



Software Output Switching

Purpose of This Solution

This application solution uses ladder logic and a second Logix5550™ controller to keep a system running if the initial Logix5550 controller experiences a problem that prevents it from controlling the system. This solution switches control to the second controller if the initial controller experiences any of these situations:

- recoverable fault (major fault)
- non-recoverable fault (e.g., the operating system of the controller fails)
- program mode
- power loss
- communications loss, such as a failure of a communication module or break in a cable

When compared to a hardware-based redundancy solution, this solution uses fewer hardware components and provides sufficient back-up for applications that do not require a bump-less switch over. It provides a viable back-up solution for applications such as:

- air-handling
- refrigeration
- material handling

For an overview of the hardware-based ControlLogix™ redundancy solution, which is currently under development, refer to “Future ControlLogix Redundancy Solution” on page 27.

When to Use This Solution

Use this solution in the following situations:

- to protect against the failure of a controller within the following limitations:
 - A delay of several seconds or longer may occur before the second controller gains full control of the system.
 - During the switch over, outputs may temporarily revert to the values for Fault mode, according to the I/O configuration for the modules.
 - The application does *not* require the 1756-M02AE motion module.
- as an interim redundancy solution until hardware-based redundancy is available

Important User Information

Because of the variety of uses for the products described in this publication, those responsible for the application and use of this control equipment must satisfy themselves that all necessary steps have been taken to assure that each application and use meets all performance and safety requirements, including any applicable laws, regulations, codes and standards.

The illustrations, charts, sample programs and layout examples shown in this guide are intended solely for purposes of example. Since there are many variables and requirements associated with any particular installation, Allen-Bradley does not assume responsibility or liability (to include intellectual property liability) for actual use based upon the examples shown in this publication.

Allen-Bradley publication SGI-1.1, *Safety Guidelines for the Application, Installation and Maintenance of Solid-State Control* (available from your local Allen-Bradley office), describes some important differences between solid-state equipment and electromechanical devices that should be taken into consideration when applying products such as those described in this publication.

Reproduction of the contents of this copyrighted publication, in whole or part, without written permission of Rockwell Automation, is prohibited.

Throughout this manual we use notes to make you aware of safety considerations:

ATTENTION



Identifies information about practices or circumstances that can lead to personal injury or death, property damage or economic loss

Attention statements help you to:

- identify a hazard
- avoid a hazard
- recognize the consequences

IMPORTANT

Identifies information that is critical for successful application and understanding of the product.

Allen-Bradley, ControlLogix, DH+, Logix5000, Logix5550, Logix5555, PLC-5, and SLC are trademarks of Rockwell Automation.

ControlNet is a trademark of ControlNet International, Ltd.

Ethernet is a trademark of Digital Equipment Corporation, Intel, and Xerox Corporation.

How to Implement This Solution

To implement software output switching, complete the actions outlined in this application solution:

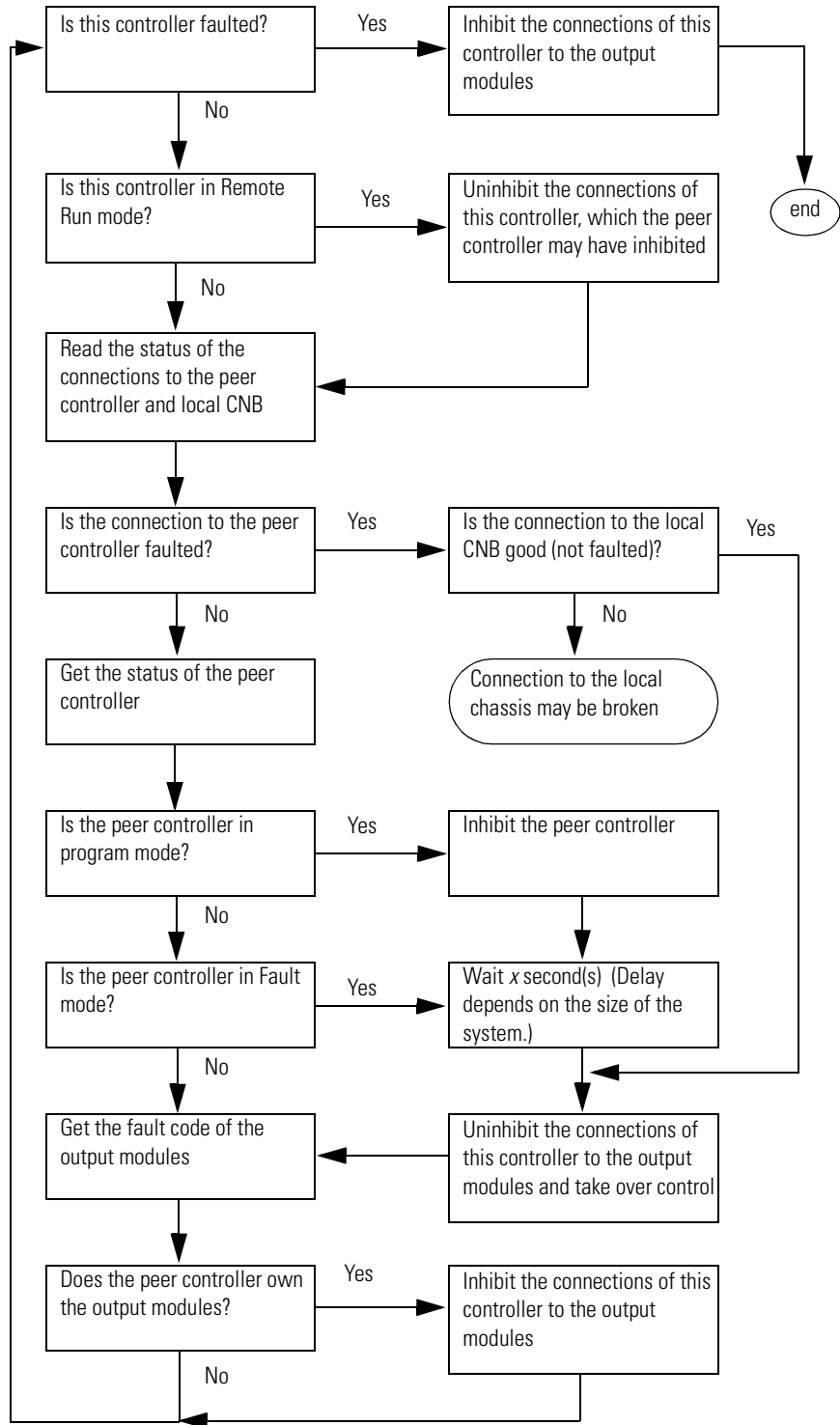
- Review the Overview of Software Switching
- Design the System for Multiple Controllers
- Develop the Project for the Initial Controller
- Develop the Project for the Second Controller
- Tune the System for Efficient Switch Over

Overview of Software Switching

To switch control from one controller to another controller, each controller performs these actions:

- monitors the status of the other controller (peer controller) and checks for these conditions:
 - key position of the peer controller
 - mode of the peer controller
 - non-faulted connection between the controllers
- establishes owner connections to the input modules:
 - Input modules broadcast input data to both controllers, enabling the programs of each controller to remain synchronized.
 - If one of the controllers become inoperative, the second controller maintains ownership of the input modules.
- maintains a configuration for each output module:
 - One controller establishes connections to the output modules and controls the outputs.
 - The other controller uses ladder logic to inhibit (disable) its connections to the output modules.
 - If the controller that owns the output modules fails, the second controller uninhibits its connections to the output modules and takes over control.
 - During the switch over, outputs revert to the values for Fault mode.

The following flowchart depicts the logic that each controller uses to monitor the system and take over control, if required.



Switch Over Delay

The switch of control from one controller to another does *not* occur immediately. The delay could range from hundreds of milliseconds to several seconds or longer, and depends on these factors:

- size of the system
- quantity of I/O in the system
- distribution of the output modules in remote chassis
- speed of the system
- type of failure
 - If a controller enters Fault or Program mode, it must close its connections to the output modules before the peer controller can take over control.
 - If a controller fails or loses communications to the system, its connections close, so the peer controller can take over control as soon as the failure is detected.

Design the System for Multiple Controllers

Software output switching requires the following design considerations:

- Controller Placement
- I/O Placement
- Communication Networks
- Electrical Wiring

Controller Placement

Software output switching requires at least two controllers. Although you can use more than two controllers, this solution uses two controllers as its model. You can place the controllers in either of these locations:

- the same chassis (single-chassis configuration)
- separate chassis (dual-chassis configuration)

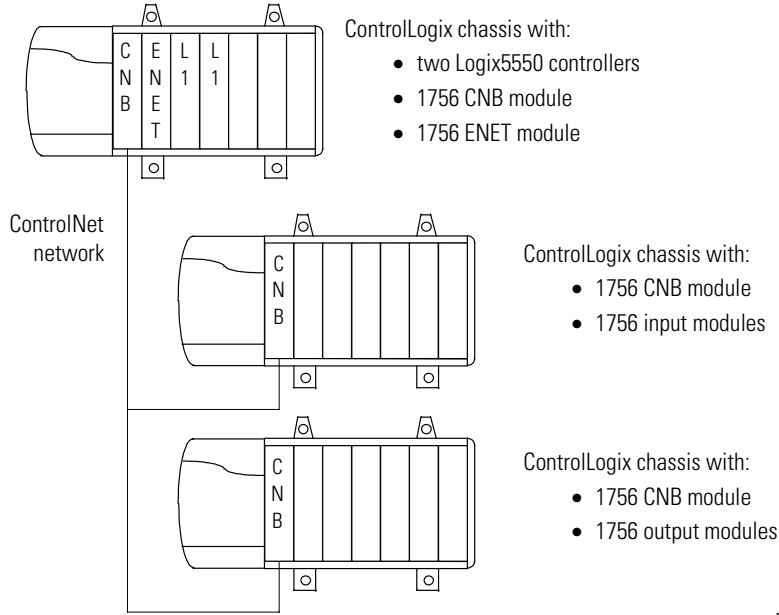
Use a dual-chassis configuration for these situations:

- to protect against either of these failures:
 - loss of power to a chassis that contains a controller
 - loss of communications to a controller, such as failure of a communication module or break in a cable
- in preparation for a transition to a hardware-based redundancy solution in the future

The following examples depict single and dual chassis configurations.

EXAMPLE

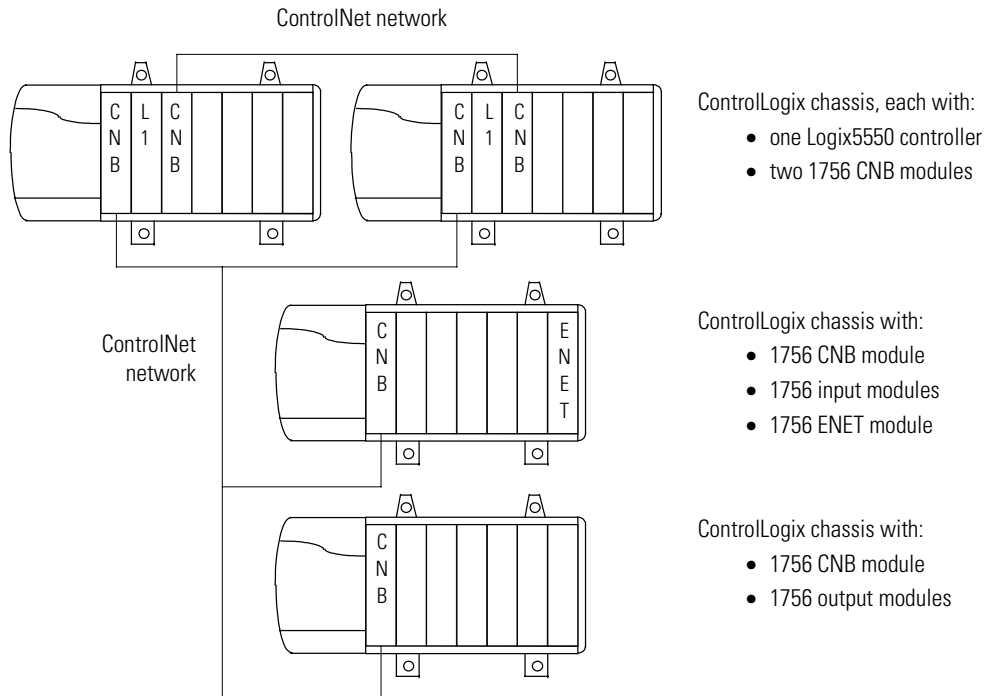
Single Chassis Configuration



42183

EXAMPLE

Dual Chassis Configuration



42184

I/O Placement

If you are using a dual-chassis configuration, place input modules and output modules in separate, remote chassis, as depicted in the “Dual Chassis Configuration” example on page 6. This lay-out reduces the amount of ladder logic required to perform the switch over:

- Since only one controller at a time can own an output module, ladder logic will inhibit or uninhibit the connection to each output module.
- If you place output modules in their own chassis, you can inhibit or uninhibit the connection to the ControlNet bridge module in the remote chassis, which inhibits or uninhibits the entire chassis.
- If you mix input and output modules in the same chassis, you will have to enter logic to inhibit or uninhibit each output module in the chassis.

IMPORTANT

I/O that is in a remote chassis updates *no* faster than the network update time (NUT) of the ControlNet network.

Communication Networks

Use the following table to select the required communication networks:

For this type of communications:	Use this network:	
	single-chassis configuration	dual-chassis configuration
I/O	ControlNet network for any remote I/O	ControlNet network for all I/O
controller-to-controller	backplane of the chassis	ControlNet network: <ul style="list-style-type: none"> • the same network that is used for I/O communications <i>or</i> • a separate network
software interface (e.g., RSLogix 5000 software, MMI software)	Either of these networks: <ul style="list-style-type: none"> • ControlNet • Ethernet 	

IMPORTANT

If you plan to transition to a hardware redundancy solution in the future, allocate ControlNet addresses as outlined in “System Configuration” on page 28.

If you use an Ethernet network in combination with a dual-chassis configuration, you can use one or two Ethernet modules:

If you want to:	Then:
minimize cost	A. Purchase one 1756 Ethernet module.
transition to a hardware redundancy system when it becomes available	B. Install the module in a chassis containing remote I/O.
protect against the failure of an Ethernet module	A. Purchase two 1756 Ethernet modules. B. Install one module in each chassis that contains a controller

Electrical Wiring

If the electrical system contains a master control relay (MCR) with an output contact that is controlled by the controller, during a switch over the output will reset and the MCR will drop out, requiring a manual restart.

- You may be able to design the electrical system to hold in the MCR while the switch over occurs.
- In either case, you can restart the system with only one controller operating.

ATTENTION

To prevent injury to personnel or damage to equipment, ensure that all electrical wiring follows local electrical codes.



For more information on wiring and grounding, refer to *Industrial Wiring and Grounding Guidelines*, publication 1770-4.1.

Develop the Project for the Initial Controller

In the project for the initial controller, complete the following tasks:

- Configure I/O
- Create Programs and Routines
- Create User-Defined Data Types
- Create Tags
- Enter Logic

Configure I/O

Configure the I/O according to the lay-out of your systems. As you configure the I/O:

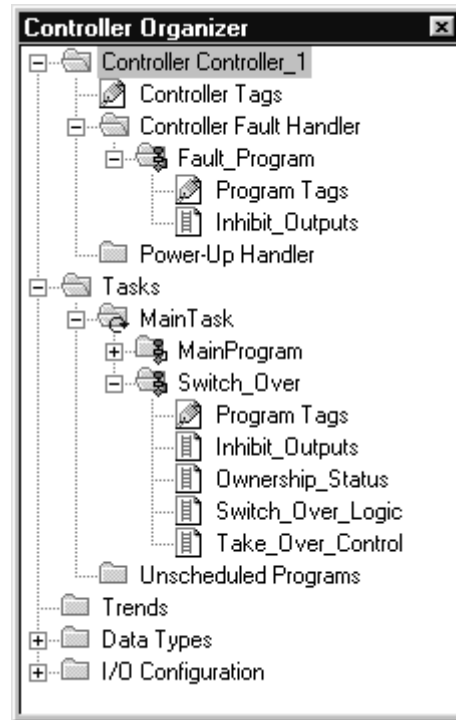
- Include the peer controller in the I/O configuration.
- Assign a name to each module.
- For each I/O module, select an owner communication format. (I.e., Do not select a listen-only communication format.)
- If viable, configure output points and channels to hold their last state when the controller enters the Fault mode. This minimizes bumps while the switch over occurs.

ATTENTION

When a controller enters the Fault mode, an output device that is configured to hold last state remains in the state in which it was in at the time of the fault until the fault is cleared or another controller takes over control of the output. Before you configure an output device to hold last state, verify that this configuration will not injure personnel or damage equipment.

Create Programs and Routines

The controller requires the following programs and routines (in addition to the tasks, programs, and routines that are required for your application):



1. Create the following programs:

In this task:	Create this program:	
	Name:	Description:
Controller Fault Handler	Fault_Program	Handles recoverable major faults
the continuous task of the project	Switch_Over	Monitors the peer controller and takes over control when required

2. Create the following routines:

In the program:	Create this ladder routine:	
	Name:	Description:
Fault_Program	Inhibit_Outputs	When the controller faults, inhibits the output modules so the peer controller can take over control
Switch_Over	Switch_Over_Logic	Monitors the peer controller and initiates a take over of control if required
	Ownership_Status	Checks for a conflict in the ownership of output modules, which signals that the peer controller has established connections to the modules
	Take_Over_Control	Uninhibits the output modules, enabling the controller to control the outputs
	Inhibit_Outputs	Inhibits the output modules

3. Assign main routines:

For this program:	Assign this routine as the main routine:
Fault_Program	Inhibit_Outputs
Switch_Over	Switch_Over_Logic

Create User-Defined Data Types

1. Create the following user-defined data type, which will be used for information about each module:

Name: MODULE

Description: Stores attributes of the MODULE object

Members:

Name:	Data Type:	Style:
FaultCode	INT	Hex
Mode	INT	Binary

2. Create the following user-defined data type, which will be used for the status of the peer controller:

Name: CONTROLLER_DEVICE

Description: Stores the attributes of the CONTROLLERDEVICE object

Members:

Name:	Data Type:	Style:
Vendor	INT	Decimal
ProductType	INT	Decimal
ProductCode	INT	Decimal
Revision	INT	Decimal
Status	INT	Binary
SerialNumber	DINT	Decimal
DeviceNameLength	SINT	Decimal
DeviceName	SINT[32]	Decimal

Create Tags

1. Create the following controller-scoped tags:

Scope:	Tag Name:	Type:	Style:	Description:
controller	Inhibit_Reset	INT[3]	Decimal	Values that uninhibit the connections of a controller
	Inhibit_Reset_IO_Map_State	INT[3]	Decimal	Holding values for message
	Inhibit_Set	INT[3]	Decimal	Values that inhibit all the connections of a controller
	Inhibit_Set_IO_Map_State	INT[3]	Decimal	Holding values for message
	Peer	CONTROLLER_DEVICE	<i>na</i>	Information about peer controller
	Peer_Dummy_Consume	DINT	Decimal	Dummy tag to establish connection with peer controller
	Peer_Dummy_Produce	DINT	Decimal	Dummy tag to establish connection with peer controller
	Peer_Inhibit	MESSAGE	<i>na</i>	Inhibits all connections of the peer controller
	Peer_Read_Status	MESSAGE	<i>na</i>	Gets information about peer controller
	This_Uninhibit	MESSAGE	<i>na</i>	If connections were inhibited by the peer controller, uninhibits all connections of this controller

2. Modify the following controller tag to produce data:

Produce this tag:	For this many consumers:
Peer_Dummy_Produce	1

3. Modify the following controller tag to consume data:

For this tag property:	Specify:
Name	Peer_Dummy_Consume
Tag Type	Consumed
Controller	<name of peer controller>
Remote Tag Name	Peer_Dummy_Produce
RPI	value of the NUT for the ControlNet network
Data Type	DINT

4. In the Inhibit_Reset array, enter the following values, which uninhibits all of the connections of the controller:

In this array element:	Enter:
Inhibit_Reset[0]	1
Inhibit_Reset[1]	7
Inhibit_Reset[2]	2#0000_0000_0000_0001

5. In the Inhibit_Set array, enter the following values, which inhibits all of the connections of the controller:

In this array element:	Enter:
Inhibit_Set[0]	1
Inhibit_Set[1]	7
Inhibit_Set[2]	2#0000_0000_0000_0101

6. Create the following program-scoped tags for the Switch_Over program:

Scope:	Tag Name:	Type:	Style:	Description:
Switch_Over	Ownership	DINT	Decimal	Status of connections to output modules
	Peer_Connection	MODULE	<i>na</i>	Information about connection to peer controller
	Peer_Delay	TIMER	<i>na</i>	Delays take over until connections from peer are inhibited.
	Peer_Failed	BOOL	Decimal	Connection to peer has failed.
	Peer_Faulted	BOOL	Decimal	Peer controller is faulted.
	Peer_In_PROG	BOOL	Decimal	Peer controller is in program mode.
	Peer_OK	BOOL	Decimal	Peer controller is operational.
	This_In_REMOTE	BOOL	Decimal	Key is in remote position.
	This_Status	INT	Binary	Mode of controller

7. For each output module and each ControlNet Bridge module in the system, create a tag that will store information about the module:

Scope:	Tag Name:	Type:	Style:	Description:
Fault_Program	<name>	MODULE	<i>na</i>	Information about connection to module

where:

name is the name of the output module or ControlNet Bridge module, as depicted in the I/O configuration of the project.

8. Copy the tags that you created in Step 7 and paste them into the program tags for the Switch_Over program.

Enter Logic

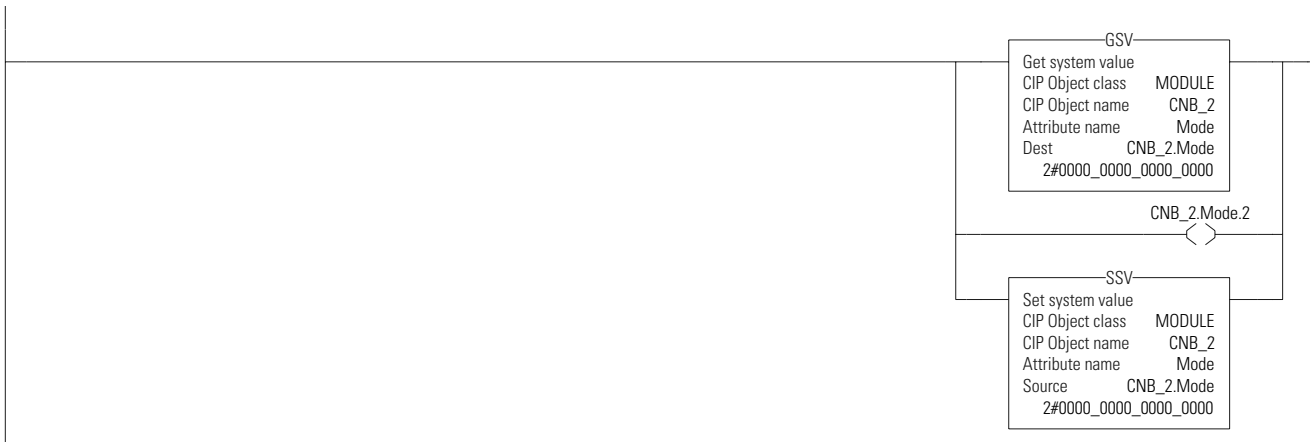
1. Open the Fault_Program program, Inhibit_Outputs routine.
2. Enter the following logic for *each* chassis that contains output modules:

If the chassis contains:	Then enter:
only output and communication modules	for the ControlNet Bridge module in the chassis, a rung from Step 2a. on page 15
both input and output modules	for each output module in the chassis, a rung from Step 2b. on page 15

- a. To inhibit the connections to all the modules in a chassis, enter this rung. For the CIP Object name parameter, select the ControlNet Bridge module in the chassis.

Inhibits the connection to the ControlNet Bridge module (1756-CNB) that is specified in the CIP Object name parameter of the GSV and SSV instructions, which inhibits the entire chassis:

- Gets the mode attribute of the module and stores it in the mode member of a tag with the same name as the module
- Sets bit 2 of the mode member, which indicates that the controller should inhibit its connection to the module
- Sends the new mode value to the mode attribute, which inhibits the connection



42095

- b. To inhibit an individual output module, enter this rung. For the CIP Object name parameter, select the name of the output module.

Inhibits the connection to the module that is specified in the CIP Object name parameter of the GSV and SSV instructions:

- Gets the mode attribute of the module and stores it in the mode member of a tag with the same name as the module
- Sets bit 2 of the mode member, which indicates that the controller should inhibit its connection to the module
- Sends the new mode value to the mode attribute, which inhibits the connection



42095

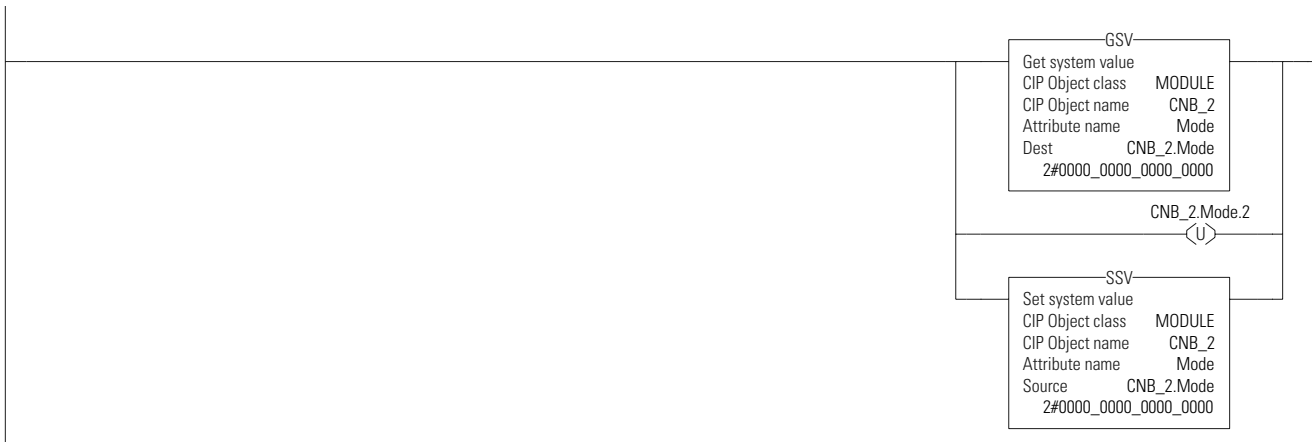
3. Copy the rungs from Step 2 and paste them into the Switch_Over program, Inhibit_Outputs routine.

4. Enter the following logic to uninhibit connections:
 - a. Copy the rungs from Step 2 and paste them into the Switch_Over program, Take_Over_Control routine.
 - b. In the Take_Over_Control routine, replace each OTE instruction with an OTU instruction.

The Take_Over_Control routine should now contain rungs similar to the following:

Uninhibits the connection to the ControlNet Bridge module (1756-CNB) that is specified in the CIP Object name parameter of the GSV and SSV instructions, which uninhibits the entire chassis:

- Gets the mode attribute of the module and stores it in the mode member of a tag with the same name as the module
- Clears bit 2 of the mode member, which indicates that the controller should *not* inhibit its connection to the module
- Sends the new mode value to the mode attribute, which uninhibits the connection



42098

Uninhibits the connection to the module that is specified in the CIP Object name parameter of the GSV and SSV instructions:

- Gets the mode attribute of the module and stores it in the mode member of a tag with the same name as the module
- Clears bit 2 of the mode member, which indicates that the controller should *not* inhibit its connection to the module
- Sends the new mode value to the mode attribute, which uninhibits the connection



42098

5. Open the Switch_Over program, Switch_Over_Logic routine.

6. Enter these rungs:

Gets the Status attribute of the CONTROLLERDEVICE object. If bits 12 and 13 are on, this controller is in a remote mode.



42096

When the controller is in Remote Run mode, uninhibits all the connections of this controller. Connections are typically uninhibited, however, in this application solution, another controller may have inhibited all the connections of this controller. To execute this type of CIP generic message, the keyswitch of this controller must be in the Remote position.



42096

7. In the This_Uninhibit MSG instruction, click  .

The Message Configuration dialog box opens.

8. On the Configuration tab, type or select the following parameters:

In this text box or drop-down list:	Type or select:
Message Type	CIP Generic
Service Code	4
Object Type	69
Object ID	1
Object Attribute	leave blank
Source	Inhibit_Reset
Num Of Elements	6
Destination	Inhibit_Reset_IO_Map_State

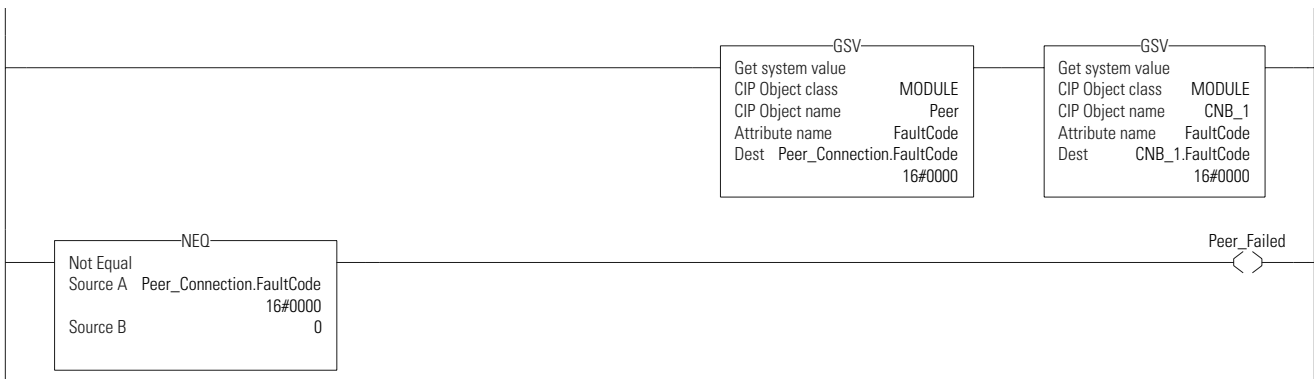
9. On the Communication tab, type or select the following parameters:

In this text box or check box:	Type or select:
Path	1, <slot number of this controller>
Cache Connections	Select <input checked="" type="checkbox"/> the check box.

10. Click **OK**.

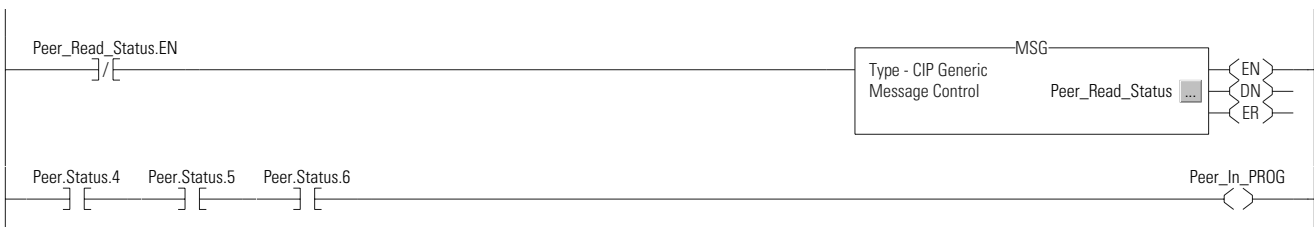
11. Enter these rungs:

Monitors the connections to the peer controller and the local ControlNet Bridge module. (The dummy produced and consumed tags establish the connection with the peer controller.) If the fault code for the connection to the peer controller is not equal to zero, the peer controller is not operational. The peer controller may have a non-recoverable fault, power to the chassis of the controller may have been interrupted, or a cable to the chassis may be broken.




42096

Continuously gets all the information contained in the CONTROLLERDEVICE object of the peer controller. If bits 4, 5, and 6 of the Status attribute are on, the peer controller is in Program mode.



42096

12. In the Peer_Read_Status MSG instruction, click  .

The Message Configuration dialog box opens.

13. On the Configuration tab, type or select the following parameters:

In this text box or drop-down list:	Type or select:
Message Type	CIP Generic
Service Code	1
Object Type	1
Object ID	1
Object Attribute	leave blank
Source	leave blank
Num Of Elements	0
Destination	Peer

14. On the Communication tab, specify the following parameters:

In this text box or check box:	Perform this action:
Path	A. Click Browse ... B. Select the peer controller.
Cache Connections	Select <input checked="" type="checkbox"/> the check box.

15. Click **OK**.

16. Enter this rung:

If the peer controller is in Program mode, inhibits all of the connections of that controller. This lets this controller take over ownership of the output modules.



42096

17. In the Peer_Inhibit MSG instruction, click .

The Message Configuration dialog box opens.

18. On the Configuration tab, type or select the following parameters:

In this text box or drop-down list:	Type or select:
Message Type	CIP Generic
Service Code	4
Object Type	69
Object ID	1
Object Attribute	leave blank
Source	Inhibit_Set
Num Of Elements	6
Destination	Inhibit_Set_IO_Map_State

19. On the Communication tab, specify the following parameters:

In this text box or check box:	Perform this action:
Path	A. Click Browse ... B. Select the peer controller.
Cache Connections	Select <input checked="" type="checkbox"/> the check box.

20. Click **OK**.

21. Enter these rungs:

If bits 4, 6, and 10 of the CONTROLLERDEVICE Status attribute for the peer controller are on, the peer controller is in Faulted mode and experiencing a recoverable major fault. The Controller Fault Handler of the peer controller will inhibit the connections of that controller to the output modules so this controller can take over control.

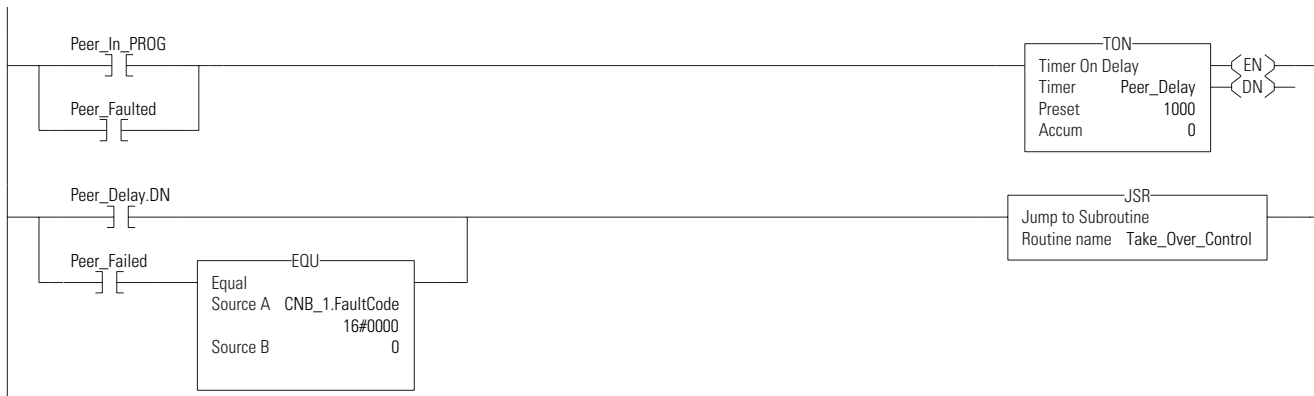


IMPORTANT

Because the time required to inhibit the connections of a controller varies with the size of the system, you may have to increase the TON preset value of the following rung. Refer to "Tune the System for Efficient Switch Over" on page 25.

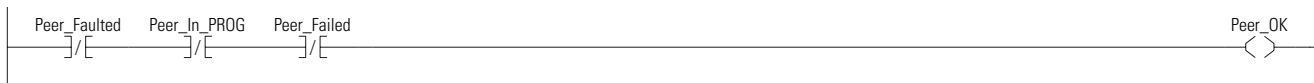
If the peer controller is in Program mode or Fault mode, starts a time delay while the connections of the peer controller are being inhibited. If the peer controller is not operational (its connection has failed) all of its connections are already broken, so no delay is required. This controller takes over control of the output modules when either of these conditions occur:

- The time delay is done.
- The connection to the peer controller is failed but the connection to the ControlNet Bridge module in the same chassis as this controller is good (i.e., no fault code for the CNB module).



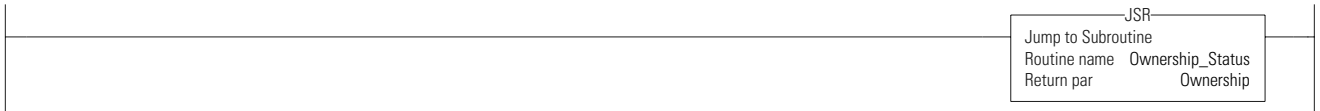
42096

If the peer controller is not in Fault mode or Program mode and the connection to the peer controller is not faulted, flags the peer controller as operational (OK)



42096

Checks for a conflict in ownership of each output module. If there is a conflict for each module, returns the value of 9999, which indicates that the peer controller owns (controls) each output module



42097

If the peer controller is operational and owns each output module, inhibits the connections from this controller to the output modules

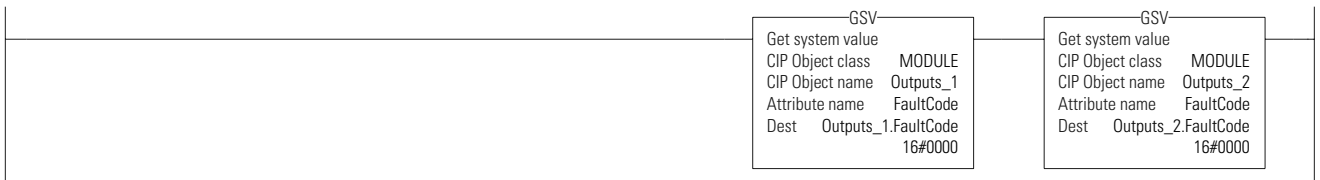


42098

22. Open the Switch_Over program, Ownership_Status routine.

23. Enter a GSV instruction, in series, for each output module.

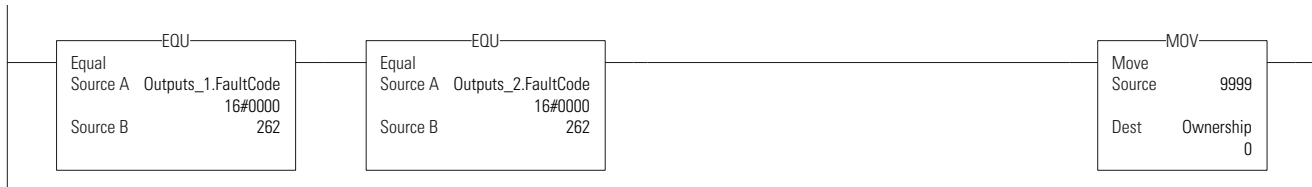
Gets the FaultCode attribute of the module that is specified in the CIP Object name parameter of the GSV instruction and stores it in the mode member of a tag with the same name as the module. There is one GSV instruction for each output module.



42099

24. Enter this rung and include an EQU instruction, in series, for each output module.

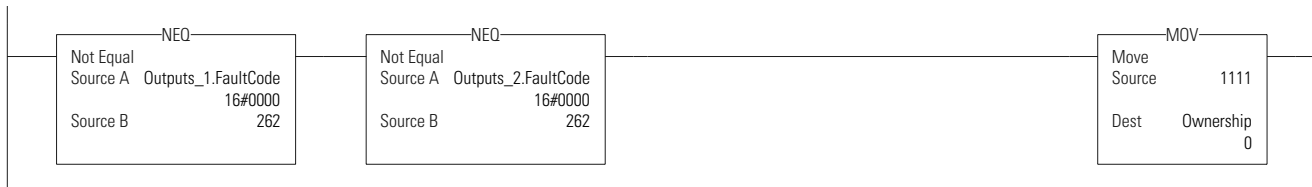
If the fault code for each output module equals 262 (16#0106), the peer controller owns the modules. (I.e., The peer controller has established a connection to each module.) Moves 9999 into the Ownership tag.



42099

25. Enter this rung and include an NEQ instruction, in series, for each output module.

If the fault code for each output module does *not* equal 262 (16#0106), the peer controller does *not* own the modules. (I.e., The peer controller has *not* established a connection to each module.) Moves 1111 into the Ownership tag.



42099

26. Enter this rung.

Returns the value of the Ownership tag to the main routine. (The value of the Ownership tag is still available to all routines in this program even without passing the parameter. The parameter is used to indicate which routine controls the value of the Ownership tag.)



42099

Develop the Project for the Second Controller

The second controller requires the same tags and logic for the switch over as the first controller. To program the second controller, complete one of the following options:

- Save the project of the first controller under a new name and use it for the second controller.
- From the project for the first controller, copy the following components and paste them into the project for the second controller:
 - User-defined data types
 - Fault_Program program
 - Switch_Over program
 - I/O configuration (except the first controller)
 - controller-scoped tags that you entered according to the “Create the following user-defined data type, which will be used for information about each module:” section of this application solution

Regardless of the option that you use to create the second project, make the following changes to the project:

- To the I/O configuration, add the first controller (which is now the peer of the second controller).
- For the following messages, type or select the Path to the correct controller:
 - This_Uninhibit
 - Peer_Read_Status
 - Peer_Inhibit

Additional Programming

As you develop and maintain the projects for both controllers, follow these additional guidelines:

- To synchronize the tag values of both controllers, use either of these methods:
 - produced/consumed tags
 - messages

Because programs execute asynchronous to data transfers, an additional exchange of data between controllers is typically required to ensure that the data is stable before the second controller uses it.

- To communicate with the controllers, any MMI applications or computers must use either separate communication paths (single-chassis configuration) or separate ControlNet node addresses (dual-chassis configuration). This also requires the MMI applications or computers to be able to perform one of the following:
 - while running, switch the controller with which it is communicating
 - communicate simultaneously with both controllers
- If you change the I/O configuration, tag values, or application code of a project, update the projects for both controllers.

Tune the System for Efficient Switch Over

After you develop the projects for both controllers, tune the system for the fastest switch over time possible by performing these actions:

- Adjust the Time Delay
- Adjust the Unscheduled Time

Adjust the Time Delay

If the peer controller faults or enters Program mode, the logic in the monitoring controller waits for one second before taking over control. This programmed time delay *reduces* switch over time, for the following reasons:

- If a controller attempts to establish an owner connection to an output module that is currently owned by another controller, the attempt to connect will fail.
- When the initial attempt to connect fails, the controller waits approximately three seconds before re-attempting to connect.
- If the previous connection is either broken or inhibited before another controller attempts to connect, the initial attempt will succeed, which avoids the three second waiting period.

Depending on the size of your system, a one second delay may *not* provide enough time to inhibit connections before the second controller attempts to take over control.

- If the switch over takes approximately four seconds or longer, you may be incurring the three second waiting period (one second delay plus three seconds waiting).
- To avoid the three second waiting period, increase the Peer_Delay.PRE value.

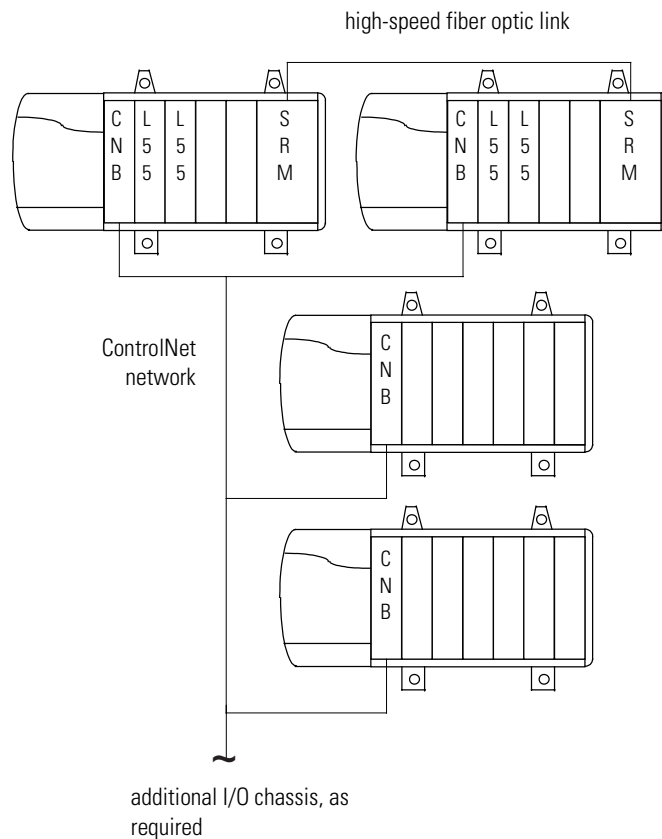
Adjust the Unscheduled Time

Since the controllers use unconnected, unscheduled messages to establish connections, increasing the amount of unscheduled time will increase the speed of the switch over. To improve switch over performance, follow these guidelines:

- Increase the Controller Overhead Time Slice.
 - The controller establishes and monitors connections during the system overhead time slice.
 - Be aware, however, that increasing this time decreases the time available for the continuous task.
- Reduce the scheduled bandwidth of the ControlNet network to 50 percent. This should provide ample unscheduled bandwidth to establish connections. To reduce the scheduled bandwidth:
 - Increase the network update time (NUT) of the network.
 - Increase the RPIs of the connections.
- Minimize the execution of MSG instructions.
 - A controller contains 10 unconnected buffers.
 - A controller uses the unconnected buffers to establish connections and send messages from MSG instructions.
 - By minimizing the use of MSG instructions, more buffers will be available to establish connections during a switch over.

Future ControlLogix Redundancy Solution

In the future, a hardware-based redundancy solution will be available for ControlLogix systems. The solution will provide controller redundancy (hot-backup) with a bump-less switch over for any program in the highest priority task. The solution is based on the ProcessLogix System Redundancy Module (1757-SRM) and uses two controller chassis, as depicted below.



ControlLogix chassis, each with identical sets of:

- one or more Logix5555 controllers
- 1756-CNB modules
- a single 1757-SRM module

ControlLogix chassis with:

- 1756-CNB module
- 1756 I/O modules
- additional communication modules, as required (e.g., 1756-ENET module)

42197

The system will provide redundancy for one or more controllers in a primary chassis (i.e., chassis that is currently controlling the system):

- No additional programming is required.
- The 1757-SRM modules and corresponding high-speed fiber optic link synchronize controllers in the secondary chassis with the status of the controllers in the primary chassis.
- All data and program changes automatically transfer from the primary controllers to the secondary controllers.
- When a failure occurs in any of the components in the primary chassis, control switches to corresponding controllers in a secondary chassis:
 - The secondary controllers can take over control within approximately 250 ms.
 - During the switch over, outputs remain in their appropriate state. (I.e., The switch over is bump-less.)

Hardware Requirements

The future ControlLogix redundancy solution will require the following hardware:

- new Logix5555™ controllers with enough memory to store two copies of all data (may require expansion memory cards)
- 1756-CNB modules, series D, for each chassis in the system
- 1757-SRM module for each controller chassis
- 1757-SCR1, -SCR3, or -SCR10 fiber optic cable to link controller chassis (part numbers correspond to 1, 3, or 10 meters, respectively)
- two redundant chassis for controllers, each with enough slots for the following:
 - primary or secondary controllers
 - 1756-CNB modules
 - 1757-SRM module, which occupies two slots
- appropriate versions of Logix5555 controller firmware and RSLogix 5000 software

System Configuration

To configure the future ControlLogix redundancy solution you will perform these actions:

- In the controller chassis, only place the following modules:
 - Logix5555 controllers
 - 1756-CNB modules
 - 1757-SRM modules
- Place the following modules in remote chassis:
 - I/O
 - 1756-CNB
 - additional communication modules, if required
- For each 1756-CNB module in the primary chassis, allocate two consecutive ControlNet addresses.
 - The matching 1756-CNB module in the secondary chassis will automatically use the primary 1756-CNB module's address plus one.
 - Do not configure any other module for either of the addresses allocated for the primary 1756-CNB module. For example, if the address of the 1756-CNB module in the primary chassis is one, no other module in the system can use addresses one or two.

Software Switching Versus Hardware Redundancy

The following table outlines the differences between software switching and the future ControllLogix hardware redundancy solution.

Attribute:	Software Switching:	Hardware Redundancy:
special hardware	<ul style="list-style-type: none"> • duplicate controller chassis • may require additional network to pass data between the primary and secondary controllers 	<ul style="list-style-type: none"> • duplicate controller chassis • Logix5555 controllers • enough extra controller RAM for two copies of all data (usually requires larger memory cards) • 1757-SRM modules and fiber optic cable
special programming	Yes, to perform these actions: <ul style="list-style-type: none"> • synchronize data • detect failures • switch control of outputs 	No, hardware detects failure and switches control automatically
project maintenance	two projects to maintain, each requiring a manual download to the appropriate controller	one project, automatically cross-loaded from the primary controller to the secondary controller
effect on program scan time	<ul style="list-style-type: none"> • Extra logic increases scan time. • Both controllers monitor inputs simultaneously. • Both controllers scan logic independently (no synchronization). 	<ul style="list-style-type: none"> • Primary controller waits until end of the scan to send data to second controller. • Secondary controller does not scan logic until it takes over control from the primary controller.
online program changes	Yes, each project requires a manual update.	Yes, edits are automatically sent to the secondary controller.
online data changes	Yes, each project requires either a manual update or logic to copy changes to the secondary controller.	Yes, data changes are automatically sent to the secondary controller.
Forcing	Yes, place forces in one controller at a time. To ensure forces remain, place forces in both controllers.	Forces in the primary controller are automatically sent to the secondary controller.
data synchronization	<ul style="list-style-type: none"> • Each project requires logic to copy information from the primary controller to the secondary controller. • Other systems may need to perform dual writes to update both controllers at the same time. 	After each program scan, changes of data in the primary controller are automatically sent to the secondary controller.
controller failure detection time	varies with each application	essentially immediate because the hardware detects failures
switch over time		≈ 250 ms
switch over if controller: <ul style="list-style-type: none"> • faults • fails • losses communications 	Yes, if programmed	automatically
switch over if controller enters Program mode		No
1756 I/O modules in controller chassis	Yes, but not easily	No

Attribute:	Software Switching:	Hardware Redundancy:
1756-M02AE motion modules in controller chassis	No	No
1756-ENET module in controller chassis	Yes, but external device must handle dual addresses.	No, must reside in a chassis of remote I/O off of a ControlNet network
1756-DHRIO module in controller chassis	<ul style="list-style-type: none"> • no RIO • DH+, if external device can accommodate dual addresses 	
1756-DNB module in controller chassis	Yes, but not easily	

Reach us now at www.rockwellautomation.com

Wherever you need us, Rockwell Automation brings together leading brands in industrial automation including Allen-Bradley controls, Reliance Electric power transmission products, Dodge mechanical power transmission components, and Rockwell Software. Rockwell Automation's unique, flexible approach to helping customers achieve a competitive advantage is supported by thousands of authorized partners, distributors and system integrators around the world.

Americas Headquarters, 1201 South Second Street, Milwaukee, WI 53204, USA, Tel: (1) 414 382-2000, Fax: (1) 414 382-4444
European Headquarters SA/NV, avenue Herrmann Debroux, 46, 1160 Brussels, Belgium, Tel: (32) 2 663 06 00, Fax: (32) 2 663 06 40
Asia Pacific Headquarters, 27/F Citicorp Centre, 18 Whitfield Road, Causeway Bay, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846



**Rockwell
Automation**