Programmable Logic Controller
Application Specification

Global Common

SD-1032

ISSUED July 01, 2007
REVISED July 14, 2014

© 2014 Nexteer Automotive Corporation

All rights reserved

TABLE OF CONTENTS

1. SCOPE 1

SPECIFICATION 1

2. GENERAL 1

3. MODES 3

4. MACHINE-IN-CYCLE (MIC) 4

5. MOTION (SOLENOID OR MOTOR STARTER) 5

6. MOTION SENSOR BACK CHECKING 8

7. PART QUALITY LOGIC 9

8. REJECT HANDLING 11

9. FAULT CONTROL & DISPLAY 12

10. MACHINE MESSAGE CONTROL & DISPLAY 14

11. MULTIPLE HMI STATION EQUIPMENT 14

12. SHIFT REGISTER / INDEXING LOGIC 15

13. CONTROLLER ORGANIZER: STRUCTURE AND NAMES 15

14. RSLOGIX 5000 TAG CONFIGURATION 18

15. INTERNAL BIT & WORD TAG NAMING 18

16. LOCAL DISCRETE I/O TAG NAMING 20

17. LOCAL ANALOG I/O TAG NAMING 20

18. DISTRIBUTED DISCRETE I/O TAG NAMING 21

19. DISTRIBUTED ANALOG I/O TAG NAMING 21

20. ADDRESS AND TAG DESCRIPTIONS 22

LOGIC TEMPLATE ROUTINES 23

21. MAINTASK PROGRAM (ROUTINES) 23

22. TOOLKIT PROGRAM (ROUTINES) 26

LIST OF LOGIC EXAMPLES / FIGURES

FIGURE A: CONDITIONS FOR AUTO CYCLE, AND MACHINE IN CYCLE 4

FIGURE B: SOLENOID CONTROL 6

FIGURE C: MOTION SENSOR BACK CHECK FAULT 8

FIGURE D: REJECT PART PRESENT MEMORY 9

FIGURE E: ANALOG BACK CHECK METHOD 10

FIGURE F: REJECT CHUTE VERIFICATION 11

FIGURE G: RESET REJECT PART PRESENT MEMORY 12

FIGURE H: EMBEDDED VARIABLE FAULT TEXT STRING 13

FIGURE I: CONTROLLER ORGANIZER: ROUTINES 16

FIGURE J: CONTROLLER ORGANIZER: I/O CONFIGURATION 17

1. SCOPE

1.1. The intent of this document is to provide specifications to Original Equipment Manufacturers (OEM) in their design of programmable controller logic. This document describes the basic requirements for the format and content of PLC logic programs.

1.2. Nexteer logic templates reflect the requirements of this specification. The templates include routine naming and structure as well as tag naming convention. Logic templates are located on www.nexteerdataexchange.com for use with PLCs that are listed on our approved components list.

1.3. This document has a companion document, SD-1020 HMI Application Specification, which details associated display and control requirements.

1.4. The general principles embodied in this guideline should be applicable to the different brands of PLCs currently available in the market.

1.5. This document is not intended for off-the-shelf equipment (refer to SD-010) such as CNC equipment, as those software applications are typically equipment specific.

1.6. 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.

1.7. This document is organized into two parts, functional specifications (sections 2 through 20), and the description of Nexteer logic template routines (sections 21 and 22).

SPECIFICATION

2. GENERAL

2.1. The PLC model and programming software version(s) specified in SD-007 Approved Components List shall be used.

2.2. The RSLogix PLC file name shall include the asset tag number (SD number) of the machine(s).

2.3. The logic design shall be created in ladder logic format.

2.3.1. The use of Structured Text, Sequential Function Chart, and Function Block Diagrams shall not be used for programming.

2.3.2. The program control instruction MCR shall not be used.

2.4. Add-on instructions (AOIs) shall not be used for base machine control. AOIs may be used when provided by the device manufacturer. AOIs should be programmed in ladder logic format.

2.5. The use of indirect addressing requires the prior approval of the Nexteer controls engineer.

2.6. Rung comments should be included to clarify the function of complex logic, logic that is not easily understood. Examples include math operations, data manipulation, analog signal conversion, and communication to servo drives.

2.7. Forces or temporary logic used for bypassing specific functions shall be removed and proper logic operation shall be verified prior to MQ1 runoff of the equipment.

2.8. Un-used routines, logic, tag names, and descriptions shall be deleted prior to shipment of the machine. Exceptions being any unused fault and message array bits.

2.9. Latch coils shall not be used in motion circuits or hazardous outputs. (Latch coils are retentive coils that if they were ON when the PLC loses power, they are turned back on immediately when the PLC powers back up.)

2.10. A Test Lights function shall be provided for all hardwired pilot lights and/or operator indicator lights.

2.11. Logic for the manual load and/or unload station multi-colored LED pilot light (refer to SD-004) shall be designed based on the following criteria at a minimum. Additional functions may be used for any color but Red.

2.11.1. "GREEN": Solid Green shall indicate a Good Part. The light shall energize when all machine motions have completed (returned Home) 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.

2.11.2. "RED": Solid Red shall indicate a Reject Part; Flashing Red shall indicate a machine fault. When indicating a Reject Part the light shall remain energized until the reject part has been handled appropriately.

2.11.3. "YELLOW": Solid Yellow shall indicate Machine-In-Cycle (*refer to the Machine-In-Cycle section of this document*).

2.12. PLC Battery Low shall be indicated as a machine message that does not prohibit machine cycle. Also, when the PLC Battery Low status is present, a resettable machine cycle stop fault shall be triggered every 60 minutes until the battery is replaced.

2.13. The controller shall have at least 25% spare (unused) memory. A CompactLogix controller shall include 25% spare (unused) memory plus an additional 100k of memory if traceability routines have not been included by the original equipment manufacturer.

3. MODES

3.1. The machine shall power up with no mode active. One, and only one, mode shall be active. Motion shall be prevented when selecting a mode or when switching between modes. A separate action by the operator is required in order for any motion to occur.

3.2. Two modes are required – Manual and Automatic. Requirements for PSDI Mode, a third mode allowed for specific applications, are detailed in SD-011.

3.3. *Manual Mode* - Manual mode allows individual motions to function. 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 when needed. The system shall not be allowed to switch to manual mode (from auto mode) while the machine is in cycle.

3.4. *Automatic Mode* - Automatic mode is the mode that prohibits manual motions and allows normal machine cycles. Automatic mode does not initiate any machine motions.

3.5. Logic that prevents machine or part damage shall be active in both manual and automatic modes.

3.6. Other operating modes that may be included on machines are subcategories of Manual or Automatic mode.

3.6.1. If a "Return All Motions" function is provided, it can be used in Manual and Automatic mode, however, Return All Motions shall not function when the machine is in cycle. A hardwired Return All PB shall include a one-shot or pulse-logic to verify that a failure of the PB does not cause motion.

3.6.2. When in Automatic or Manual, the "Return All Motions" may function as long as the Return All pushbutton is held.

3.6.3. An automatic or sequenced Return All Cycle (initiated by pressing a pushbutton momentarily) may only be used in automatic mode. A return all cycle overtime shall be provided.

3.6.4. If a "Calibration" cycle is provided, it shall only function in Automatic mode.

4. MACHINE-IN-CYCLE (MIC)

4.1. MIC is the control function which makes a machine capable of producing automatic (sequenced) motions. Motion shall occur only when MIC is energized.

*To clarify: MIC is the one internal coil that enables automatic motions for the duration of the cycle. Auto mode is **not** the internal coil used throughout the logic to allow motions.*

Refer to the Fault Control section of this document for MIC overtime requirements.

4.2. The machine shall be allowed to enter into cycle (MIC shall energize) only if the following conditions are met:

4.2.1. Automatic mode is selected.

4.2.2. All sensors are verified to be in their proper initial state (typically, this means all of the returned sensors are activated and all of the advanced sensors are deactivated).

4.2.3. No faults are present on the machine.

4.2.4. All safety devices are in the "safe" condition (light curtain clear, safety gates closed).

4.2.5. A new part has been loaded or has entered into the machine.

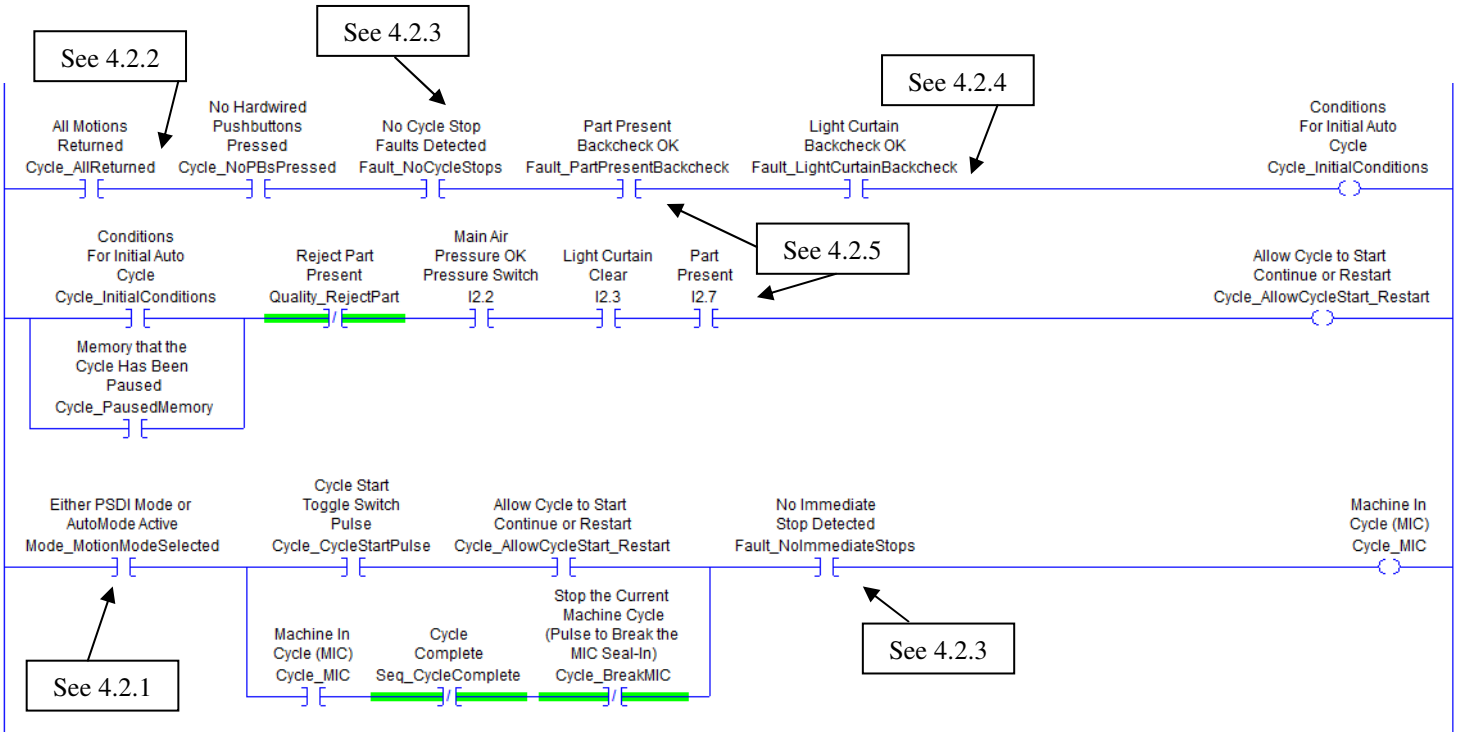


Figure A: Conditions for Auto Cycle, and Machine In Cycle

4.3. 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.

4.3.1. Single cycle machines shall execute one complete cycle for each initiation by the operator.

4.3.2. Continuous cycle machines execute repetitive cycles until halted by operator action or a fault condition. The first cycle shall be initiated by the operator.

4.3.3. A Cycle Stop pushbutton (either hardwired or on the HMI) shall be provided on continuous cycle machines. When the Cycle Stop pushbutton is pressed MIC does not instantly drop out. The machine is allowed to finish processing the part, return the machine to its normal starting position, and then drop MIC.

4.3.4. 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.

5. MOTION (SOLENOID OR MOTOR STARTER)

5.1. Each solenoid motion control rung, or process output rung, should be preceded by the manual collision avoidance rung and the auto sequence conditional rung as shown below.

5.2. Motions shall have minimal collision avoidance logic. Collision avoidance shall only be used to prevent damage to the equipment or damage to the part. Other terms used for "collision avoidance" include "clear to move", "motion interlocks", or "motion constraints".

5.3. The top branch in a solenoid motion rung normally includes MIC and auto sequence conditions to initiate the output. If this output is required multiple times during the cycle, these conditions are programmed in the auto sequence rung prior to the solenoid-control rung.

5.4. The second branch of each solenoid motion or process output rung contains the manual initiation of the output.

5.5. All outputs that initiate opposing motions shall be logically linked such that both motions cannot be active at the same time. Solenoids shall remain energized (sealed-in) until the opposite motion is initiated.

5.6. Solenoids shall remain energized (sealed-in) upon the actuation of the positional sensor indicating completion of the associated motion. The bottom branch of each solenoid motion rung contains the solenoid seal-in logic.

5.6.1. Exception 1: Certain hydraulic motion solenoids may not require solenoid sealed-in, thus preventing overheating of the hydraulic fluid.

5.6.2. Exception 2: The positional sensor contact is not required when a sensor does not exist. Also, the positional sensor contact shall not be used when a double-solenoid, detented valve is being controlled. A detented valve mechanically seals-in as soon as electrically commanded and the logic should match the actual condition.

5.6.3. The positional sensor contact cannot be replaced with an internal coil of "Advanced" or "Dwell" where the internal coil contains the normally closed (opposite state) input. This requirement exists because the logic would break the seal-in while an electrician is adjusting the opposite sensor.

5.7. Conditions to break the solenoid seal-in also include logical conditions indicating that hardwired power has been removed from the solenoid such as the following:

5.7.1. An E-Stop condition occurs (which breaks the seal-in on power down).

5.7.2. The light curtain is interrupted (required for Category 3 pneumatic applications when the safety blocking valve muting occurs, and implemented on Category 4 for consistency).

5.8. Valve solenoid seal-in control logic shall be designed as follows:

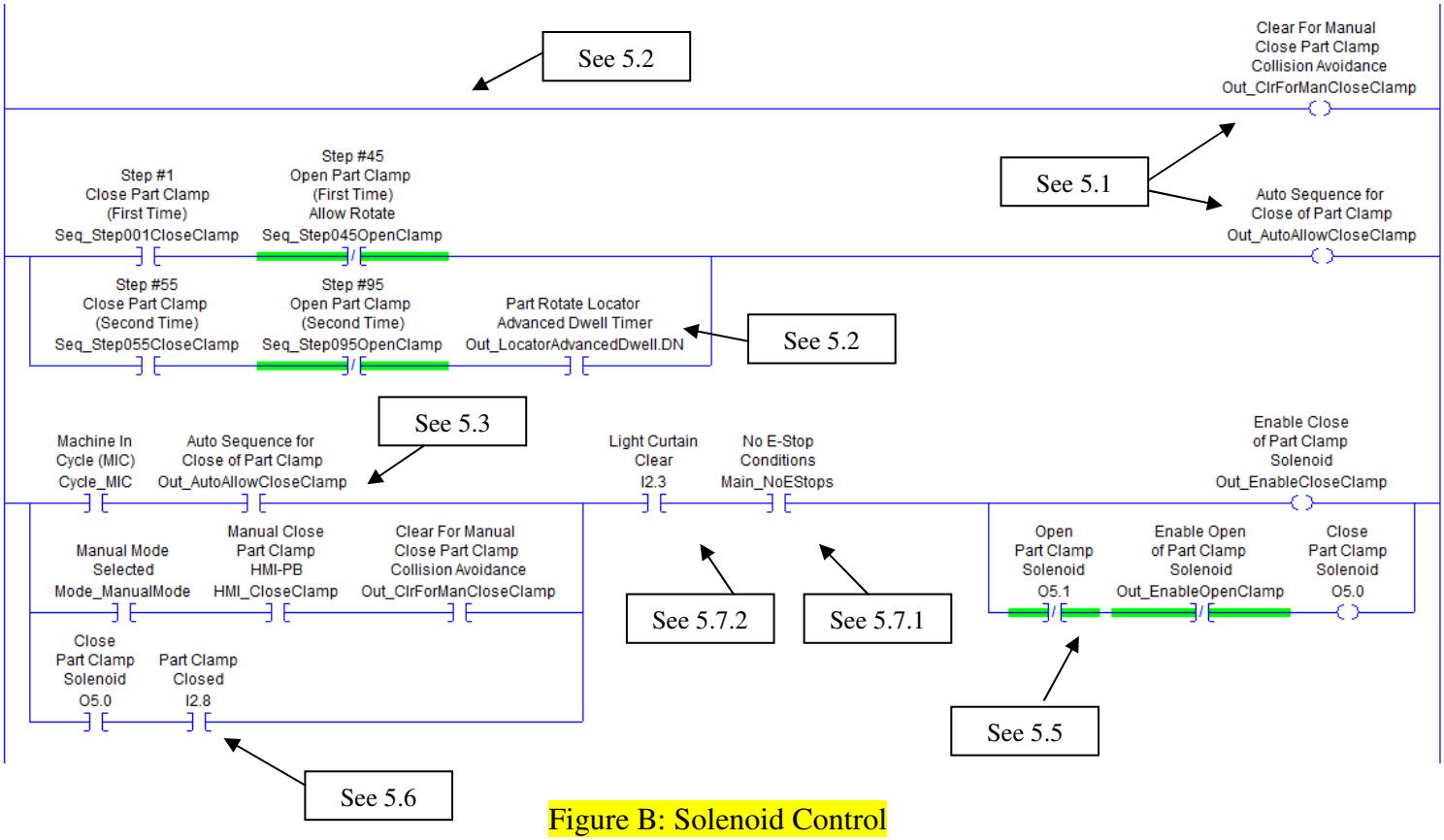


Figure B: Solenoid Control

5.9. Motor starter logic should be consistent with the solenoid control rungs shown above. Logic shall be included on the motor starters such that resetting of the overload device does not restart the motor.

5.9.1. A normally open auxiliary contact from the motor starter shall be wired to an input. A normally open contact from this input shall be used in the seal-in branch of the motor starter output coil.

5.9.2. On applications where the motor starter overload is not hardwired in series with the starter, an auxiliary contact from the overload shall be wired to an input. A normally open contact from this input shall be used as a condition to energize the starter coil.

6. MOTION SENSOR BACK CHECKING

6.1. All motion sensors shall be back checked.

To clarify: Back checking is a Nexteer phrase, a phrase meaning logic that both verifies proper input operation and detects input failure. Input failures may either stop the machine mid-process, or disallow the start of the next cycle. Back checking verifies both the function of the input sensor and the function of the electronics of the PLC input card.

6.2. Nexteer has developed standard back checking logic formats for motion sensors. Several methods are promoted / accepted.

6.2.1. Detecting an advanced sensor is "Off" prior to commanding the advance motion, such that the solenoid doesn't seal-in improperly (safety and part quality concern).

6.2.2. A simple scheme for motion sensors is to include all normally open "Retracted" inputs and all normally closed "Advanced" inputs in the logic to allow the initiation of cycle.

6.2.3. Another back checking scheme is to design the logic to include an "advanced" internal coil with a normally open sensor advanced input and a normally closed sensor retracted input (converse for retracted internal coil). This is the same scheme as referenced in the solenoid seal-in requirements in the Motion section. For this scheme to work, the internal coil shall be used throughout the logic (exception being that the solenoid seal-in requires the actual input to be used).

6.2.4. Another scheme for back checking motion sensors is to create a fault if the logic sees both the "advanced" and "retracted" inputs at the same time. This scheme requires an immediate process stop (loss of MIC) upon a back check fault.

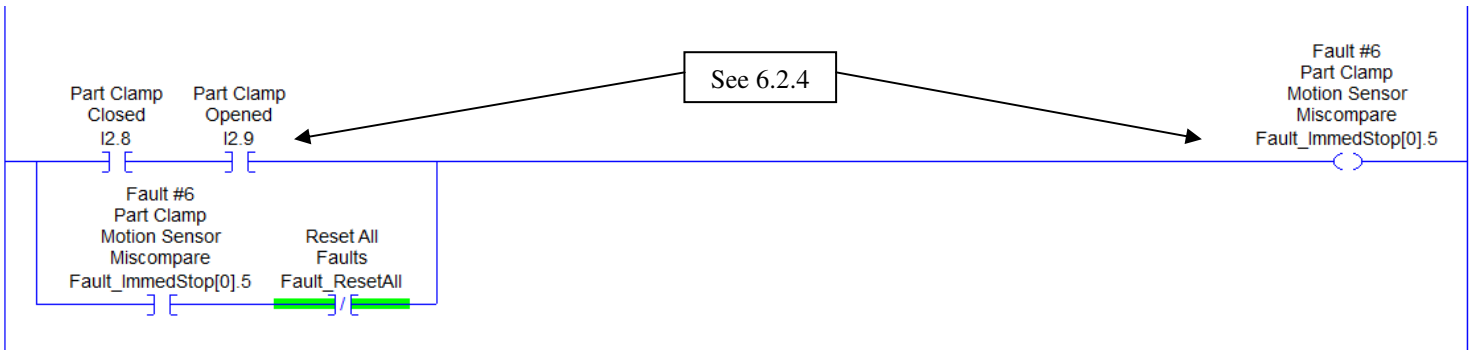


Figure C: Motion Sensor Back Check Fault

7. PART QUALITY LOGIC

7.1. Part quality logic shall be designed to prevent qualifying a Reject Part as a Good Part. At a minimum the machine logic shall reset Good Part status upon removal of the part. Where no Part Present signal is provided, the logic shall:

7.1.1. reset Good Part status upon change of mode.

7.1.2. reset Good Part status upon opening of an interlocked safety gate on automatic part-transfer systems where a part can be removed. *Note: On these automatic part-transfer systems, solenoid-locking interlock switches should be considered to minimize the number of rejected (scrapped) parts.*

7.2. Devices used to confirm a part is good and that protect the customer from getting a reject part shall also be back checked to verify they are operating as required. These devices can include sensors, auxiliary equipment (such as instrumentation), analog signals, and communication signals. *Refer to the Back Checking Clarification Note under the Motion Sensor Back Checking section above.*

7.3. Sensors and auxiliary equipment must turn on a PLC "Good Part" input when the desirable component, dimension, or feature is detected.

7.3.1. The sensor shall be detected to transition from "Off" to "On".

7.3.2. The logic shall default to calling a part "Reject" until it is proven to be "Good".

7.3.3. Cycling of power shall have no effect on the part quality decision. If a "Good" signal is detected "On" at power up, the logic must prevent the part from being called "Good".

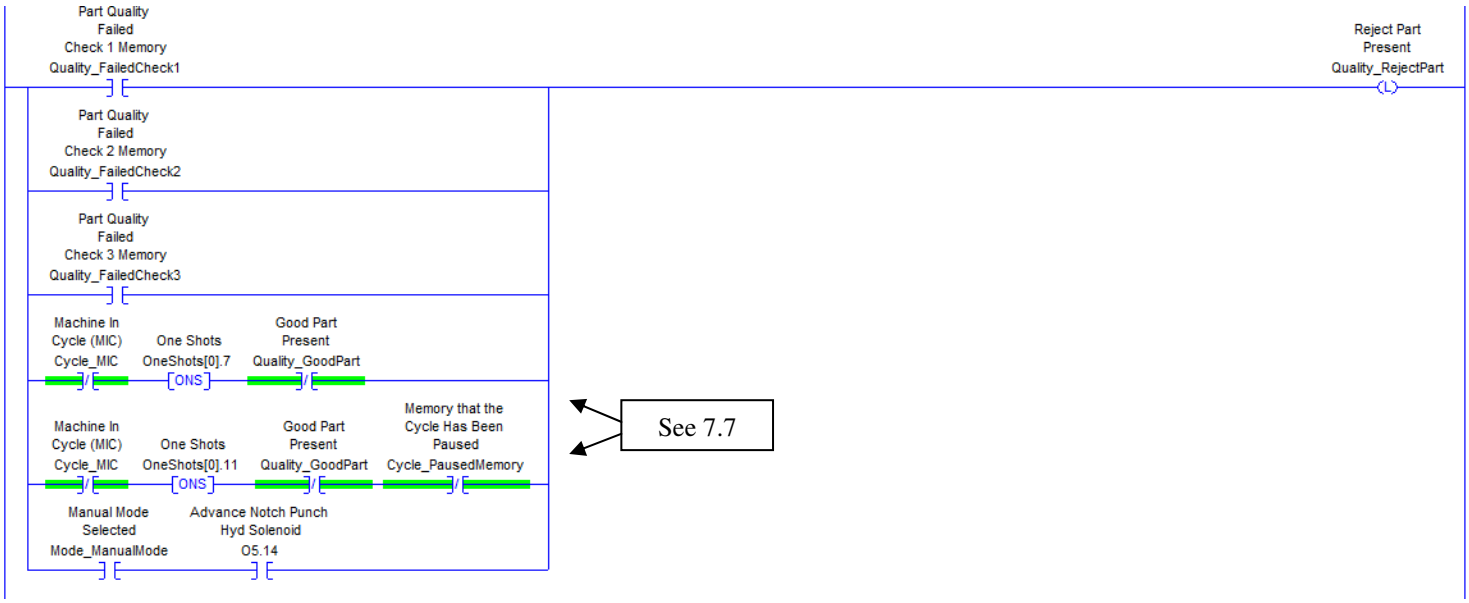


Figure D: Reject Part Present Memory

7.4. Analog signals shall be verified to move from a "Reject" to a "Good" range. The signal shall be verified to have returned to the reject range (typically a known initial position) as part of initial conditions to perform the next cycle. As an example, if for some reason the cable is unplugged from the device and the signal drifts into the good range the logic must prevent the part from being considered good through the use of back checking.

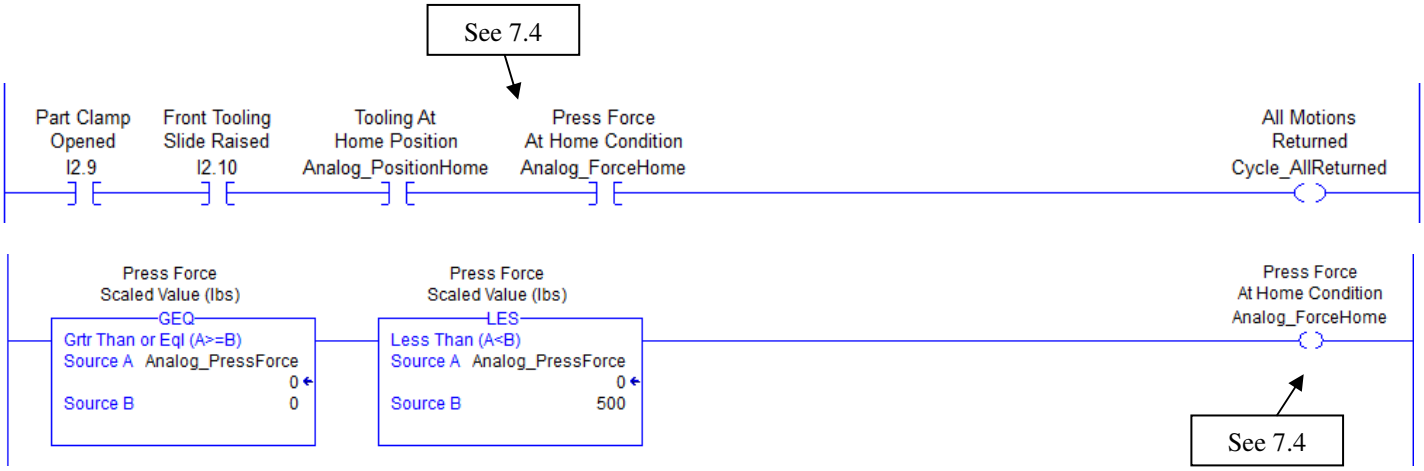


Figure E: Analog Back Check Method

7.5. Logic for other signals such as serial communications or parallel wire communications shall verify unplugging or broken wires will not classify the part as "Good".

7.6. Other quality sensors, such as part presence and pick bin, shall be verified to transition each cycle (often between machine cycles).

7.7. Applications may require a part to be rejected when there is a loss of MIC. Depending on the process, and when the MIC is lost, the part may be required to be rejected even though a quality check may not yet have occurred during the process. Examples include heat treating, welding, and other processes.

7.8. 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.

7.8.1. The logic shall reset part-quality memories upon loss of pallet present, upon pallet present, and upon power-up.

7.8.2. The logic shall prevent storing (or memory of) inaccurate pallet number(s) which can occur after the non-controlled transfer of pallets (such as the pushing the solenoid override of a valve).

7.8.3. The system shall store the pallet data table values of Last Station Number and Part Status on the pallet identification RFID tag (refer to SD-1033).

8. REJECT HANDLING

8.1. When a rejected part is present a means shall be provided to assure that the reject is disposed of or handled properly.

8.2. The reject part present internal coil shall seal-in until the reset sequence is completed. This internal coil shall remain sealed in upon power loss.

8.3. When error proofing includes hand-unload to a reject chute or bin, the logic shall require the following reject control sequence:

8.3.1. The logic shall annunciate a rejected part is present and keep the machine from cycling again until the acknowledgement process has been completed.

8.3.2. The machine shall drop out of cycle upon a Reject part.

8.3.3. The Reject part shall remain clamped or, where no part clamp is provided, at least one machine motion shall stay Advanced to prevent removal of part.

8.3.4. The logic shall require the operator put the machine in Manual mode and unclamp the part (or retract the appropriate motion).

8.3.5. The logic shall verify that the part has been removed from this machine. This 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.

8.3.6. 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 functionality of the reject chute sensor shall be verified (such as logic requiring a transition from an "Off" state to an "On" state).

Figure F: Reject Chute Verification

8.3.7. Upon completion of this reset sequence the rejected part present internal coil may be reset. The machine is then permitted to go into automatic mode.

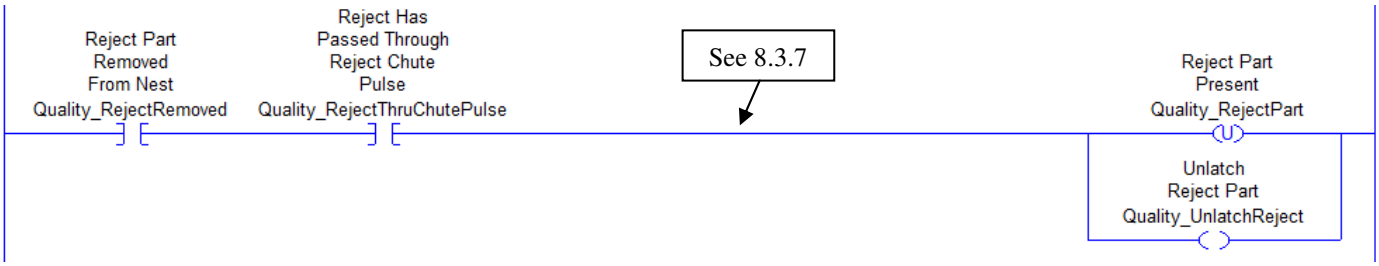


Figure G: Reset Reject Part Present Memory

8.4. For machines that include automatic unload of reject parts the logic shall meet the following reset sequence.

8.4.1. The part shall be placed in a reject chute or repair loop, depending on the application.

8.4.2. 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.

8.4.3. Upon completion of this reset sequence the rejected part present internal coil may be reset and the machine may now be permitted to begin another cycle.

9. FAULT CONTROL & DISPLAY

Any condition that prevents the cycle from starting or stops a cycle shall be displayed as a fault, message, or operator prompt. The following are fault control requirements.

9.1. All faults shall seal-in until the fault is intentionally reset by the operator by actuating the "Reset Fault" button/switch. The "Reset Fault" button/switch shall reset all fault conditions that no longer exist, but only when the machine is not in cycle.

Reference: To view all faults, when multiple faults exist, the HMI fault history screen is selected. This function is not controlled by the logic. Refer to SD-1020.

9.2. All faults shall be classified as "Cycle Stop" faults or "Immediate Stop" faults. A "Cycle Stop" fault allows the machine to finish the cycle, returning the machine to its normal start position, and then drops MIC. An "Immediate Stop" fault drops MIC immediately along with all motions and processing of the part. Emergency Stop inputs, safety gate inputs, and indications of situations that could cause harm to an operator or the machine (such as a fatal fault from servo system) shall be classified as immediate stop faults.

9.3. The logic shall include a cycle overtime timer that is active whenever the equipment is in cycle. If the cycle overtime timer times out, an immediate stop shall occur, and auto cycle shall be deactivated by dropping out MIC. All machines are required to have a cycle overtime fault. On single-cycle machines the timer for this fault is driven by MIC (only). For continuous cycle machines, MIC still drives the fault timer, but the fault is inhibited between cycles. The cycle

overtime fault is not inhibited by conditions such as a single limit switch that can fail, because such a failure can lead to inhibiting this safety-related fault.

Reference: Fault display text is stored in the HMI, through both an indicator and alarm list on every screen, and use of an alarm history screen. Refer to SD-1020.

9.4. The fault display text is allowed to include an Embedded Variable text string stored within the PLC for equipment delivered to Nexteer's USA or Mexico facilities. If Embedded Variable fault text strings are used for fault display, an STOD instruction shall be added to all faults.

Nexteer's logic template includes logic for both display methods. For Embedded Variable display the text string is stored in the associated Fault_ImmedStopText[x] and Fault_CycleStopText[x] tags.

The STOD instruction is a simple method for design-aid-display of the Embedded Variable text string associated with the fault. The STOD instruction does not perform any logical function in this application; it is not required for the fault control logic.

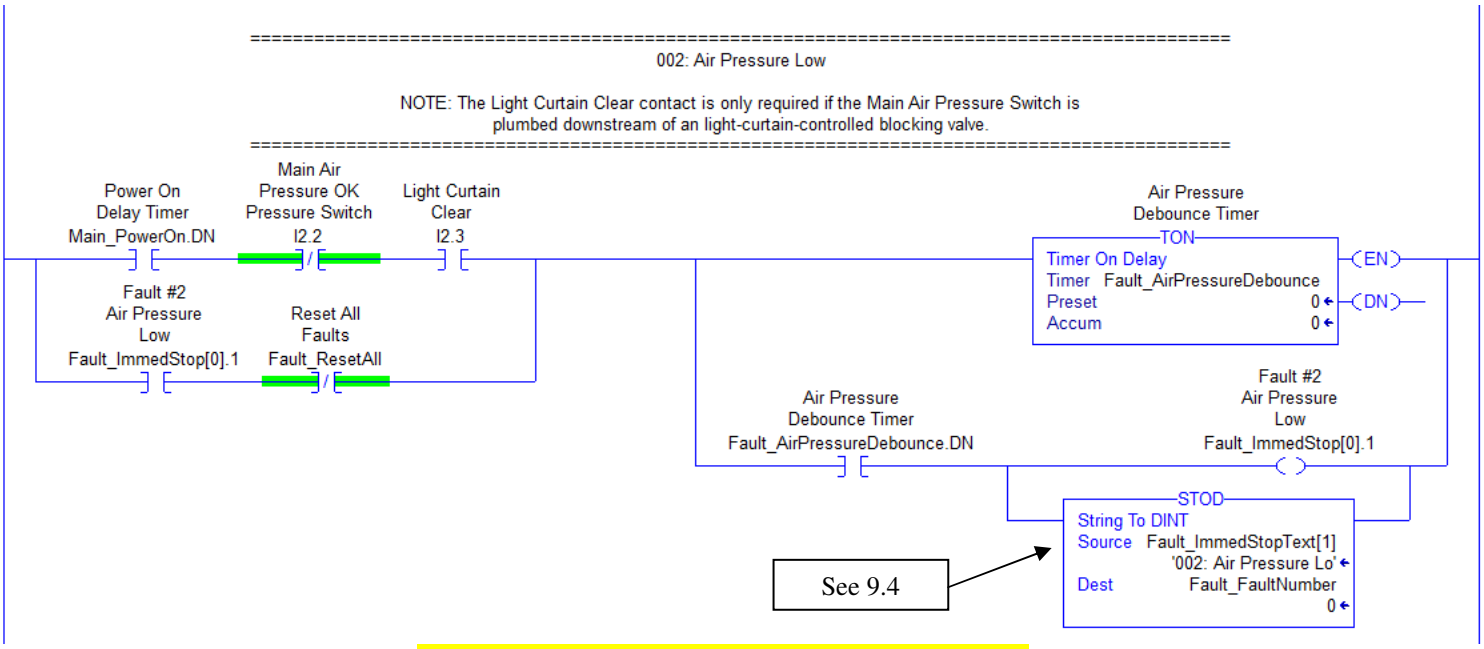


Figure H: Embedded Variable Fault Text String

10. MACHINE MESSAGE CONTROL & DISPLAY

Any condition that prevents the cycle from starting or stops a cycle shall be displayed as a fault, message, or operator prompt. The following are message control requirements.

10.1. When machine messages are used, the control and display of these messages is different from fault displays. These messages are displayed in a separate display box above the fault display box.

10.2. When a machine message is present, it should be displayed on the HMI only as long as the condition remains. The "Fault Reset" button/switch is not required to clear messages.

10.3. Machine messages are used to display information that may inhibit machine cycles. Messages like "PLC Battery Low", "Coolant Level Low", "Barrel Heat Zone Not at Temperature", and "Bowl Feeder Low" are examples.

10.4. When multiple machine messages exist, the logic shall automatically scroll through the messages, displaying each for 3 seconds. Once the last message has been displayed, the scrolling starts again at the first message.

11. MULTIPLE HMI STATION EQUIPMENT

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

11.1. Manual/Off/Auto selection is required on each HMI station that can initiate machine motion.

11.2. 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 selection from station HMIs shall disable automatic and manual mode selection at the main control HMI.

11.3. 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 Station Equipment 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.

12. SHIFT REGISTER / INDEXING LOGIC

12.1. Data transfer in a shift register shall be initiated by a shift pulse conforming to the following requirements.

12.1.1. 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 Index routine within the Toolkit program of the Nexteer logic template file.

12.1.2. When index mechanisms are used, the shift pulse should occur when the indexing mechanism begins to transfer parts from one station to another.

12.2. Shift register logic shall provide a Nexteer controls engineer approved method of controlling part quality data (to prevent qualifying Reject part as Good). Logic shall prevent against accepting Good Part status based on the reloading of memory (reloading of old or stale data). 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). Three approved methods of controlling this part quality data include:

12.2.1. clearing the shift register (classifying all parts as Reject) on PLC power-up (first scan logic), or

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

12.2.3. 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 if the part at unload is not the part that had been loaded to that pallet or fixture.

13. CONTROLLER ORGANIZER: STRUCTURE AND NAMES

This section of the specification uses the RSLogix 5000 term "Routine". These same specifications apply to the program LAD files of RSLogix 500 applications. *Note: Logic templates are located on nexteerdataexchange.com in the Toolkits, Templates and Forms selection under Vendor Documents.*

13.1. The PLC name (the Name field under Controller Properties) shall include the asset tag number (SD number) of the machine(s).

13.2. Logic should be organized, by logic function, into routines named to match their logic function.

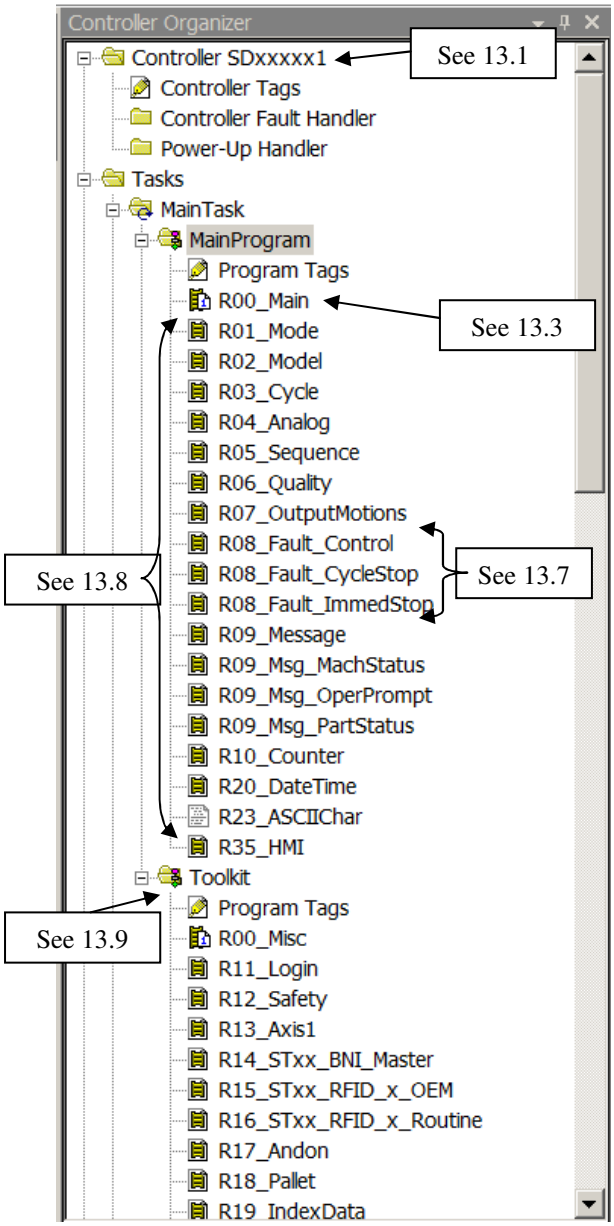


Figure I: Controller Organizer: Routines

13.3. All routines shall be organized under a program named "Main" for a PLC controlling a single station. The program Main should be the only scheduled program under the continuous task named MainTask.

13.4. For a PLC controlling several stations such as an assembly line, the structure should include a separate program for each station (typically named as the station number) under the continuous task named MainTask. Each station should include routines consistent with the single station routines depicted in the Nexteer logic template. The conveyor control logic should be included either as a routine under the Main program, or as routines under a program named Conveyor.

13.5. Logic shall be structured using routines that are unconditionally called from the Main routine. The Nexteer logic templates show the required structure and routine naming, typical on many Nexteer machines.

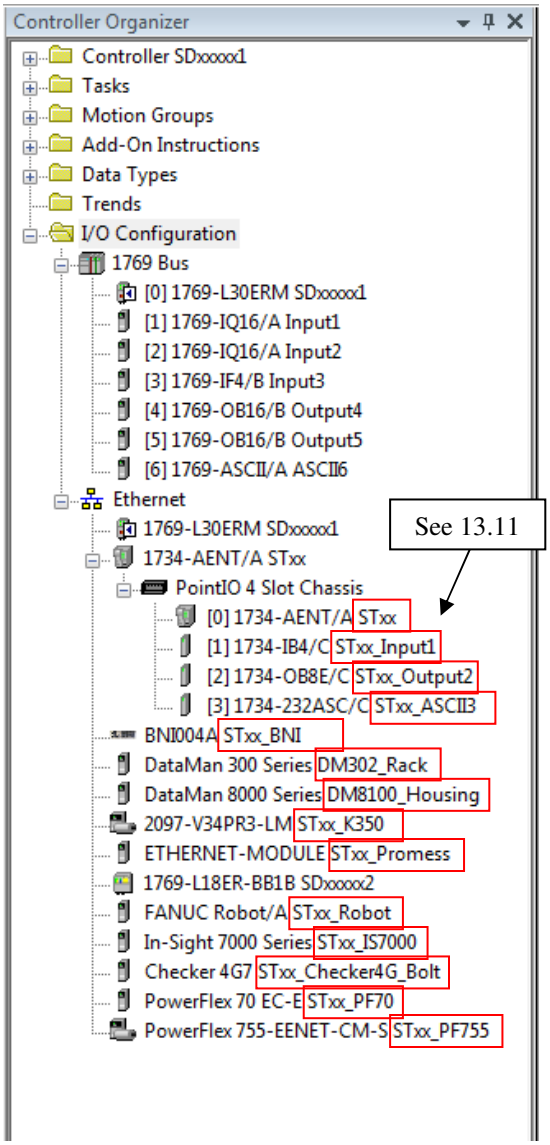
13.6. Use of periodic tasks requires the prior approval of the Nexteer controls engineer. For systems which use periodic tasks, processor utilization shall be kept to less than 80%. *Note: Example applications which have used periodic tasks include a complex assembly system with a large amount of communication traffic, or a system requiring instrumentation inputs read at a fixed-rate.*

13.7. The Controller Organizer view shall represent the PLC logic solve-order. Therefore, the routine names shall include the prefix Rxx (xx being in numerical order, however duplicate numbers are allowed to group related functions). The routines shall be called by the routine Main in this numerical order.

13.8. The routines provided in the Main Program of Nexteer's logic template shall be used for all applications. These routines are to be included with all equipment even when the equipment does not include the associate function (such as routines with no logic rungs). *Note: Descriptions of these logic routines are included in the Logic Template Routines section below.*

13.9. Additional routines are available for use depending on the application (routines provided in the Toolkit program of Nexteer's logic template). Toolkit routines, when used, shall be copied or moved to the MainProgram, and unconditionally called from the Main routine. Exception: Toolkit logic may be copied into the appropriate MainTask routine when noted within the toolkit routine description. *Note: Examples of routines currently available in the Toolkit are included in the Logic Template Routine section below.*

13.10. Additional routines are allowed for functions not illustrated in the Nexteer logic templates. Additional routines should follow the naming and functional details of this specification.



13.11. Non-local devices in the I/O Configuration (the Name field within the I/O Configuration / Properties) shall be named using the convention Xxx_Yyy_ZzzZzz. These names shall be unique for each device. Xxx, Yyy, and ZzzZzz should be combinations of the following:

13.11.1. Device type or model number.

13.11.2. Application or functional use description.

13.11.3. Station number (STxx) or OP number (for multi-station equipment).

Device name examples:

STxx_Output2
Station xx Output Card in Point I/O slot 2

DM8100_Housing
Dataman 8100 Housing Barcode reader

STxx_Checker4G_Bolt
Station xx Bolt Orientation Cognex 4G Checker

13.11.4. Device name length should be a maximum of 20 characters, abbreviated only when needed. The use of upper case characters to start each word is recommended.

Figure J: Controller Organizer: I/O Configuration

14. RSLOGIX 5000 TAG CONFIGURATION

A tag is a text-based name for an area within the controller's memory where data is stored. *Note: These tag names are stored in the controller while the tag descriptions are only stored in the offline file.*

14.1. All tags shall be defined as Controller scope. Exceptions: Program scope tags are allowed when approved by the Nexteer controls engineer. An example of allowed program scope tags may be a single PLC controlling several stations that require identical logic routines for each station and therefore can have identically named Program scope tags.

14.2. The use of double integer (DINT) data types is recommended for numerical value tags. The Logix CPU reads and manipulates 32-bit data values. To manipulate SINT or INT data, the controller converts the values to DINT, performs the programmed manipulation, and then returns the result to a SINT or INT value. Therefore DINT data types use less memory and execute faster than other data types.

14.3. Boolean (BOOL) arrays shall not be used. Exceptions: Boolean arrays are required for functions detailed in existing template routines such as Faults and Messages.

14.4. User-defined structures also referred to as user-defined data types (UDT) are allowed. Examples of their use are documented in the template Model and RFID routines. UDT members should be grouped by data type.

15. INTERNAL BIT & WORD TAG NAMING

15.1. Internal bit & word tag naming convention shall follow the format YYYYY_ZZZZZZZZZZ. Maximum overall tag name length should be 30 characters.

15.1.1. **YYYYY** = Routine name (or an abbreviation of the routine name) where the coil originated or device that the contact or signal is being controlled from.

Routine names (abbreviations) shall be:

Main	*(Main)*	Quality	*(Quality)*
Mode	*(Mode)*	Out	*(OutputMotions)*
Model	*(Model)*	Faults	*(Faults)*
Cycle	*(Cycle)*	Msg	*(Message)*
Analog	*(Analog)*	Count	*(HMI Display Counters)*
Seq	*(Sequence)*		

Application specific routines not shown above shall follow the same naming conventions.

Device name examples when not covered by above routines:

HMI *Tags "SET" by the HMI.*
Promess *Tags "SET" by a Promess unit.*
Vision *Tags "SET" by a vision system.*

15.1.2. **ZZZZZZZZZ =** A distinct functional description, abbreviated only when needed. The use of upper case characters to start each word is recommended.

15.2. User defined data types (UDT) shall follow a similar convention with the first underscore being the automatically generated ".", and the Z's being the member functional names within the structure (XXXXX.ZZZZZZZZ).

15.3. Example tag names are listed below.

Mode_ManualMode
Internal coil (Bool) set in the Mode routine that indicates manual mode has been selected.

Cycle_AllowPausedCycle
Internal coil (Bool) set in the Cycle routine that allows the machine cycle to remain in the paused state.

HMI_AutoModeSelectPB
Internal bit (Bool) set by a pushbutton from the HMI that selects Automatic Mode.

Fault_CycleStopText
Internal array (String) programmed in the Fault_CycleStop routine and used as an embedded variable string in the HMI message portion of each Cycle Stop alarm state.

Out_PressRetractedDwell
Internal timer set in the Output routine that indicates the press motion has been retracted.

Seq_Step005TriggerDM300
Internal coil (Bool) set in the Sequence routine that is Step #5 and triggers the Dataman 300 code reader.

Model_Setup.Postion1LowLimit
Internal tag (Float) that is part of a UDT structure set in the Model routine. This tag sets the low limit for quality check position 1 for the current model being run.

16. LOCAL DISCRETE I/O TAG NAMING

16.1. Hardwired local discrete I/O tag naming convention shall follow the format XY.ZZ.

16.1.1. **X** = "I" or "O" representing an Input or Output module.

16.1.2. **Y** = The input or output module slot location. Applications with more than 9 slots Y shall be two digits (YY) and include a leading zero such that tag names are in proper numerical order.

16.1.3. **ZZ** = The input or output module bit address. Generally these will be numbered 0 to 15, depending on the data type. These bit references are created automatically when the tag name XY is created.

16.2. Example tag names are listed below.

I2 *Tag written to in the main routine representing the input status for the 16pt input module in slot #2, Referenced in logic as* ***I2.0***, ***I2.1***…***I2.15***.

O3 *Tag written to in the main routine representing the output status for the 16pt output module in slot #3, Referenced in logic as* ***O3.0***, ***O3.1***…***O3.15***.

17. LOCAL ANALOG I/O TAG NAMING

17.1. Hardwired local analog I/O tag naming convention shall follow the format VWChX.

17.1.1. **V** = "I" or "O" representing an Input or Output module.

17.1.2. **W** = The input or output module slot location. Applications with more than 9 slots W shall be two digits (WW) and include a leading zero such that tag names are in proper numerical order.

17.1.3. **ChX** = The channel of the analog signal on a particular module.

17.2. Example tag names are listed below.

I3Ch0 *Tag written to in the main routine representing a transducer, such as the tooling position, wired to the analog input module in slot #3, channel 0.*

I3Ch1 *Tag written to in the main routine representing a load cell, such as the tooling force, wired to the analog input module in slot #3, channel 1.*

Note: Per SD-004, I/O conductors shall have the same identification as the I/O – in this example Cable I3Ch1 or conductors I3Ch1A and I3Ch1B.

18. DISTRIBUTED DISCRETE I/O TAG NAMING

18.1. Hardwired distributed discrete I/O tag naming convention shall follow the format WXXX_YY.ZZ.

18.1.1. **W** = "I" or "O" representing an Input or Output module.

18.1.2. **X** = The name, or an abbreviation of the name of the I/O module as configured in the controller organizer I/O Configuration area of RSLogix 5000.

18.1.3. **Y** = The input or output module slot location. Applications with more than 9 slots Y shall be two digits (YY) and include a leading zero such that tag names are in proper numerical order.

18.1.4. **ZZ** = The input or output module bit address. Generally these will be numbered 0 to 15, depending on the data type. These bit references are created automatically when the tag name XY is created.

18.2. Example tag name listed below.

OSta1_2 *Tag written to in the main routine representing the I/O status for the 8pt output module "O" of "Sta10" distributed I/O rack, slot #4, referenced in logic as* ***OSta1_2.0****,* ***OSta1_2.1****…****OSta1_2.7****.*

19. DISTRIBUTED ANALOG I/O TAG NAMING

19.1. Hardwired distributed analog I/O tag naming convention shall follow the format WXXX_YChZ.

19.1.1. **W** = "I" or "O" representing an Input or Output module.

19.1.2. **X** = The name, or an abbreviation of the name of the I/O module as configured in the controller organizer I/O Configuration area of RSLogix 5000.

19.1.3. **Y** = The input or output module slot location. Applications with more than 9 slots Y shall be two digits (YY) and include a leading zero such that tag names are in proper numerical order.

19.1.4. **ChZ** = The channel of the analog signal on a particular module.

19.2. Example tag name listed below.

IST20_7Ch0 *Tag written to in the main routine representing a transducer, as an example a slide position, wired to the analog input module "I" of Station 20 "ST20" distributed I/O rack, slot #7, channel 0.*

20. ADDRESS AND TAG DESCRIPTIONS

20.1. All addresses and tags shall have detailed descriptions that use full English words (no abbreviations). *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.*

20.2. The descriptions should be 5 lines or less with a maximum of 20 characters for each line.

20.3. The descriptions for all I/O points shall be consistent with the wording on the hardware drawings. Documentation for any unused addresses shall be deleted or noted as spare prior to shipment of the equipment.

20.4. The descriptions for all controller tag names or addresses shall be consistent with the text appearing on HMI objects such as pushbuttons and indicator lights.

20.5. Example descriptions are listed below.

Tag Name	**Description**
Axis1_AutoAllowMoveRetPos	Axis-1Auto Allow Move To Return Position
Analog_ToolingPosition	Tooling Position Scaled (inches)
Quality_RejectRemoved	Reject Part Removed From Nest
B3/100 (Logix500 only)	Raise Front Tooling Slide

LOGIC TEMPLATE ROUTINES

21. MAINTASK PROGRAM (ROUTINES)

The following routines are required for all equipment per the *Specification* section titled *Logic Structure and Routine Names* (above).

21.1. MAIN

21.1.1. Logic that controls the deterministic (once-per-scan) update of I/O status and the call all other routines.

21.1.2. Includes miscellaneous general logic such as standard flash / blinker logic.

21.1.3. May include general logic for indicator lights.

21.2. MODE

21.2.1. Logic that controls the selection of Automatic and Manual modes.

21.2.2. May include the programming of other machine modes when applicable.

21.2.3. Relates to Specification item *Modes* (above).

21.3. MODEL

21.3.1. Logic that controls selection or addition of new models or part numbers and their associated model setup parameters.

21.3.2. May include logic for automatic selection of model number.

21.4. CYCLE

21.4.1. Logic that controls initial conditions for machine cycle, initial positions, and Machine In Cycle (MIC) logic, including the Return All (Homing) function for machine motions.

21.4.2. This revision has been modified to include a Pause Cycle PB, cycle restart function, and Abort Cycle PB.

21.4.3. May include logic for cycle-related indicator lights.

21.4.4. Relates to Specification items *Modes, Machine-In-Cycle (MIC), and Motion Sensor Back Checking (above).*

21.5. ANALOG

21.5.1. Logic that controls the scaling and calculating of all analog and similar signals.

21.5.2. May include logic to compare the signals to limits, including back check logic.

21.5.3. Relates to Specification item *Part Quality Logic* (above).

21.6. SEQUENCE

21.6.1. Logic that controls the automatic, stepped, sequence of the machine.

21.6.2. This revision has been modified to include a method to allow the sequence to restart (where process-appropriate).

21.6.3. The sequence routine should be limited to the basic steps of the machine process. Strict adherence to this logic format should assist the destination plant support personnel to more-readily understand the machine process.

21.6.4. Relates to Specification items *Machine-In-Cycle (MIC), Motion (Solenoid or Motor Starter), and Reject Handling (above).*

21.7. QUALITY

21.7.1. Logic that determines the part quality, indicates the part quality status, and controls part handling based on the part quality status.

21.7.2. May include back check logic for part quality-related inputs including discrete, analog, or communication-based.

21.7.3. Relates to Specification items *Part Quality Logic*, and *Reject Handling* (above).

21.8. OUTPUTMOTIONS

21.8.1. Logic that controls all machine solenoid-controlled motions.

21.8.2. This revision has been modified to include clear-to-move and collision avoidance functions, as well as motion complete / dwell logic.

21.8.3. May include conventional motor-starter controlled motions. Other general outputs may be programmed here. *Note: Machine motions related to servos are typically programmed in a separate servo routine. Indicator lights are typically programmed in the routines related to their function.*

21.8.4. Relates to Specification items *Modes* and *Motion (Solenoid or Motor Starter)* (above).

21.9. FAULT

21.9.1. Multiple routines that detect machine faults including logic that controls the display, seal-in, and reset of faults on the HMI.

21.9.2. Sensor back check logic is typically included in these routinee.

21.9.3. This revision has been modified to reset all faults with a single button actuation. The logic also allows Embedded Variables for fault display text.

21.9.4. Relates to Specification items *Motion Sensor Back Checking, Part Quality Logic, and Fault Control & Display* (above).

21.10. MESSAGE

21.10.1. Multiple routines that control machine messages, the HMI display of those messages, plus display of machines status, part status, and operator prompts.

21.10.2. Relates to Specification items *Fault Control & Display, and Machine Message Control & Display* (above).

21.11. COUNTERS

21.11.1. Logic that controls good part and reject part production counter, and/or tool counter for display on the HMI.

21.11.2. It is not intended that all PLC counters are programmed in this routine. This routine should not include PLC counters used in sequence control, or fault control (such as a counter used to count the retry attempts on reading from a camera).

21.12. DATETIME

21.12.1. Logic that controls date and time information used within the fault routine.

21.13. ASCIICHAR

21.13.1. Logic for populating standard ASCII number, letter, and symbol strings.

21.14. HMI

21.14.1. Logic for HMI Screen control such as for the clean-screen function, and mode screens.

22. TOOLKIT PROGRAM (ROUTINES)

The following routines are available for use on all equipment per the *Specification* section titled *Logic Structure and Routine Names* (above). Not all available routines are described below. Example Toolkit routines include:

22.1. SAFETY

22.1.1. Logic for all safety related functions (refer to SD-011).

22.2. AXIS

22.2.1. Logic for servo axis logic.

22.2.2. If a machine has more than one axis each axis should have its own routine.

22.3. RFID

22.3.1. Multiple routines exist for both on-the-part RFID and pallet RFID functions (refer to SD-1033).

22.4. ANDON

22.4.1. Logic for the functions of preparing data and communicating with the plant's andon system.

22.5. INDEX

22.5.1. Logic for controlling the index function of a dial table or similar.

22.6. SHIFTTIMES

22.6.1. Logic for Shift Time functions and operation.

22.7. PARTCODE

22.7.1. Logic for generating Serial Number based on Julian Date.

22.8. TRACEABILITY

22.8.1. Multiple routines for traceability and electronic error-proofing applications.

22.8.2. This revision has been modified and updated.

22.9. INTERLOCK

22.9.1. Routine(s) related to the interconnection between machinery, done reliably (fail-safe), through produced-consumed tag communications.

22.9.2. Logic that controls the deterministic (once-per-scan) update of consumed tags based on a non-faulted connection status.

22.9.3. The logic depicts reliable communication practices that should be followed for other interconnections such as discrete I/O and serial communications.

22.10. ROBOT

22.10.1. Routine to demonstrate reliable Ethernet communications and diagnostics logic.

22.10.2. Logic from these toolkit routine should be imported as a single routine, a starting point to additionally include logic relevant for the proper interfacing of a robot.

22.11. COGNEXCHECKER

22.11.1. Routine to demonstrate reliable Ethernet communications and diagnostics logic.

22.11.2. Since implementation of a Cognex Checker typically involves minimal logic in the Sequence or Quality routines, logic from this toolkit routine should not be imported as a single routine, but copied into an existing routine within the MainProgram.

22.12. Device routines

22.12.1. Routines related to particular devices such as a Camera, Printer, HydPwerUnit (hydraulic power unit), and other fluid power devices.

RECORD OF REVISIONS

Revision #	Date	Section	Description
001	01JL04	All	Original Issue
002	28AU08	All	Major Revision
003	06NO09	All	Company name updated.
004	17DE10	All	Major Rev based on updated tag naming convention & RSLogix 5000.
005	14JL14	All	Major Revision based on Nexteer Central CSE 2014 BPI 2
006			
007			
008			
009			
010			
011			
012			
013			
014			
015			
016			
017			
018			
019			
020			