



APDL Programmable Logic Controller Application Specification

SD-2032

ISSUED July 14, 2025
REVISED

© 2025 Nexteer Automotive

All rights reserved.

This page intentionally blank.

Table of Contents

1.	Scope and Purpose.....	7
1.1	Scope.....	7
1.2	Purpose and Objective	8
1.3	Critical Principle - Control Functions in the Event of Failure	8
2.	Requirements (associated routine name).....	9
2.1	Main Program Control (R00_Main routine)	9
2.2	Mode Selection (R01_Mode Routine).....	10
2.3	Precondition and Initiate Machine Cycle (R03_Cycle Routine).....	11
2.4	Signal Conditioning (R04_Analog Routine).....	15
2.5	Machine Sequence (R05_Sequence Routine)	16
2.6	Part Quality Logic (R06_Quality Routine).....	20
2.7	Solenoid Control (R07_OutputMotions Routine)	24
2.8	Machine Diagnostics – Display Control (multiple routines)	31
2.9	Machine Diagnostics – Conditions and Detection logic.....	32
2.10	Routines Required On All Machines	34
3.	Required Logic Design - Application Specific	36
3.1	Light Curtain Interruption.....	36
3.2	Motor Starter Control.....	37
3.3	Shift Register / Indexing Logic	38
3.4	Pallet Release / Pallet Memory	39
3.5	Indicator Lights	39
3.6	HMI Requirements for Synchronous Transfer Systems (Multiple HMIs)	40
A.	Annex A - Machine Diagnostics Scheme and Hierarchy	42
B.	Annex B - Controller: Organizer, Structure, Names, and Instructions	44
C.	Annex C – Complex or Special Sequence Considerations	57
D.	Annex D - Cycle Pause - Pausing a Cycle.....	60
E.	Annex E - Glossary	62
F.	Annex F - References	63

List of Figures

Figure 1: Deterministic Update of Inputs	9
Figure 2: Reset All Memories Examples.....	13
Figure 3: Reset String Memory Example	13
Figure 4: Reset All Memories Logic Rung.....	14
Figure 5: Sequence Structure - Preferred – Two Machine Tasks / Two Rungs.....	16
Figure 6: Sequence Structure – Allowed – Two Machine Tasks / Two Instructions	17
Figure 7: Sequence Structure – NOT ALLOWED – Two Machine Tasks / One Instruction...	17
Figure 8: Reset Sequence Logic.....	18
Figure 9: Examples of Minimum Collision Avoidance Logic.....	24
Figure 10: Example Use Of Collision Avoidance Logic	25
Figure 11: Example Auto Allow Logic (Motion Towards the Work Position)	25
Figure 12: Example Auto Allow Logic (Motion Toward Home Position).....	25
Figure 13: Example Solenoid Control Rung	26
Figure 14: Remove Solenoid Power	27
Figure 15: Example Single Solenoid Motion Valve Logic.....	28
Figure 16: Example Motor Starter (Start/Stop) Logic	30
Figure 17: Example Sequence-Controlled Motor Starter Logic.....	30
Figure 18: Nexteer_Library Controller Organizer.....	34
Figure 19: Multiple Station Controller Organizer Views	35
Figure 20: Light Curtain Interruption	36
Figure 21: Example Motor Starter Logic	37
Figure 22: Controller Organizer Routines.....	45
Figure 23: Controller Organization – I/O Configuration for Single Station.....	47
Figure 24: Controller Organization – I/O Configuration for Multiple Station	Error!
Bookmark not defined.	
Figure 25: Library UDTs	49
Figure 26: Torque Process Example (Support must remain <i>lowered</i>).....	Error! Bookmark not defined.
Figure 27: Oetiker Clamp Process Example (Lift must remain <i>lowered</i>).....	Error! Bookmark not defined.
Figure 28: Auto Allow Collision Avoidance for Cycle Pause	60

List of Tables

Table 1:	I/O Tag Consistency – Local and Distributed I/O.....	51
Table 2:	I/O Tag Consistency – Distributed I/O.....	52
Table 3:	I/O Tag Consistency – Code Reader	53
Table 4:	Routine and Tag Name Consistencies (Set 1).....	54
Table 5:	Routine and Tag Name Consistencies (Set 2).....	55
Table 6:	Tag Description Examples.....	56

1. Scope and Purpose

1.1 Scope

- 1.1.1 This specification describes programmable logic controller (PLC) logic design functional requirements and format for Nexteer Automotive AP DriveLine facilities. This specification shall be used by the Original Equipment Manufacturers (OEM) in their design of PLC systems.
- 1.1.2 This specification applies to the purchase of new equipment and control system rebuilds. It should not be implied that any existing equipment is required to be retrofitted to comply with this specification.
- 1.1.3 This specification references two associated PLC logic files:
Nexteer_TIAxx_PLC_HMI_rev_date and *Nexteer_TIAxx_Library*. These Nexteer logic files (collectively referred to as templates) reflect the requirements of this specification (*Nexteer Library*); they provide additional logic routines for specific applications (*Nexteer Library*); The PLC logic files are available at www.nexteerdatabase.com.
- 1.1.4 Additional applications specific guidelines that include PLC logic-related topics (such as HMI operator interface, RFID, or traceability) are also available at www.nexteerdatabase.com.
- 1.1.5 The use of the word “shall” indicates requirements and the use of the word “should” indicates recommendations. The use of the word “may” indicates permission or allowance and the use of the word “can” indicates a possibility.
- 1.1.6 This specification is structured as follows.
 - 1. Requirements and guidance are detailed within Section 2.
 - 2. Additional application specific requirements are detailed in Section 3.
 - 3. Nexteer’s machine diagnostic philosophy, scheme, and hierarchy are described in Annex A. An understanding first of Annex A’s philosophies will aid in understanding the machine diagnostics requirements of Section 2.
 - 4. The PLC logic structure and organization is summarized in Annex B.

1.2 Purpose and Objective

- 1.2.1 The purpose of this specification is to provide Nexteer requirements and guidance to Original Equipment Manufacturers (OEM) for use in their design of PLC logic.
- 1.2.2 The objective of this specification is to provide common, maintainable, and cost effective controls systems that enhance both the productivity and ease-of-use of the systems, plus ensure the quality of Nexteer products produced. The application of this specification will result in common controls systems software that:
 - 1. ensures the machine processes the part correctly. To correctly process the part, the machine logic design needs to include significant consideration for the control functions in the event of failure such that the machine is not capable of processing the part incorrectly. Aspects of control functions in the event of failure are discussed in detail throughout this specification.
 - 2. provides ease of customer use. Ease of customer use relates to logic that provides plant personnel a quick understanding as to how the machine processes the part, logic that can be quickly used to troubleshoot failures, and logic that can be easily used to verify part quality. The Nexteer templates provide common structure and naming conventions for the purpose of improved plant production, independent of which OEM supplied the equipment.
 - 3. facilitates the OEM design and Nexteer logic approval process. Nexteer's specifications require logic/software approval prior to MQ1. Nexteer's approval process, adherence to this specification, and use of the templates, provides an opportunity for the OEM to demonstrate compliance to the requirements.

1.3 Critical Principle - Control Functions in the Event of Failure

- 1.3.1 Control Functions in the Event of Failure: the controls systems software design shall include appropriate measures such that failures within the electrical equipment do not cause the system to incorrectly process the part, and failures within the electrical equipment shall not cause the system to qualify a Reject Part as a Good Part. Appropriate measures shall include detection of, and indication of, such failures.

To clarify: Nexteer specifications and templates use the terms "Back check", "Back checking", or "Back checked" to indicate the logic that takes appropriate measures to protect against such failures.

Back checking is a phrase that Nexteer uses relating to logic that both verifies the proper input device operation and also detects input failure.

Back checking verifies the operation (action) of the input device. Back checking also verifies the operation (action) of the PLC input card electronics.

Back checking also verifies the operation (action) of communications, whether parallel or serial, such that part process and quality is based upon up-to-date (actual and current) data, not based upon stale data (retained, old, or previous-part data).

Failure detection (back checking) may either stop the machine immediately, or disallow the start of the next cycle, depending on the application.

2. Requirements (associated routine name)

Nexteer's functional requirements are described within this chapter. Each clause of this chapter details a logic topic and typically indicates which Nexteer routine(s) is associated with that logic topic.

The routines provided in the Main Program of the logic Nexteer_Library file shall be used for all applications; programmed on all equipment even when the equipment does not include any logic within the routine. Additional routines from the Nexteer_Library program are detailed elsewhere within this specification. *Note: The MultiStation example uses all of the required routines, distributing them appropriately between the Main Program and the station programs.*

Nexteer's logic organization, structure, and naming conventions are described in Annex B.

2.1 Main Program Control (R00_Main[OBx] routine)

Requirements:

- 2.1.1 The routine named Main shall be assigned as OB (within the main program's configuration properties, and within the configurations properties for all station programs for a multiple station system).
- 2.1.2 The Main routine shall include logic that controls the deterministic (once-per-scan) update of discrete and analog I/O tags for all module and slot-based signals (module data). Input tags shall be mapped **from** the module data; output tags shall be mapped **to** the module data. The I/O tags shall be used throughout the logic.

Note: Aliasing does not accomplish deterministic, once-per-scan updates.

Note: Communications with auxiliary devices such as cameras and servos, when mapped within a device-associated routine, are not required to be mapped in the Main routine.

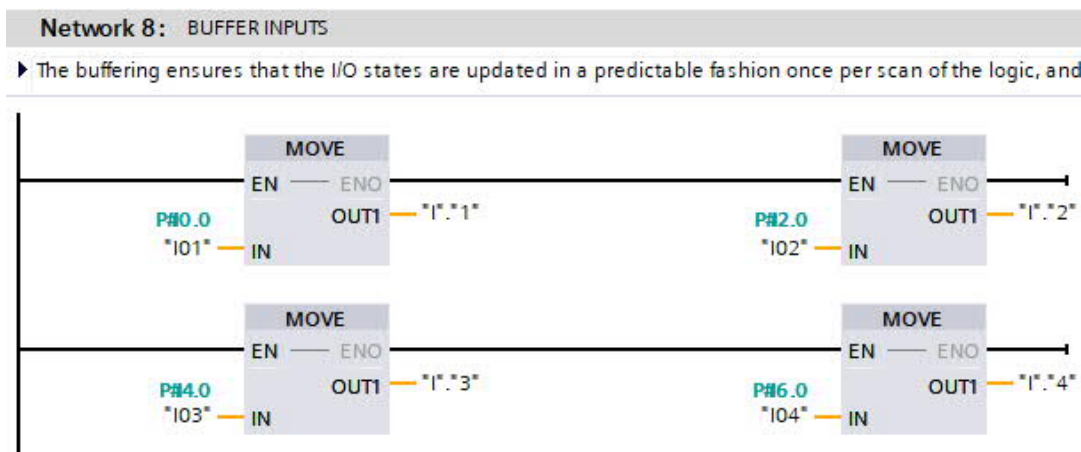


Figure 1: Deterministic Update of Inputs

- 2.1.3 Each program's Main routine shall include logic that unconditionally calls (jumps to) all other routines of the program. The routines shall be called in the same rung-order as is visible in the controller organizer.

Guidance:

- 2.1.4 The only additional logic that should be included in the Main routine is miscellaneous logic, such as the logic shown within the templates' Main routines.
- 2.1.5 The Main routine may include general logic for indicator lights.
- 2.1.6 The deterministic (once-per-scan) update of station I/O tags on multiple station equipment can be programmed within the controller's Main program or within each associated station's Main program.

2.2 Mode Selection (R01_Mode Routine)**Requirements:**

- 2.2.1 The Mode routine shall include all logic that controls the selection of modes.
- 2.2.2 The machine shall power up with no mode active. After selection, one mode, and only one mode, shall be active. *Note: Being in an E-Stop condition should not deactivate a mode selection.*
- 2.2.3 At a minimum, all machines shall include two modes – Manual and Automatic. *Note: Machine motion shall only be enabled when a mode is active.*
 - 1. *Manual Mode* - Manual mode allows individual motions to be commanded. Manual mode is not a forced logical "step through the machine sequence", but is a means for operators and maintenance to exercise any individual motion. The system shall not be allowed to switch to manual mode (from auto mode) while the machine is in cycle.
 - 2. *Automatic Mode* - Automatic mode is the mode that allows normal machine cycles and prohibits manual motions. Automatic mode does not initiate any machine motions.

Guidance:

- 2.2.4 Other operating modes may be included on the machine. However many additional machine processes are typically subcategories of Manual mode or Automatic mode and are not as such an additional type of mode. Refer to the Cycle routine section below for examples of Return All and Calibration which are both a type of cycle – not a mode.
- 2.2.5 The requirements for PSDI, an additional type of cycle that is allowed for specific applications, are detailed in SD-011.
- 2.2.6 For multiple station equipment with multiple programs, mode selection logic should be programmed within both the Main program's Mode routine and within each station's Mode routine.

2.3 Precondition and Initiate Machine Cycle (R03_Cycle Routine)

Requirements:

- 2.3.1 The Cycle routine shall include the logic that is a precondition to the machine sequence, including logic that indicates initial conditions, indicates initial positions, controls the initiation of Machine In Cycle (MIC), and controls the Return All (Homing) logic for machine motions.
- 2.3.2 The Cycle routine shall include the following functions and output-energize instructions:
1. Cycle_ResetAllMemories – output-energize instruction that resets memories affecting, storing, or otherwise relating to part status and part quality. Requirements are detailed below.
 2. Cycle_AllReturned – output-energize instruction indicating that all motions are returned or retracted to the typical home position, based on positional indication consisting of XIO contacts from each returned sensor input and returned position for analog sensors.
Note: For any actuator that is sensed with only one motion-direction sensor, and that sensor is an advanced sensor, then an XIC contact from the advanced sensor input shall be included in the Cycle_AllReturned logic.
 3. Cycle_MemoriesAreReset – output-energize instruction indicating that all memories affecting, storing, or otherwise relating to part status and part quality have been reset or nullified.
 4. Cycle_InitialConditions – output-energize instruction indicating the combination of conditions required to allow the initiation of machine cycle.
 5. Cycle_StationArmed – output-energize instruction, used for each station on an asynchronous assembly line, indicating and allowing the station to go into cycle. Station Armed shall be operator-initiated by an HMI pushbutton.
Note: Station Armed allows the machine in cycle (MIC); MIC is a separate requirement detailed below. MIC for each station on an assembly line is typically initiated upon Pallet Presence with a part ok to be worked on (but only when the StationArmed is energized). Operator de-energizing StationArmed can be used to hold a pallet from cycling until the station is, again, intentionally Armed.
 6. Cycle_AllowCycleStart – output-energize instruction indicating combined returned, initial, and other conditions required to allow machine cycle start or allow the machine cycle to be restarted.
Note: Machine conditions such as the main air pressure or hydraulic oil temperature can be programmed in the initial conditions above, here in the allow cycle start, or in machine faults depending on the application.
 7. Cycle_CycleStartPulse. The cycle start logic is application specific.
 8. Cycle_AbortPulse. An HMI button-initiate pulse to abort the current machine cycle; the abort cycle logic shall be included on all machines.

9. Cycle_MIC – output-energize instruction indicating and allowing machine in cycle with requirements as detailed below.
 10. Cycle_ReturnAll – output-energize instruction initiating and allowing return-to-home, or return to initial positions as detailed below.
- 2.3.3 Machine In Cycle (MIC) shall be the control function, and output-energize instruction, that makes a machine capable of producing automatic (sequenced) motions. Motion shall occur only when MIC is energized.

To clarify: MIC shall be the one output-energize instruction that enables sequenced motions for the duration of the cycle. Auto mode shall not be used throughout the logic to allow motions. Non-motion processes (such as reading an RFID tag or communicating with traceability) may be initiated prior to, and/or independent of, MIC.

Note: On an asynchronous assembly line MIC is typically not required to produce conveyance and non-hazardous pallet control motions.

Note: The Nexteer_Library includes multiple MIC output-energize instructions; all but one output-energize instruction shall be removed from the logic.

- 2.3.4 The machine shall be allowed to enter into cycle (MIC shall energize) only when all of the following conditions are met:
1. Automatic mode is selected.
 2. All motions and devices are in their initial state (typically, indicated by all of the returned sensors being ON).
 3. No faults are present on the machine.
 4. All safety devices are in the “safe” condition.
 5. A new part has been loaded or has entered into the machines.
 6. The station is Armed (for stations on an asynchronous assembly line).
- 2.3.5 Machine cycle shall be initiated by the operator. On single cycle machines, machine cycle shall be initiated by operator actuation of a hardwired device(s). On continuous cycle machines, machine cycle should be initiated by an HMI pushbutton. Exception: for each station on an asynchronous assembly line, Station Armed shall be operator-initiated by an HMI pushbutton; machine cycle shall be initiated upon the presence of a pallet and the part is ok to be worked on.
- Note: Machine motion shall not occur based on mode selection.*
1. Single cycle machines execute one complete cycle for each initiation by the operator.
 2. Continuous cycle machines execute repetitive cycles until halted by operator action or a fault condition. The first cycle shall be initiated by the operator.
 3. A Cycle Stop pushbutton (either hardwired or on the HMI) shall be provided on continuous cycle machines. When the Cycle Stop pushbutton is pressed the machine is allowed to finish processing the part, return the machine to its normal starting position, and then MIC is de-energized.

4. Logic controlled by a hardwired cycle start device shall include a one-shot to verify that a failure of the device ON does not cause consecutive cycles to occur.
- 2.3.6 Reset All Memories shall be the control function, and output-energize instruction, that resets memories affecting, storing, or otherwise relating to part status and part quality.

To clarify: Memories include all data types (such as DINTs and STRINGS), as well as BOOL tags that have been sealed-in.

To clarify: Part status and part quality memories include Good Part, Reject Part, part test results, and any part data storage from the previous cycle. Part status and part quality memories also include previous cycle data from a pallet on pallet transfer systems, and shift registers on other part-indexed systems.

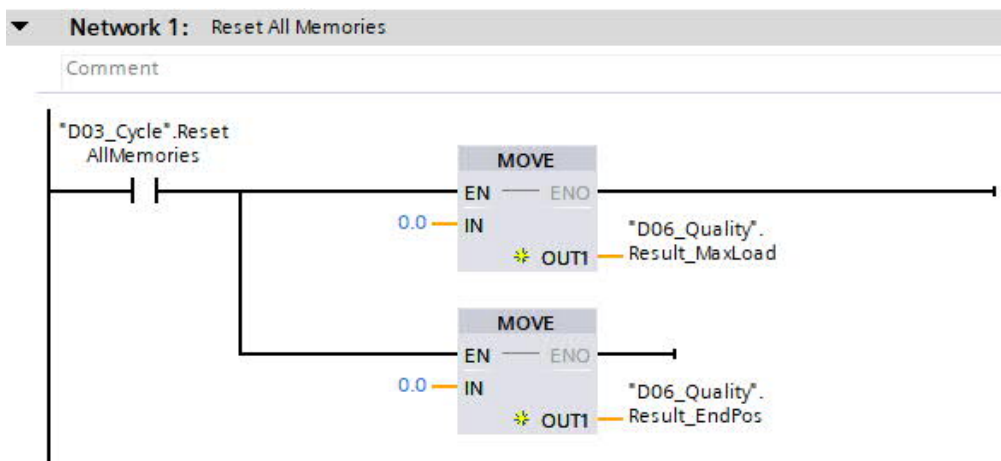


Figure 2: Reset All Memories Examples

*To clarify: Resetting STRING quality memories is accomplished by filling the string tag with null characters (ASCII \$00) in the data, but is **not** accomplished by merely resetting the string length to zero.*

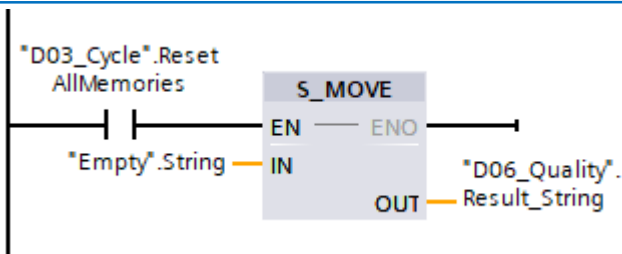


Figure 3: Reset String Memory Example

- 2.3.7 Logic to initiate Reset All Memories is application specific. At a minimum, Reset All Memories shall be initiated upon power-up and upon removal of the part (see Figure 4 below).

To clarify: Removal of the part includes the release of a pallet on conveyor lines, or the initiation of index (shift pulse) on dial tables and indexing machines.

To clarify: Removal of the part includes interruption of a light curtain on operator unload machines that do not include Part Presence sensing. (It shall be presumed that the part has been removed).

To clarify: On an automatic part-transfer system that does not include Part Presence sensing at fixtures, and mechanically allows the part to be removed from the fixture, removal of the part includes interruption of an interlocked safety gate. (It shall be presumed that the part has been removed).

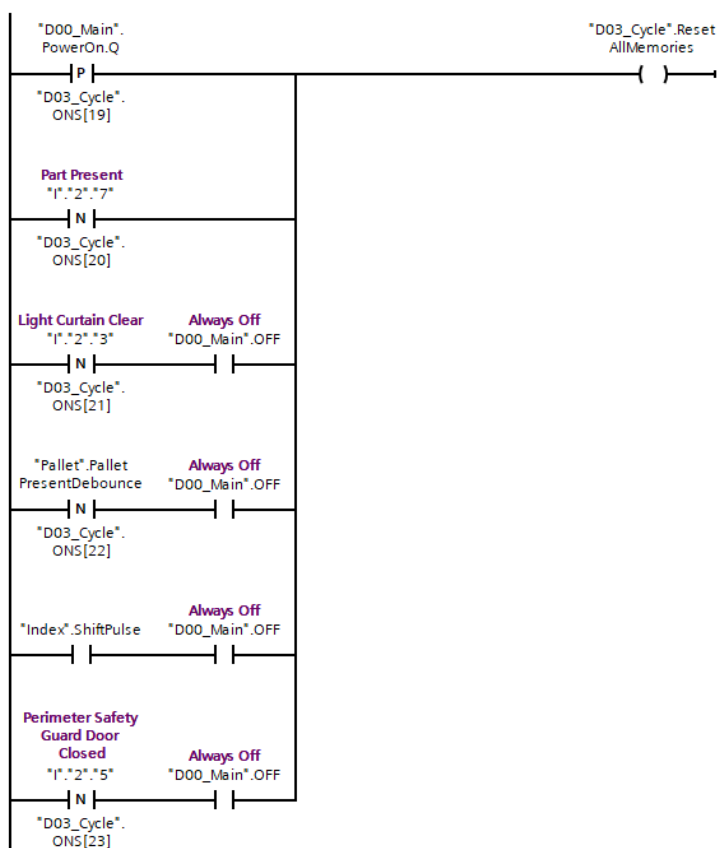


Figure 4: Reset All Memories Logic Rung

2.3.8 The Return All output-energize instruction shall only be enabled when a mode is active.

1. The Return All output-energize instruction can be enabled in Manual and Automatic mode, however, the Return All output-energize instruction shall not be enabled when Machine In Cycle is active.
2. When in Manual mode the Return All output-energize instruction shall only be enabled when Return All pushbutton input remains enabled.
3. When in Automatic mode, the Return All output-energize instruction may be enabled through one of two methods:

- a. when the Return All pushbutton input remains enabled, or
 - b. as an automatic or sequenced Return All Cycle (initiated by momentarily pressing the Return All pushbutton).
4. Logic controlled by a hardwired Return All pushbutton shall include a one-shot or pulse to verify that a failure of the pushbutton or input does not cause machine motion.
- 2.3.9 Additionally, special-purpose machine cycles such as a Calibration Cycle, shall only be enabled when in automatic mode.

Guidance:

- 2.3.10 The Cycle routine may include logic for cycle-related indicator lights.
- 2.3.11 The retentive CycleTime timer provided for HMI display may be programmed in the Cycle routine or an HMI routine.
- 2.3.12 Logic to verify operator tasks such as part pre-assembly may be programmed within the Cycle routine or within the Sequence routine. However, operator tasks that are required to occur in a specific sequence or specific order shall be programmed within a sequence routine. Refer to the Machine Sequence details below.

2.4 Signal Conditioning (R04_Analog Routine)**Requirements:**

- 2.4.1 Analog signals shall be verified to move from a “reject” value to a “within-limits” value. The signal shall be verified to have returned to the reject value or range (typically to a known initial position) as part of initial conditions in order to allow the start of the next cycle.

To clarify: The logic shall detect and prevent a common failure from classifying the part as a Good Part, such as a broken wire that allows a signal to drift into the good part range.

Guidance:

- 2.4.2 The analog routine should include the logic that controls the scaling and calculating of all analog and similar signals.
- 2.4.3 The analog routine may include the logic to compare the signals to limits, including the back check logic.

2.5 Machine Sequence (R05_Sequence Routine)

Requirements:

- 2.5.1 The Sequence routine shall include all of the logic that steps through the machine cycle. The machine sequence includes stepping through all sequenced motions, stepping through the machine processes, and initiating each process-based action. *Note: Examples of process-based action steps include initiating a quality check, or initiating a communication.*

*To clarify: All sequence control logic shall be in the sequence routine. As an example: The logic for raising a cylinder and then engaging a rod-lock shall include two separate sequence steps programmed within the sequence routine – the rod-lock shall **not** be controlled solely by the cylinder controlling sequence step plus time-delay logic located within the OutputMotions routine.*

- 2.5.2 The Sequence routine shall include an output-energize instruction for each sequence-driven task.

To clarify: Each sequence-driven task includes: each machine function, each step, each process, and each command originated from the sequence. Each shall have an individual output-energize instruction in the Sequence routine. One output-energize instruction shall not initiate more than one machine task. Two output-energize instructions are required even when identical machine conditions initiate each task.

As an example: The example process requires that the same conditions both Step202 and raise Step203. The sequence routine includes a separate output-energize instruction for both Step202 and Step203 (see Figure 5, 6, and 7 as follows).

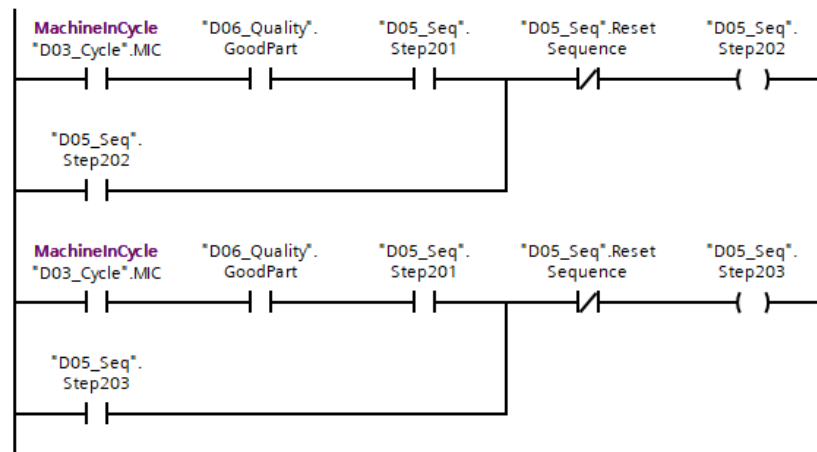


Figure 5: Sequence Structure - **Preferred** – Two Machine Tasks / Two Rungs

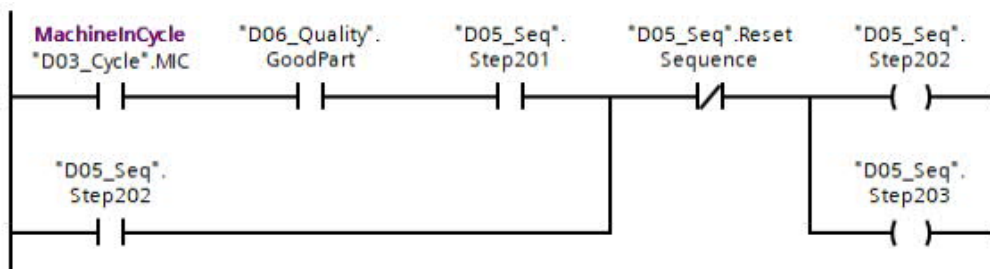


Figure 6: Sequence Structure – **Allowed** – Two Machine Tasks / Two Instructions

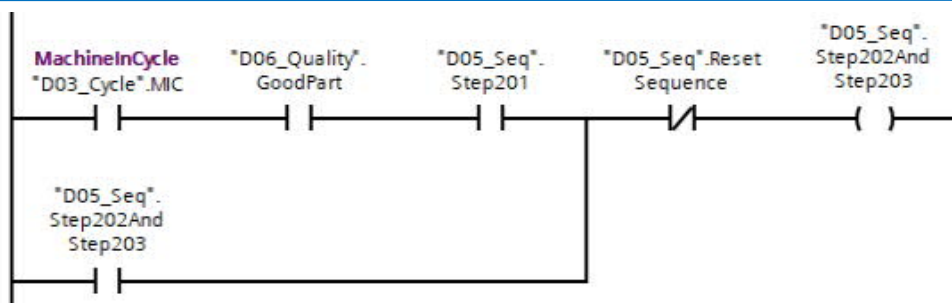


Figure 7: Sequence Structure – **NOT ALLOWED** – Two Machine Tasks / One Instruction

2.5.3 The Sequence routine shall include only sequence and process steps.

*To clarify: Sequence logic includes the output-energize instructions for process steps such as initiating data monitoring, initiating sending traceability data, or initiating quality limit checks. However, the logic that performs the quality limit checks is to be programmed within the *Quality* routine - **not** within the sequence routine.*

2.5.4 Each sequence-driven task shall be verified to have been completed within the sequence routine.

*To clarify: Each output-energize instruction within the sequence routine can be considered as an “output” **from** the sequence routine to another routine. Similarly then, each contact or signal from other routines (or input devices) **into** the sequence routine can be considered as a “completed” input **to** the sequence from the other routines. Therefore the sequence routine logic needs to verify that each output **from** the sequence routine receives a completed input **into** the sequence.*

As an example: When an RFID write is initiated by the sequence, a write-completed contact from the RFID routine must be used within the sequence routine to ensure a properly cycled part.

- 2.5.5 Each sequence step rung shall include a normally-opened (XIC) contact from Machine In Cycle MIC. The first automatic sequence step is nearly always initiated only by MIC. Exception: Logic included in the sequence routine that verifies operator tasks such as part pre-assembly, and are not required to occur in a specific sequence, typically should not include the MIC contact.
- 2.5.6 The Sequence routine shall include a reset sequence output-energize instruction named “Seq_ResetSequence”. The reset sequence output-energize instruction shall be the first solved rung within the Sequence routine (for solve-order reasons). The reset sequence output-energize instruction shall have a normally-closed (XIO) contact included in each sequence step rung, to reset the entire sequence. The ResetSequence rung shall be structured as follows (see Figure 8 below):

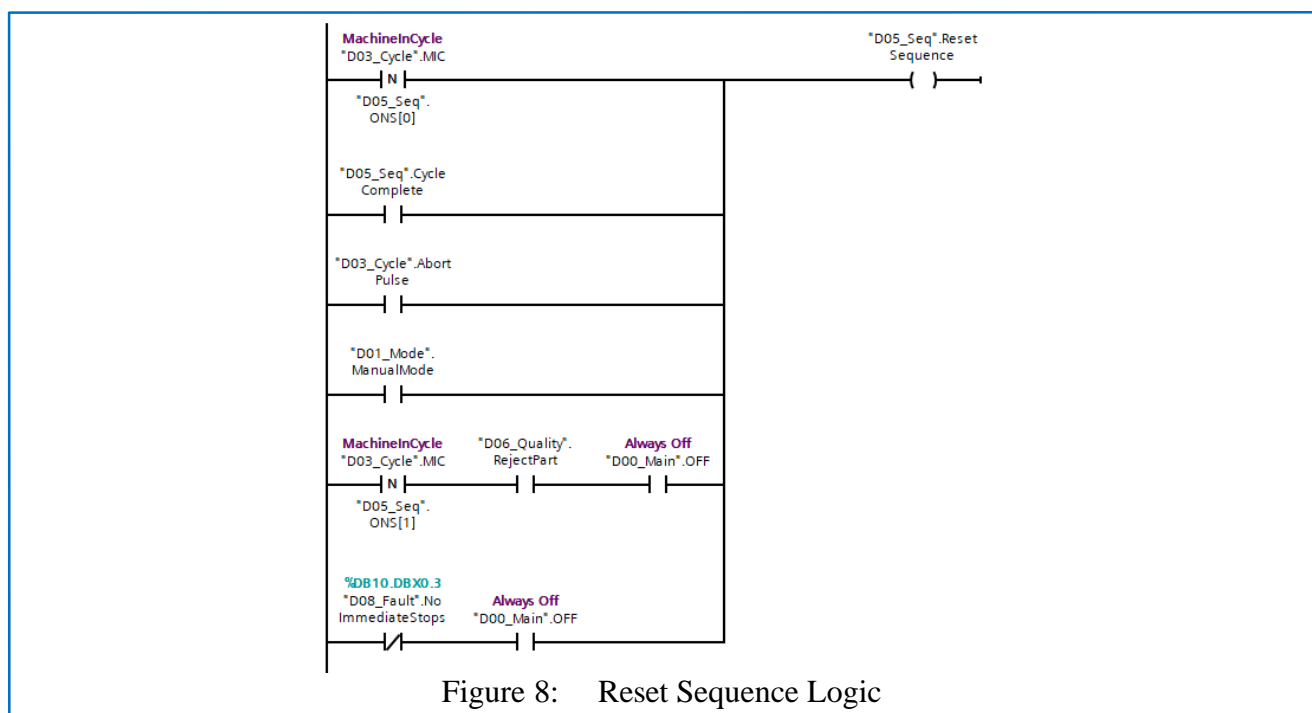


Figure 8: Reset Sequence Logic

- 2.5.7 The sequence shall be reset upon:
- loss of MIC.
 - after cycle complete.
 - the first scan after a cycle is aborted (an AbortPulse).
 - selection of manual mode.
- 2.5.8 From the Nexteer_Library logic, branches that include an Always-Off instruction are shown as optional and are dependent upon the application. The sequence can be required to reset upon:
- paused cycle (loss of MIC) while a reject part is present.
 - detection of an Immediate Stop fault.

2.5.9 Operator tasks (such as part pre-assembly) that are required to occur in a specific sequence shall be programmed within a sequence routine(s).

1. The assembly sequence shall follow the correct assembly order.

Note: Nexteer's manufacturing engineer purchasing the equipment details the required assembly sequence.

2. All associated sensors and error-proofing shall be monitored during the entire process step(s). Refer to the *Hand Assembly of Parts* portion of Annex C.

Guidance:

2.5.10 Logic in the sequence routine should be kept simple.

Note: Adherence to simple logic and format will assist the destination plant support personnel to more-readily understand the machine process.

2.5.11 Refer to Annex C for application details relating to more-complex, special, multiple, or customized sequence examples.

2.5.12 Step numbers should be in ascending, consecutive order. Gaps in the numbering scheme are allowed (such as step numbering 10, 20, 30). Sequence steps that execute simultaneously may have the same step number.

2.5.13 The sequence routine can include reject-control logic as detailed in the reject-handling requirements of the Part Quality Logic section below.

2.5.14 The logic templates include common reset-sequence examples. Additional sequence reset conditions that should be included in the sequence reset logic depend on each machine's application. Branches (from the Nexteer_Library logic) that include an Always-Off instruction are shown as optional and may be removed when not implemented by the application.

2.6 Part Quality Logic (R06_Quality Routine)

Requirements:

- 2.6.1 The Quality routine shall include all of the logic that determines the part quality, logically indicates the part quality status, and controls operator reject-part handling (based on the part quality status).
- 2.6.2 Part quality logic shall be designed to prevent qualifying a Reject Part as a Good Part.
- 2.6.3 The part quality logic shall reset Good Part status upon removal of the part.

Note: Use of a contact from the Reset All Memories as detailed in the Cycle routine above meets this specification requirement.

- 1. Where no Part Present signal is provided, the part quality logic shall reset a Good Part status upon change of mode.
- 2. Where no Part Present signal is provided at fixtures of an automatic part-transfer system that mechanically allow the part to be removed, the part quality logic shall reset Good Part status upon the opening of an interlocked safety gate for each fixture that can be accessed. *Note: On automatic part-transfer systems, solenoid-locking interlock switches should be considered to minimize the number of rejected (scrapped) parts.*
- 2.6.4 The part quality logic that interfaces with a part-quality-determining-device shall be designed to verify the device is operating as required. Devices and interface signals include sensors, auxiliary equipment (such as instrumentation), analog signals, and communication signals.

Refer to the Control Functions in the Event of Failure items under the Critical Principle section above.

- 2.6.5 The control system, including instrumentation and sensors, shall provide a PLC Good Part input when the desirable component, dimension, or feature is detected.
 - 1. The logic shall detect that the Good Part input transitions from OFF to ON, or from an out-of-limit value to a within-limit value. *Note: This input transition shall occur during the part-process, not at power-up of the system.*
 - 2. The logic design shall give the highest priority to classifying a part as a Reject Part, over classification as a Good Part.

*To clarify: At the logic scan for part quality check, the logic shall check for a reject first. As an example: If the instrumentation provides the PLC with both a Good Part and a Reject Part input, at the part quality check scan the logic shall classify a part as a Reject Part because the Reject Part input is ON. At the part quality check scan the logic shall then only classify as a Good Part if **not** a Reject Part and if the Good Part input transitions to ON. At the part quality check the logic shall also classify a part as a Reject Part if neither input transitions ON.*

- 2.6.6 Logic for part-quality data storage (the logic that stores or saves the part quality data into tags for use by traceability, or onto RFID) shall be located in the Quality routine.

Note: The logic that transfers the status byte(s) to traceability or writes the RFID data shall be in the appropriate traceability or RFID routines respectively.

- 2.6.7 Means shall be provided to confirm that reject parts are disposed of or handled properly. Reject part status and data shall be written to traceability or to RFID, and the write-complete verified, prior to the logic allowing any reject disposal, reject-handling, and reject removal as detailed below.
- 2.6.8 The Reject Part Present output-energize instruction shall seal-in until the reject-handling reset sequence is completed. The Reject Part Present output-energize instruction shall be maintained during power loss (typically through use of an OTL).
- 2.6.9 When error proofing includes communication of part status (such as communication to traceability or to on-the-part RFID) the logic shall require the following reject-handling:
1. The logic shall annunciate a rejected part is present.
 2. The communication of reject part status shall be included in the machine sequence logic.
 3. The machine should (typically) stop processing the part upon a reject and return to the home position.
 4. The logic shall verify that the part status has been communicated to the traceability or on-the-part RFID system.
 5. Upon completion of the reject part communication, the Reject Part Present output-energize instruction may be reset.
- 2.6.10 When error proofing includes hand-unload to a reject chute or bin, the logic shall require the following reject-handling reset sequence:
1. The logic shall annunciate a rejected part is present and prohibit the machine from cycling again until the acknowledgement process has been completed.
 2. The machine should (typically) stop processing the part upon a reject. *Note: Most machine motions are allowed to return to the home position.*
 3. The reject part shall remain clamped, or, where no part clamp is provided, at least one machine motion shall stay advanced to mechanically prevent the removal of the part.
 4. The logic shall require the operator to put the machine in Manual mode and unclamp the part (or retract the appropriate motion).
 5. The logic shall verify that the part has been removed from the machine, which may be accomplished by verifying that the part present sensor switches to an OFF state. If a part present sensor does not exist, other means, such as a light curtain being broken, may be used to indicate part removal.

6. The logic shall not permit the machine to be switched back into Automatic mode until the reject part has been placed into the reject chute or bin. The correct operation of the reject chute sensor shall be verified (such as logic requiring a transition from an OFF state to an ON state).
 7. Upon completion of the reject-handling reset sequence the Reject Part Present output-energize instruction may be reset. The machine is then permitted to go into Automatic mode.
- 2.6.11 For machines that include automatic unload of reject parts the logic shall meet the following reject-handling reset sequence.
1. The part shall be placed in a reject chute or repair loop, depending on the application.
 2. The reject chute sensor must transition from an OFF state to an ON state when the part passes down the chute. The sensor shall transition back to an OFF state in order to complete the acknowledgement process.
 3. Upon completion of the reject-handling reset sequence the Reject Part Present output-energize instruction may be reset and the machine may now be permitted to begin another cycle.

Guidance:

- 2.6.12 Applications may require a part to be rejected when there is a loss of MIC. Depending on the process, or depending on what point in the process the machine drops out of cycle, the part may be required to be rejected even though a quality check may not have occurred. Applications may require a part to be rejected if the cycle is interrupted after a particular process step has been started but not completed. Examples include heat treating, welding, and other processes.
- 2.6.13 Back check logic for part quality-related inputs including discrete, analog, or communication values may be programmed in the Quality routine although they are typically programmed in a fault routine.
- 2.6.14 Applications may require the part status to be classified as Part In Process (typically upon cycle initiation) until classified as a Reject Part or Good Part.
- 2.6.15 Logic to communicate the part status to traceability or RFID is typically not programmed in the Quality routine.
- 2.6.16 Logic for quality-related indicator lights may be programmed in the quality routine or in an output-related routine.

This page intentionally blank.

2.7 Solenoid Control (R07_OutputMotions Routine)

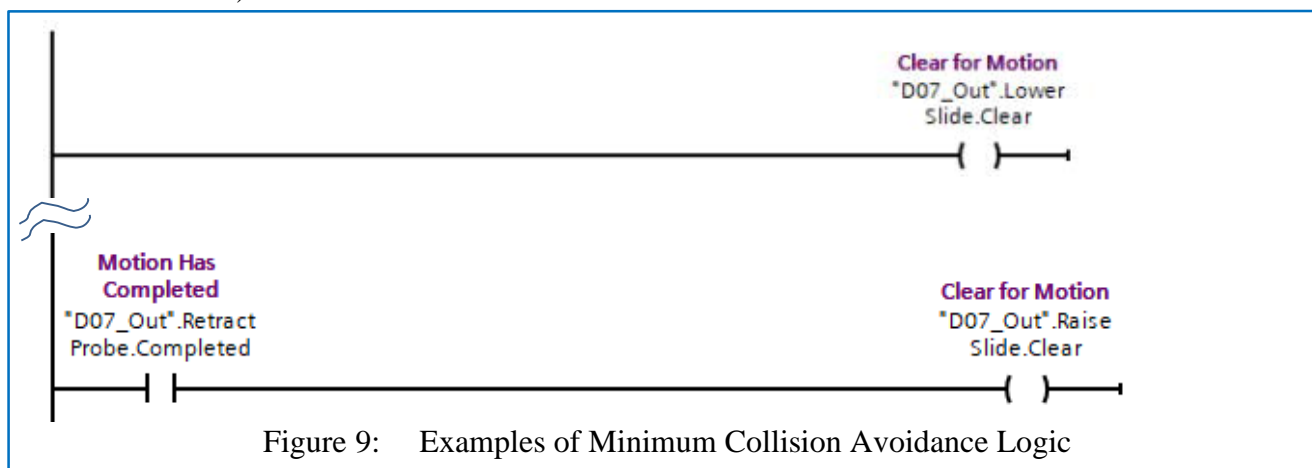
Requirements:

- 2.7.1 The OutputMotions routine shall include the logic that controls all machine solenoid-controlled motion.
1. Logic that controls machine solenoid motion includes collision avoidance, auto allow, enable motion, dwell timers, fault motion timers, and the motion not clear HMI display output-energize instructions as detailed within this solenoid control section.
 2. The OutputMotions routine shall **not** include logic that coordinates the sequential control of two or more related outputs. Sequence logic shall be in the Sequence routine.
- 2.7.2 Motion shall be prevented when selecting a mode, and motion shall be prevented when switching between modes. A separate action by the operator (clearly identified as a motion initiating action) is required in order for any motion to occur.
- 2.7.3 Logic for each motion shall include collision avoidance output-energize instruction(s).

Note: Collision avoidance may be one output-energize instruction per actuator, or two output-energize instructions (one for each direction of an actuator).

Note: Other terms used for “collision avoidance” include “clear to move”, “motion interlocks,” or “motion constraints.”

1. Motions shall have minimal collision avoidance logic. Collision avoidance shall only be used to prevent damage to the equipment or to prevent damage to the part (see Figure 9 below).



2. Collision avoidance logic shall be active in both manual and automatic modes.

Note: Collision avoidance includes a “Clear To” contact in all motion-initiation branches of motion-control logic (see Figure 10 next page).

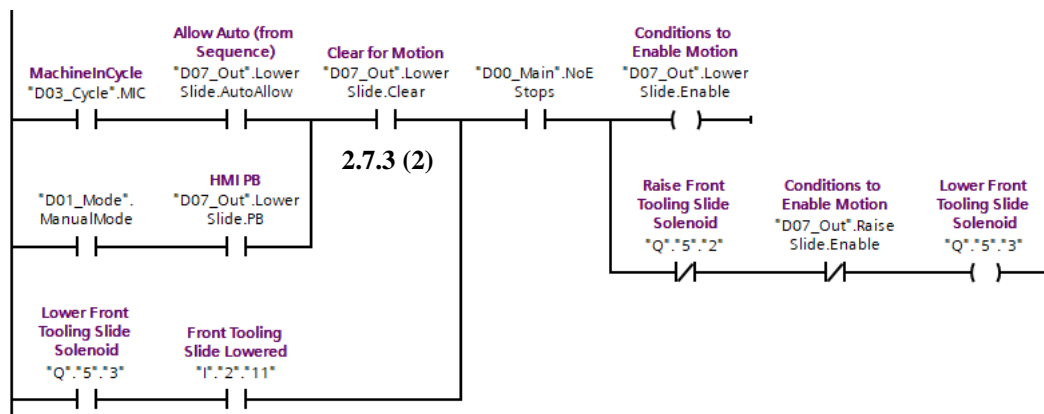


Figure 10: Example Use Of Collision Avoidance Logic

2.7.4 Logic for each direction of motion shall include an auto allow output-energize instruction. The auto allow logic should be kept simple per the following:

1. The auto allow for the motion towards the work position (moves away from the home position) is typically enabled by a normally-open (XIC) contact from the motion-initiating sequence step (a) and a normally-closed (XIO) contact from the sequence step that initiates the motion towards the home position (b) (see Figure 11 below).

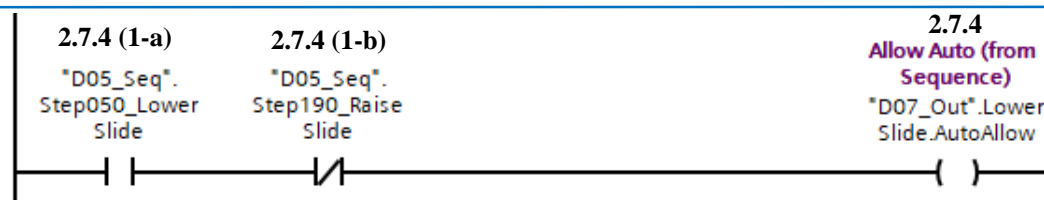


Figure 11: Example Auto Allow Logic (Motion Towards the Work Position)

2. The auto allow for the motion that returns to the home position is typically enabled by a normally-open (XIC) contact from the motion-initiating sequence step and a normally-closed (XIO) contact from Cycle Complete (see Figure 12 below).

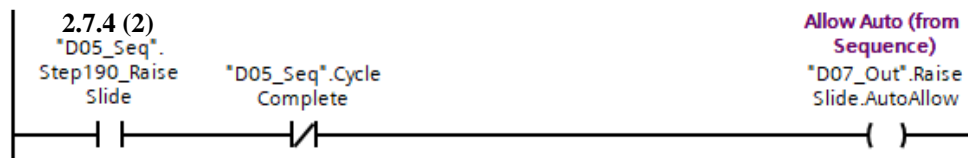
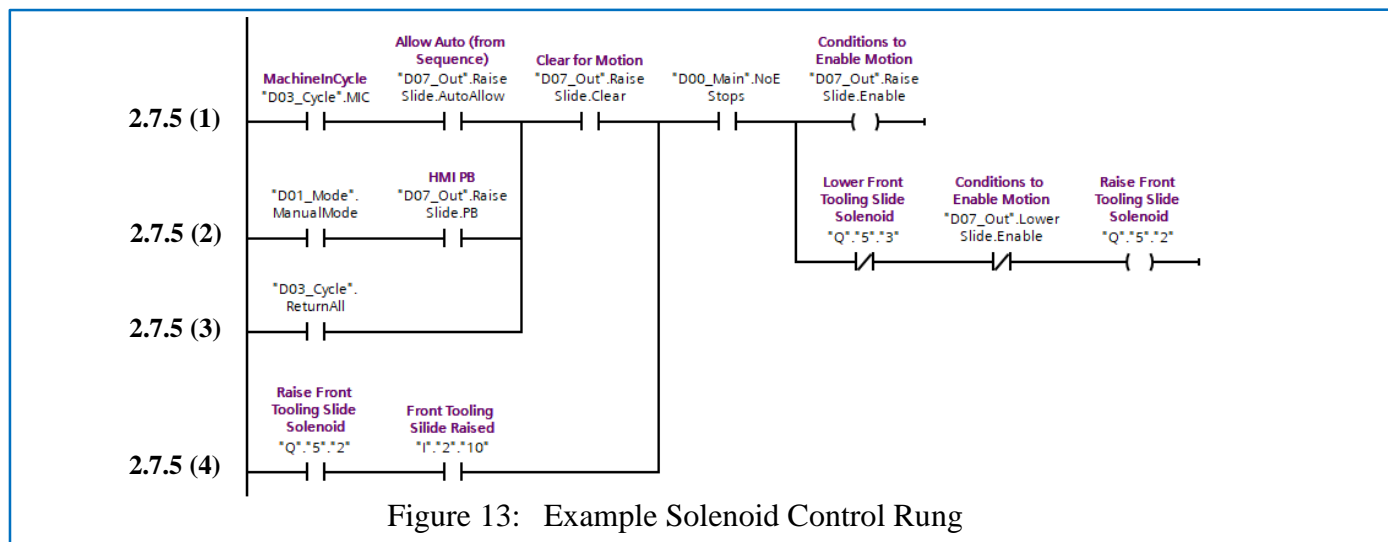


Figure 12: Example Auto Allow Logic (Motion Toward Home Position)

3. Machine sequences that require a motion to be enabled multiple times within the sequence, or requires enabling a motion at varying process steps dependent on such conditions as model selection or part-reprocess, require additional contacts in each motion's auto allow logic.

2.7.5 Each solenoid control rung shall be structured as follows (see Figure 13 below):



1. The top branch of the solenoid control rung shall include MIC and the auto allow to initiate the output.

Note: If the output is required for multiple sequence steps during the cycle, these multiple sequence conditions shall be programmed in the auto allow rung prior to the solenoid-control rung.

2. The second branch of the solenoid control rung shall include the manual initiation of the output.
3. The third branch of the solenoid control rung shall include the Return All initiation logic (for return-direction motions).
4. The bottom branch of the solenoid control rung shall include the solenoid seal-in logic, including a seal-in around the Clear To move contact.

2.7.6 Solenoids shall remain energized (seal-in) until the opposite motion is initiated.

1. Exception 1: A sequence step that needs to de-pressurize a cylinder by de-energizing an output without energizing the opposite motion direction.
2. Exception 2: Certain hydraulic motion solenoids may require the solenoid to **not** seal-in, since a continual energized solenoid can overheat the hydraulic fluid.

2.7.7 Solenoids shall remain energized (seal-in) based upon the actuation of the positional sensor indicating completion of the associated motion.

Note: Solenoid seal-in is required for all motions; the exceptions listed below are exceptions to the "positional sensor" portion of this specification item.

1. Exception 1: A normally-closed (XIO) contact from the opposing-direction positional sensor should be used, indicating a start of the motion, when the positional sensor indicating completion of the motion does not exist.

2. Exception 2: Positional sensor contacts shall not be used for solenoid seal-in when a double-solenoid, detented valve is being controlled. Since a detented valve mechanically seals-in when electrically energized, the logic should seal-in upon command such that the logical-state of the output, and the electrical-state of the output, are consistent with the actual mechanical condition of the valve.
- 2.7.8 Conditions that remove power to the output include logical E-Stop conditions that remove logical-power when hardwired power has been removed from the solenoid. The logical E-Stop contact also breaks the solenoid seal-in when the machine is powered down (see Figure 14 below).
- 2.7.9 All outputs that initiate opposing motions shall be logically linked such that both motions cannot be energized at the same time (see Figure 14 below).

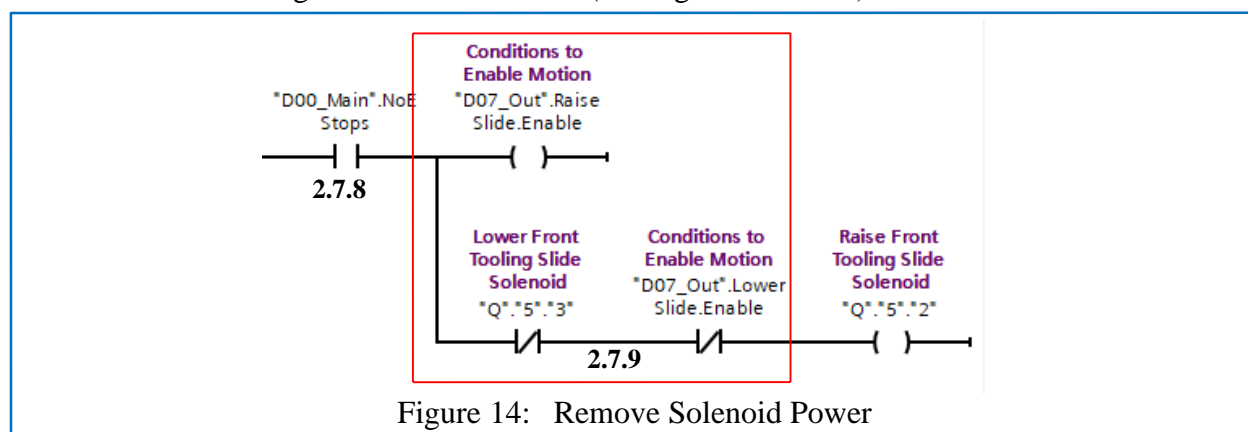


Figure 14: Remove Solenoid Power

- 2.7.10 A motion dwell timer for each position of motion shall be programmed in the rung immediately following the solenoid control rung. Timer presets may be adjusted for sensor debounce as needed.

Note: The Nexteer_Library dwell timers are initially preset to 0000 to indicate that timers can be set as low as practicable based on the application, thus minimizing cycle time.

2.7.11 Logic that controls single-solenoid motion valves shall include all rungs (collision avoidance, auto allow, enable motion, dwell timers, fault motion timers, and motion not clear HMI display output-energize instructions) for **each** motion direction, consistent with logic controlling double-solenoid valves (see Figure 15 next page).

1. The branch controlling the solenoid output-energize instruction does not include a normally-closed (XIO) contact driven by the opposite-direction output (since the opposite-direction output does not exist).
2. The rung enabling the non-solenoid direction of motion contains the Out_Enable output-energize instruction to logically link (disable) the solenoid.
3. The rung enabling the non-solenoid direction of motion does not contain seal-in logic.

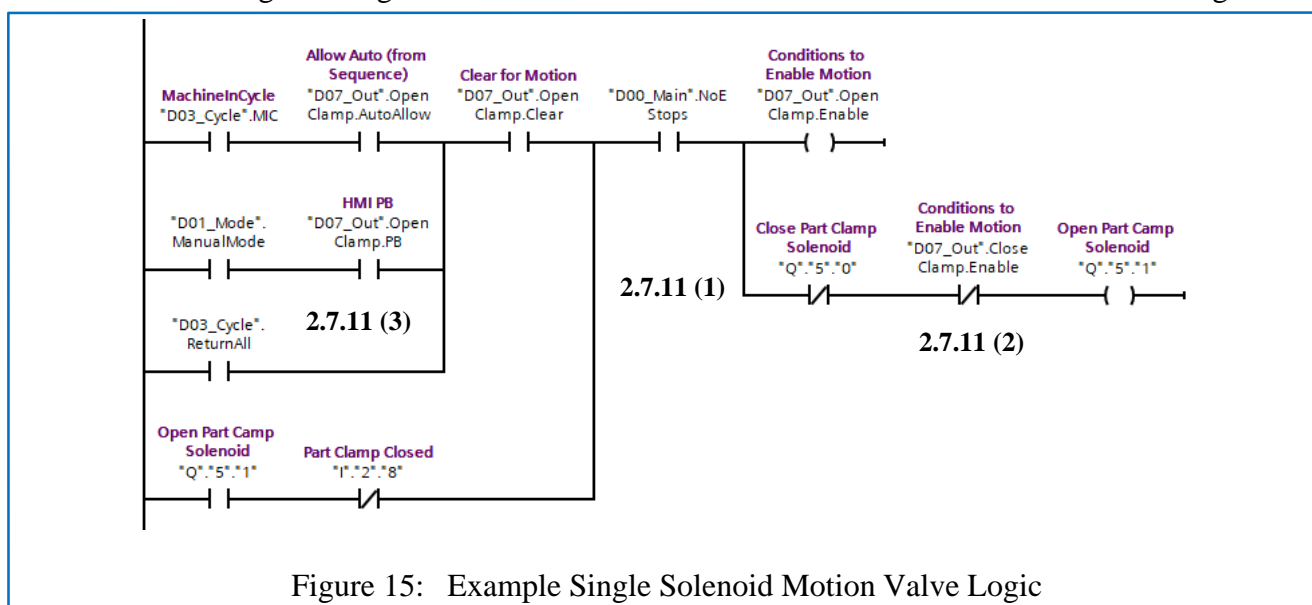


Figure 15: Example Single Solenoid Motion Valve Logic

Guidance:

- 2.7.12 In addition to the motion dwell timers (provided for sensor debounce as required above), process delay or process dwell timers may be programmed in the OutputMotion routine, or they may be programmed within the Sequence routine.
- 2.7.13 Diagnostic logic, including motion fault timers, as shown in the templates, may be programmed in rungs following the solenoid control rungs, or within the Immediate Stop fault routine.
- 2.7.14 The auto allow output-energize instructions may be eliminated for machines that require only minimal (or simple) auto allow logic meeting all of the following:
 - 1. The machine has minimal motions
 - 2. The auto allow logic for all motions includes only two sequence step contacts per motion (consistent with item 2.7.4 above), and
 - 3. All auto allow output-energize instructions are eliminated. The two sequence step contacts per motion (consistent with items 2.7.4 above) shall be programmed in each appropriate solenoid control rung.

2.7.15 The OutputMotions routine may include conventional motor-starter controlled motions, or the motor-starter logic may be in another output-related or device related routine.

2.7.16 Motor starter logic format for conventional motors (run independent of machine sequence) should be consistent with the format shown below (see Figure 16 below).

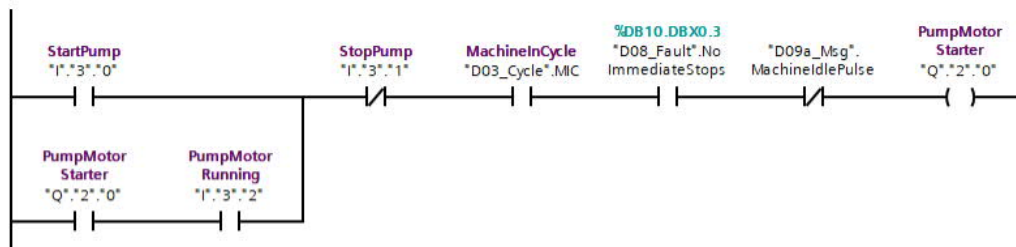


Figure 16: Example Motor Starter (Start/Stop) Logic

2.7.17 Motor starter logic format for sequence-controlled motors should be consistent with the solenoid control format detailed in this Solenoid Control section, including rungs such as the collision avoidance, auto allow, solenoid control, and motion timer rungs where applicable (see Figure 17 below).

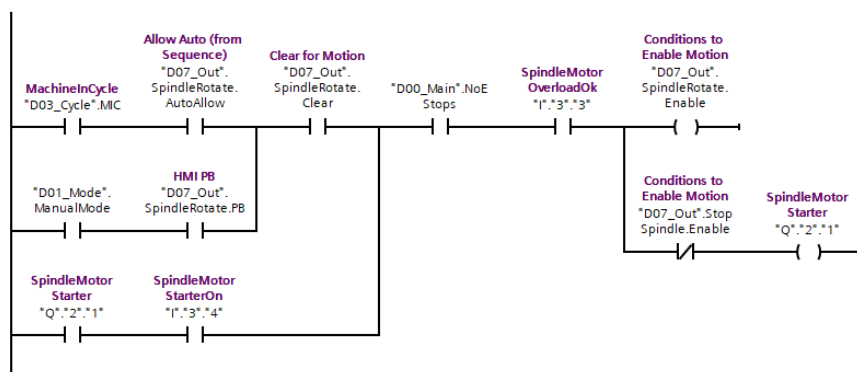


Figure 17: Example Sequence-Controlled Motor Starter Logic

2.7.18 Other general outputs may be programmed in the OutputMotions routine.

Note: Machine motions related to servos are typically programmed in a separate servo routine. Indicator lights are typically programmed in the routines that relates to the light's purpose.

2.8 Machine Diagnostics – Display Control (multiple routines)

Nexteer's machine diagnostics, display hierarchy, and diagnostic control philosophy are described in Annex A, entitled "Machine Diagnostic Scheme and Hierarchy." An understanding of the Annex A's terms will aid in an understanding the following machine diagnostics requirements. The use of Nexteer fault and message routines plus Nexteer HMI screens support compliance with the requirements of this clause.

Requirements:

- 2.8.1 The logic for fault display control shall be located in the Fault Control routine. The logic for message display control shall be located in the Message routine.
- 2.8.2 All faults shall seal-in until the fault is intentionally reset by actuating the "Reset Fault" button/switch. The "Reset Fault" button/switch shall reset all fault conditions that no longer exist. The "Reset Fault" button/switch shall only reset fault conditions when the machine is not in cycle.

Reference: To view all faults when multiple faults exist, the HMI fault history screen must be selected. Selection of the fault history screen is not controlled by the logic. Refer to SD-2020.

- 2.8.3 When multiple machine messages exist, the logic shall automatically scroll through the messages, displaying each for 3 seconds. After the last message has been displayed the scrolling shall start again at the first message.

Guidance:

- 2.8.4 Fault display text requirements, naming and numbering, is described in SD-2020, Nexteer Automotive Human Machine Interface Application Specification.
- 2.8.5 For equipment with a PLC that supports the character set in the language of the country of destination, the fault and message HMI display text is allowed to be a text string stored within the PLC. If these PLC text strings are used for display, an STOD instruction shall be added to each fault and message rung.

Note: The dual language requirements from SD-2020 apply to use of these text string displays.

2.9 Machine Diagnostics – Conditions and Detection logic

Requirements:

- 2.9.1 Any condition that stops a cycle shall create and display a machine fault.
- 2.9.2 Any condition that prevents the cycle from starting shall be displayed as a fault, message, or status.
- 2.9.3 All faults shall seal-in until the fault is intentionally reset by actuating the “Reset Fault” button/switch.
- 2.9.4 All faults shall be classified as either “Immediate Stop” faults or “Cycle Stop” faults. The logic for Immediate Stop fault conditions and detection shall be located in the Fault_ImmedStop routine. The logic for Cycle Stop fault conditions and detection shall be located in the Fault_CycleStop routine.
- 2.9.5 Immediate Stop faults shall immediately remove power to MIC, stop commanding all motions, and stop processing of the part.
- 2.9.6 Cycle Stop faults shall allow the machine to finish the current cycle, returning the machine to its normal start position, and then drop MIC. Cycle Stop faults that are detected prior to the start of a cycle shall prohibit the start of cycle.
- 2.9.7 Emergency Stop inputs, safety gate inputs, and indication of situations that could cause harm to an operator or cause harm to the machine (such as a fault from a servo system) shall be classified as Immediate Stop faults.
- 2.9.8 All machines, and every station on a multiple station system, shall include an immediate stop cycle overtime fault.
 - 1. The cycle overtime timer shall be active whenever the machine is in cycle.
 - 2. On single-cycle machines the timer for the cycle overtime fault is driven by MIC (only).
 - 3. For continuous cycle machines, MIC also drives the fault timer, but the fault is inhibited between cycles. The cycle overtime fault shall not be inhibited by a single condition that can fail, such as a single limit switch input contact, because such a failure can lead to inhibiting this safety-related cycle overtime fault.

Note: Resetting the cycle overtime timer with a pulse from a limit switch input is an allowed method to implement a single condition into this cycle overtime fault.
 - 4. Machines which have multiple cycles (such as MIC and Return All Cycle) shall include a cycle overtime fault for each cycle.

Note: All cycles are allowed to enable one cycle overtime timer and fault.

2.9.9 All motion sensors shall be back checked and generate an Immediate Stop fault upon failure detection.

1. For all motions that have more than one positional sensor, a Motion Sensor Error fault shall be included. The fault shall be detected if the motion is both “advanced” and “retracted”, or if the sensors indicate the motion to be at more than one position (such as mid-position and advanced).

Note: Motion Sensor Error faults shall be provided for operator hand-powered motions such as a hand-clamp.

2. For all motions that have only one positional sensor, back checking shall be accomplished by two opposite-state uses of the sensor. A sensor contact shall be used in the machine’s all returned logic, and an opposite-state contact shall be used in the motion dwell timer. This motion dwell timer shall be included in the sequence logic to indicate the completion of a sequence step. Thus, upon a sensor failure, two opposite-state uses of the sensor will either generate a motion timer fault or a Not Returned indication.

2.9.10 An Immediate Stop motion overtime fault shall be included for each logic-controlled actuator. Exception: A motion overtime fault is not required for actuators that include following-error faults, such as servo controlled actuators.

Note: The motion overtime timers may enable a single fault per actuator or two faults, one for each direction of motion.

2.9.11 All non-motion sensors (such as part quality sensors, part present, and pick-bin sensors) shall be back checked. Failures that are detected during a machine cycle shall cause an Immediate Stop fault upon failure detection. Failures that are detected between machine cycles shall cause either a Cycle Stop fault (preferred) or an Immediate Stop fault.

2.9.12 The logic for message conditions and detection shall be located in the Message routine.

2.9.13 The logic for machine status control shall be located in the MachStatus routine. The logic for part status control shall be located in the PartStatus routine. The logic for operator prompt control shall be located in the OperPrompt routine.

Guidance:

2.9.14 PLC Battery Low should be indicated as a machine message that does not prohibit machine cycle. Also, when the PLC Battery Low condition is present, a resettable Cycle Stop fault shall be triggered every 60 minutes until the battery is replaced.

2.9.15 The Nexteer_Library’s Fault_ImmedStop and Fault_CycleStop arrays (sized for 128 immediate stop faults and 64 cycle stop faults) may be increased in sized for applications with numerous faults.

2.10 Routines Required On All Machines

Requirements:

- 2.10.1 The routines provided in the Main Program of the Nexteer_Library file shall be used for all applications (see Figure 18).
- 2.10.2 These required routines are to be programmed on all equipment, even when the equipment does not include any logic within the routine.
- 2.10.3 For single station equipment, the routines provided in the Main Program of the Nexteer_Library file shall be programmed in the MainProgram.
- 2.10.4 For multiple station equipment with multiple programs, the routines provided in the Main Program of the Nexteer_Library file shall all be programmed, but are allowed to be distributed between the MainProgram (sometimes called the ConveyanceProgram) and the Station programs.
 - Refer to Figure 19 on the following page for an example of routine distribution within a multi-station controller organizer.
 - The routines provided in the Main Program of the Nexteer_Library file are typically duplicated within each station's program.



Figure 18: Nexteer_Library
Controller Organizer

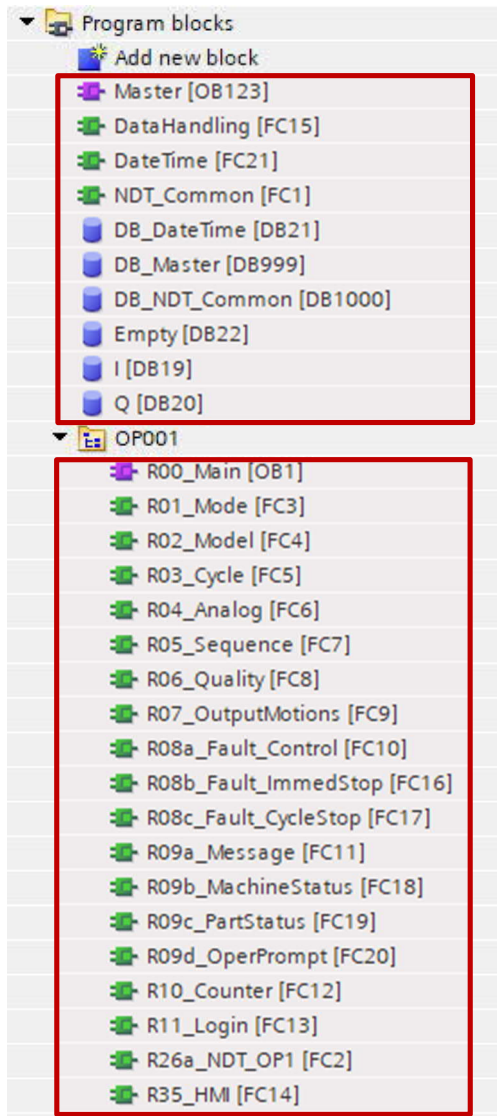


Figure 19: Multiple Station Controller Organizer Views

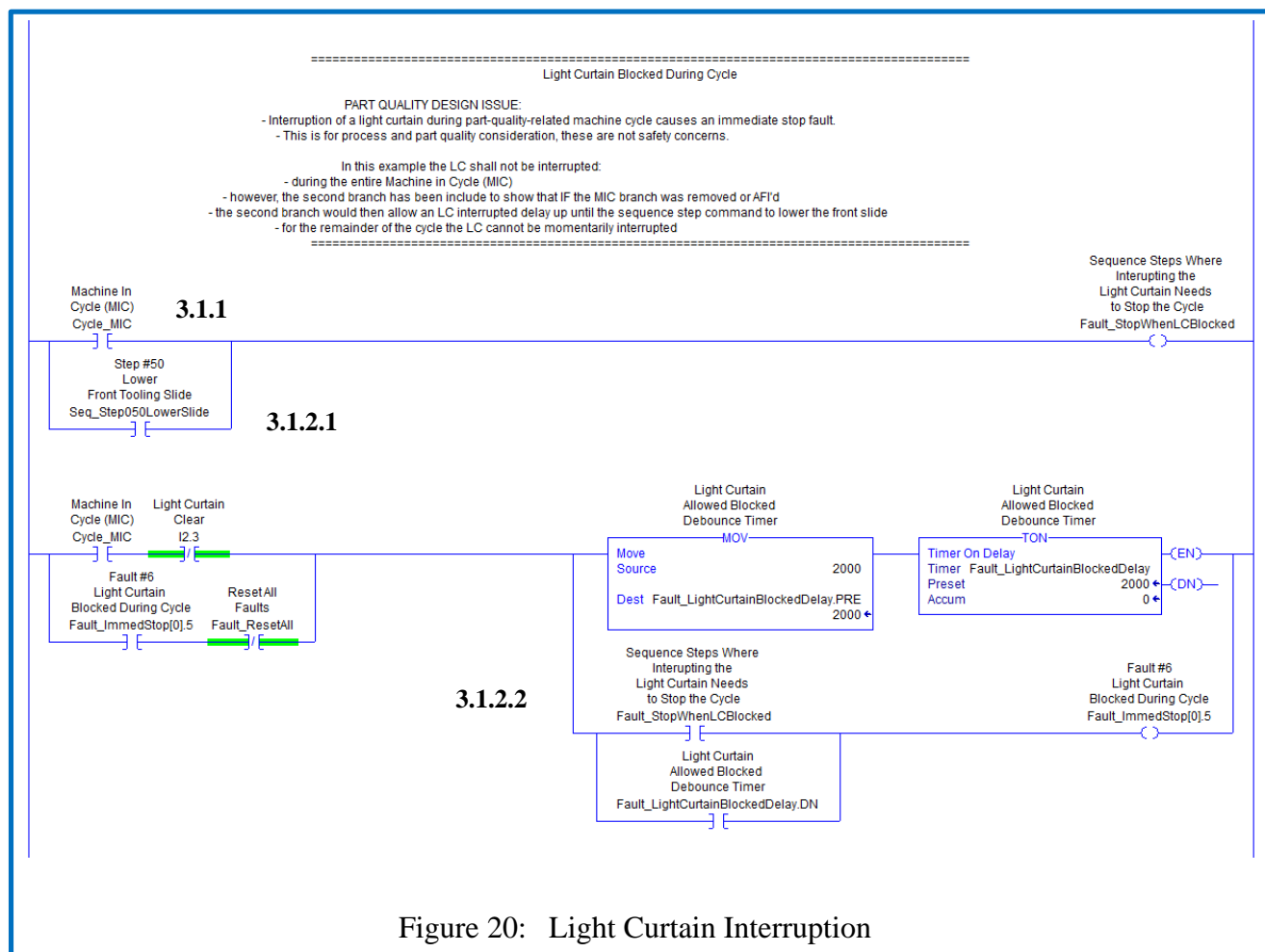
3. Required Logic Design - Application Specific

Nexteer logic requirements that are associated with a routine are detailed in Section 2 above. The following topics are additional application-specific requirements.

3.1 Light Curtain Interruption

Operator and machine safety is ensured by appropriately applied safety devices and hardwired safety circuits. Therefore, the logic based on an interruption of the light curtain does not relate to safety, but instead relates to part quality and properly processing the part since the logic needs to consider that physical power has been removed from hazardous actuators and devices.

- 3.1.1 Interruption of a light curtain during the machine cycle shall cause an Immediate Stop fault. Typically the Light Curtain Blocked During Cycle fault is energized upon the first scan that the light curtain is not clear.



3.1.2 A deviation that delays the fault, and allows the light curtain to be momentarily interrupted during a cycle, shall meet all of the following requirements:

1. The logic design shall ensure part quality and proper machine sequence. Design consideration needs to be given to ensure that the light curtain may be momentarily interrupted **only** during those process steps where the part will still be properly processed with hardware power momentarily removed.

To clarify: Most process steps that effect or work directly on the part will make a reject part if hardware power is momentarily removed, and therefore light curtain interruption during these steps shall be detected immediately and cause an immediate stop fault.

To clarify: To avoid nuisance stops and loss of production, momentary interruption of the light curtain during initial part-positioning machine motions (such as part clamps and part shuttles) can be considered for inclusion in any logic that delays the Light Curtain Blocked immediate stop fault.

2. The MIC contact and branch enabling the Fault_StopWhenLCBlocked output-energize instruction may be removed or ignored with an AFI.
3. The Fault_LightCurtainBlockedDelay timer preset shall be set to 2000 or less (a maximum of 2 seconds).

3.2 Motor Starter Control

3.2.1 Motor start control logic shall be designed such that resetting of an overload device does not restart the motor (see Figure 21 below).

1. For applications where a normally open contact from the motor starter is wired to an input, a normally-open (XIC) contact from this input shall be used in the seal-in branch of the motor starter's output-energize instruction.
2. For applications where the motor starter overload is wired to an input, a normally-open (XIC) contact from this input shall be used as a condition to prohibit energizing the motor starter's output-energize instruction.

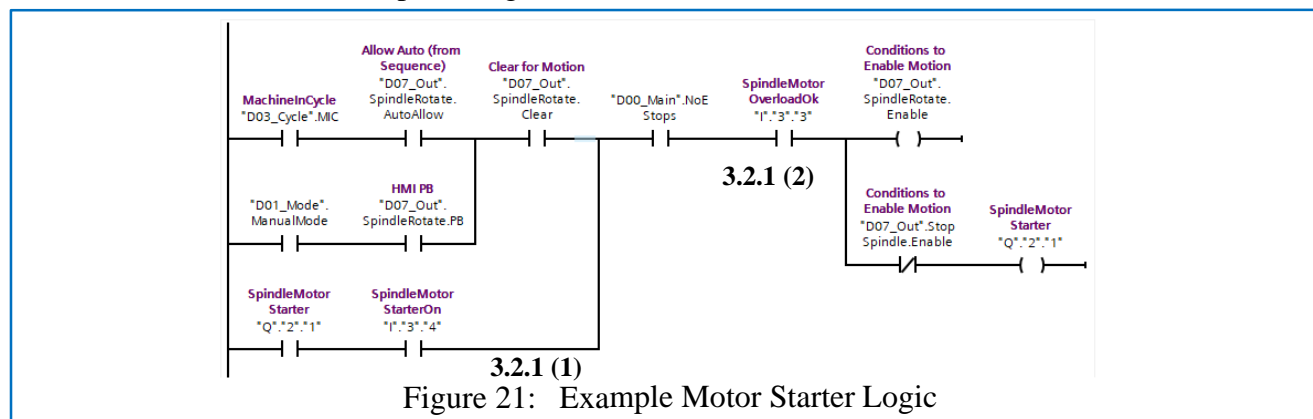


Figure 21: Example Motor Starter Logic

3.3 Shift Register / Indexing Logic

- 3.3.1 Data transfer in a shift register, whether stored in an array or a bit-register such as a DINT, shall be initiated by a shift pulse conforming to the following requirements.
1. The shift pulse shall occur once per index cycle. Multiple shift pulses shall not occur because of sensor contact bounce, programmable device power up/down, or any other unintended cause. Refer to the IndexData routine within the Library_Routine program of the Nexteer_Library file.
 2. When index mechanisms are used, the shift pulse should occur when the indexing mechanism begins to transfer parts from one station to another.
- 3.3.2 Logic shall be included to ensure the correctness of shift register part quality data; the shift register logic shall prevent qualifying a Reject Part as a Good Part under the following conditions.
1. Logic shall prevent against accepting Good Part status based on the reloading of PLC memory (reloading of old or stale data).
 2. Logic shall also prevent against accepting Good Part status based on memory which can be invalid due to an unknown index (such as occurs when a dial table is indexed while power is off).

Four approved methods of ensuring the correctness of shift register part quality data include:

- resetting the shift register data (classifying all parts as rejects) on PLC power-up (first scan logic), or

- use of a 10-turn encoder connected to a dial table indexer (to detect an index without power), classifying all parts as rejects when powered-up out of the last known position, or

- dial table fixture identification (such as RFID or barcode), read at a minimum of one location, such that the shift register (or array pointer) can be reliably established even under such conditions as clutch overload or manual-index with power removed, or

- part identification (such as on-the-part RFID or barcode) read at load, tracked with all other shift register part data, and read again at unload prior to unload. The part shall be classified as a Reject Part if the part at unload is not the part that had been loaded to that pallet or fixture.

3.4 Pallet Release / Pallet Memory

- 3.4.1 Logic for part-quality data storage, on asynchronous assembly lines with data tables (or arrays) that use the pallet number as the table-pointer, shall prevent part-quality data from being written to inaccurate locations in the data table.
1. The logic shall prevent storing (or memory of) a pallet number(s) which can become inaccurate after the non-controlled transfer of pallets (such as the pushing the solenoid override of a valve).
 2. The logic shall reset the station's part-quality memories (both within the station logic and within conveyance pallet control logic) consistent with the Reset All Memories requirements within the Cycle routine section of Section 2 above.

3.5 Indicator Lights

- 3.5.1 Logic controlling indicator lights is typically programmed in routines associated with the purpose of the light.
- 3.5.2 Test Lights logic shall be provided for all operator indicator lights. *Note: Operator indicator lights include lights on the operator control station and multi-colored pilot lights provided at manual load / unload locations.*
- 3.5.3 Logic for the manual load /unload station multi-colored LED pilot light (refer to SD-004) shall be designed based on the following criteria at a minimum. *Note: GREEN and YELLOW may be used to indicate additional conditions; however RED shall only indicate those conditions stated below.*
1. "GREEN": Solid Green shall indicate a Good Part. The light shall energize when the machine cycle has completed, and stay energized until any of the following conditions occur: either the part is unloaded, or the machine is put into Manual Mode, or the machine is powered-down.
 2. "RED": Solid Red shall indicate a Reject Part; Flashing Red shall indicate the machine has stopped because of a fault. When indicating a Reject Part the light shall remain energized until the reject part has been handled appropriately. *Note: Flashing Red typically indicates an Immediate Stop fault. However, on continuous cycle machines Flashing Red may indicate a Cycle Stop fault.*
 3. "YELLOW": Solid Yellow shall indicate Machine-In-Cycle (refer to the Machine-In-Cycle section of this specification).

3.6 HMI Requirements for Synchronous Transfer Systems (Multiple HMIs)

This Multiple HMI section details the motion-control logic requirements for multi-station synchronous-transfer systems with multiple HMIs, where machine motion can be initiated from more than one HMI.

- 3.6.1 Manual/Off/Auto selection is required on each HMI station that can initiate machine motion.
- 3.6.2 All station HMIs must have Auto selected to allow initiation of any automatic or manual motion on the machine from the main control HMI. Manual and Off selections from station HMIs shall disable automatic and manual mode selection at the main control HMI.
- 3.6.3 When Off is selected at a station, no motion for that station shall be permitted (whether main or local initiated), and machine index or transfer shall not be permitted (whether main or local initiated).

Note: This Multiple HMI section does not apply to asynchronous transfer systems such as pallet-and-free conveyor lines. This section also does not apply to HMI stations that have been included solely for remote display purposes.

This page intentionally blank.

A. Annex A - Machine Diagnostics Scheme and Hierarchy

A.1 General

Nexteer's Production and Operations require machine diagnostics and display consistent between all machines. Any condition that stops a cycle, or prohibits a cycle from starting, needs to be detected and displayed for operations on the operator interface. Conditions which stop or prohibit a cycle can be relative to the part, the machine, auxiliary devices, and the operator. Design consideration needs be given to failures related to the part, failures related to the machine, failures related to auxiliary devices, and incorrect operator action.

All machine diagnostics, from simple to complex, inherently use logical hierarchy for collecting and displaying information. Nexteer specifications use the following terminology and hierarchy within its specifications, guidelines and templates.

A.2 Terminology & Hierarchy:

Faults:

Faults are machine and device conditions. Faults require operator intervention; faults require operator reset (via a Fault Reset button) – so therefore logic for faults seals-in and captures the fault condition for machine diagnostics and potentially for part diagnostics. The fault display object is programmed in a location on the global common screen, and placed in the same location on every HMI screen. The display text is stored in the HMI, through an alarm list object, plus use of an alarm history screen (refer to SD-2020). Nexteer's specification require faults to be grouped into either an Immediate Stop or a Cycle Stop

Immediate Stop Faults:

Immediate Stop Faults are those machine conditions that require the machine logic to instantly stop part processing, immediately stop the machine cycle, and/or immediately stop all machine motion.

Immediate stop faults can include part conditions that prohibit the part from being processed further. If the part condition indicates that there is no value in further processing the part, then the part condition can be an immediate stop fault. A reject part can be either an immediate stop fault or, more-typically, a cycle stop fault depending on the machine's reject handling.

Example Immediate Stop Faults: Light Curtain Blocked During the Cycle, Machine Cycle Overtime, sensor error faults, motion overtime faults, and certain part-quality faults.

Cycle Stop Faults:

Cycle Stop Faults are those machine conditions that do not require the machine to instantly stop, therefore, the logic allows the machine to finish processing the part, or finish the current cycle, and/or return the machine to its normal start position.

Cycle stop faults can include part conditions that allow a part to be completely processed and return the machine to its home position. Cycle stop faults can also include part quality faults that allow the machine to return to the home position without further processing of the part. A reject part is typically a cycle stop fault, although it can be an immediate stop fault, depending on the machine's reject handling.

Example Cycle Stop Faults: Between-cycle back check faults, feed track low level, traceability PC Heartbeat Timeout, and certain part-quality faults.

Machine Messages:

Machine messages (typically referred to just as “messages”) are machine and device conditions. Messages are general machine operating conditions with less significant impact concerning the machine or the part and are not considered faults. A message indicates a condition that should be corrected by operator intervention such that subsequent nuisance machine cycle stop faults can be avoided. Messages are allowed to cycle stop the machine, but they typically do not. The message display object is programmed in a location on the global common screen and placed in every screen, typically separate from fault display such that messages are displayed on every HMI screen. The display text is stored in the HMI through a multi-state indicator (refer to SD-2020).

Messages should be displayed as long as the associated machine condition exists. The fault reset button or switch is not required to clear messages. Messages often require operator intervention with the equipment such that the message is cleared after operator intervention. Without operator intervention, the message condition may lead to an additional condition which can cause a cycle stop fault.

Examples: Bowl Feeder Low Level - the message is associated with a need for the operator to add parts to the bowl feeder. The Bowl Feeder Low Level message (and the logic displaying the message) will no longer be displayed on the operator interface when the operator re-fills the bowl. However, ignoring the message may lead to a Cycle Stop fault for No Parts in the feeder track, which would require the operator to reset the fault display. Messages such as “PLC Battery Low”, “Coolant Level Low”, “Barrel Heat Zone Not at Temperature”, and “Bowl Feeder Low” are additional examples.

Machine Status and Part Status:

Machine Status and Part Status conditions have a lower hierarchy for display. There are several fixed, standard, basic, and status conditions displayed in two dedicated multi-state indicators on Nexteer’s Automatic HMI screen. The logic enabling these status displays is already programmed in the logic and HMI templates consistent with the expectations for nearly all machines. The machine status and part status logic typically do not need to be modified by the OEM.

Operator Prompts:

Operator Prompts (typically referred to as just “prompts”) are part and operator-related conditions with less significant impact than faults or messages. Prompts are based on operator interaction with the part being processed. Prompts indicate a condition for operator intervention such that subsequent nuisance machine faults can be avoided. Prompt conditions are allowed to cycle stop the machine, but typically do not, although the condition typically does prohibit a cycle from initiating. Prompts are displayed in a dedicated multistate indicator on the Automatic screen(s). Prompts should be displayed as long as the condition exists. A reset is not required to clear prompts; they are cleared after operator intervention. Ignoring the prompt may lead to an additional condition which can cause a cycle stop fault.

Prompts are application specific. Fully-automatic machines may require no prompts, while operator-based hand-build assembly stations may require many prompts.

Example Prompt: Use Hand-Tool to Position Snap-Ring. The operator must use the hand-tool to properly pre-position the snap-ring on the shaft prior to cycle. If the operator attempts to cycle start the machine without correctly pre-assembling the snap-ring, a machine fault will occur.

B. Annex B - Controller: Organizer, Structure, Names, and Instructions

GENERAL

- B.1 The TIA Portal PLC/HMI file name shall include the asset tag number (SD number) of the machine(s).
- B.2 The logic design shall be created using the FC with Ladder Diagram language type.
- STL(Statement List) and Function Block Diagram(FBD) shall not be used.
 - The use of Structured Control Language(SCL) requires Nexteer Controls deviation approval prior to the start of logic design. SCL shall not be used for basic machine control logic that is used for machine support, such as cycle, sequence, motion and outputs, part quality, and machine diagnostics. Note: A deviation can and may be granted for SCL that Nexteer Controls has determined will not need to be altered by plant support personnel, nor used for machine support. Two examples where Structured Text can be the appropriate language type include (1) logic to accomplish sophisticated calculations, or (2) logic provided by the device manufacturer that is not modified for the application).
- B.3 The Master Control Reset (MCR) instruction shall not be used.
- B.4 Output Latch instructions shall not be used in motion outputs.
- B.5 FBs may be used when provided by the device manufacturer. FBs should be programmed in ladder logic format.
- B.6 Network comments shall be included to clarify the purpose or design intent of complex logic that is not easily understood. Examples of complex logic can include math operations, data manipulation, analog signal conversion, and communication to auxiliary devices.
- B.7 The controller shall have at least 25% spare (unused) memory.
- B.8 Forces or temporary logic used for bypassing logic shall be removed prior to MQ1 runoff of the equipment. Proper logic operation shall be verified at MQ1.
- B.9 Un-used logic, tag names, and descriptions shall be deleted prior to shipment of the machine.
Exception: Descriptions for un-used fault and message array bits should not be deleted.

Clarification: Un-used FBs and UDTs (PLC Data Types defined by OEM) are allowed to be deleted from the delivered machine logic.

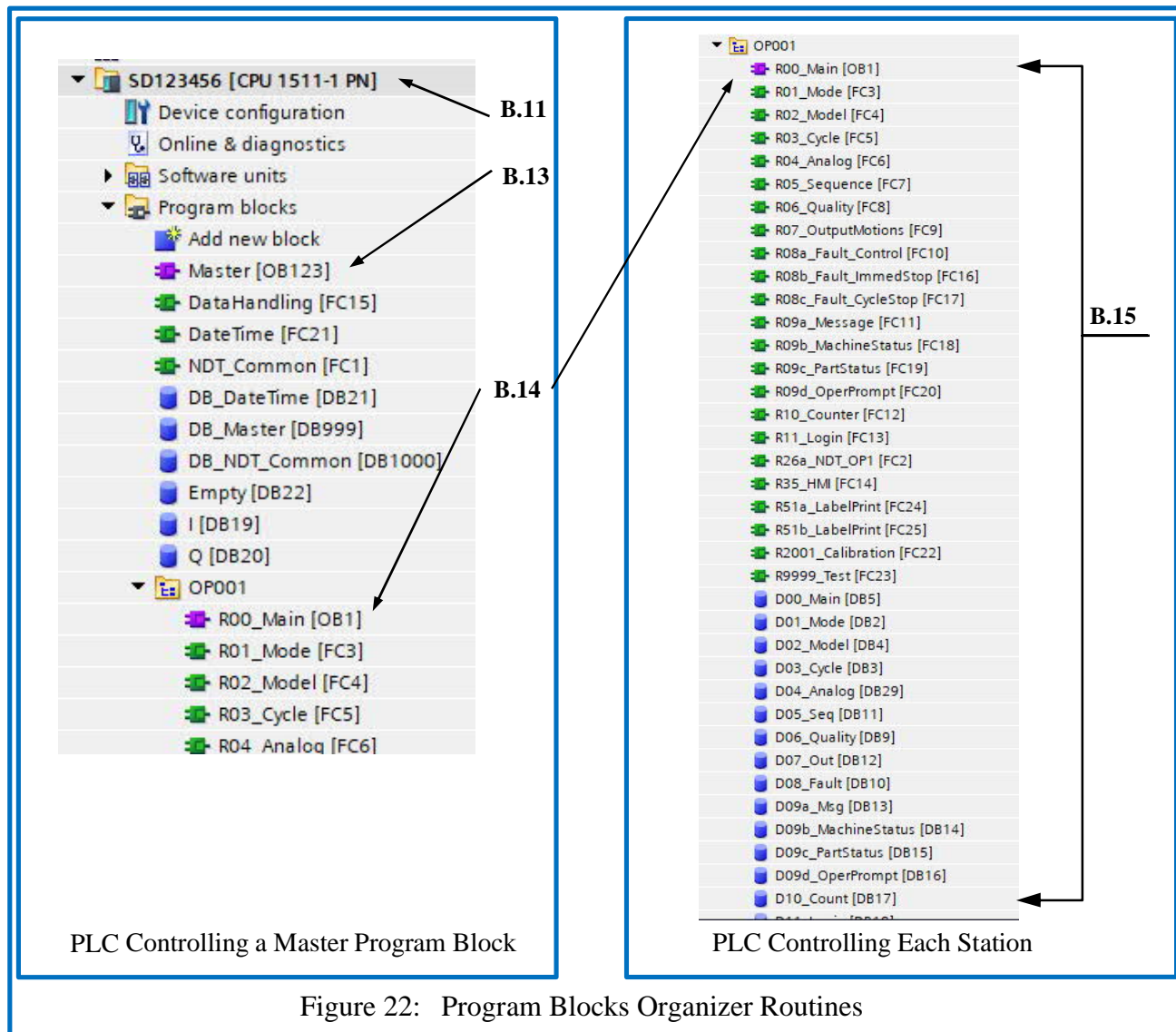
*Clarification: The OEM is expected to remove unused Nexteer Library routines. However, when a library routine (such as an RFID routine, or traceability routine) has been used within the main program, the OEM is **not** required to remove un-used portions of **logic** from that routine.*

CONTROLLER ORGANIZER: STRUCTURE AND NAMES

- B.10 Nexteer logic files, showing the controller organizer structure and naming conventions as described within the following sections, are located on www.nexteerdatabase.com in the Toolkits, Templates and Forms selection under Vendor Documents.
- B.11 The PLC name (the Name field under Controller Properties) shall include the asset tag number (SD number) of the machine(s) (see Figure 22 below).
- B.12 Names shall be consistent between the routine names, I/O configuration devices names, and tag names, as described within the following sections and as shown in the tables at the end of this annex.

TASKS / PROGRAMS / ROUTINES

- B.13 All programs and routines shall be organized under a program block Master. (*Refer also to sub routines deviations listed below*).
- B.14 For a PLC controlling all routines shall be organized under a program block named “Master[OBx]”. And each station shall be organized under a program R00_Main[OBx] (see Figure 22 below).
- B.15 For a PLC controlling multiple stations such as an assembly line, the program structure should include a general-purpose Master Program and separate programs for each station under the Main Program. The program for each station should be named as the station number. Each station program should include routines consistent with the routines required from the Nexteer_Library MainProgram (as noted elsewhere within this specification). And each program shall have its own DB. The conveyance control logic should be included either as routines under the Main program (see Figure 22 below), or as routines under a program named Conveyor.



- B.16 A deviation can and may be granted allowing program blocks per the following:
- The Master OB shall be configured as a priority number 1.
 - The Main OB shall be configured as a priority number 2.
 - Enable minimum cycle time for the cyclic OBs.
- B.17 For all projects, the name for each routine shall represent the functions or machine tasks that are accomplished by the logic within the routine.
- B.18 The Program blocks tree view shall represent the PLC logic solve-order. Therefore, the routine names shall include the prefix Rxx. Note: xx represents the logic solving order. Duplicate numbers are allowed such that related routines are grouped; however duplicate numbers should include an alpha character suffix such that the controller organizer view represents the PLC logic solve-order. The routines shall be called by logic within the routine Main in this numerical order.
- B.19 Additional routines are available for use as determined by the machine application, such as routines provided in the Library_Routines program of the Nexteer_Library. Logic from the library, when used, shall be moved, copied, or imported into the MainProgram or station programs. Routines shall be unconditionally called from the MainProgram or station programs. When the routine is copied, it shall be renamed. Each program block shall have correspondent Data Block. The data block shall include the prefix Dxx.
- Note: Nexteer Library routines with a routine name appended with the text “_TOOLS” contain specific rungs of logic that are intended to be copied or rung-imported into existing MainProgram or station program routines of the same name.*
- B.20 Additional routines are allowed for applications not illustrated in the logic templates. Additional routines should follow the requirements and naming conventions of this specification.

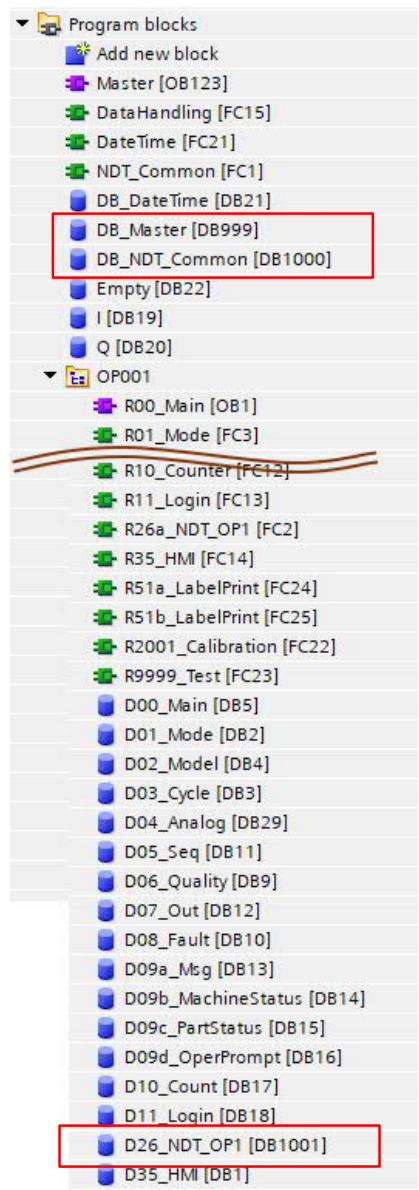
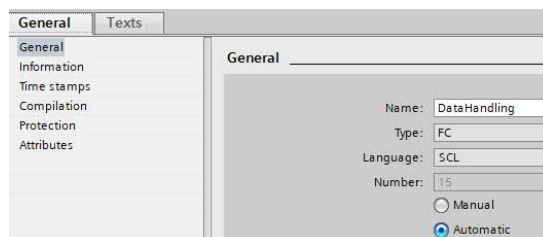


Figure 23: Program block tree view

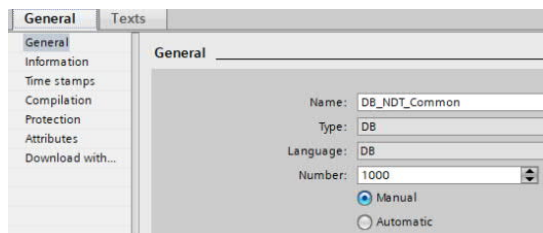
BLOCK NUMBER AND OPTIMIZED BLOCK ACCESS

B.21 Default program block number is set as “automatic”.



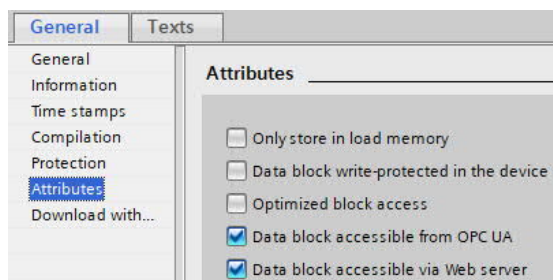
B.22 DB_Master, DB_NDT_Common and D26_NDT_OPx shall be set as “manual”. And the number shall be 999, 1000 and 1001. If the project has multiple stations, D26_NDT_OPx shall be incremental from 1001.

Note: The number of DB_NDT_Common shall be matched with the setting NDT app. Default number is 1000.



B.23 Optimized block access shall be unchecked for DB_Master, DB_NDT_Common and D26_NDT_OPx.

Note: All other program blocks shall be checked.



DATA BLOCK TAGS

TAG NAMING

B.24 This section establishes a hierarchy for **tag naming**; **the tables at the end of this annex show examples** with clarification notes.

B.25 I/O tag names shall start with the character I or Q for input or output respectively, then be built as follows:

- a. for distributed I/O only (such as remote I/O and distributed on-machine blocks) followed by additional text.
 - i. **D** and a unique device **number**, or **ST** and a station **number** for the associated station, or **B** and a unique block **number**, or **BNI** and a unique Balluff BNI module **number**.
 - ii. followed by an **underscore** character.
- b. followed by the slot **number** (does not apply to distributed on-machine blocks).
- c. for analog I/O only: followed by the text **CH** and the signal channel **number**.

Note: The additional text required on distributed I/O may be a combination of one or more device, block, and station identification alpha-numeric characters. For the additional text required on distributed I/O, the number may be omitted on small systems which contain only one distributed I/O device, block, or module.

Note: Per SD-004, I/O conductors shall have the same identification as the I/O, including cables for analog signals.

B.26 Non I/O tags with contents modified (set or enabled) within a routine shall be named based on the routine name, an underscore, and the tag function (purpose or use within the routine). *Note: The templates establish the required routine-name portion of the tag names that are associated with the required routines. An abbreviation of the routine name may be used for tags associated with routines and devices not specifically shown in the templates.*

B.27 Non I/O tags with contents modified by a device shall be named based on the device name, an underscore, and the tag function (purpose or use from the device).

B.28 Maximum overall tag name length should be 30 characters. It is recommended that upper case characters be used to start each word in the name. The use of abbreviations should be minimized.

USER-DEFINED DATA TYPES (UDT / PLC data types)

- B.29 This section describes both the application of Nexteer-provided UDTs, and establishes requirements for OEM-created UDTs.
- B.30 Nexteer-provided UDTs have a property name prefixed with either a “u_” or an “x_” (see Figure 25).
- UDTs that have a u_ prefix typically require modification to match the application.
 - UDTs that have an x_ prefix shall not be modified.
- B.31 Tag names shall follow a similar convention to the non-I/O tag-naming section covered previously in this specification. *Note: Tags created by UDT usage will have a format: Tag_Name.Device_Member.*
- The Tag_Name shall be a unique name for each use of the UDT, using the following naming hierarchy:
 - Tags with contents modified (set or enabled) within a routine shall be named based on the routine name. Example UDT usage tag name: Out_CloseClamp, with a complete tag and member name Out_CloseClamp.Enable. Example UDT usage tag name: CodeReaderHsg, with a complete tag and member name CodeReaderHsg.CommActive.
 - Tags with contents modified by a non-PLC device shall be named based on the device name. Example UDT usage tag name: FANUC_DataIn, with a complete tag and member name FANUC_DataIn.TPENBL.

Note: Tag names are required to include a prefix based on the routine in which the tag's contents are modified. It therefore follows that, in order for an OEM-created UDT to be acceptable, each use of the UDT shall have all member-modifying instructions, such as OTEs and timers, set in one routine.

- B.32 The Device_Member names shall be based on the member's control function (purpose or use).
- B.33 All tags shall have detailed description consistent with the Tag Descriptions section of this specification.

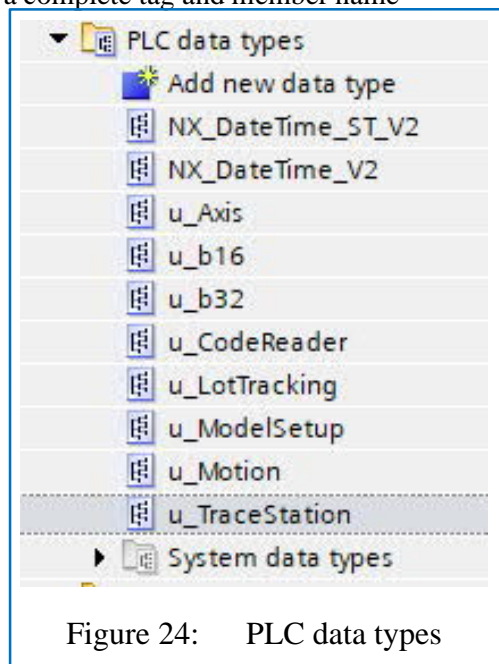


Figure 24: PLC data types

- B.34 The use of additional, OEM-created UDTs, requires Nexteer Controls deviation approval prior to the start of logic design.
- a. OEM-created UDTs shall not be created for logic routines used for sequence, part quality, and machine diagnostics (faults and messages).
 - b. OEM-created UDTs should not be created for logic routines used for mode and cycle.
 - c. OEM-created UDTs shall have a property name prefixed with the characters **oem_name**.
- B.35 OEM-created UDTs, and OEM modification to Nexteer UDTs should keep UDT members grouped by data type.
- B.36 OEM-created UDTs, and OEM modification to Nexteer UDTs shall include detailed descriptions for all UDT members consistent with the Tag Descriptions section of this specification.
- Note: Tag member descriptions will be appended to the base tag description using the pass-through display feature under the controller properties project tab. Refer to tag description examples shown in the Tag Descriptions section at the end of this annex.*
- B.37 UDTs are also allowed when device-created by third-party device applications. The device-created property name for these device-created UDTs should not be altered.

I/O CONFIGURATION, ROUTINE, AND TAG NAMING TABLES

Naming – Consistency Between I/O Configuration Device Names and Tags			
Name – I/O Configuration	Tag Name Examples	Device	Explanation and Clarification Notes
I1	I1	Local Discrete Input Card Slot 1	I/O configuration name is consistent with the I/O device tag(s).
I1	I1.2	Local Discrete Input Card Slot 1, Bit 2	
Q4		Local Discrete Output Card Slot 4	I/O configuration name is consistent with the I/O device tag(s).
Q4	Q4.7	Local Discrete Output Card Slot 4, Bit 7	Most outputs will be energized within the OutputMotions routine; however, discrete outputs are NOT named by their associated routine. Naming by I/O takes priority.
I3	I3Ch1	Local Analog Input Card Slot 3, Channel 1	Although an analog input will be associated with the Analog routine, naming by I/O takes priority. This signal's cable number should be I3CH1.
B2	IB2.8	Siemens Distributed Input Block Block 2, Bit 8	I/O configuration name is consistent with the I/O device tag(s).
ST70_B1	IST70_B1.7	Siemens Distributed Input Block Station 70, Block 1, Bit 7	I/O configuration name is consistent with the I/O device tag(s). This input device's wire number should be "ST70_IBK01.7".

Table 1: I/O Tag Consistency – Local and Distributed I/O

Naming – Consistency Between I/O Configuration Device Names and Tags			
Name – I/O Configuration	Tag Name Examples	Device	Explanation and Clarification Notes
D1		Distributed Point I/O AENT Module 1	Point I/O AENT Module number 1 either within a single station, or in the main enclosure of a multi-station machine.
ID1_4	ID1_4.1	Point I/O Input Card Module 1, Slot 4, Bit 1	I/O configuration name is consistent with the I/O device tag(s).
ST20		Distributed Point I/O AENT Module ST20	Point I/O AENT Module associated with Station Number 20.
IST20_1	IST20_1.5	Point I/O Input Card Module ST20, Slot 1, Bit 5	Station 20 (only) inputs.
ST20_30		Distributed Point I/O AENT Module ST20 & ST30	Point I/O AENT Module associated with Station Numbers 20 and 30, the I/O Configuration name can include all Stations.
IST20_1	IST20_1.5	Point I/O Input Card Module ST20, Slot 1, Bit 5	Station 20 (only) inputs. The I/O Configuration name for the card is allowed to be just the Station Number associated with the card when the Station Number is also part of the AENT Module configuration name.
IST30_3	IST30_3.5	Point I/O Input Card Module ST30, Slot 3, Bit 5	Station 30 (only) inputs. The I/O Configuration name for the card is allowed to be just the Station Number associated with the card when the Station Number is also part of the AENT Module configuration name.
BNI2	IBNI2.4	Balluff BNI Module Module 2, Input 4	I/O configuration name is consistent with the I/O device tag(s).

Table 2: I/O Tag Consistency – Distributed I/O

Naming – Consistency Between I/O Configuration Device Names and Tags			
Name – I/O Configuration	Tag Name Examples	Device	Explanation and Clarification Notes
DM302	CodeReaderRack_DM302CommActive	Cognex Dataman 300 Series Code Reader	Cognex Dataman Module 302, associated with reading the rack code, communication active OTE instruction programmed in the CodeReader_Rack routine.
DM1155	CodeReaderRack_DM1155CommActive	Cognex Dataman 300 Series Code Reader	An example of a Cognex Dataman Model 302 that has a device number 1155 (appears on Sheet 11, Line 55), tag names based on device number are allowed when used consistently throughout the project.
DM302_Pinion	CodeReaderPinion_DM302Status	Cognex Dataman 300 Series Code Reader	Cognex Dataman Model 302, associated with reading the pinion code, additional naming required if multiple code readers exist on the machine.
DM302_Sleeve	CodeReaderSleeve_DM302Status	Cognex Dataman 300 Series Code Reader	Cognex Dataman Model 302, associated with reading the sleeve code, additional naming required if multiple code readers exist on a machine.

Table 3: I/O Tag Consistency – Code Reader

Naming – Consistency Between Routine Names and Tags			
Routine Name	Tag Name Examples	Device	Explanation and Clarification Notes
R00_Main	Main_NoEstops		The OTE instruction programmed in the Main routine indicating that no emergency stop conditions exist.
R01_Mode	Mode_AutoMode		Automatic Mode Selected OTE instruction programmed in the Mode routine.
R03_Cycle	Cycle_MIC		Machine In Cycle (MIC) OTE instruction programmed in the Cycle routine.
R05_Sequence	Seq_Step025PartQualityCheck1		The OTE instruction programmed in the Sequence routine that initiates the first quality check, at step number 25 of the sequence.
R07_OutputMotions	Out_ClearToRetractPress		The OTE instruction programmed in the OutputMotions routine that indicates the conditions are clear to retract the press.
R07_OutputMotions	HMI_CloseClamp	HMI	A tag used in the R07_OtuputMotions routine, but set by the HMI, indicating a command from the HMI to close the clamp.
R08_Fault_Control	Fault_NoCycleStops		The OTE instruction programmed in the Fault_Control routine that indicates there are no cycle stop conditions. Note that there are multiple routines with tag names called "Fault_."
R08_Fault_CycleStop	Fault_PartPresentBackcheck		The OTE instruction programmed in the Fault_CycleStop routine that maintains memory of the Part Present switch being OFF. Note that there are multiple routines with tag names called "Fault_."
R08_Fault_ImmedStop	Fault_ImmedStop[0].1		Immediate Stop fault number 2; the OTE instruction programmed in the Fault_ImmedStop routine. The tag's description indicates the fault display text. Note that there are multiple routines with ta names called "Faults_."

Table 4: Routine and Tag Name Consistencies (Set 1)

Naming – Consistency Between Routine Names and Tags			
Routine Name	Tag Name Examples	Device	Explanation and Clarification Notes
R14_BNI_Master	BNI_Faults.Port2_Connection	Balluff BNI Module	A tag set in the BNI_Master routine. The Port2_Connection member of the UDT named UDT_BNI_Master_Faults_v2. In this example, the system has only one master BNI and therefore the UDT predecessor name (BNI_Faults) indicates no device number. The tag description for Port2_Connection includes the text "I/O Link Port #2 (physically port #3) Device Not Connected – Check Cable."
R14_BNI_Master	BNI_Faults.Port2_Connection	Balluff BNI Module	A tag set in the BNI_Master routine. The Port2_Connection member of the UDT named UDT_BNI_Master_Faults_v2. In this example, the system has only one master BNI and therefore the UDT predecessor name (BNI_Faults) indicates no device number. The tag description for Port2_Connection includes the text "I/O Link Port #2 (physically port #3) Device Not Connected – Check Cable."
R91_HyperCyl	Out_HypercylPowerTimer.DN		The done bit for the HyperCyl Power Stroke Raise / Lower Motion Fault Timer. The done bit is used in the Fault routine, however, the ta name includes the routine name "Out" since the timer is to be moved from the R91_HyperCyl routine and programmed in the OutputMotions routine.
R91_HyperCyl	Fault_ImmedStop[3].11		Immediate Stop fault number 108, HyperCyl Approach Motion Overtime, included in the library routine R91_HyperCyl, however, the logic is to be moved to the Fault_ImmedStop routine. The tag's description is to be consistent with the fault display text. Note that there are multiple routines with the tag names called "Fault_".

Table 5: Routine and Tag Name Consistencies (Set 2)

TAG DESCRIPTIONS

- B.38 All tags shall have detailed descriptions. The descriptions shall not be a copy of the tag name. The descriptions should use full English words. To clarify: Descriptions should include more detail than the tag name. The purpose of the description is to provide additional information to clarify tag names that, due to length constraint, are not easily understood. Therefore abbreviations should also be avoided.
- B.39 The descriptions should be 5 lines or less with a maximum of 20 characters for each line.
- B.40 The descriptions for all I/O tags shall be consistent with the wording on the hardware drawings. Documentation for any unused I/O tags shall be deleted or noted as spare prior to shipment of the equipment.
- B.41 Example descriptions that provide clarification detail are listed below.

Tag Name	Description
Axis1_AutoAllowMoveRetPos	Axis-1 Auto Allow Move to Return Position
Analog_ToolingPosition	Tooling Position Scaled Value (inches)
Quality_RejectRemoved	Reject Part Removed from Nest
Out_AdvancePunch (u_Motion tag)	Advance the Hydraulic Notch Punch
.Enable (u_Motion member)	Conditions to Enable Motion
Out_AdvancePunch.Enable	Advance the Hydraulic Notch Punch Conditions to Enable Motion

Table 6: Tag Description Examples

C. Annex C – Complex or Special Sequence Considerations

C.1 General

Nexteer's logic philosophies and requirements for basic machine sequence are detailed above within the Sequence routine section of this specification.

The purpose of this annex is to explain how to incorporate several complex, special, or customized sequences into the Nexteer format and philosophy. For all applications, the design of the sequence logic needs to provide the Nexteer plant personnel with a quick understanding of how the machine processes the part.

Four variants of sequences are detailed in this annex:

- Variance in step-order, such as different sequences based on model selection
- Multiple simultaneous sequences, such as processes occurring at the same time within an over-all machine sequence
- Machine that repeats processes
- Use of the Sequence routine for machines specifically designed for hand-assembly of parts

The routines related to special sequence applications typically include the Main and Sequence routines.

Requirements:

Simple variance in machine sequence can be accomplished within one sequence routine. However, for more complex sequences as described within this annex, multiple sequence routines may be programmed.

Simple machine example: When running Model L and W it is required that two screws be tightened in the order of screw 1 then 2, but for Model R these two screws are to be tightened in the order of screw 2 then 1. This simple sequence variance may be accomplished with just a few logic contacts within the one sequence routine.

The Main routine requirements section of this specification states that the Main routine shall include logic that unconditionally calls (jumps to) all other routines. The routines shall be called in the same rung-order as is visible in the controller organizer.

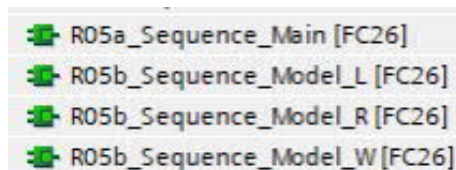


Figure C.3: Multiple Model-Dependent Sequence Routines

C.3 Variance in step-order:

For more complicated sequences, multiple sequence routines may be programmed.

Machine example, the Cover Screw station on a Solder Line: The station requires a differing number of screws for each model cover, as well as a differing screw order. In this example the station steps through its main sequence, and then proceeds to the correct Model L, W, or R screw sequence before returning to the main sequence routine to complete the cycle.

When multiple sequence routines are programmed for a single station, each sequence routine may include a Reset_Sequence output-energize instruction.

Note: In this example Reset_Sequence output-energize instruction from the Sequence_Main routine resets the steps from all routines.

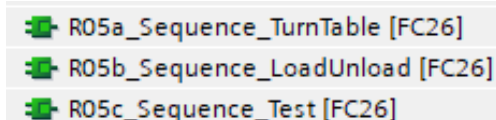


Figure C.4: Multiple Simultaneous Sequence Routines

C.4 Multiple simultaneous sequences:

Use of multiple sequences is a convenient programming method when multiple processes need to occur at the same time, such as when multiple stations on a dial table all process their respective part at the same time.

In the controller organizer shown in Figure C.4 to the left, the dial table index sequence is controlled by the TurnTable sequence. After index, the load and unload station sequence is controlled by the Load_Unload routine, while simultaneously the test station sequence is controlled by the combined Test routine.

Multiple sequences may be programmed on an individual station when multiple processes need to occur at the same time.

When multiple sequence routines are programmed for a single station, each sequence routine may include a Reset_Sequence output-energize instruction.

Note: In this example a separate Reset_Sequence output-energize instruction for each routine should be programmed to reset just that routine's steps from all routines.

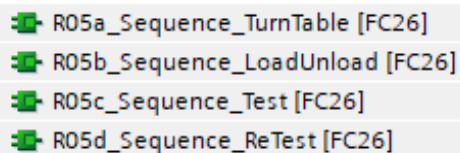


Figure C.5: Multiple Sequence Routines – Repeat Sequence

C.5 Repeat process steps:

Use of multiple sequence routines is also a convenient programming method when a part process needs to repeat a major portion of the machine sequence.

Typically, a station will step through the normal sequence of advancing motions, but when a process needs to be repeated, the logic should proceed to a separate sequence routine that includes both steps to retract motions and steps that then re-advance motions until reaching the position to repeat a process (reaching the steps to be repeated in the normal sequence routine).

Example: In the program blocks tree shown in Figure C.5 to the left, the test station's typical sequence (Test) engages the part and steps through the test process. After the test, if the part is allowed to be re-tested, the logic proceeds to the ReTest sequence which retracts the engage motion and resets the Test sequence such that the normal Test sequence

is re-run. The Test sequence would again engage and test.

Note: In this example Reset_TestSequence output-energize instruction has been programmed in the SequenceC_Test routine to reset all steps in both the SequenceC_Test and SequenceD_ReTest routines, which includes resetting the sequence when the ReTest sequence has retracted the engage motion.

D. Annex D - Cycle Pause - Pausing a Cycle

D.1 General

Pausing a cycle is **not** typical for Nexteer's production processes. Typically it is **not** appropriate to pause a cycle (and then restart the cycle). Therefore most machines should **not** include a Cycle Pause feature because of the complex logic that would be needed to ensure part quality and proper machine sequence. The logic would be complex when taking into account every possible incorrect or inadvertent machine or operator action that could occur while the machine is paused.

Logic that includes the ability to pause and then restart a cycle adds a level of complication to the Nexteer logic philosophies and formats established within this specification. Even so, the purpose of this Cycle Pause annex is to explain how to incorporate an operator initiated Cycle Pause (through an HMI pushbutton Cycle Pause) feature into the Nexteer format and philosophy, since the ability to pause a cycle is occasionally allowed for specific applications.

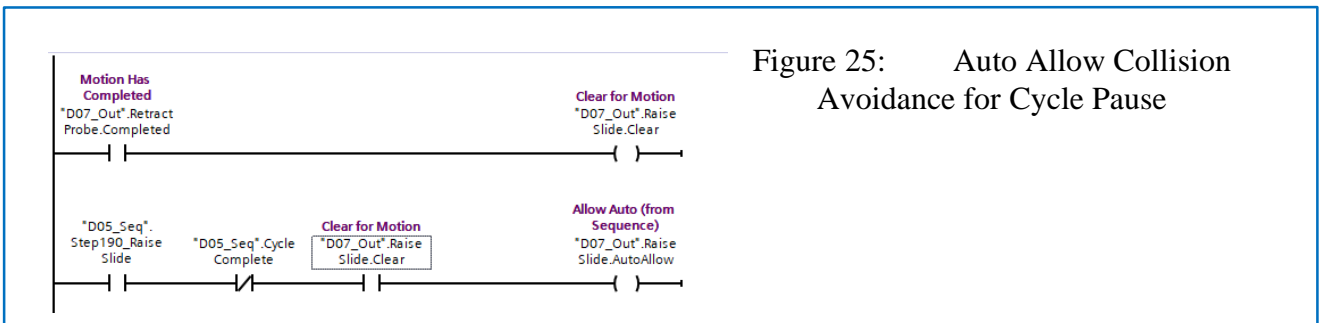
D.2 Requirements

A Pause Cycle momentary push button shall be included on the HMI Automatic screen.

Example logic for a Cycle Pause feature is available within the template routines. The Cycle routine will require additional rungs to break the Machine In Cycle, provide a pause request pulse, specify conditions that will allow a cycle to remain paused, and seal-in the paused state. The Immediate Stop fault routine will include a Maximum Paused Time fault if the cycle has been paused for more than two minutes.

D.3 Application Specific

Logic shall be included in the OutputMotions routine as revisions to the Auto Allow rung for each motion. Typically a contact from the motion's collision avoidance output-energize instruction (a), and a contact indicating that the previous machine sequence motion has been completed (b), are required. The first motion of the sequence does not require these contacts.



The application specific detail that **cannot** be shown in an example includes the complex logic required to ensure part quality and proper machine sequence. The most complex logic that shall be considered addresses these two questions for each and every motion and process:

- Will the part process occur correctly if any motion that has already fully advanced be hand-forced out of position while paused, and then be re-advanced by restarting a paused cycle?
- Will the part process occur correctly if any motion that has already fully advanced simply be de-pressurized (such as via interruption of a light curtain) while paused, and then be re-pressurized by restarting a paused cycle?

E. Annex E - Glossary

- E.1 **Abort Cycle:** An operator-initiated command to immediately stop the current machine cycle.
- E.2 **Auto allow:** A Nexteer phrase referring to the one output-energize instruction that provides a common method of interfacing the auto mode sequence logic into the standard solenoid output control rung of logic. One Auto Allow output-energize instruction is provided per direction of motion; and one contact is used from this output-energize instruction. Auto Allow is enabled by commands from the sequence logic.
- E.3 **Back Check:** Back check, back checked, or back checking, are terms or phrases that originated from logic that “checked” that an input was OFF (or went “back” to OFF after a cycle) such that the input would then be ensured to transition to ON during the cycle in order classify a part as a Good Part. Nexteer’s use of the term has evolved to be associated with any and all logic that ensures the function of inputs and input devices (including discrete, analog, and communication-based).
- E.4 **Collision avoidance:** A Nexteer phrase related to logic included to prevent damage to the tooling or part; logic that “avoids” a damaging “collision.” Similar terms include clear to move, motion interlocks, and motion constraints.
- E.5 **Control Function in the Event of Failure:** A term referenced from international machine standards such as IEC 60204-1. The term refers to how the machine control system is designed to detect, react, and function when a failure occurs.
- E.6 **Debounce (sensor debounce):** Bounce is a common industry term for the tendency of a contact in devices to generate multiple signals as the contact closes or opens, including potential multiple signals from the bounce of machine mechanics; “debounce” is any logic that ensures that only a single signal will be acted upon for a single opening or closing of a contact.
- E.7 **Error proofing:** An automatic device or method that either makes it impossible for an error to occur or makes the error immediately obvious once it has occurred.
- E.8 **MIC:** A Nexteer acronym used for the term Machine In Cycle.
- E.9 **OEM:** An acronym used for the Original Equipment Manufacturer; another term used for the machine builder.
- E.10 **PSDI:** An acronym for Presence Sensing Device Initiation, referenced from international machine standards such as ANSI. PSDI is the machine control function for starting a machine cycle based upon the loss of a signal from a presence-sensing safety device (or the absence of an operator within the safety device presence-sensing envelope, safely clear of the hazardous area).
- E.11 **Reset All Memories:** The control function and output-energize instruction that resets memories affecting, storing, or otherwise relating to part status and part quality.
- E.12 **Seal-in logic:** A common phrase in ladder logic referring to parallel contacts that keep an output-energize instruction in the ON state (“seal-in” the output). Although similar to the function of an output-latch instruction, Nexteer typical requires an output-energize instruction with parallel contacts seal-in such that all of the logic controlling the state of the output can be viewed within one rung of logic.
- E.13 **Template:** The Nexteer HMI and PLC logic files provided as examples of both (1) basic format, and (2) methods of compliance to Nexteer specifications.
- E.14 **Unconditionally called:** Logic that powers the output command directly from the left-hand power rail such that the command is executed each logic scan.

F. Annex F - References

- F.1 IEC 60204-1: Electrical Equipment of Machinery – Part 1: General Requirements
- F.2 SD-000: Nexteer Automotive Machinery and Equipment Specification
- F.3 SD-004: Nexteer Automotive Electrical Specification for Industrial Machinery Addendum to IEC 60204-1
- F.4 SD-2007: Nexteer Automotive Approved Components List
- F.5 SD-010: Nexteer Automotive Standard Equipment Specification
- F.6 SD-011: Nexteer Automotive Specification for Safety Circuits

- F.7 SD-2020: Nexteer Automotive Human Machine Interface Application Specification
- F.8 SD-1033: Nexteer Automotive RFID Application Specification
- F.9 SD-2034: Machine Controls Traceability
- F.10 TIA_v17_PLC_HMI_Template_Files_rev_date.zip: Nexteer Automotive Siemens TIA Portal files

NOTE: To obtain a copy of Nexteer Automotive specifications and templates visit our vendor document web-site currently at www.nexteerdatabexchange.com . Copies of any other referenced specification can be purchased, typically from the originating organization or at various industry specification web-sites.

RECORD OF REVISIONS

Revision #	Date	Section	Description
001	14JU25	All	Original Approval & Issue Date.
002			
003			
004			
005			
006			
007			
008			
009			
010			
011			
012			
013			
014			
015			
016			
017			
018			
019			
020			