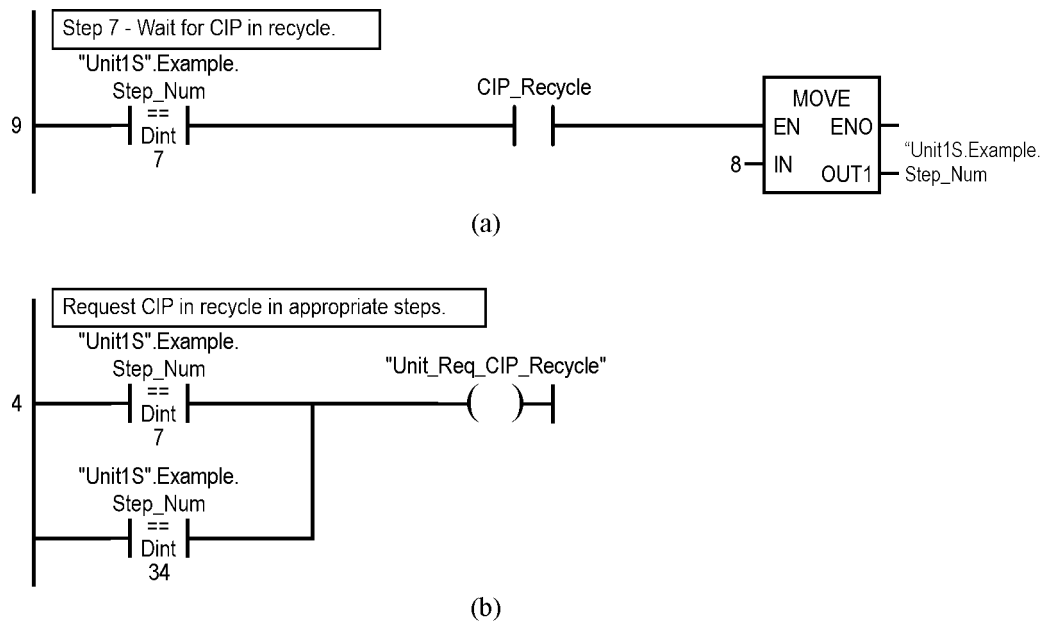


Figure 32. Code for sending request to another unit: (a) in sequence code; (b) in Communications (Comms block).

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 33a. The PT and ET connections to the TON are TIME types and hence the need for MOVE blocks to convert the data type to DINT. 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 33b. The network that clears the time should be the "Unit_Misc" FC block.

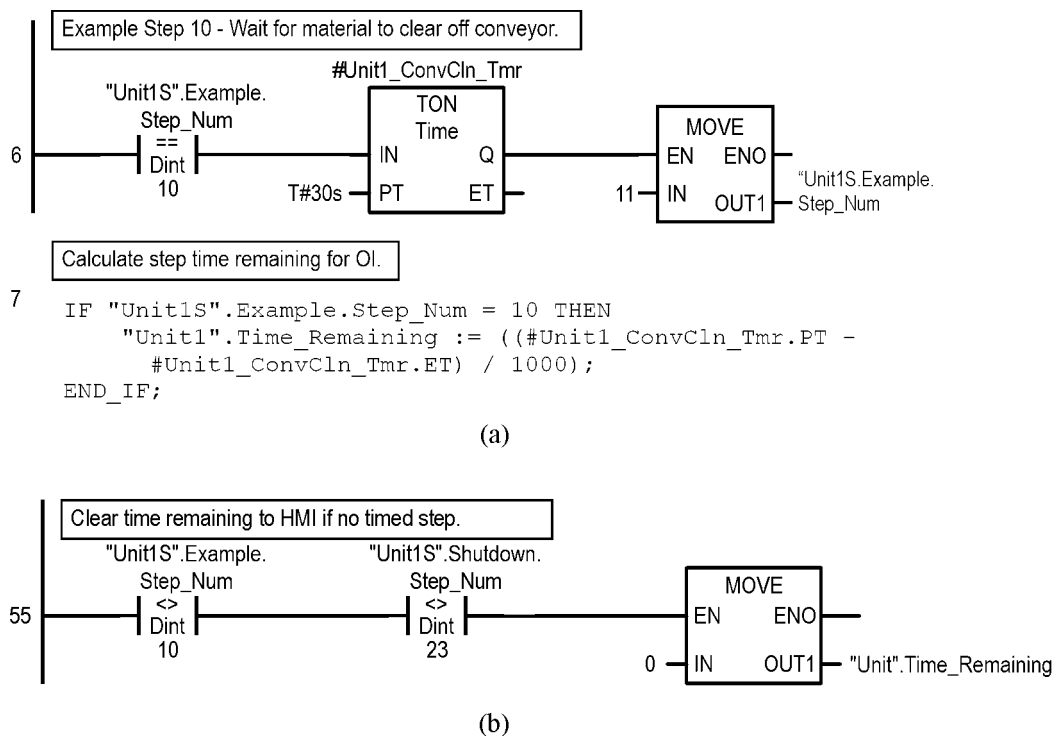


Figure 33. Display of remaining time: (a) calculation in step transition logic; (b) clearing time.

7. Continuous Control

A simple PID loop, FIC101, is shown in Figure 34. 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 0.25 second. If implemented on the S7-300/400, one can use "Fic_Tic".Q in place of "Fic_Tic_Q" and the coil attached to the timer output can be deleted. The sequences interact with the loop via the following locations that located in a UDT (UDT94) named PIDData_Type and defined as:

<u>Name</u>	<u>Type</u>	<u>Description</u>
SW_Auto	Bool	Operator-set mode (1=Auto, 0=Manual)
Loop_Auto	Bool	Loop mode (1=Auto, 0=Manual)
PV_LO	Real	Low range value for PV
PV_HI	Real	High range value for PV
SP	Real	Operator-adjusted setpoint
PV	Real	Scaled PV for operator display
Man_Out	Real	Operator-adjusted output when in manual
SPS	Real	Setpoint scaled to percent
ManO	Real	Manual output to CONT_C
Loop_LMN	Real	Controller output, in percent

If the PID data block is named "FIC101_Data", the block is used as follows

"FIC101_Data".SWAuto	Latch to set mode to auto; unlatch to set mode to manual
"FIC101_Data".Man_Out	Copy (MOVE) the desired output value (0-100) to this location when the loop is in the manual mode (when SWAuto unlatched)
"FIC101_Data".SP	Copy (MOVE) the desired setpoint to this location.

Note that the PV_LO and PV_HI locations in the data block should be set to the proper values. The sequence commands of Figure 9 of the "Sequence Diagram Guidelines" are implemented as shown in Figure 35.

An operator manual station (text Figure 10.65) is assumed present. The operator manual station sets the FY101_Md bit and the FIC101_Tiebk value (as in text Figure 10.65). The FIC101_Tiebk value is copied to the controller manual input (MAN) when FY101_Md is off. If "FIC101_Data".SWAuto is off (and FY101_Md is on), the "FIC101_Data".Man_Out is copied to the controller manual input (MAN). The FY101_Md bit overrides the "FIC101_Data".SWAuto bit. In order for the sequences to control the controller mode, the FY101_Md bit must be on. If an operator manual station is absent, then the NC contact controlled by FY101_Md is absent on the MAN_ON input of the CONT_C blocks and "FIC101_Data".Man_Out is tied to the MAN input of the CONT_C block.

Since the SP_INT input to the CONT_C block must be scaled into the range of 0 to 100 to match the units of the PV_PER input. The internal scaling of the CONT_C block assumes that zero corresponds to the lowest value of the analog input. For 4 to 20 mA analog input signals, the 4 mA signal corresponds to an analog input value of 5530 (not zero) and the "FIC101_Data".PV_LO and "FIC101_Data".PV_HI values must be set accordingly. For example, if the flow sensor range is 0 - 2000 gpm, then "FIC101_Data".PV_HI = 2000.0 and "FIC101_Data".PV_LO = 0 - 0.25(2000-0) = -500.0.

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 to order to obtain the totalization. Ladder logic for a totalizer, FQI101, is shown in Figure 36. In this case, the flow from the PID loop of Figure 34 (FT101) is being totalized. Note that the analog input is scaled to the proper units since the CONT_C block does not provide this scaling. The totalization is performed with rectangular integration, executed every second. The totalizer may be reset by the operator through the FQI101_Oper_Reset variable, but only when the unit is in the maintenance mode. Sequences reset the totalizer by moving a zero into the totalizer (for example, FQI101=0.0).

If implemented on the S7-300/400, one can use "FQI101_Tmr".Q in place of "FQI101_Tmr_Q" and the coil attached to the timer output can be deleted in Figure 36.

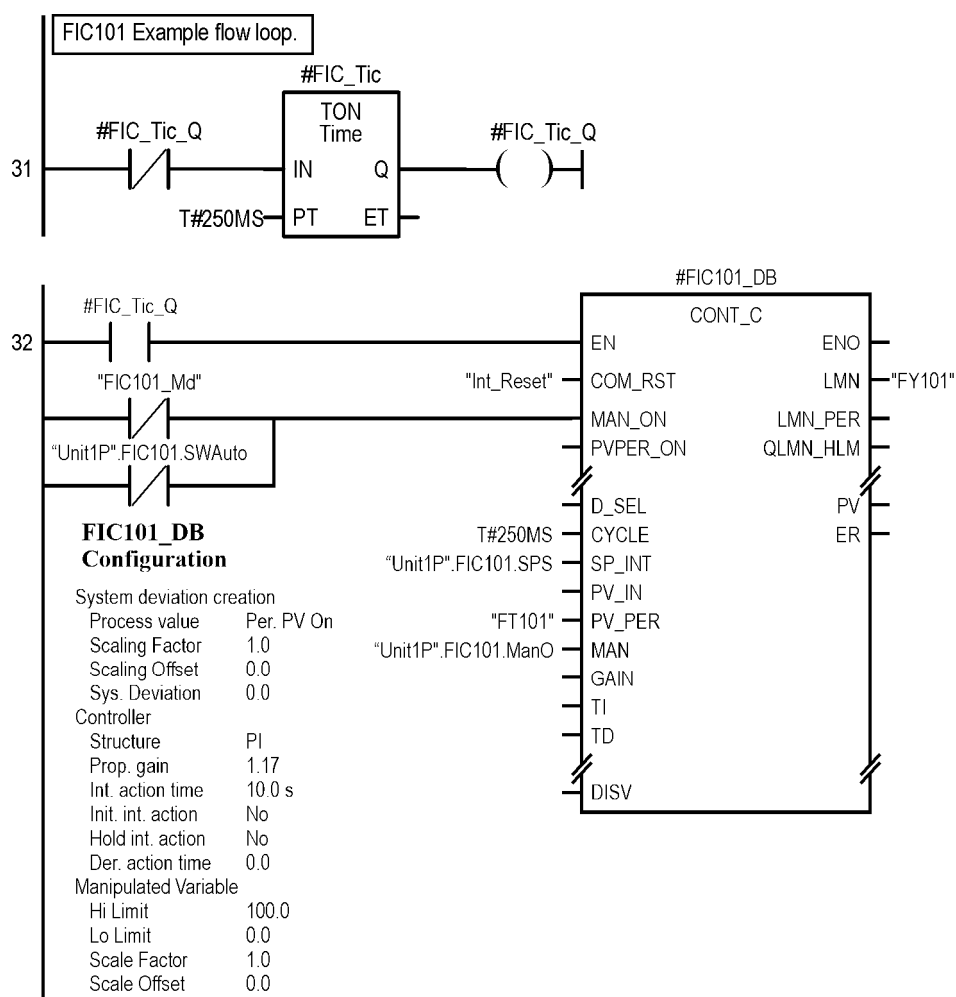
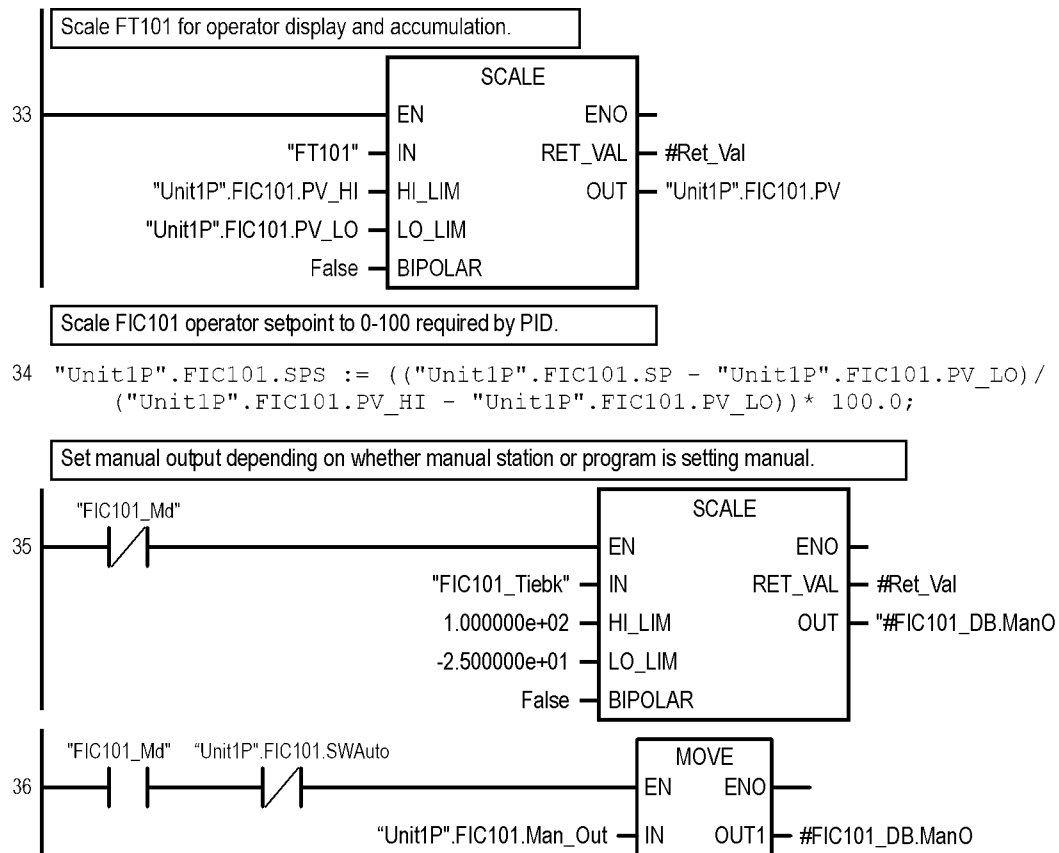


Figure 34. Simple PID loop (*continued*).

**Figure 34.** (continued).

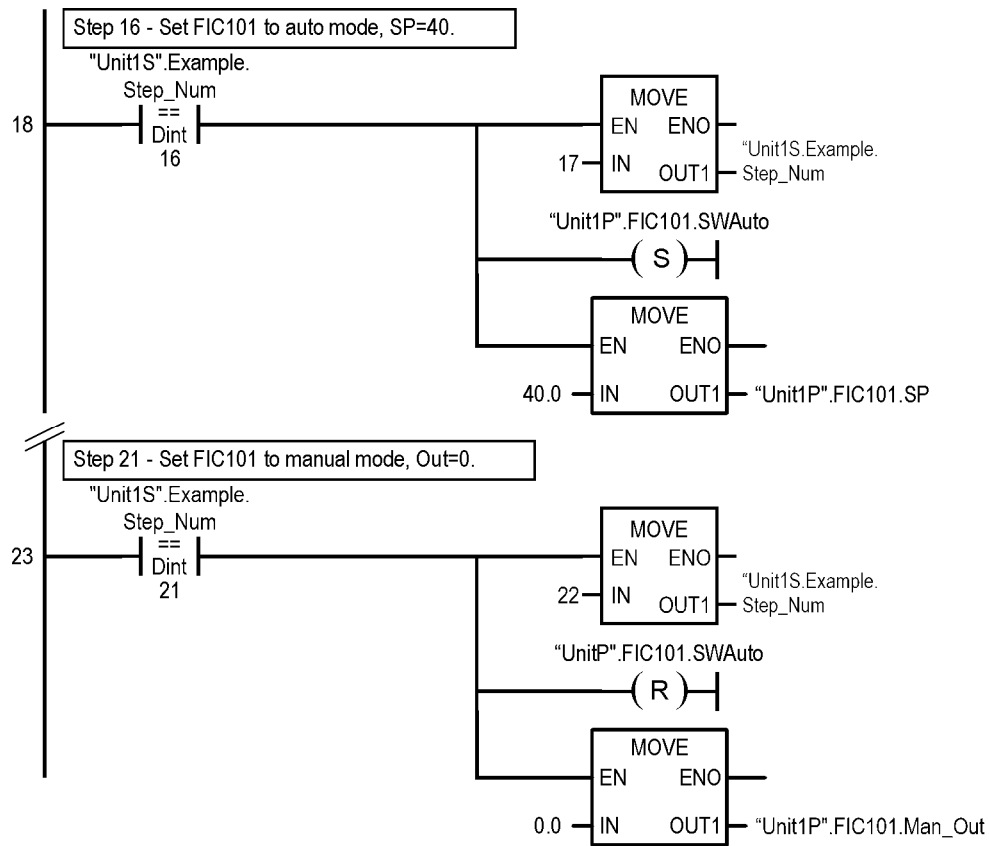


Figure 35. PID sequence commands.

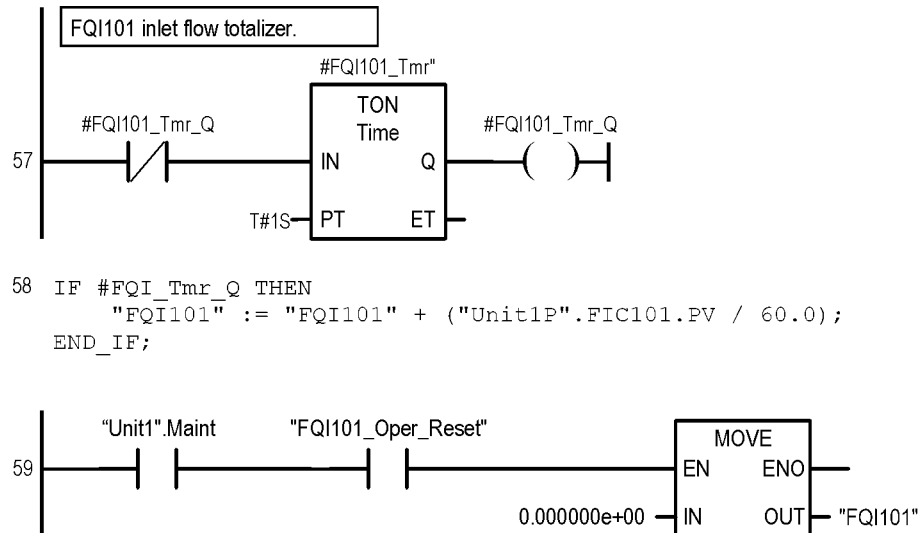


Figure 36. Flow totalization.

8. Analog Inputs and Outputs

The SCALE function block scales an integer (INT) from an analog input module to the desired engineering units. Note that if an analog value is used by a CONT_C function block, the CONT_C can perform the scaling. A typical scaling (for FT101) is shown in Figure 37. The example shown in Figure 37 assumes that the transmitter measures temperatures in the range of 90 to 700 °C and has an output range of 4 to 20 mA, but the analog input range is 0 to 20 mA. Therefore, one must artificially lower the LO_LIM by 25% of the normal range to account for the 4 mA low value of the transmitter corresponding to 90 °C. Hence, the LO_LIM of Figure 37 is $90 - (700-90)/4 = -62.5$. With this change, when TT101 is 5530 (4 mA), TI101 is 90.

In a similar manner, the values for an analog output channel may need to be de-scaled to the units of the actuator (if not driven by a CONT_C block). The UNSCALE function block scales a real number in engineering units to an integer (INT) for an analog output module. Except for the IN and OUT data types, the UNSCALE function block has the same input and output connections as the SCALE function block. The IN real number is scaled to the OUT integer. An example de-scaling (for FY101) is shown in Figure 38. As for the scaling example, if the analog actuator has a range of 4 to 20 mA, the LO_LIM must be lowered by 25% of the normal range.

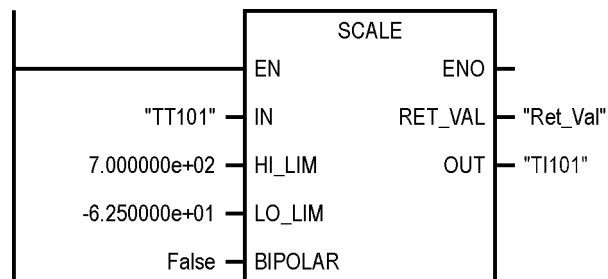


Figure 37. Example SCALE function block.

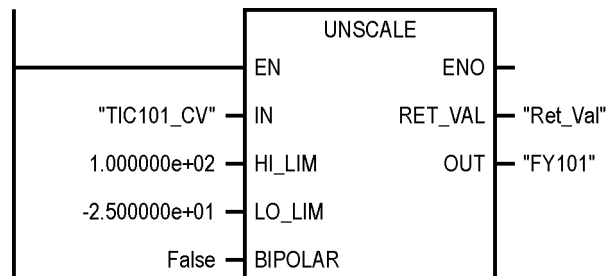


Figure 38. Example UNSCALE function block.

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 39 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 40 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 40 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%.

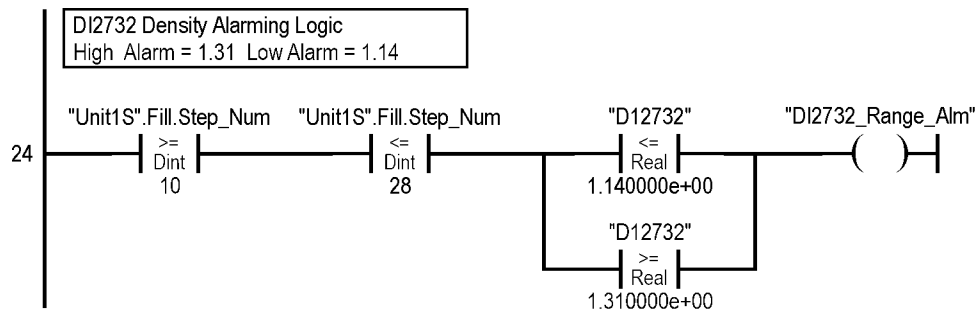


Figure 39. Density range alarm logic.

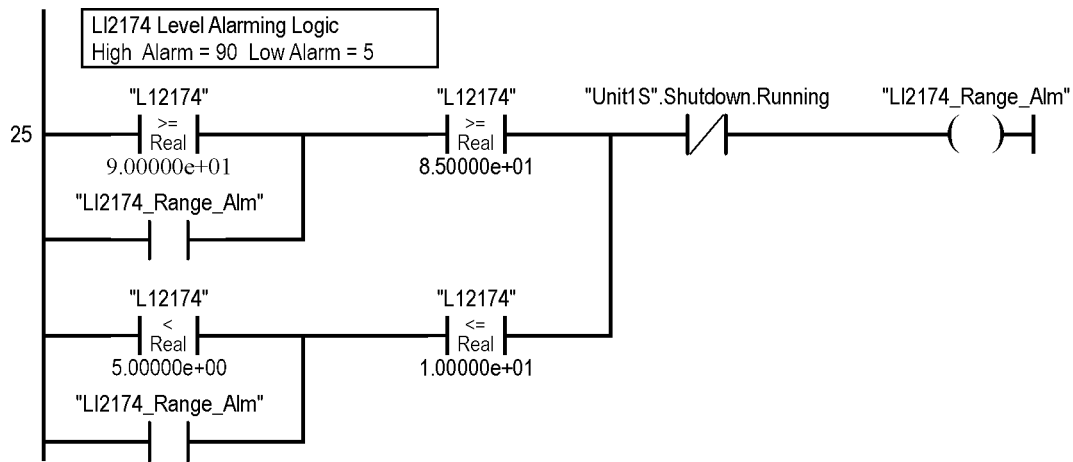


Figure 40. Level high and low alarm logic with deadbands.

The temperature alarm in Figure 41 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.

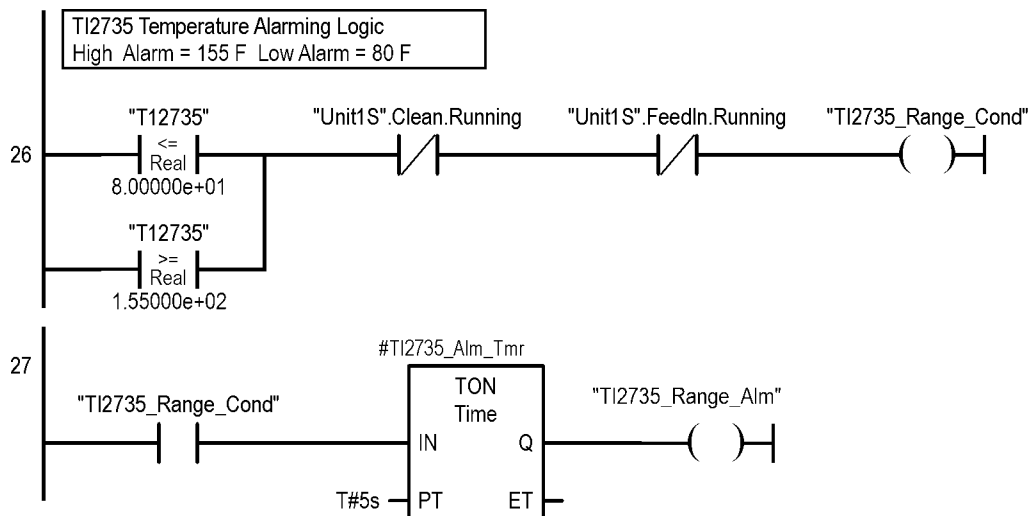


Figure 41. Temperature alarm logic with debounce timer.

10. Variable Name 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 variable 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 variable. The comment associated with the variable name is not displayed with the program, however, it should be descriptive. For example, a pump, with a tag number P-2700, used in a cooling tower operation in the reactor area shall be commented as:

“P-2700 Reactor Cooling Tower Pump A”

Other variable naming conventions:

Sequences

The sequence-related tags are:

Unit_Tag + "S." + Seq_Name	Defined as UDT91
Unit_Tag + "S." + Seq_Name + ".Step_Num"	Sequence step number
Unit_Tag + "S." + Seq_Name + ".Auto_Start"	Sequence auto start bit
Unit_Tag + "S." + Seq_Name + ".Auto_Ons"	Sequence auto-start one-shot storage bit
Unit_Tag + "S." + Seq_Name + ".Req_Tmr"	Sequence auto-start timer
Unit_Tag + "S." + Seq_Name + ".Running"	Sequence running indication
Unit_Tag + "S." + Seq_Name + ".Ons"	Sequence transition storage to start
Unit_Tag + "S." + Seq_Name + ".PC_Start"	Sequence start request from PC (remote)
Unit_Tag + "S." + 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, on when controlled from LTP; off when controlled from remote PC.
Unit_Tag + ".Maint"	For a group of sequences, on 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:

"Unit1M". + Equip_Tag + ".Seq_Start"	Start motor with sequence step
"Unit1M". + 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:

"Unit1M". + Equip_Tag + ".Run_Status"	Running status of motor
---------------------------------------	-------------------------

Conveyors

Commands:

"Unit1M". + Equip_Tag + ".Seq_Start" Start conveyor with sequence step
 "Unit1M". + 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:

"Unit1M". + Equip_Tag + ".Run_Status" Running status of conveyor

Valves**Commands:**

"Unit1V". + Equip_Tag + ".Seq_Open" Open valve with sequence step
 "Unit1V". + 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:

"Unit1V". + Equip_Tag + ".Open_Status" Open status for valve
 "Unit1V". + Equip_Tag + ".Close_Status" Closed status for valve

Slide Gates**Commands:**

"Unit1M". + Equip_Tag + ".Seq_Open" Open slide gate with sequence step
 "Unit1M". + Equip_Tag + ".Seq_Close" 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:

"Unit1M". + Equip_Tag + ".Open_Status" Open status for gate
 "Unit1M". + Equip_Tag + ".Close_Status" Closed status for gate
 "Unit1M". + Equip_Tag + ".Run_Status" Running status of motor

Flop Gates**Commands:**

"Unit1G". + Equip_Tag + ".Seq_Left" Move flop gate to divert left with seq. step
 "Unit1G". + Equip_Tag + ".Seq_Right" Move flop gate to divert right with seq. 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:

"Unit1G". + Equip_Tag + ".Left_Status" Left position status for gate
 "Unit1G". + Equip_Tag + ".Right_Status" Right position status for gate

PID Loops**Commands:**

"Unit1P". + Loop_Tag + ".SWAuto" Set: loop set to auto mode

"Unit1P". + Loop_Tag + ".Man_Out"	Reset: loop set to manual mode
"Unit1P". + Loop_Tag + ".SP"	Set loop output in manual mode
Indications:	Set loop setpoint

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. Revision History

Rev 0	2005 May 31	First release
Rev 1	2010 Sep 26	Added Motor_Conv device. Revised sequence standard
Rev 2	2011 Jun 19	Added AG_SEND/AG_RECV to FCs. Moved PID_Loops FC to FC7.
Rev 3	2014 Jly 05	Changes for Step7 TIA
Rev 4	2016 Jan 07	Changed to use TSEND_C/TRCV_C for S7-1200/1500 communication. Code using timers corrected for S7-1200/1500.
Rev 5	2024 Jan 29	Revised to use move-based sequencer and only for S7-1500