Logix 5000 Controllers Tasks, Programs, and Routines

1756 ControlLogix, 1756 GuardLogix, 1769 CompactLogix, 1769 Compact GuardLogix, 1789 SoftLogix, 5069 CompactLogix, 5069 Compact GuardLogix, Studio 5000 Logix Emulate

Rockwell Automation Publication 1756-PM005K-EN-P - November 2022
Supercedes Publication 1756-PM005J-EN-P - March 2022

Programming Manual

Original Instructions

Important User Information

Read this document and the documents listed in the additional resources section about installation, configuration, and operation of this equipment before you install, configure, operate, or maintain this product. Users are required to familiarize themselves with installation and wiring instructions in addition to requirements of all applicable codes, laws, and standards.

Activities including installation, adjustments, putting into service, use, assembly, disassembly, and maintenance are required to be carried out by suitably trained personnel in accordance with applicable code of practice.

If this equipment is used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

In no event will Rockwell Automation, Inc. be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment.

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular installation, Rockwell Automation, Inc. cannot assume responsibility or liability for actual use based on the examples and diagrams.

No patent liability is assumed by Rockwell Automation, Inc. with respect to use of information, circuits, equipment, or software described in this manual.

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

Throughout this manual, when necessary, we use notes to make you aware of safety considerations.

WARNING: Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss.

ATTENTION: Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you identify a hazard, avoid a hazard, and recognize the consequence.

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

Labels may also be on or inside the equipment to provide specific precautions.

SHOCK HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present.

BURN HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.

ARC FLASH HAZARD: Labels may be on or inside the equipment, for example, a motor control center, to alert people to potential Arc Flash. Arc Flash will cause severe injury or death. Wear proper Personal Protective Equipment (PPE). Follow ALL Regulatory requirements for safe work practices and for Personal Protective Equipment (PPE).

Rockwell Automation recognizes that some of the terms that are currently used in our industry and in this publication are not in alignment with the movement toward inclusive language in technology. We are proactively collaborating with industry peers to find alternatives to such terms and making changes to our products and content. Please excuse the use of such terms in our content while we implement these changes.

Summary of changes

This manual includes new and updated information. Use these reference tables to locate changed information.

Grammatical and editorial style changes are not included in this summary.

Global changes

This table identifies changes that apply to all information about a subject in the manual and the reason for the change. For example, the addition of new supported hardware, a software design change, or additional reference material would result in changes to all the topics that deal with that subject.

Change	Topic
Added inclusive language notice	Preface on page 7

New or enhanced features

None in this release.

Table of Contents

Index

This manual is one of a set of related manuals that show common procedures for programming and operating Logix 5000 controllers.

For a complete list of common procedures manuals, refer to the [Logix 5000 Controllers Common Procedures Programming Manual](), publication [1756-PM001]().

The term Logix 5000 controller refers to any controller based on the Logix 5000 operating system.

Rockwell Automation recognizes that some of the terms that are currently used in our industry and in this publication are not in alignment with the movement toward inclusive language in technology. We are proactively collaborating with industry peers to find alternatives to such terms and making changes to our products and content. Please excuse the use of such terms in our content while we implement these changes.

Studio 5000 environment

The Studio 5000 Automation Engineering & Design Environment® combines engineering and design elements into a common environment. The first element is the Studio 5000 Logix Designer® application. The Logix Designer application is the rebranding of RSLogix 5000® software and will continue to be the product to program Logix 5000™ controllers for discrete, process, batch, motion, safety, and drive-based solutions.

The Studio 5000® environment is the foundation for the future of Rockwell Automation® engineering design tools and capabilities. The Studio 5000 environment is the one place for design engineers to develop all elements of their control system.

Additional resources

These documents contain additional information concerning related Rockwell Automation products.

Resource	Description
Industrial Automation Wiring and Grounding Guidelines, publication 1770-4.1	Provides general guidelines for installing a Rockwell Automation industrial system.
Product Certifications webpage, available at http://ab.rockwellautomation.com	Provides declarations of conformity, certificates, and other certification details.

View or download publications at http://www.rockwellautomation.com/literature. To order paper copies of technical documentation, contact the local Rockwell Automation distributor or sales representative.

Legal Notices

Rockwell Automation publishes legal notices, such as privacy policies, license agreements, trademark disclosures, and other terms and conditions on the Legal Notices page of the Rockwell Automation website.

End User License Agreement (EULA)

You can view the Rockwell Automation End-User License Agreement ("EULA") by opening the License.rtf file located in your product's install folder on your hard drive.

Open Source Licenses

The software included in this product contains copyrighted software that is licensed under one or more open source licenses. Copies of those licenses are included with the software. Corresponding Source code for open source packages included in this product are located at their respective web site(s).

Alternately, obtain complete Corresponding Source code by contacting Rockwell Automation via the Contact form on the Rockwell Automation website:
http://www.rockwellautomation.com/global/about-us/contact/contact.page
Please include "Open Source" as part of the request text.

A full list of all open source software used in this product and their corresponding licenses can be found in the OPENSOURCE folder. The default installed location of these licenses is `C:\Program Files (x86)\Common Files\Rockwell\Help\<Product Name>\Release Notes\OPENSOURCE\index.htm`.

Manage Tasks

Introduction

The default project provides one task for all your logic. Although this is sufficient for many applications, some situations may require multiple tasks.

Select Controller Tasks

A Logix 5000 controller supports multiple tasks to schedule and prioritize the running of your programs based on specific criteria. This balances the processing time of the controller.

- The controller runs only one task at one time.
- Another task can interrupt a task that is running and take control.
- In any given task, only one program runs at one time.

A Logix 5000 controller supports the following types of tasks.

If you want to run a section of your logic	Then use this type of task	Description
All the time	Continuous Task	The continuous task runs in the background. Any CPU time not allocated to other operations (such as motion, communication, and periodic or event tasks) is used to run the programs within the continuous task. • The continuous task runs all the time. When the continuous task completes a full scan, it restarts immediately. • A project does not require a continuous task. If used, there can be only one continuous task.
• At a constant period (for example, every 100 ms) • Multiple times within the scan of your other logic	Periodic Task	A periodic task performs a function at a specific period. When the time for the periodic task expires, the periodic task: • Interrupts any lower priority tasks. • Runs one time. • Returns control to where the previous task left off. You can configure the time period from 0.1 ms...2000 s. The default is 10 ms.
Immediately when an event occurs	Event Task	An event task performs a function only when a specific event (trigger) occurs. When the trigger for the event task occurs, the event task: • Interrupts any lower priority tasks. • Runs one time. • Returns control to where the previous task left off. The trigger can be a: • Change of a digital input. • New sample of analog data. • Certain motion operations. • Consumed tag. • EVENT instruction. **Important:** Some Logix 5000 controllers do not support all triggers.

The following table lists example situations for the tasks.

For this example situation	Use this type of task
Fill a tank to its maximum level and then open a drain valve.	Continuous task
Collect and process system parameters and send them to a display.	Continuous task
Complete step 3 in a control sequence—reposition the bin diverter.	Continuous task
Your system must check the position of a field arm each 0.1 s and calculate the average rate of change in its position. This is used to determine braking pressure.	Periodic task
Read the thickness of a paper roll every 20 ms.	Periodic task
A packaging line glues boxes closed. When a box arrives at the gluing position, the controller must immediately run the gluing routine.	Event task
In a high-speed assembly operation, an optical sensor detects a certain type of reject. When the sensor detects a reject, the machine must immediately divert the reject.	Event task
In an engine test stand, you want to capture and archive each analog data immediately after each sample of data.	Event task
Immediately after receiving new production data, load the data into the station.	Event task
In a line that packages candy bars, you have to make sure that the perforation occurs in the correct location on each bar. Each time the registration sensor detects the registration mark, check the accuracy of an axis and perform any required adjustment.	Event task
A gluing station must adjust the amount of glue it applies to compensate for changes in the speed of the axis. After the motion planner runs, check the command speed of the axis and vary the amount of glue, if needed.	Event task
In a production line, if any of the programs detect an unsafe condition the entire line must shut down. The shutdown procedure is the same regardless of the unsafe condition.	Event task

The number of tasks supported depends on the controller.

This controller	Supports this number of tasks	Notes
ControlLogix 1756-L71 1756-L72 1756-L73 1756-L74 1756-L75	32	Only one task can be continuous.
GuardLogix 1756-L71S 1756-L72S 1756-L73S	32	
SoftLogix5800 1756-L7SP	32	

This controller	Supports this number of tasks	Notes
CompactLogix		
1769-L2x	3	
1769-L31	4	
1769-L32x	6	
1769-L35x	8	
1768-L43	16	
1768-L45		
1769-L16x	32	
1769-L18x		
1769-L19x		
1769-L24x		
1769-L27x		
1769-L30x		
1769-L33x		
1769-L36x		
1769-L37x		

Use Caution in the Number of Tasks That You Use

Typically, each task takes controller time away from the other tasks. If you have too many tasks, then:

- The continuous task may take too long to complete.
- Another task may overlap. If a task is interrupted too frequently or too long, it may not finish running before it is triggered again.

Prioritize Periodic and Event tasks

Although a project can contain multiple tasks, the controller runs only one task at a time. If a periodic or event task is triggered while another task is running, the priority of each task indicates what the controller should do.

The number of priority levels depends on the controller.

This Logix 5000 controller	Has this many priority levels
CompactLogix	15
ControlLogix	15
DriveLogix	15
FlexLogix	15
SoftLogix5800	3

To assign a priority to a task, use the following guidelines.

If you want	Then	Notes
This task to interrupt another task	Assign a priority number that is less than (higher priority) the priority number of the other task.	• A higher priority task interrupts all lower priority tasks. • A higher priority task can interrupt a lower priority task multiple times.
Another task to interrupt this task	Assign a priority number that is greater than (lower priority) the priority number of the other task.	
This task to share controller time with another task	Assign the same priority number to both tasks.	The controller switches back and forth between each task and runs each task for 1 ms.

Additional Considerations

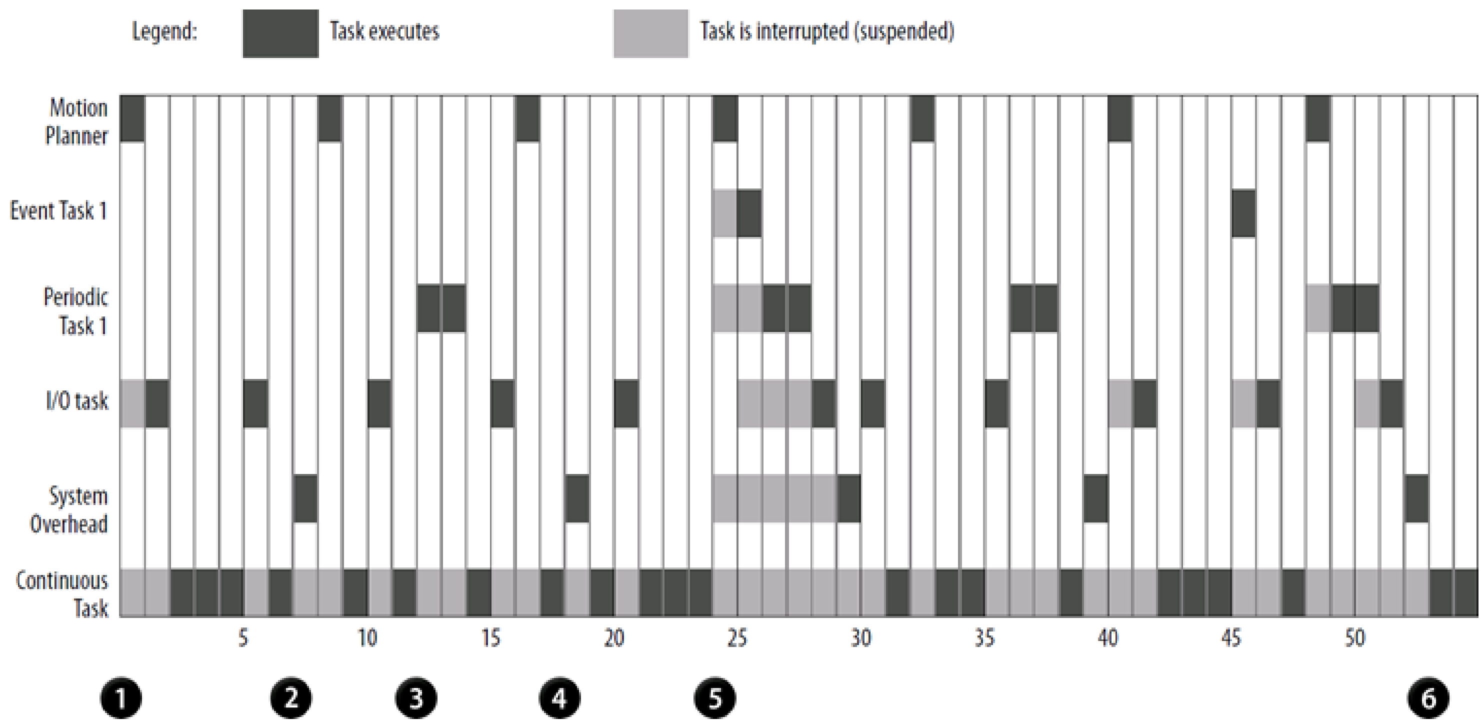
As you estimate the execution interrupts for a task, consider the following.

<table>
<tr><th>Consideration</th><th colspan="2">Description</th></tr>
<tr><td>Motion planner</td><td colspan="2">The motion planner interrupts all user tasks, regardless of their priority.
• The number of axes and coarse update period for the motion group affect how long and how often the motion planner runs.
• If the motion planner is running when a task is triggered, the task waits until the motion planner is done.
• If the coarse update period occurs while a task is running, the task pauses to let the motion planner run.</td></tr>
<tr><td rowspan="5">I/O task</td><td colspan="2">Tip: CompactLogix controllers do not have I/O tasks.
FlexLogix, and DriveLogix controllers use a dedicated periodic task to process I/O data. This I/O task:
• Does not show up in the Tasks folder of the controller.
• Does not count toward the task limits for the controller.
• Operates at priority 6.
• Runs at the fastest RPI you have scheduled for the system.
• Runs for as long as it takes to scan the configured I/O modules.
As you assign priorities to your tasks, consider the I/O task.</td></tr>
<tr><td>If you want a task to</td><td>Then assign one of these priorities</td></tr>
<tr><td>Interrupt or delay I/O processing</td><td>1…5</td></tr>
<tr><td>Share controller time with I/O processing</td><td>6</td></tr>
<tr><td>Let I/O processing interrupt or delay the task</td><td>7…15</td></tr>
<tr><td>System overhead</td><td colspan="2">System overhead is the time that the controller spends on unscheduled communication.
• Unscheduled communication is any communication that you do not configure through the I/O configuration folder of the project, such as Message (MSG) instructions and communication with HMIs or workstations.
• System overhead interrupts only the continuous task.
• The system overhead time slice specifies the percentage of time (excluding the time for periodic or event tasks) that the controller devotes to unscheduled communication.
• The controller performs unscheduled communication for up to 1 ms at a time and then resumes the continuous task.</td></tr>
<tr><td>Continuous task</td><td colspan="2">You do not assign a priority to the continuous task. It always runs at the lowest priority. All other tasks interrupt the continuous task.</td></tr>
</table>

Example

The following example shows the execution of a project with three user tasks.

Task	**Priority**	**Period**	**Execution time**	**Duration**
Motion planner	N/A	8 ms (base update rate)	1 ms	1 ms
Event task 1	1	N/A	1 ms	1…2 ms
Periodic task 1	2	12 ms	2 ms	2…4 ms
I/O task—n/a to CompactLogix, ControlLogix and SoftLogix controllers. See *Additional Considerations*.	7	5 ms (fastest RPI)	1 ms	1…5 ms
System overhead	N/A	Time slice = 20%	1 ms	1…6 ms
Continuous task	N/A	N/A	20 ms	48 ms



Description	
1	Initially, the controller runs the motion planner and the I/O task (if one exists).
2	After running the continuous task for 4 ms, the controller triggers the system overhead.
3	The period for periodic task 1 expires (12 ms), so the task interrupts the continuous task.
4	After running the continuous task again for 4 ms, the controller triggers the system overhead.
5	The trigger occurs for event task 1. Event task 1 waits until the motion planner is done. Lower priority tasks are delayed.
6	The continuous task automatically restarts.

The Studio 5000 environment includes a task monitor tool on the distribution CD. You can use this tool to analyze how tasks are running.

See also

Additional Considerations on page 12

Leave Enough Time for Unscheduled Communication

Unscheduled communication occurs only when a periodic or event task is not running. If you use multiple tasks, make sure that the scan times and execution intervals leave enough time for unscheduled communication. Use the following methods to plan enough unscheduled communication time.

1. Verify that the execution time of a highest priority task is significantly less than its specified period.

2. Verify that the total execution time of all your tasks is significantly less than the period of the lowest priority tasks.

For example, the following is true in this configuration.

Task	Priority	Execution Time	Period Specified
1	Higher	20 ms	80 ms
2	Lower	30 ms	100 ms
	Total execution time:	50 ms	

- The execution time of the highest priority task (Task 1) is significantly less than its specified period (20 ms is less than 80 ms).
- The total execution time of all tasks is significantly less than the specified period of the lowest priority task (50 ms is less than 100 ms).

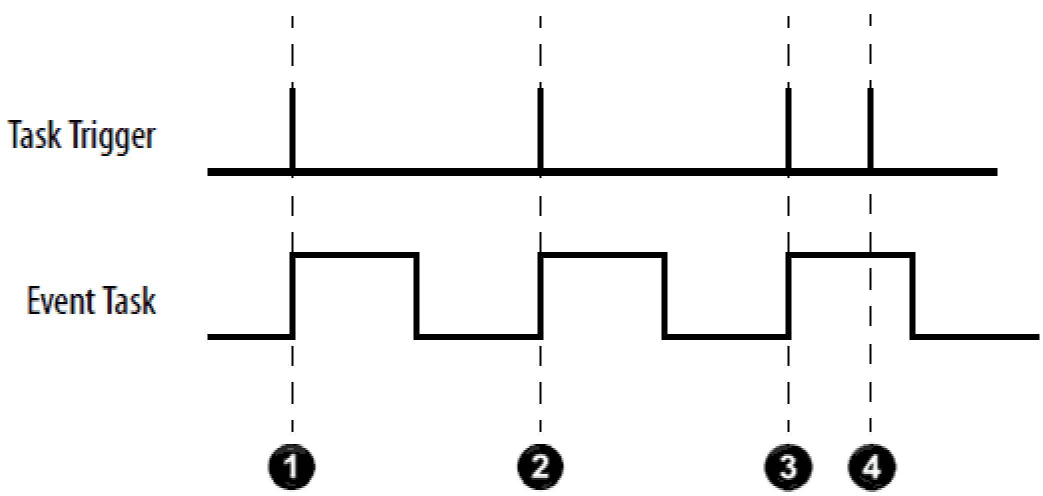
The following guidelines generally leave enough time for unscheduled communication.

- Adjust the period of the tasks as needed to get the best balance between running your logic and servicing unscheduled communication.
- If your project has a continuous task, unscheduled communication occurs as a percentage of controller time (excluding the time for periodic or event tasks).

Avoid Overlaps

An overlap is a condition where a task (periodic or event) is triggered while the task is still running from the previous trigger.

IMPORTANT If an overlap occurs, the controller disregards the trigger that caused the overlap. In other words, you might miss an important execution of the task.



Description	
1	Task trigger occurs. Task runs.
2	Task trigger occurs. Task runs.
3	Task trigger occurs. Task runs.

	Description
4	Overlap occurs. Task is triggered while it is still running. The trigger does not restart the task. The trigger is ignored.

Each task requires enough time to finish before it is triggered again. Make sure that the scan time of the task is significantly less than the rate at which the trigger occurs. If an overlap occurs, reduce the frequency at which you trigger the task.

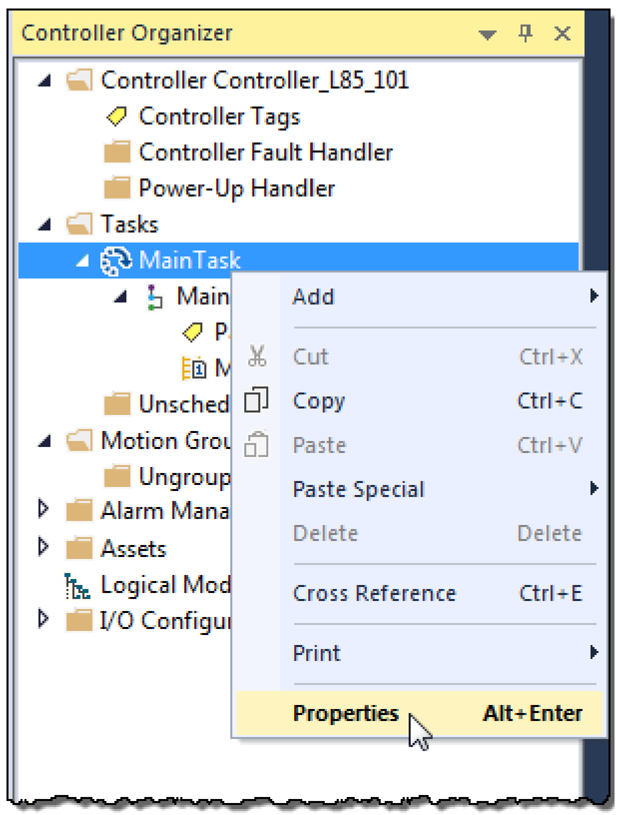
If the type of task is	Then
Periodic	Increase the period of the task.
Event	Adjust the configuration of your system to trigger the task less frequently.

Manually Check for Overlaps

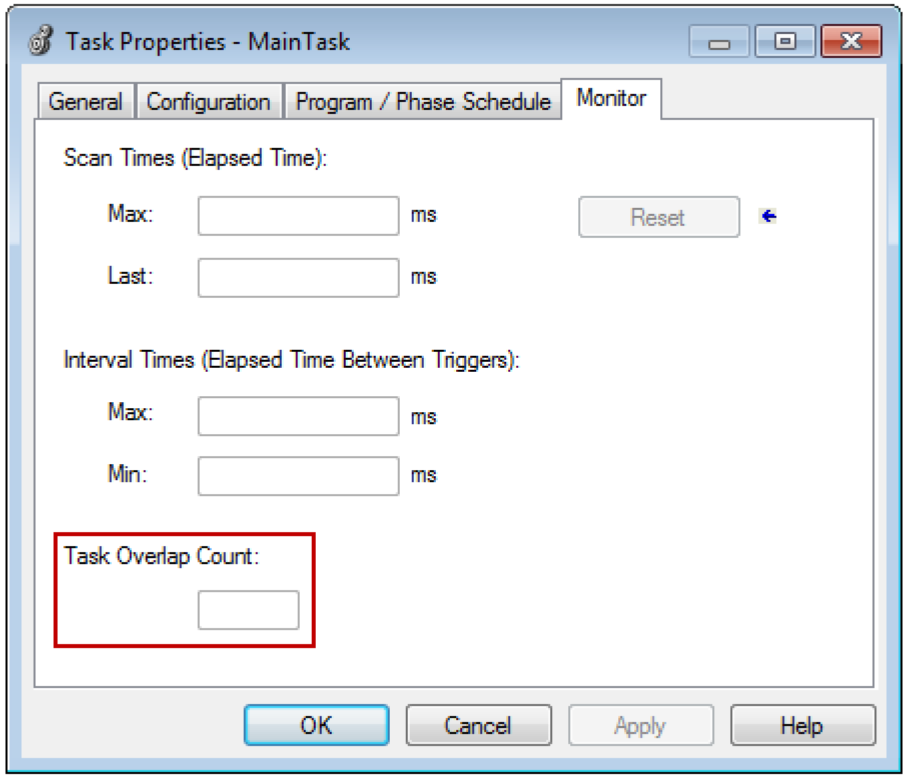
Follow these steps to manually see if overlaps are occurring for a task.

To manually check for overlaps

1. In the Controller Organizer, right-click **MainTask** and choose **Properties**.



2. On the **Task Properties** dialog box, select the **Monitor** tab.



The **Task Overlap Count** shows the number of overlaps since the counter was last reset.

3. Select **OK**.

See also

Manually Configure Output Processing on page 19

Manually Inhibit or Uninhibit a Task on page 22

Programmatically Check for Overlaps

When an overlap occurs, the controller:

- Logs a minor fault to the FAULTLOG object.
- Stores overlap information in the Task object for the task.

To write logic to check for an overlap, use a Get System Value (GSV) instruction to monitor either of these objects.

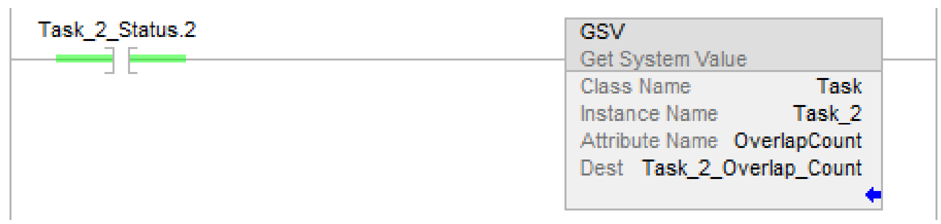
If you want to	Then access the object and attribute				
	Object	**Attribute**	**Data Type**	**Description**	
Determine if an overlap occurred for any task	FaultLog	MinorFaultBits	DINT	Individual bits that indicate a minor fault:	
				To determine if	**Examine this bit**
				An instruction produced a minor fault.	4
				An overlap occurred for a task.	6
				The serial port produced a minor fault.	9
				The battery/ESM is not present or needs replacement.(1)	10

If you want to	Then access the object and attribute				
	Object	Attribute	Data Type	Description	
Determine if an overlap occurred for a specific task	Task	Status	DINT	Status information about the task. Once the controller sets one of these bits, you must manually clear the bit.	
				To determine if	**Examine this bit**
				An EVENT instruction triggered the task (event task only).	0
				A timeout triggered the task (event task only).	1
				An overlap occurred for this task.	2
Determine the number of times that an overlap occurred.	Task	OverlapCount	DINT	Valid for an event or a periodic task.	
				To clear the count, set the attribute to 0.	

1. Battery for 1756-L6X, 1769-L2X, and 1769-L3X controllers. ESM for 1756-L7X and CompactLogix 5370 series controllers.

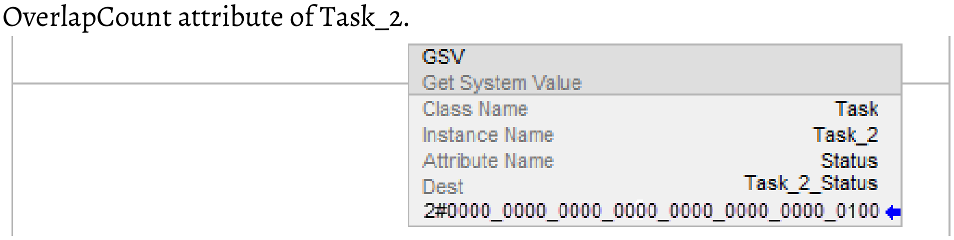
Example

1. The GSV instruction sets Task_2_Status = Status attribute for Task_2 (DINT value).



2. If Task_2_Status.2 = 1, then an overlap occurred, so get the count of overlaps:

   The GSV instruction sets Task_2_Overlap_Count (DINT tag) = OverlapCount attribute of Task_2.



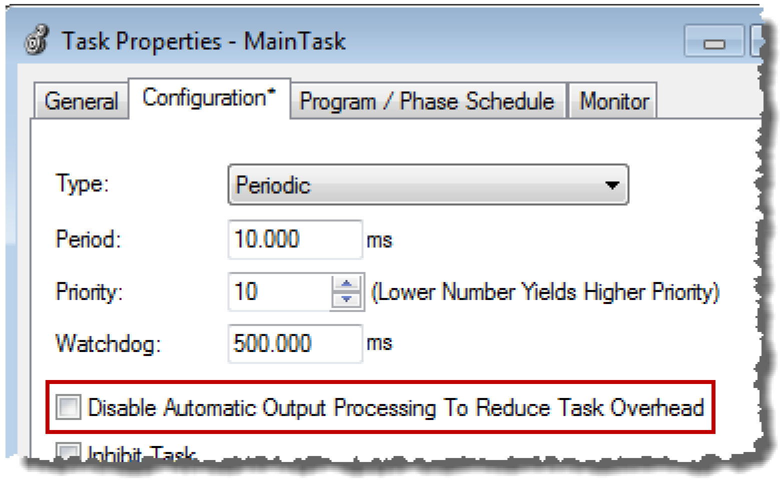
3. If Condition_1 = 1, then clear the bits of the Status attribute for Task_2:

   The SSV instruction sets the Status attribute of Task_2 = Zero. Zero is a DINT tag with a value of 0.

Configure Output Processing for a Task

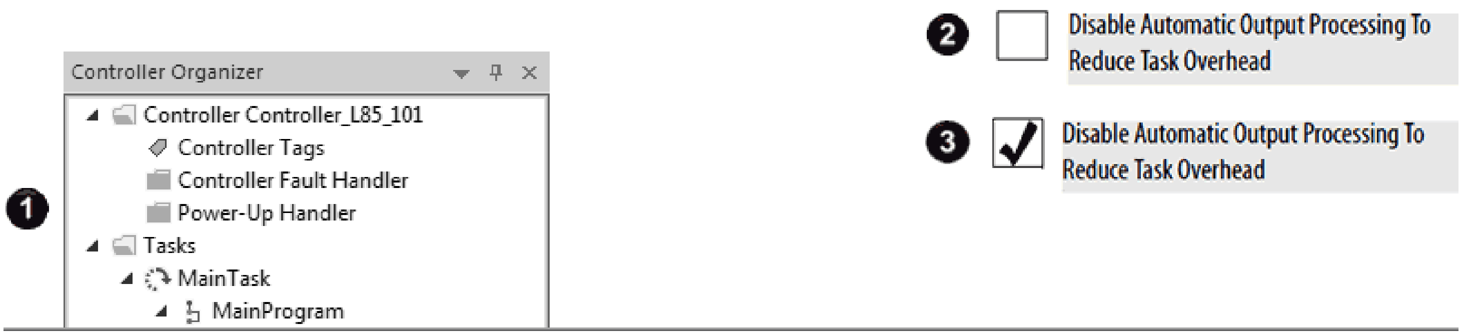
At the end of a task, the controller performs overhead operations (output processing) for the I/O modules in your system. Although these operations are not the same as updating the modules, the output processing may affect the update of the I/O modules in your system.

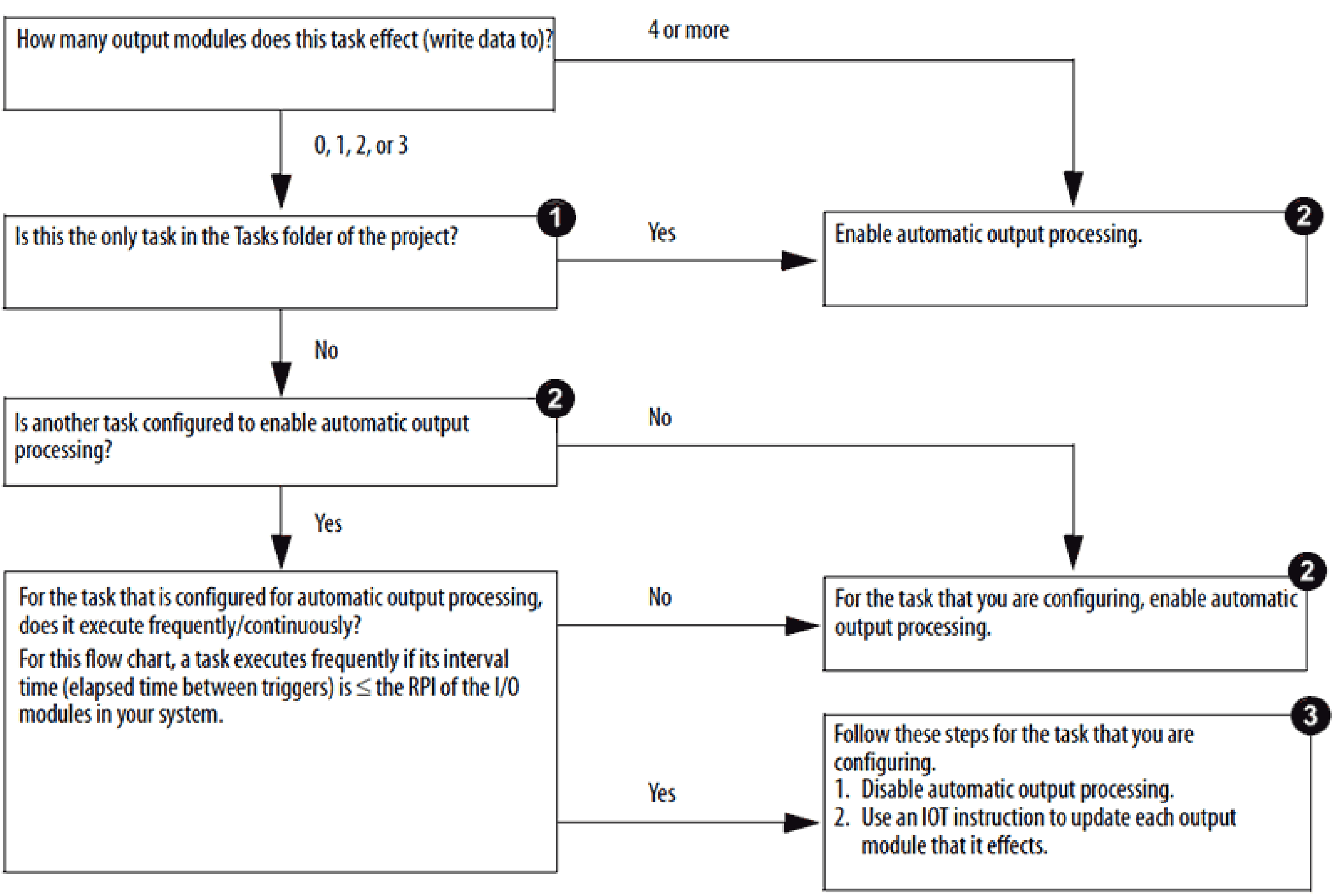
As an option, you can turn off this output processing for a specific task, which reduces the elapsed time of that task.



Select **Disable Automatic Output Processing To Reduce Task Overhead** to disable the processing of outputs at the end of the task.

Choose how to configure output processing for a task.

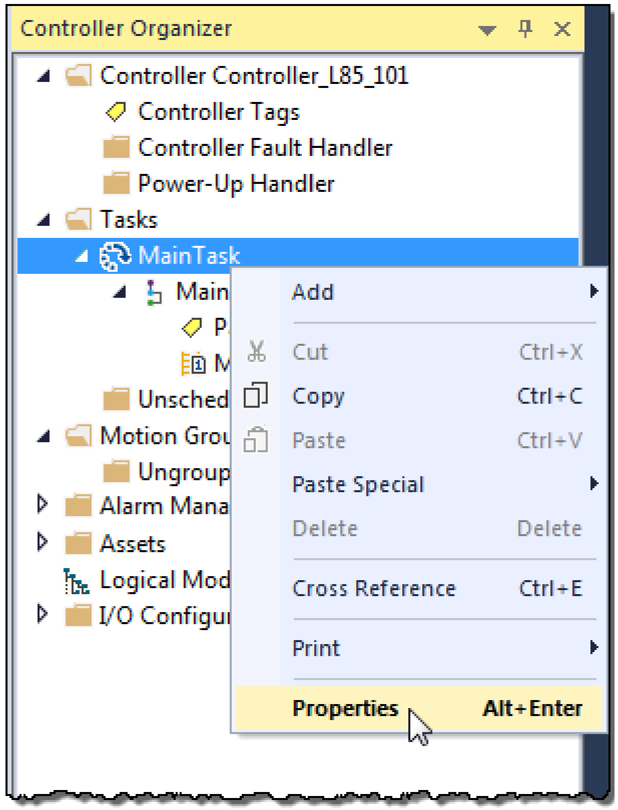




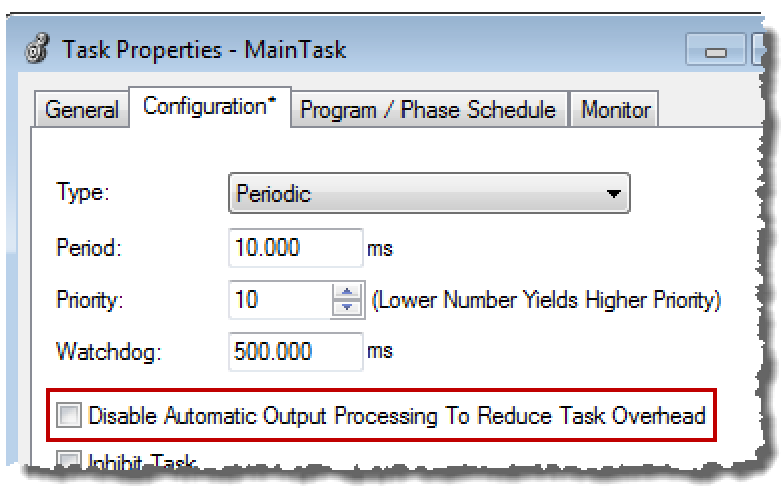
Manually Configure Output Processing

Follow these steps to manually configure output processing.

1. In the Controller Organizer, right-click **MainTask** and choose **Properties**.



2. On the **Task Properties** dialog box, click the **Configuration** tab.



3. Configure output processing for the task.

If you want to	Then
Enable the processing of outputs at the end of the task	Clear **Disable Automatic Output Processing To Reduce Task Overhead** (default).
Disable the processing of outputs at the end of the task	Check **Disable Automatic Output Processing To Reduce Task Overhead**.

4. Click **OK**.

Programmatically Configure Output Processing

To write logic to configure output processing for a task, use a Set System Value (SSV) instruction. Access the attribute of the Task object for the task.

If You Want to	Access This Attribute	Data Type	Instruction	Description	
				To	**Set the attribute to**
Enable or disable the processing of outputs at the end of a task	DisableUpdateOutputs	DINT	GSV SSV	Enable the processing of outputs at the end of the task	0
				Disable the processing of outputs at the end of the task	1 (or any non-zero value)

Example

If Condition_1 = 0 then let Task_2 process outputs when it is done.

1. The ONS instruction limits the true run of the SSV instruction to one scan.
2. The SSV instruction sets the DisableUpdateOutputs attribute of Task_2 = 0. This lets the task automatically process outputs when it finishes its run.

If Condition_1 = 1 then do not let Task_2 process outputs when it is done.

1. The ONS instruction limits the true run of the SSV instruction to one scan.
2. The SSV instruction sets the DisableUpdateOutputs attribute of Task_2 = 1. This prevents the task from automatically processing outputs when it finishes its run.

Inhibit a Task

By default, each task runs based on its trigger (event, periodic, or continuous). As an option, you can prevent a task from running when its trigger occurs (that is, inhibit the task). This is useful when you test, diagnose, or start up your project.

If You Want to	Then
Let the task run when its trigger occurs	Uninhibit the task (default).
Prevent the task from running when its trigger occurs	Inhibit the task.

EXAMPLE During the commissioning of a system that uses several tasks, you can first test each task individually.

- Inhibit all the tasks except one, and then test that task.
- Once the task meets your requirements, inhibit it and uninhibit a different task.

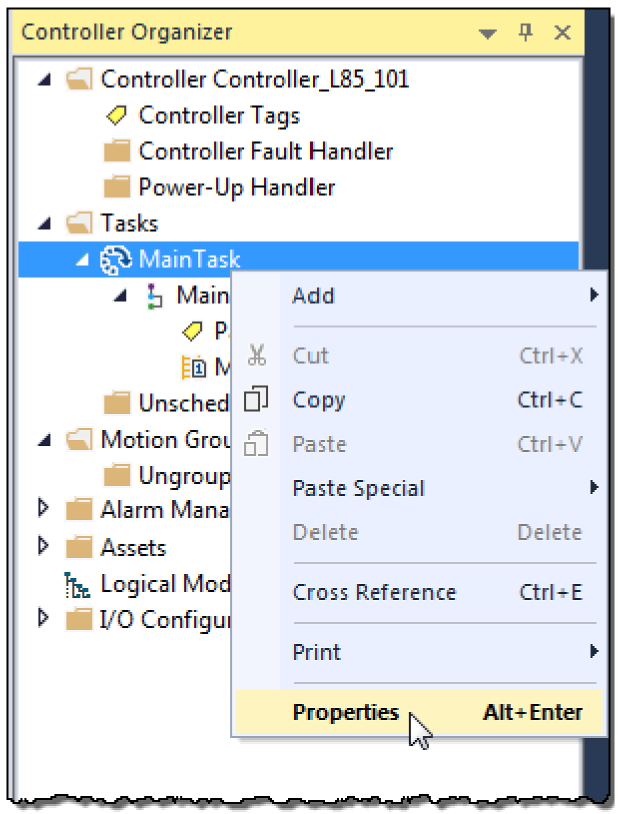
Continue this process until you have tested all your tasks.

If a task is inhibited, the controller still prescans the task when the controller transitions from Program to Run or Test mode.

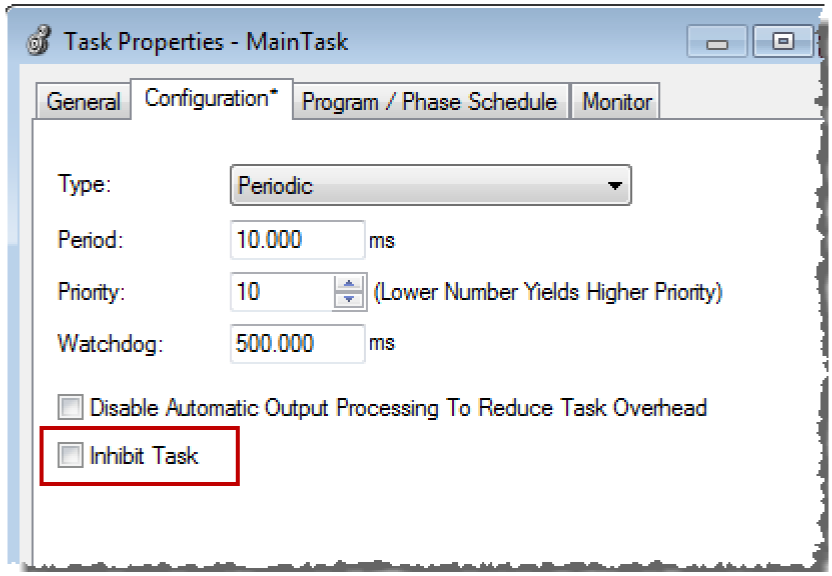
Manually Inhibit or Uninhibit a Task

Follow these steps to manually inhibit or uninhibit the running of a task.

1. In the Controller Organizer, right-click **MainTask** and choose **Properties**.



2. On the **Task Properties** dialog box, click the **Configuration** tab.



3. Do one of these steps to inhibit or uninhibit the task.

If You Want to	Then
Let the task run when its trigger occurs	Clear **Inhibit Task** (default).
Prevent the task from running when its trigger occurs	Check **Inhibit Task**.

4. Click **OK**.

Programmatically Inhibit or Uninhibit a Task

To write logic to inhibit or uninhibit a task, use a Set System Value (SSV) instruction to access the attribute of the Task object for the task.

<table>
<tr><th>Attribute</th><th>Data Type</th><th>Instruction</th><th colspan="2">Description</th></tr>
<tr><td rowspan="4">InhibitTask</td><td rowspan="4">DINT</td><td rowspan="4">GSV
SSV</td><td colspan="2">Prevents the task from running.</td></tr>
<tr><td>To</td><td>Set the attribute to</td></tr>
<tr><td>Enable the task</td><td>0 (default)</td></tr>
<tr><td>Inhibit (disable) the task</td><td>1 (or any non-zero value)</td></tr>
</table>

Example

If Condition_1 = 0 then let Task_2 run.

1. The ONS instruction limits the true run of the SSV instruction to one scan.
2. The SSV instruction sets the InhibitTask attribute of Task_2 = 0. This uninhibits the task.

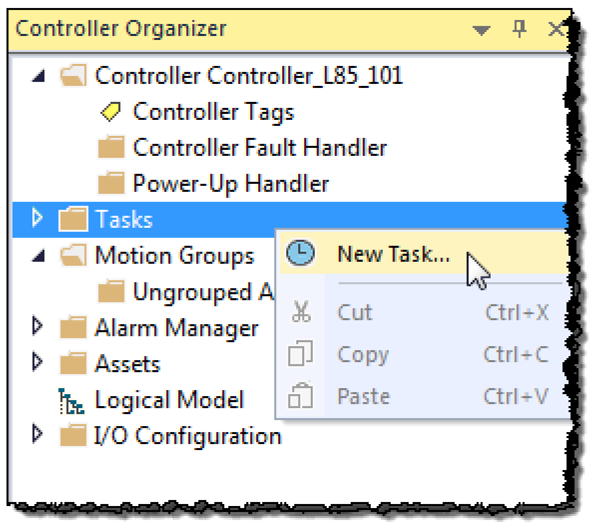
If Condition_1 = 1 then do not let Task_2 run.

1. The ONS instruction limits the true run of the SSV instruction to one scan.
2. The SSV instruction sets the InhibitTask attribute of Task_2 = 1. This inhibits the task.

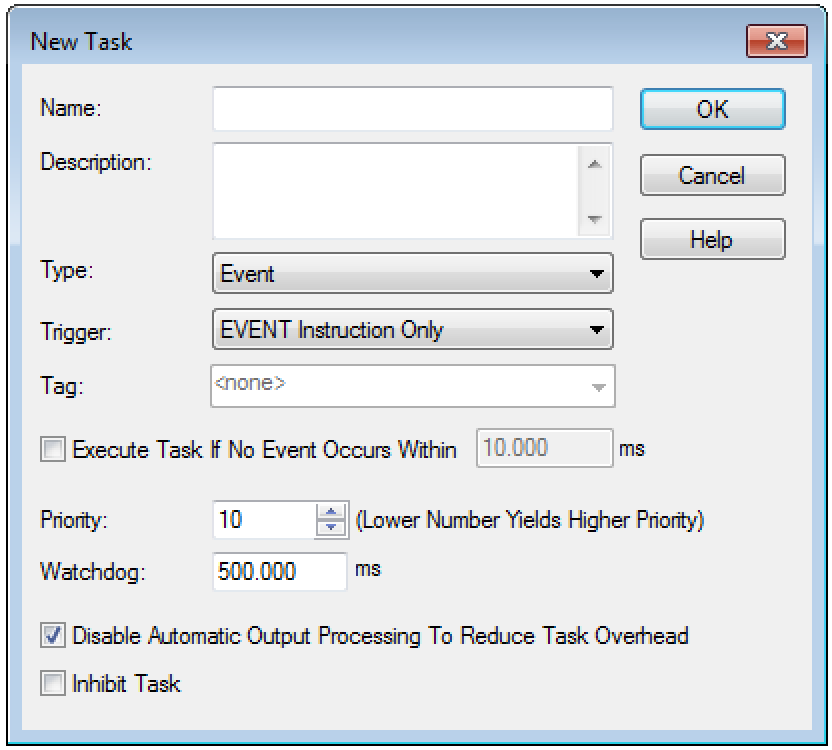
Create a Task

Follow these steps to create an event task.

1. In the Controller Organizer, right-click the **Tasks** folder and choose **New Task**.



2. Enter task information in the **New Task** dialog box.



Topic	Description
Name	Type a name for the task.
Description	Type an optional description for the task.
Type	Choose **Event** for the task type.
Trigger	Choose a trigger for the task.
Tag	Choose a tag if the field is active for the selected trigger.
Execute Task If No Event Occurs Within	Check the box and type a time period that must elapse before a task can run.
Priority	Enter the task priority value.
Watchdog	Type the watchdog time for the task.

3. Click **OK**.

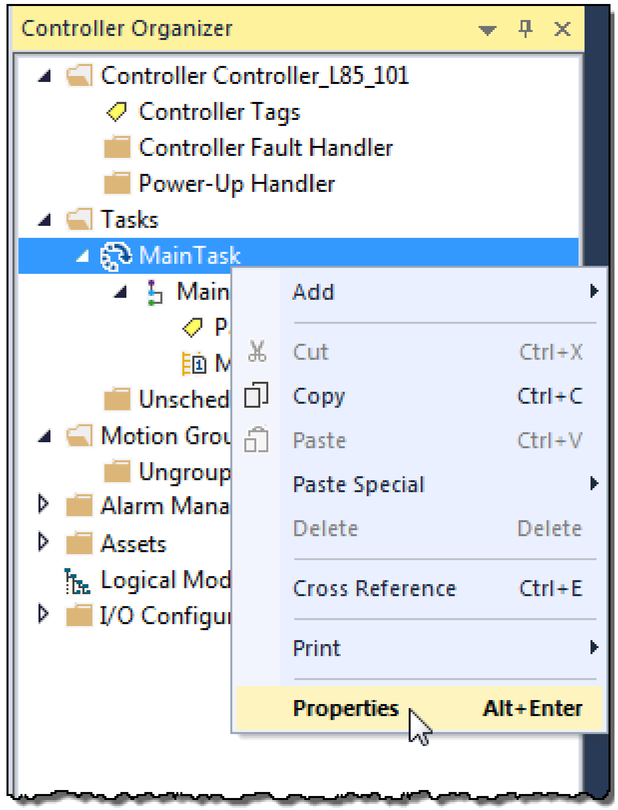
Create a Periodic Task

A periodic task performs a function or functions at a specific rate.

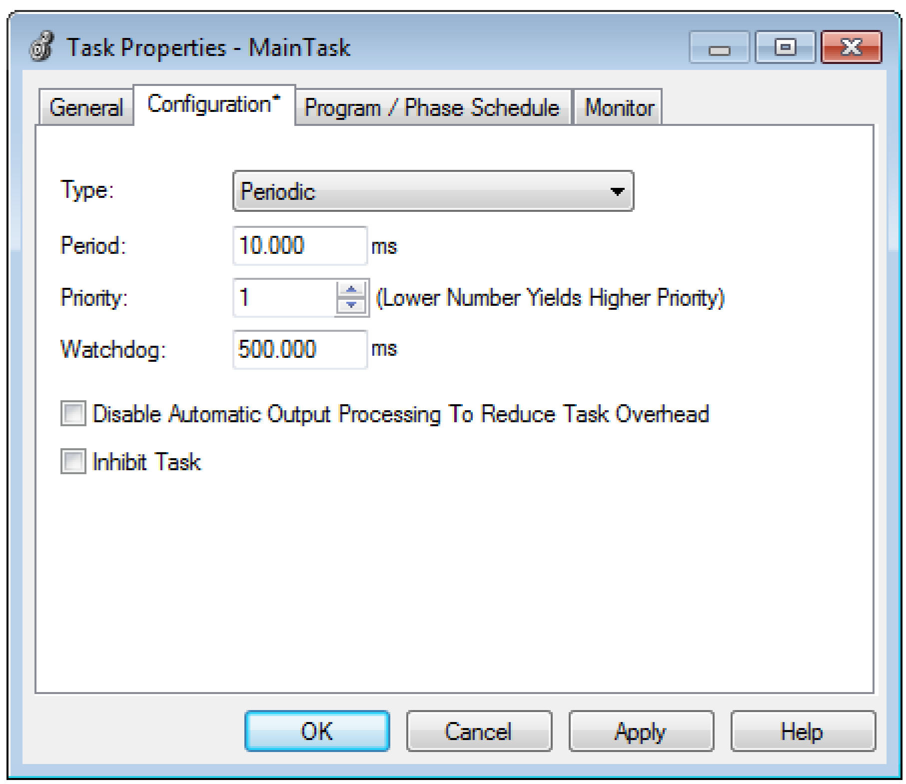
IMPORTANT Be sure that the time period is longer than the sum of the run times of all the programs assigned to the task.

- If the controller detects that a periodic task trigger occurs for a task that is already operating, a minor fault occurs (overlap).
- Priorities and run times of other tasks may also cause an overlap.

1. In the Controller Organizer, right-click the **MainTask** folder and choose **Properties**.



2. On the **Task Properties** dialog box, click the **Configuration** tab.



3. Enter this information in the **Task Properties** dialog box.

Topic	Description
Type	Choose **Periodic** (default) for the type of task.
Period	Type a value for when (or at what time interval) you want the task to run.
Priority	Enter the task priority value.
Watchdog	Type the watchdog time for the task.

4. Click **OK**.

Language Switching

In versions 17 and later of the application, you can display project documentation, such as tag descriptions and rung comments, for any supported localized language. You can store project documentation for multiple languages in a single project file rather than in language-specific project files. You define all the localized languages that the project will support and set the current, default, and optional custom-localized language. The software uses the default language if the current language's content is blank for a particular component of the project. However, you can use a custom language to tailor documentation to a specific type of project file user.

Enter the localized descriptions in your project, either when programming in that language or by using the import/export utility to translate the documentation offline and then import it back into the project. When you enable language switching, you can dynamically switch between languages as you use the software.

Project documentation that supports multiple translations within a project includes the following:

- Component descriptions in tags, routines, programs, user-defined data types, and Add-On Instructions
- Equipment phases
- Trends
- Controllers
- Alarm Messages (in ALARM_ANALOG and ALARM_DIGITAL configuration)
- Tasks
- Property descriptions for modules in the Controller Organizer
- Rung comments, SFC text boxes, and FBD text boxes

Adjust the System-overhead Time Slice

A Logix 5000 controller communicates with other devices, I/O modules, controllers, HMI terminals, and so forth, at either a specified rate (scheduled) or when there is processing time available to service the communication (unscheduled).

This type of communication	Is
Update I/O data (not including block-transfers)	Scheduled Communication
Produce or consume tags	
Communicate with programming devices (that is, the Logix Designer application)	Service Communication
Communicate with HMI devices	
Run Message (MSG) instructions, including block-transfers	

This type of communication	Is
Respond to messages from other controllers	
Synchronize the secondary controller of a redundant system	
Re-establish and monitor I/O connections (such as Removal and Insertion Under Power conditions); this does not include normal I/O updates that occur during the running of logic	

Service communication is any communication that you do not configure through the I/O configuration folder of the project.

The system-overhead time slice specifies the percentage of time a controller devotes to service communication. However, if there is no continuous task, the overhead time slice has no effect. If you have both a periodic and a continuous task, the value selected on the **Advanced** tab of the **Controller Properties** dialog box determines the ratio of running the continuous task and service communication.

The following table shows the ratio between the continuous task and service communication at various system overhead time slices.

At this time slice	The continuous tasks runs	Service communication occurs for up to
10%	9 ms	1 ms
20%	4 ms	1 ms
25%	3 ms	1 ms
33%	2 ms	1 ms
50%	1 ms	1 ms
66%	1 ms	2 ms
75%	1 ms	3 ms
80%	1 ms	4 ms
90%	1 ms	9 ms

As shown in the table, for version 16 and later of the application, the system overhead time slice at 50% stays fixed at 1 ms.

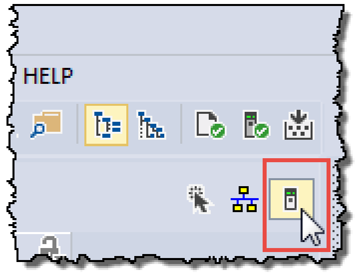
The same applies for 66% and higher, except that there are multiple 1 ms intervals. For example, at 66% there are two 1 ms intervals of consecutive time and at 90% there are nine 1 ms intervals of consecutive time.

Configure the System-overhead Time Slice

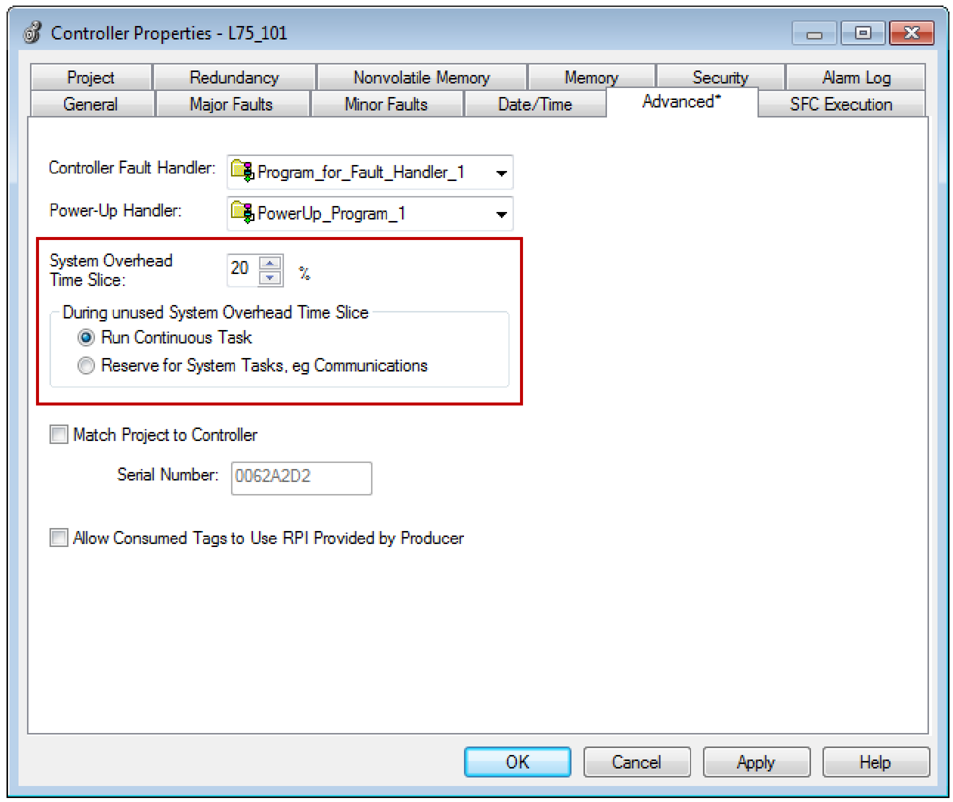
Follow these steps to configure the system-overhead time slice.

To configure the system-overhead time slice

1. On the Online toolbar, select the controller properties icon.



2. On the **Controller Properties** dialog box, select the **Advanced** tab.



3. Enter a number in the **System Overhead Time Slice** box.
4. Use either **Run Continuous Task** (default) or **Reserve for System Tasks**.
   - Select the **Run Continuous Task** radio button when there are no communication or background tasks to process; the controller immediately returns to the continuous task.
   - Select the **Reserve for System Task** radio button to allocate the entire 1 ms of the system-overhead time slice when the controller has communication or background tasks to perform before returning to the continuous task. This lets you simulate a communication load on the controller during design and

programming before HMIs, controller to controller messaging, and so forth, are set up. Use this setting for testing purposes only.

5. Select **OK**.

Adjust the System Watchdog Time

Each task contains a watchdog timer that specifies how long a task can run before triggering a major fault.

> ATTENTION: If the watchdog timer reaches a configurable preset, a major fault occurs. Depending on the controller fault handler, the controller might shut down.

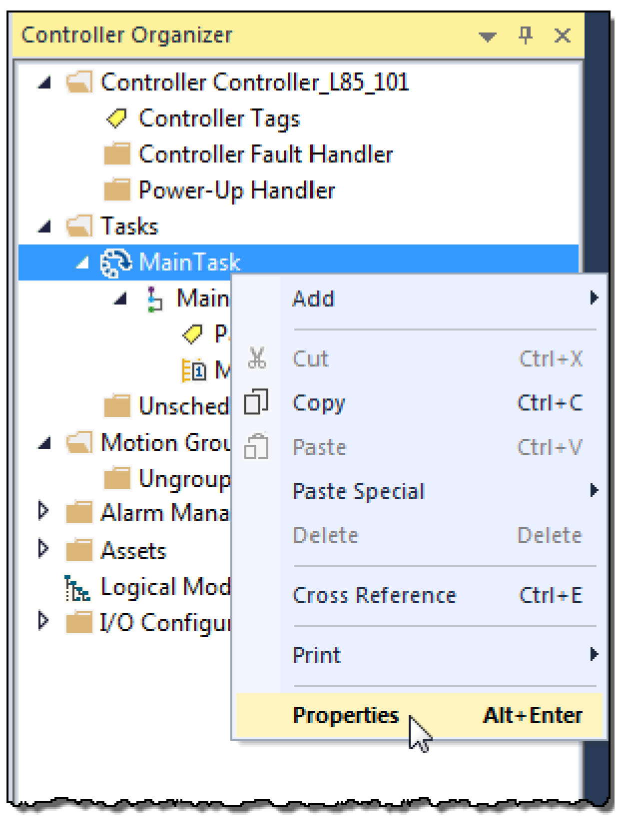
- A watchdog time can range from 1...2,000,000 ms (2000 seconds). The default is 500 ms.
- The watchdog timer begins to run when the task is initiated and stops when all the programs within the task have run.
- If the task takes longer than the watchdog time, a major fault occurs. (The time includes interruptions by other tasks.)
- You can use the controller fault handler to clear a watchdog fault. If the same watchdog fault occurs a second time during the same logic scan, the controller enters Faulted mode, regardless of whether the controller fault handler clears the watchdog fault.

Adjust the Watchdog Timer for a Task

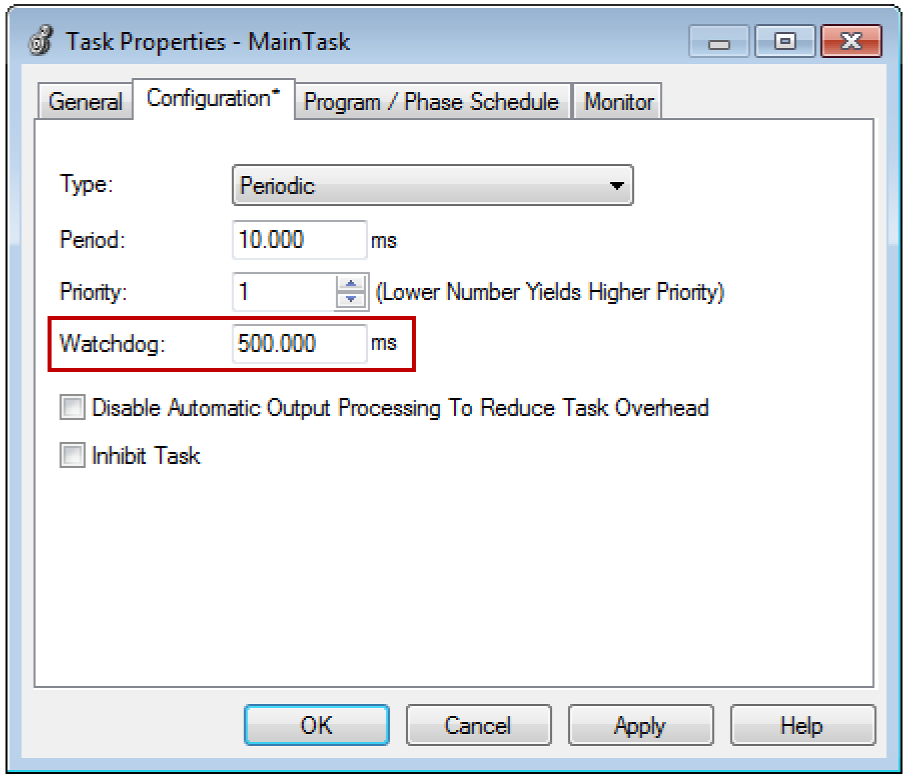
Follow these steps to change the watchdog time of a task.

To adjust the watchdog timer for a task

1. In the Controller Organizer, right-click **Main Task** and choose **Properties**.



2. On the **Task Properties** dialog box, select the **Configuration** tab.



3. Type a numeric value for the watchdog timeout for the task.
4. Select **OK**.

See also

Adjust the System Watchdog Time on page 30

Adjust the System-overhead Time Slice on page 27

Manage Event Tasks

Introduction

An event task, if configured correctly, interrupts all other tasks for the minimum amount of time required to respond to the event.

This section describes how to set up event tasks and lists considerations, such as a higher priority task, that can affect the execution of an event task.

Choose the trigger for an event task

Each event task requires a specific trigger that defines when the task is to run. The table reviews some of these triggers.

To trigger an event task when	Use this trigger	With these considerations
Digital input turns On or Off	Module Input Data State Change	• Only one input module can trigger a specific event task. • The input module triggers the event task based on the change of state (COS) configuration for the module. The COS configuration defines which points prompt the module to produce data if they turn On or Off. This production of data (due to COS) triggers the event task. • Typically, enable COS for only one point on the module. If you enable COS for multiple points, a task overlap of the event task may occur.
Analog module samples data	Module Input Data State Change	• Only one input module can trigger a specific event task. • The analog module triggers the event task after each real time sample (RTS) of the channels. • All the channels of the module use the same RTS.
Controller gets new data via a consumed tag	Consumed Tag	• Only one consumed can trigger a specific event task. • Typically, use an IOT instruction in the producing controller to signal the production of new data. The IOT instruction sets an event trigger in the producing tag. This trigger passes to the consumed tag and triggers the event task. • When a consumed tag triggers an event task, the event task waits for all the data to arrive before the event task runs.
Registration input for an axis turns On (or Off)	Axis Registration 1 or 2	• For the registration input to trigger the event task, first run a Motion Arm Registration (MAR) instruction. This lets the axis detect the registration input and in turn trigger the event task. • Once the registration input triggers the event task, run the MAR instruction again to re-arm the axis for the next registration input. • If the scan time of your normal logic is **not** fast enough to re-arm the axis for the next registration input, consider placing the MAR instruction within the event task.

To trigger an event task when	Use this trigger	With these considerations
Axis reaches the position that is defined as the watch point	Axis Watch	• For the registration input to trigger the event task, first run a Motion Arm Watch (MAW) instruction. This lets the axis detect the watch position and in turn trigger the event task. • Once the watch position triggers the event task, run the MAW instruction again to re-arm the axis for the next watch position. • If the scan time of your normal logic is not fast enough to re-arm the axis for the next watch position, consider placing the MAW instruction within the event task.
Motion planner completes its execution	Motion Group Execution	• The base update period for the motion group triggers both the motion planner and the event task. • Because the motion planner interrupts all other tasks, it runs first. If you assign the event task as the highest priority task, it runs after the motion planner.
Specific condition or conditions occur within the logic of a program	EVENT instruction	Multiple EVENT instructions can trigger the same task. This lets you run a task from different programs.

The table lists some example situations for event tasks and the corresponding triggers.

For this example situation	Use an event task with this trigger
A packaging line glues boxes closed. When a box arrives at the gluing position, the controller must immediately run the gluing routine.	Module Input Data State Change
A production line uses a proximity sensor to detect the presence of a part. Because the proximity sensor is on for only a very short time (pulse), the continuous task might miss the off to on transition of the sensor.	Module Input Data State Change
In an engine test stand, you must capture and archive each sample of analog data.	Module Input Data State Change
Controller A produces an array of production data for Controller B. You want to make sure that Controller B does not use the values while Controller A is updating the array.	Consumed Tag
In a line that packages candy bars, you have to make sure that the perforation occurs in the correct location on each bar. Each time the registration sensor detects the registration mark, check the accuracy of an axis and perform any required adjustment.	Axis Registration 1 or 2
At the labeling station of a bottling line, you want to check the position of the label on the bottle. When the axis reaches the position that is defined as the watch point, check the label.	Axis Watch
A gluing station must adjust the amount of glue it applies to compensate for changes in the speed of the axis. After the motion planner runs, check the command speed of the axis and vary the amount of glue, if needed.	Motion Group Execution
In a production line, if any of the programs detect an unsafe condition the entire line must shut down. The shutdown procedure is the same regardless of the unsafe condition.	EVENT instruction

The triggers that you can use for an event task vary depending on your controller type.

IMPORTANT The Logix Designer application may let you configure a trigger for an event task that your controller does not support. The project verifies and successfully downloads, but the event task does not run.

Controller	Applicable event task triggers					
	Module Input Data State Change	Consumed Tag	Axis Registration 1 or 2	Axis Watch	Motion Group Execution	EVENT instruction
CompactLogix 5370		X	X[3]	X[3]	X[3]	X
CompactLogix 5380	X	X	X[3]	X[3]	X[3]	X
FlexLogix		X				X
ControlLogix	X	X	X	X	X	X
DriveLogix		X	X	X	X	X
SoftLogix 5800	X[1]	X[2]				X
Compact GuardLogix 5370		X	X[3]	X[3]	X[3]	X
Compact GuardLogix 5380	X		X[3]	X[3]	X[3]	X
CompactLogix 5480	X	X	X[3]	X[3]	X[3]	X

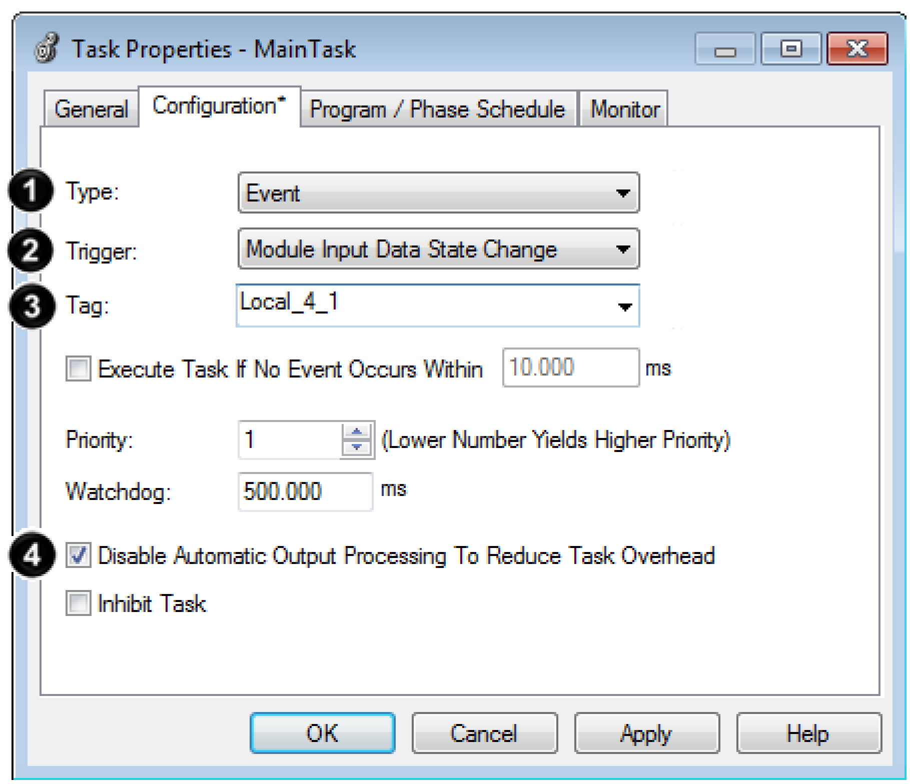
(1) Requires a 1756 I/O module or a virtual backplane.

(2) A SoftLogix5800 controller produces and consumes tags only over a ControlNet network.

(3) Motion controller support only.

Module Input Data State Change Trigger

To trigger an event task based on data from an input module, use the **Module Input Data State Change** trigger.



1	Let an event trigger this task.
2	Let data from an input module trigger the task.
3	Let this input tag trigger the task.

❶	Let an event trigger this task.
❹	When the task is done, do not update digital outputs in the local chassis.

How an I/O Module Triggers an Event Task

These terms apply to the operation of an input module.

Term	Definition
Multicast	A mechanism where a module sends data on a network that is simultaneously received by more than one listener (device). Describes the feature of the Logix 5000 I/O line that supports multiple controllers receiving input data from the same I/O module at the same time.
Requested packet interval (RPI)	The RPI specifies the interval that a module multicasts its data. For example, an input module sends data to a controller at the RPI that you assign to the module.
	• The range is 0.2…750 ms. • When the specified time frame elapses, the module multicasts its data. This is also called a cyclic update.
Real time sample (RTS)	The RTS specifies when an analog module scans its channels and multicasts the data (update the input data buffer then multicast).
	• The RPI specifies when the module multicasts the current contents of the input data buffer without scanning (updating) the channels. • The module resets the RPI timer each time an RTS transfer occurs.
Change of state (COS)	The COS parameter instructs a digital input module to multicast data whenever a specified input point transitions from On → Off or Off → On.
	• You enable COS on a per-point basis. • When any point that is enabled for COS receives the specified change, the module multicasts the data for all its points. • By default, COS is enabled for both On → Off and Off → On changes for all points. • You must specify an RPI regardless of whether you enable COS. If a change does not occur within the RPI, the module sends its data at the RPI.

The table summarizes when an input module multicasts its data and triggers an event task within its own chassis.

If the input module is	And	Then it multicasts data	And it triggers an event task
Digital	COS is enabled for any point on the module	• When any point that is enabled for COS receives the specified change • At the RPI	When any point that is enabled for COS receives the specified change
	COS is not enabled for any point on the module	At the RPI	Never
Analog	RTS ≤ RPI	At the RTS (newly updated channel data)	At the RTS for the module
	RTS > RPI	• At the RTS (newly updated channel data) • At the RPI (does not contain updated data from the channels)	At the RTS for the module

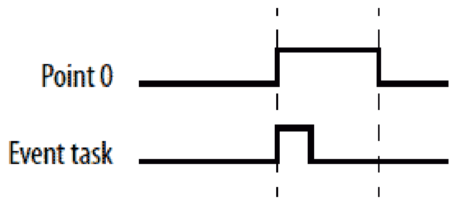
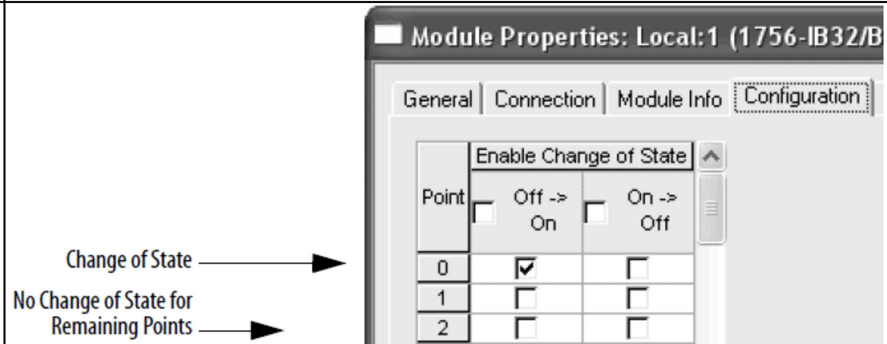
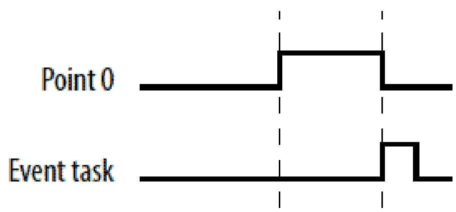
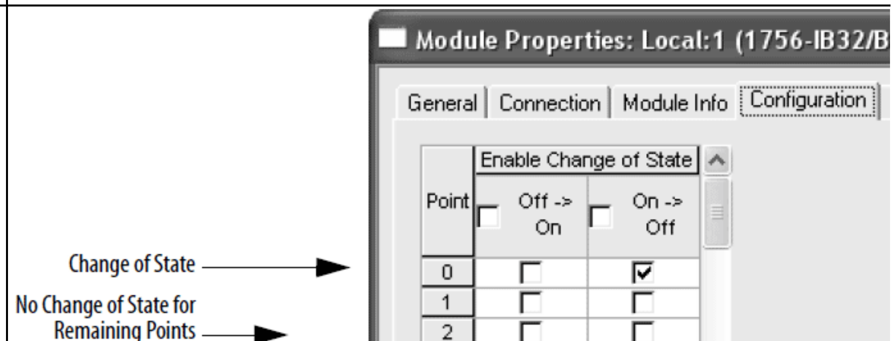
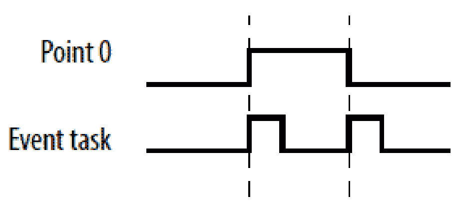
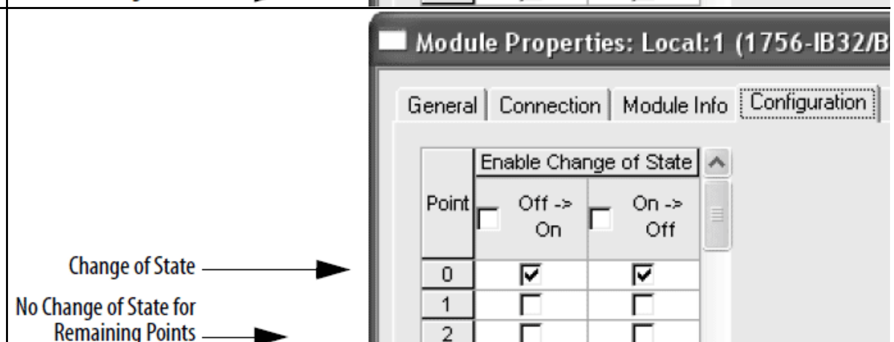
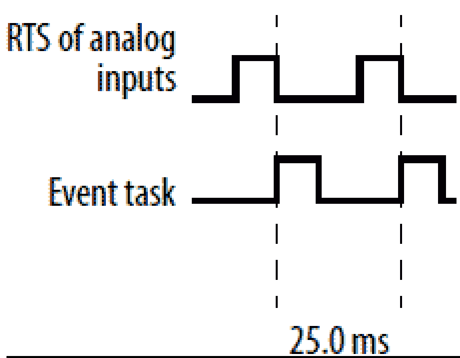
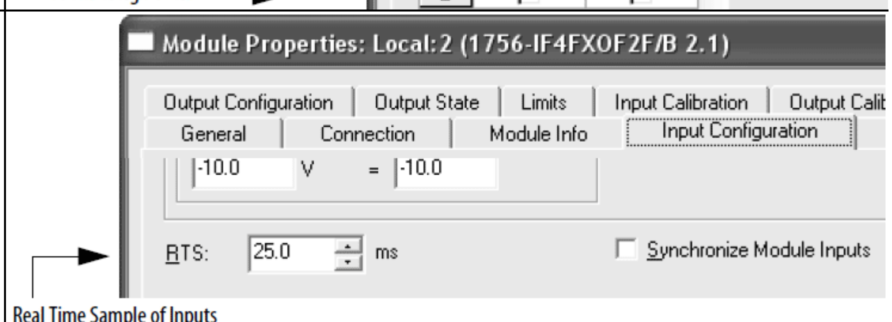
If the module is in a remote chassis, only the RPI determines when the controller receives the data and event trigger over the network.

Over this network	Controller receives the data
EtherNet/IP	Close to the RPI, on average
ControlNet	At the actual packet interval (≤ RPI)

The examples show COS and RTS configurations.

IMPORTANT	If you use a digital module to trigger an event task, configure only one point on the module for COS. If you configure multiple points, a task overlap could occur.

COS and RTS Configuration Examples

<table>
<tr><th>If you want this</th><th>Then configure the input module like this (Point 0 is an example)</th></tr>
<tr><td>
</td><td>
</td></tr>
<tr><td>
</td><td>
</td></tr>
<tr><td>
</td><td>
</td></tr>
<tr><td>
</td><td>
</td></tr>
</table>

Make Sure Your Module Can Trigger an Event Task

To use an input module to trigger an event task, the module must support event task triggering. If the module is in a remote location, the associated communication modules must also support event triggering.

The following table lists Rockwell Automation modules that have been tested for event task triggering. Some third-party modules may also support event task triggering. Before you use a third-party module, check with the supplier to validate the operation of the module.

Category	Modules	
Digital I/O modules that support change of state	1756-IA8D	1756-IA16
	1756-IA16I	1756-IA32
	1756-IB16	1756-IB16D
	1756-IB16I	1756-IB16ISOE
	1756-IB32	1756-IC16
	1756-IG16	1756-IH16I
	1756-IH16ISOE	1756-IM16I
	1756-IN16	1756-IV16
	1756-IV32	
Analog I/O modules that support real time sample	1756-IF16	1756-IF4FXOF2F/A
	1756-IF6CIS	1756-IF6I
	1756-IF8	1756-IR6I
	1756-IT6I	1756-IT6I2
Communication modules that provide rack-optimized connections	1756-CNB/A	1756-CNB/B
	1756-CNB/D	1756-CNBR/A
	1756-CNBR/B	1756-CNBR/D
	1756-DNB	1756-ENBT/A
	1756-SYNCH/A	1784-PCIDS/A
Generic I/O modules that conform to CIP event communication	1756-MODULE	
	1789-MODULE	

Checklist for an Input Event Task

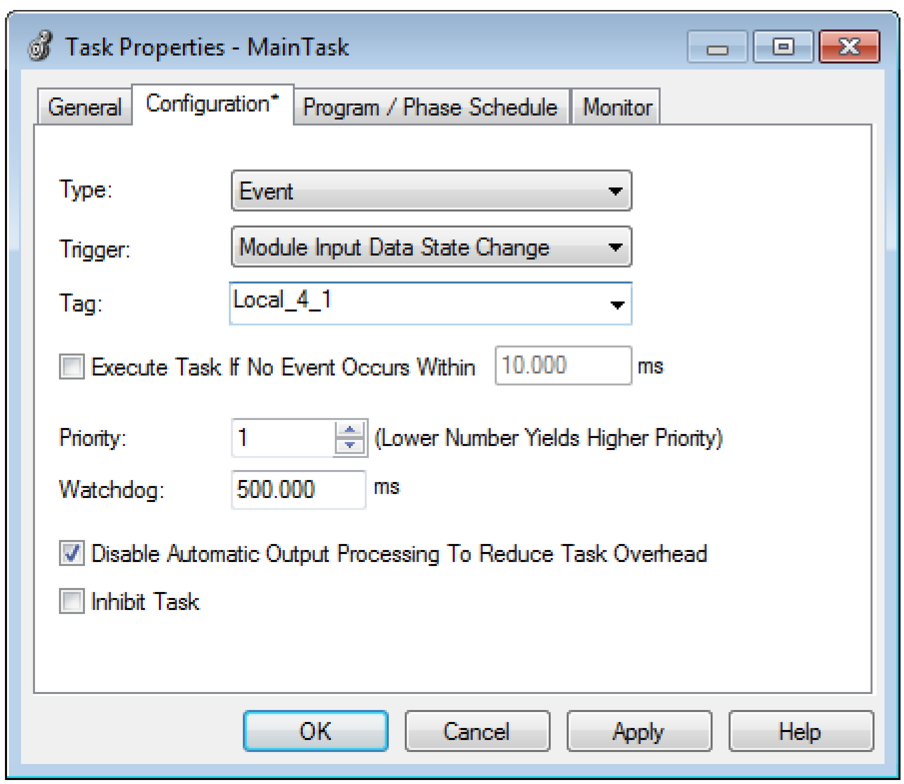
Use the following checklist when creating an Input Event Task.

For This			Make Sure You
☐	1.	Input module type	For the fastest response, use these modules: • For fastest digital response, use a 1756-IB32/B module. • For fastest analog response, use a 1756-IF4FXOF2F module.
☐	2.	I/O module location	Place the module that triggers the event and the modules that respond to the event (outputs) in the same chassis as the controller. Remote modules add network communication to the response time.
☐	3.	Number of local modules	Limit the number of modules in the local chassis. Additional modules increase the potential for backplane delays.
☐	4.	Change of state (COS)	If a digital device triggers the event, enable COS for only the point that triggers the event task. • Enable change of state for the type of transition that triggers the task, either Off → On, On → Off, or both. • If you configure COS for both Off → On and On → Off, the point triggers an event task whenever the point turns on or off. Make sure the duration of the input is longer than the scan time of the task. Otherwise an overlap could occur. • Disable (clear) COS for the remaining points on the input module. If you configure multiple points on a module for COS, each point could trigger the event task. This could cause an overlap.

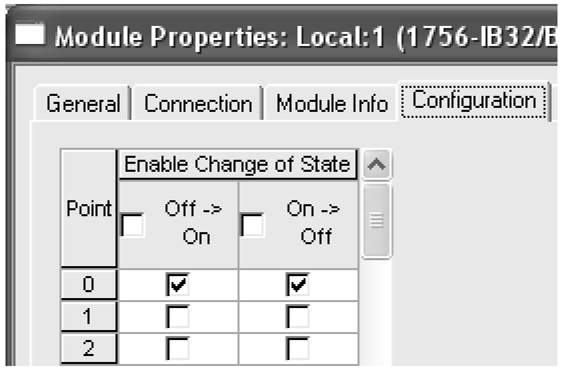
☐	5.	Task priority	Configure the event task as the highest priority task. If a periodic task has a higher priority, the event task may have to wait until the periodic task is done.
☐	6.	Motion planner	The motion planner interrupts all other tasks, regardless of their priority. • The number of axes and coarse update period for the motion group affect how long and how often the motion planner executes. • If the motion planner is executing when a task is triggered, the task waits until the motion planner is done. • If the coarse update period occurs while a task is executing, the task pauses to let the motion planner execute.
☐	7.	Number of event tasks	Limit the number of event tasks. Each additional task reduces the processing time that is available for other tasks. This could cause an overlap.
☐	8.	Automatic Output Processing	For an event task, you can typically disable automatic output processing (default). This reduces the elapsed time of the task.
☐	9.	IOT instruction	Use an IOT instruction for each output module that you reference in the event task. The IOT instruction overrides the RPI for the module and immediately sends the data.

Example – Input Event Task

As parts move past a diverter location, the controller logic determines whether to turn on the diverter. Once the diverter is on, the controller must also turn it off before the next part is in that position. Because of the speed of the line, an event task controls the diverter.



A photoeye at the diverter position indicates when a part is in the diverter position. In this example, the input is wired to the module in slot 4 of the local chassis.



The diverter photoeye (point 0) is configured for change of state for both Off and On. This lets the photoeye trigger the event task when it turns on and when it turns off.

The event task uses the following logic to control the diverter.

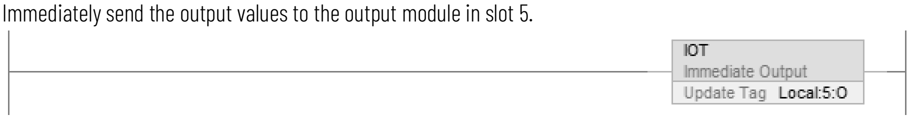
If Diverter_Photoeye = 1 (part is in the diverter position)

and Divert_Part = 1 (divert this part)

then Diverter = 1 (turn on the diverter)

otherwise Diverter = 0 (turn off the diverter)

Immediately send the output values to the output module in slot 5.



Estimate Throughput

To estimate the throughput time from input to output (screw to screw), use the following worksheet.

<table>
<tr><th></th><th>Consideration</th><th>Value</th></tr>
<tr><td>1.</td><td>What is the input filter time of the module that triggers the event task?
This is typically shown in milliseconds. Convert it to microseconds (µs).</td><td>µs</td></tr>
<tr><td>2.</td><td>What is the hardware response time for the input module that triggers the event task?
Make sure you use the appropriate type of transition (Off → On or On → Off). See Nominal hardware response times for the 1756 I/O modules most commonly used with Event tasks later in this section.</td><td>µs</td></tr>
<tr><td>3.</td><td>What is the backplane communication time?<table><tr><th>If chassis size is</th><th>Use this value (worst case)</th></tr><tr><td>4 slot</td><td>13 µs</td></tr><tr><td>7 slot</td><td>22 µs</td></tr><tr><td>10 slot</td><td>32 µs</td></tr><tr><td>13 slot</td><td>42 µs</td></tr><tr><td>17 slot</td><td>54 µs</td></tr></table></td><td>µs</td></tr>
<tr><td>4.</td><td>What is the total execution time of the programs of the event task?</td><td>µs</td></tr>
</table>

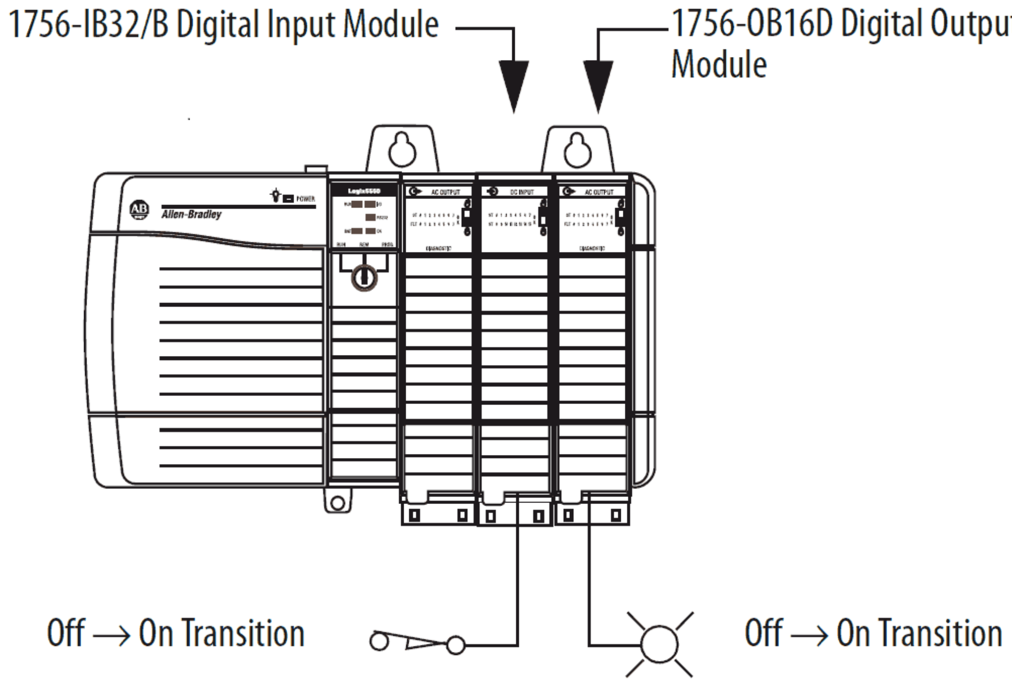
Consideration		Value
5.	What is the backplane communication time? (Same value as step 3.)	µs
6.	What is the hardware response time of the output module?	µs
7.	Add steps 1…6. This is the minimum estimated throughput, where execution of the motion planner or other tasks do **not** delay or interrupt the event task.	µs
8.	What is the scan time of the motion group?	µs
9.	What is the total scan time of the tasks that have a higher priority than this event task (if any)?	µs
10.	Add steps 7…9. This is the nominal estimated throughput, where execution of the motion planner or other tasks delay or interrupt the event task.	µs

The following table lists nominal hardware response times for 1756 I/O modules with event tasks.

Cat. No.	Nominal response time µs			
	25 °C		60 °C	
	Off → On	On → Off	Off → On	On → Off
1756-IB16	265	582	265	638
1756-IB16D	303	613	305	673
1756-IB32/B	330	359	345	378
1756-IV16	257	435	254	489
1756-IV32	381	476	319	536
1756-OB16D	48	519	51	573
1756-OB16E	60	290	61	324
1756-OB32	38	160	49	179
1756-OV16E	67	260	65	326
1756-OV32E	65	174	66	210

Example - Estimate Throughput

The following example shows the throughput considerations for the system shown in the following illustration. In this example, the throughput is the time from when the input turns on to when the output turns on.



<table>
<tr><th colspan="2">Consideration</th><th>Value</th></tr>
<tr><td>1.</td><td>What is the input filter time of the module that triggers the event task?
This is typically shown in milliseconds. Convert it to microseconds (μs).</td><td>0 μs</td></tr>
<tr><td>2.</td><td>What is the hardware response time for the input module that triggers the event task?
Make sure you use the appropriate type of transition (Off → On or On → Off). See the table, earlier in this section, that lists nominal hardware response times for the 1756 I/O modules most commonly used with Event tasks.</td><td>330 μs</td></tr>
<tr><td>3.</td><td>What is the backplane communication time?
<table>
<tr><th>If chassis size is</th><th>Use this value (worst case)</th></tr>
<tr><td>4 slot</td><td>13 μs</td></tr>
<tr><td>7 slot</td><td>22 μs</td></tr>
<tr><td>10 slot</td><td>32 μs</td></tr>
<tr><td>13 slot</td><td>42 μs</td></tr>
<tr><td>17 slot</td><td>54 μs</td></tr>
</table></td><td>13 μs</td></tr>
<tr><td>4.</td><td>What is the total run time of the programs of the event task?</td><td>400 μs</td></tr>
<tr><td>5.</td><td>What is the backplane communication time? (Same value as step 3.)</td><td>13 μs</td></tr>
<tr><td>6.</td><td>What is the hardware response time of the output module?</td><td>51 μs</td></tr>
<tr><td>7.</td><td>Add steps 1...6. This is the minimum estimated throughput, where execution of the motion planner or other tasks do not delay or interrupt the event task.</td><td>807 μs</td></tr>
<tr><td>8.</td><td>What is the scan time of the motion group?</td><td>1130 μs</td></tr>
<tr><td>9.</td><td>What is the total scan time of the tasks that have a higher priority than this event task (if any)?</td><td>0 μs</td></tr>
<tr><td>10.</td><td>Add steps 7...9. This is the nominal estimated throughput, where execution of the motion planner or other tasks delay or interrupt the event task.</td><td>1937 μs</td></tr>
</table>

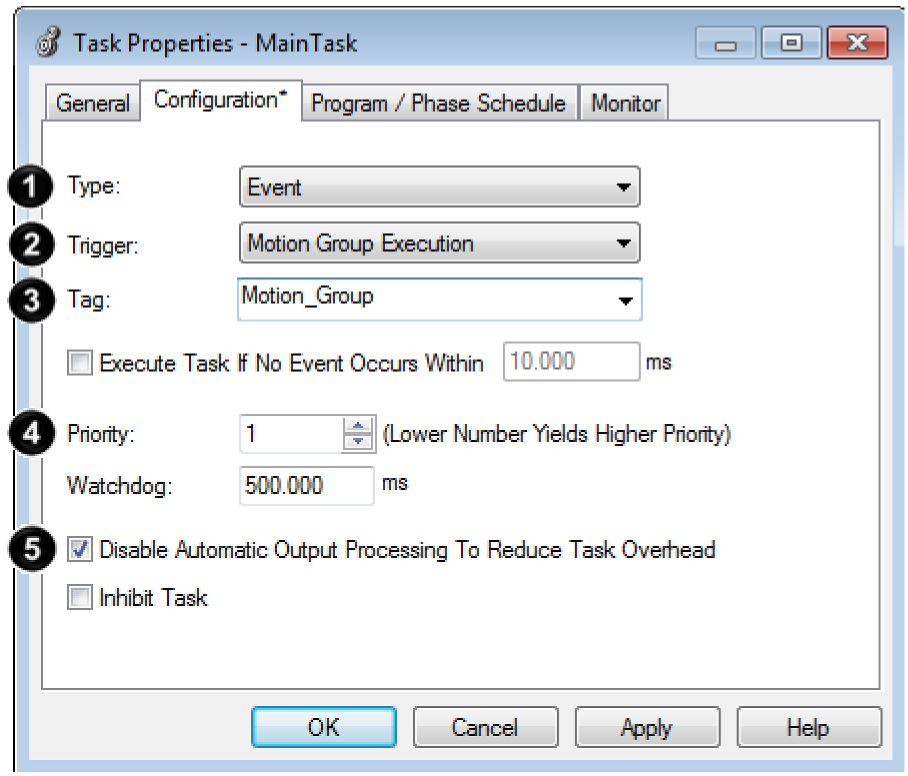
Additional Considerations

The following considerations affect the scan time of the event task, which affects the speed at which it can respond to the input signal.

Consideration	Description
Amount of code in the event task	Each logic element (rung, instruction, Structured Text construct, and so forth) adds scan time to the task.
Task priority	If the event task is not the highest priority task, a higher priority task may delay or interrupt the execution of the event task.
CPS and UID instructions	If one of these instructions are active, the event task cannot interrupt the currently running task. (The task with the CPS or UID.)

Motion Group Trigger

To couple the running of an event task with the running of the motion planner, use the **Motion Group Execution** trigger.

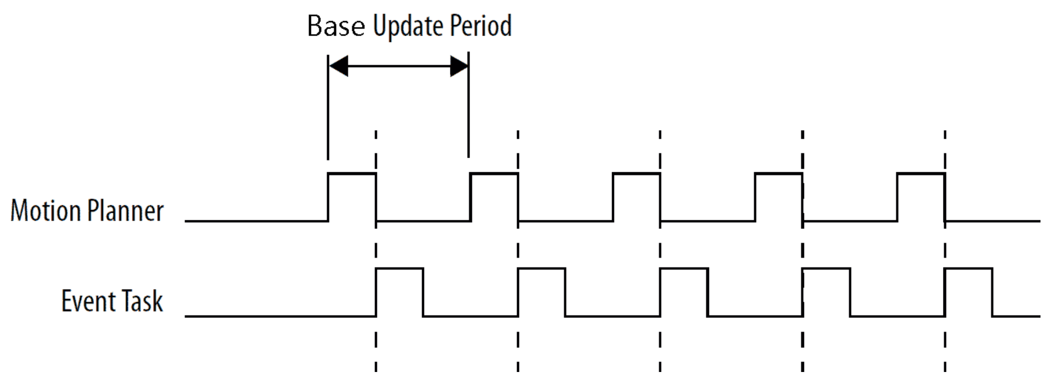


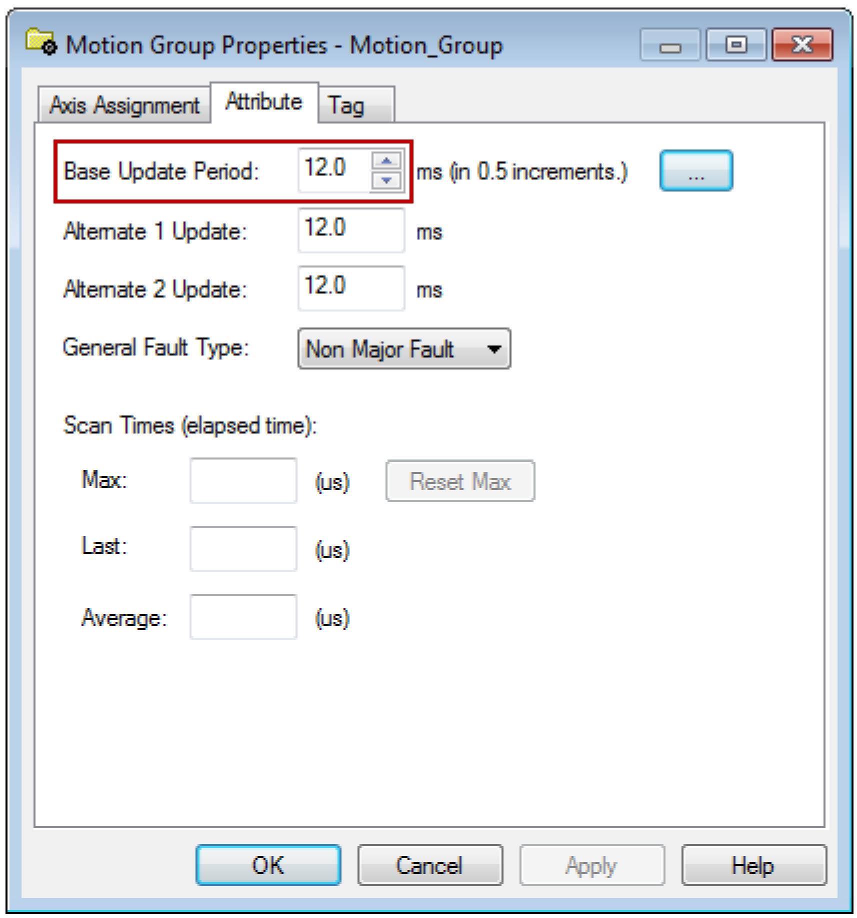
1	Let an event trigger this task.
2	Let the motion planner trigger the task.
3	This is the name of the motion group tag.
4	Interrupt all other tasks.
5	When the task is done, do not update digital outputs in the local chassis.

The **Motion Group Execution** trigger works as follows:

- The base update period for the motion group triggers the running of both the motion planner and the event task.
- Because the motion planner interrupts all other tasks, it runs first. If you assign the event task as the highest priority task, it runs immediately after the motion planner.

This timing diagram shows the relationship between the motion planner and the event task.





The **Base Update Period** for the motion group triggers both the motion planner and the event task. See the online help for more information on the **Motion Group Properties** dialog box.

Checklist for a Motion Group Task

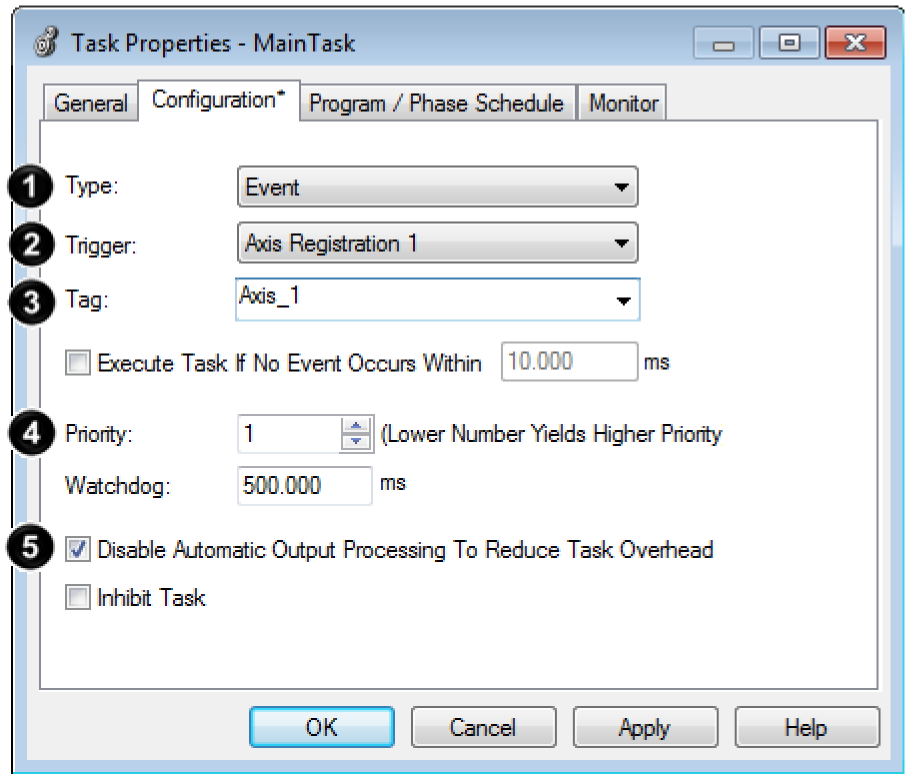
The following is the checklist for a motion group task:

For This		Make Sure You
☐	1. Scan time	Make sure the scan time of the event task is significantly less than the base update period of the motion group. Otherwise, a task overlap could occur.

☐	2.	Task priority	Configure the event task as the highest priority task. If a periodic task has a higher priority, the event task may have to wait until the periodic task is finished.
☐	3.	Number of event tasks	Limit the number of event tasks. Each additional task reduces the processing time that is available for other tasks. This could cause an overlap.
☐	4.	Automatic output processing	For an event task, you can typically disable automatic output processing (default). This reduces the elapsed time of the task.

Axis Registration Trigger

To let the registration input of an axis trigger an event task, use the **Axis Registration 1** or **Axis Registration 2** triggers.



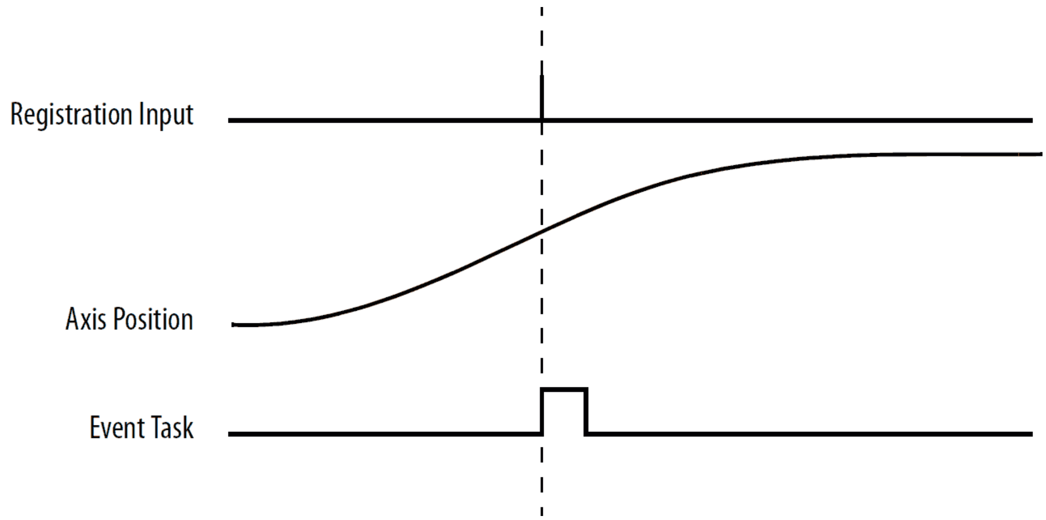
1	Let an event trigger this task.
2	Let registration input 1....
3	...of this axis trigger the task.
4	Interrupt all other tasks.
5	When the task is done, do not update digital outputs in the local chassis.

When the specified registration input reaches its trigger condition, it triggers the event task.

- In the configuration of the event task, specify which registration input you want to trigger the task. Choose either **Axis Registration 1** or **Axis Registration 2**.
- You must first arm the registration input using a **Motion Arm Registration** (MAR) instruction.

- In the MAR instruction, the Trigger Condition operand defines which transition of the registration input (Off → On or On → Off) triggers the event task.
- Once the registration input triggers the task, you have to re-arm the registration input.

This timing diagram shows the relationship between the registration input and the event task.



Checklist for an Axis Registration Task

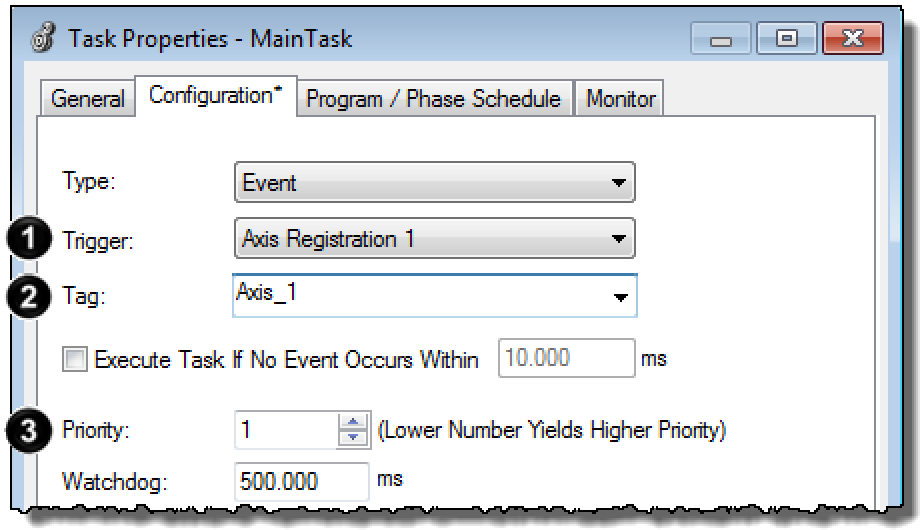
The following is a checklist for an axis registration task:

<table>
<tr><th colspan="2">For This</th><th colspan="2">Make Sure You</th></tr>
<tr><td rowspan="4">☐</td><td rowspan="4">1. Registration input</td><td colspan="2">Arm the registration input (MAR instruction). This lets the axis detect the registration input and trigger the event task.
• Initially, arm the registration input to detect the first trigger condition.
• Re-arm the registration input after each execution of the event task.
• Re-arm the registration input fast enough to detect each trigger condition.</td></tr>
<tr><td>If your normal logic is</td><td>Then</td></tr>
<tr><td>Fast enough to re-arm the registration input between intervals of the trigger condition
For example, normal logic always completes at least two scans between registration inputs.</td><td>Arm the registration input within your normal logic, if desired.</td></tr>
<tr><td>Not fast enough to re-arm the registration input</td><td>Arm the registration input within the event task.</td></tr>
<tr><td>☐</td><td>2. Task priority</td><td colspan="2">Configure the event task as the highest priority task.
If a periodic task has a higher priority, the event task may have to wait until the periodic task is finished.</td></tr>
<tr><td>☐</td><td>3. Number of event tasks</td><td colspan="2">Limit the number of event tasks.
Each additional task reduces the processing time that is available for other tasks. This could cause an overlap.</td></tr>
<tr><td>☐</td><td>4. Automatic output processing</td><td colspan="2">For an event task, you can typically disable automatic output processing (default). This reduces the elapsed time of the task.</td></tr>
</table>

Example - Axis Registration Trigger

In a line that packages candy bars, you have to make sure that the perforation occurs in the correct location on each bar.

- Each time the registration sensor detects the registration mark, check the accuracy of an axis and perform any required adjustment.
- Due to the speed of the line, you have to arm the registration input within the event task.



1	A registration sensor is wired as registration input 1...
2	...for the axis named *Axis_1*.
3	This event task interrupts all other tasks.

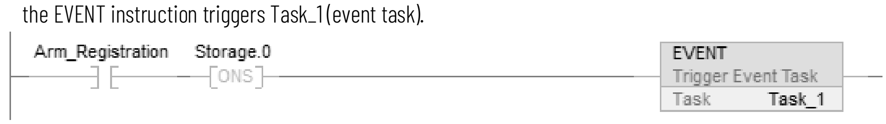
The following logic arms and re-arms the registration input.

Continuous task

If Arm_Registration = 1 (system is ready to look for the registration mark) then

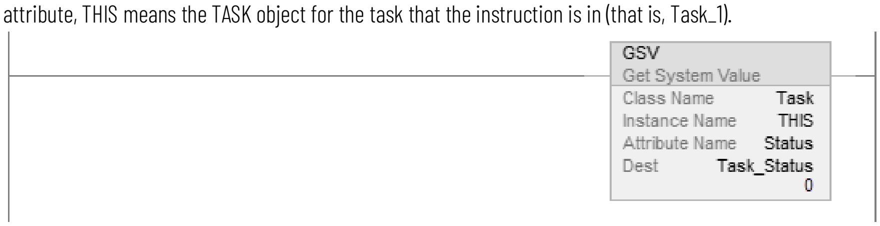
the ONS instruction limits the EVENT instruction to one scan.

the EVENT instruction triggers Task_1 (event task).



Task_1 (event task)

The GSV instruction sets Task_Status (DINT tag) = Status attribute for the event task. In the Instance Name attribute, THIS means the TASK object for the task that the instruction is in (that is, Task_1).



If Task_Status.0 = 1 then an EVENT instruction triggered the event task. In the continuous task, the EVENT runs to arm registration for the first time.

The JMP instruction causes the controller to jump to the Arm LBL instruction. This skips all the logic of the routine except the rung that arms registration for the axis.

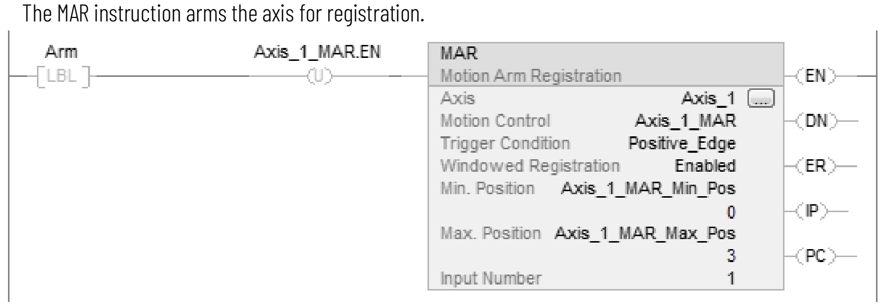
- Other logic

The MAR instruction runs each time the task runs and arms Axis_1 for registration.

The OTU instruction sets the EN bit of the MAR instruction = 0.

- The MAR instruction is a transitional instruction.
- For the MAR instruction to run, its rung-condition-in must go from false to true.
- By first clearing the EN bit, the instruction responds as if its rung-condition-in changed from false to true.

The MAR instruction arms the axis for registration.

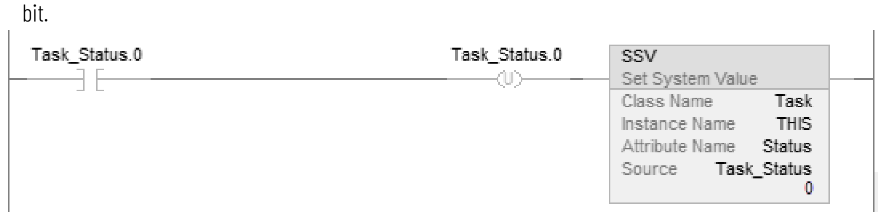


The controller does not clear the bits of the Status attribute once they are set. To use a bit for new status information, you must manually clear the bit.

If Task_Status.0 = 1 then clear that bit.

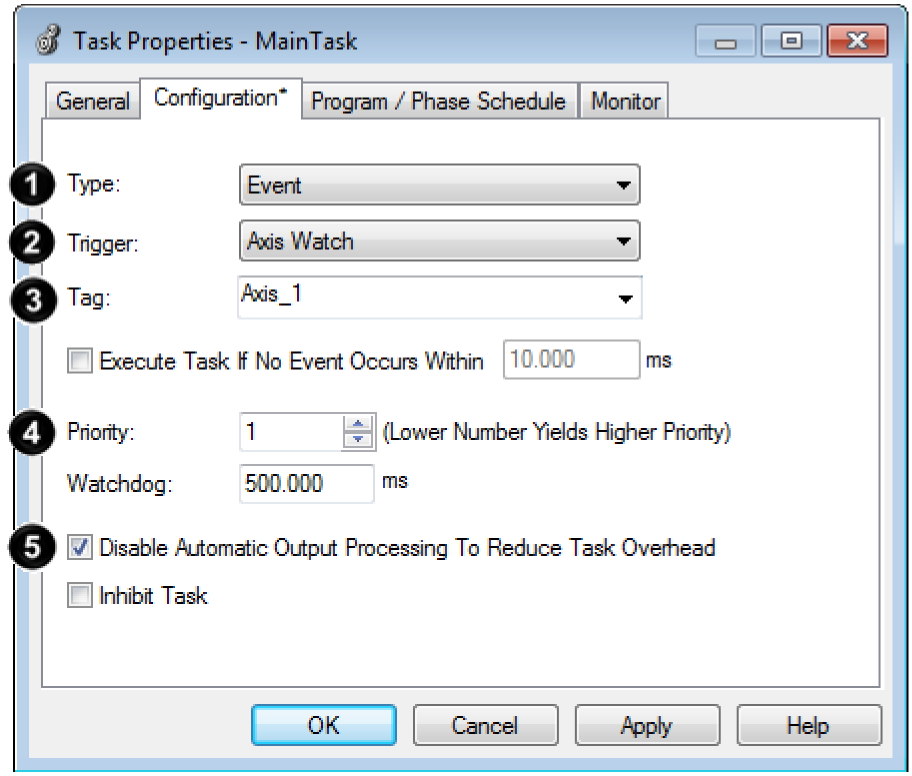
The OTU instruction sets Task_Status.0 = 0.

The SSV instruction sets the Status attribute of THIS task (Task_1) = Task_Status. This includes the cleared bit.



Axis Watch Trigger

To configure the watch position of an axis to trigger an event task, use the **Axis Watch** trigger.

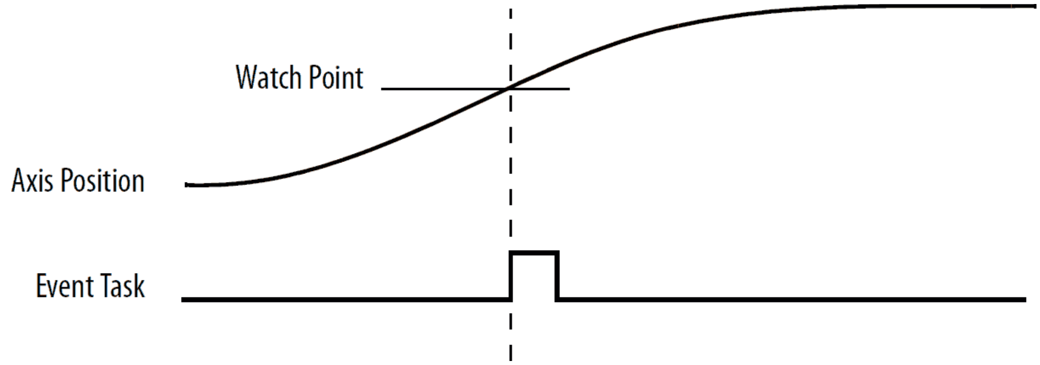


1	Let an event trigger this task.
2	Let the watch position….
3	…of this axis trigger the task.
4	Interrupt all other tasks.
5	When the task is done, do not update digital outputs in the local chassis.

When the axis reaches the position that is specified as the watch position, it triggers the event task.

- You must first arm the axis for the watch position by using a Motion Arm Watch (MAW) instruction.
- In the MAW instruction, the Trigger Condition operand defines the direction in which the axis must be moving to trigger the event task.
- Once the axis reaches the watch position and triggers the event task, you have to re-arm the axis for the next watch position.

This timing diagram shows the relationship between the watch position and the event task.



Checklist for an Axis Watch Task

The following is a checklist for an axis watch task:

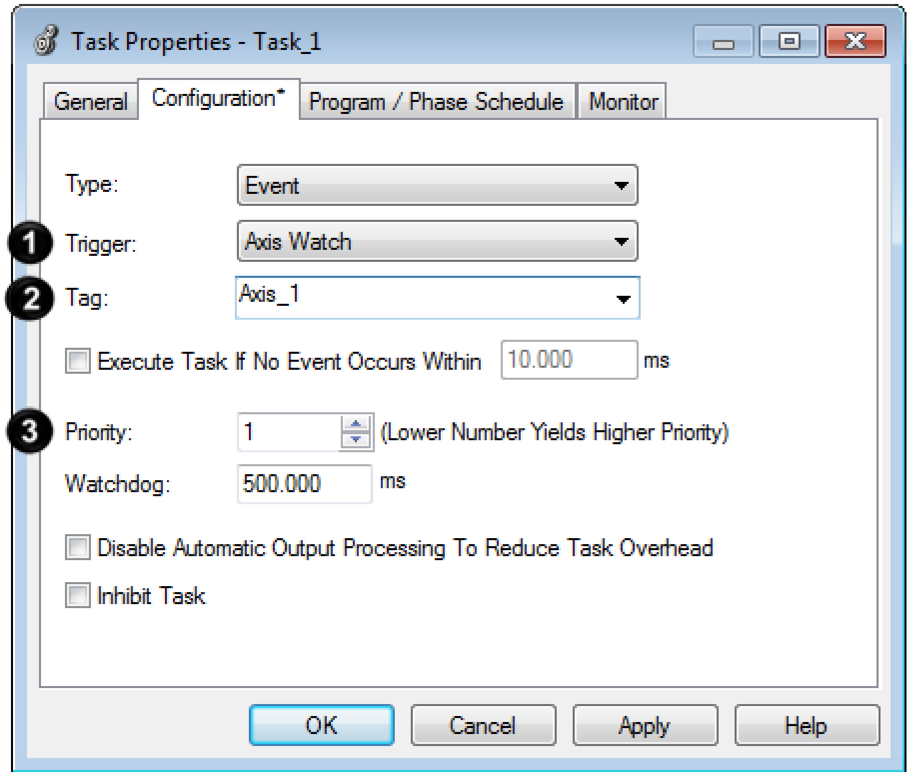
<table>
<tr><th colspan="3">For This</th><th colspan="2">Make Sure You</th></tr>
<tr><td rowspan="4">☐</td><td rowspan="4">1.</td><td rowspan="4">Watch position</td><td colspan="2">Use a MAW instruction to set up a watch position. This lets the axis trigger the event task when it reaches the watch position.
• Initially, arm the axis to detect the first watch position.
• When the axis reaches the watch position and triggers the event task, re-arm the axis for the next watch position.
• Re-arm the axis fast enough to detect each watch position.</td></tr>
<tr><td>If your normal logic is</td><td>Then</td></tr>
<tr><td>Fast enough to re-arm the axis between intervals of the watch position
(For example, normal logic always completes at least two scans between watch positions.)</td><td>Arm the axis within your normal logic, if desired.</td></tr>
<tr><td>Not fast enough to re-arm the axis</td><td>Arm the axis within the event task.</td></tr>
<tr><td>☐</td><td>2.</td><td>Task priority</td><td colspan="2">Configure the event task as the highest priority task.
If a periodic task has a higher priority, the event task may have to wait until the periodic task is finished.</td></tr>
<tr><td>☐</td><td>3.</td><td>Number of event tasks</td><td colspan="2">Limit the number of event tasks.
Each additional task reduces the processing time that is available for other tasks. This could cause an overlap.</td></tr>
<tr><td>☐</td><td>4.</td><td>Automatic output processing</td><td colspan="2">For an event task, you can typically disable automatic output processing (default). This reduces the elapsed time of the task.</td></tr>
</table>

Example - Axis Watch Trigger

At the labeling station of a bottling line, you want to check the position of the label on the bottle.

- When the axis reaches the position that is defined as the watch point, check the label and perform any required adjustment.

- Due to the speed of the line, you have to arm axis for the watch position within the event task.



1	Let the watch position...
2	...for the axis named Axis_1 trigger the event task..
3	This event task interrupts all other tasks.

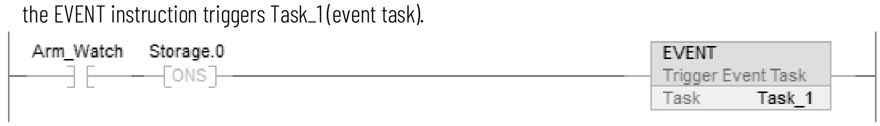
The following logic arms and re-arms the axis for the watch position.

Continuous task

If Arm_Watch = 1 (system is ready to set up a watch position) then

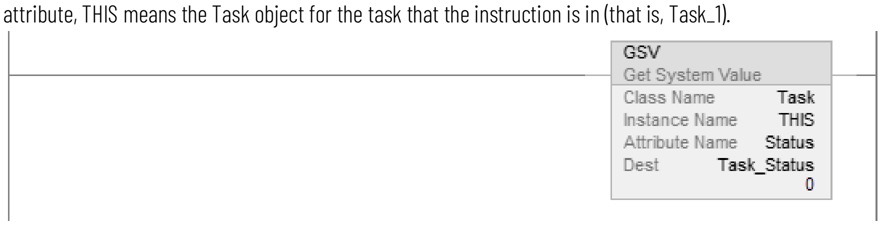
the ONS instruction limits the EVENT instruction to one scan.

the EVENT instruction triggers Task_1 (event task).



Task_1 (event task)

The GSV instruction sets Task_Status (DINT tag) = Status attribute for the event task. In the Instance Name attribute, THIS means the Task object for the task that the instruction is in (that is, Task_1).



If Task_Status.0 = 1 then an EVENT instruction triggered the event task. In the continuous task, the EVENT runs to set up the watch position for the first time.

The JMP instruction causes the controller to jump to the Arm LBL instruction. This skips all the logic of the routine except the rung that arms the axis for the watch position (MAW instruction).

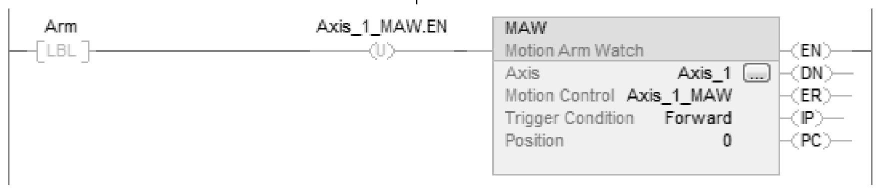
- Other logic

The MAW instruction runs each time the task runs and arms Axis_1 for the watch position.

The OTU instruction sets the EN bit of the MAW instruction = 0.

- The MAW instruction is a transitional instruction.
- To run the MAW instruction, its rung-condition-in must go from false to true.
- By first clearing the EN bit, the instruction responds as if its rung-condition-in changed from false to true.

The MAW instruction arms the axis for the watch position.

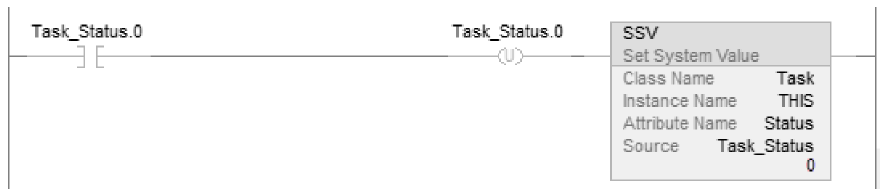


The controller does not clear the bits of the Status attribute once they are set. To use a bit for new status information, you must manually clear the bit.

If Task_Status.0 = 1 then clear that bit.

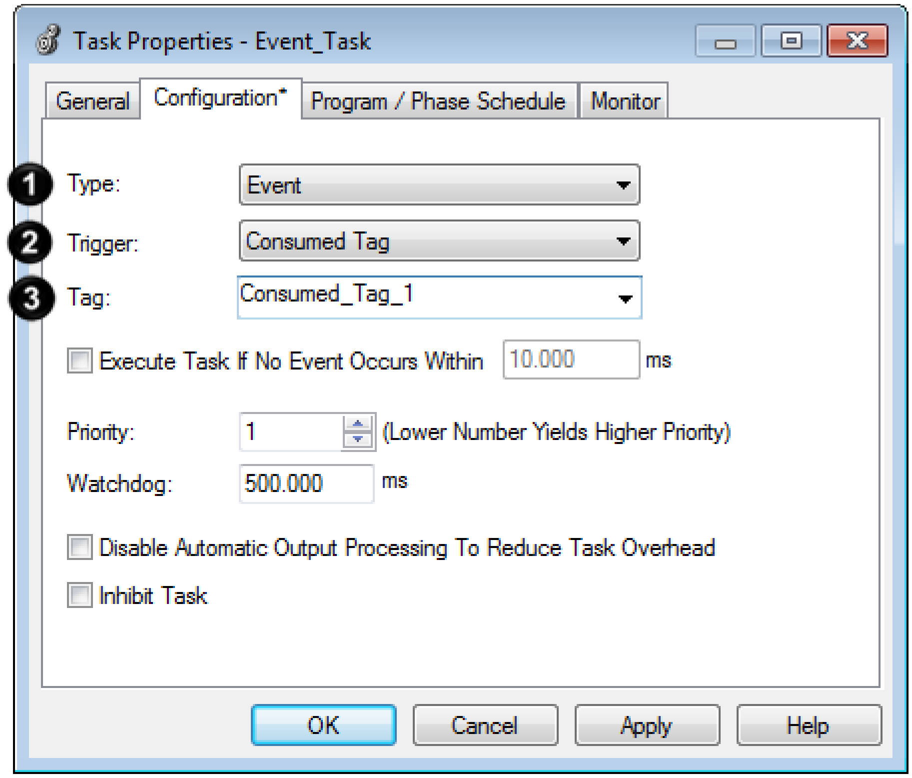
The OTU instruction sets Task_Status.0 = 0.

The SSV instruction sets the Status attribute of THIS task (Task_1) = Task_Status. This includes the cleared bit.



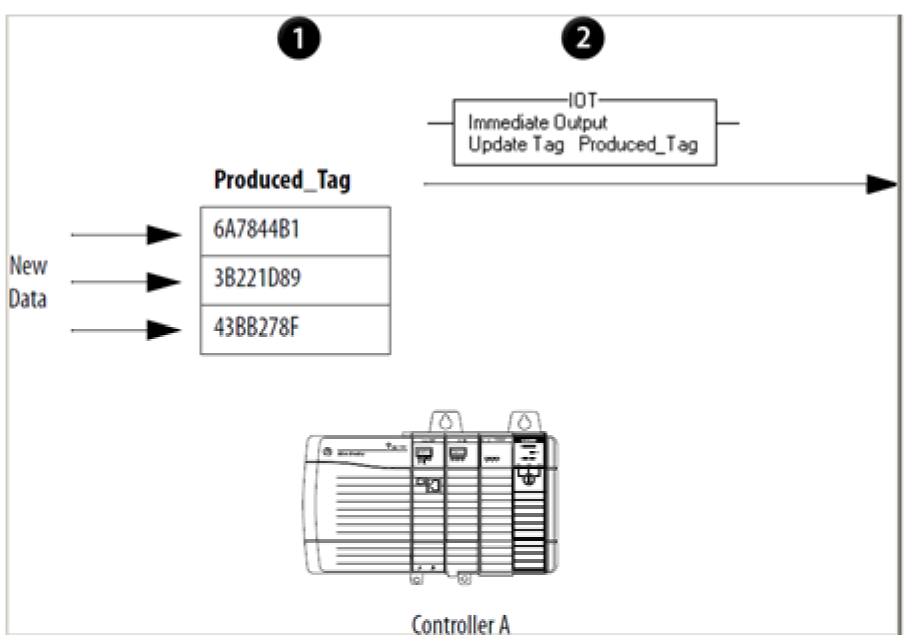
Consumed Tag Trigger

To trigger an event task based on data from a consumed tag, use the **Consumed Tag** trigger.



1	Let an event trigger this task.
2	Let a consumed tag trigger the task.
3	Let this consumed tag trigger the task.

A produced/consumed tag relationship can pass an event trigger along with data to a consumer controller. Typically, you use an Immediate Output (IOT) instruction to send the event trigger to the consumer controller.



Description	
1	In Controller A, logic updates the values of a produced tag.
2	Once the update is complete, the Controller A runs an IOT instruction to send the data and an event trigger to Controller B.
3	Controller B consumes the new data.
4	After Controller B updates the consumed tag, it runs the event task.

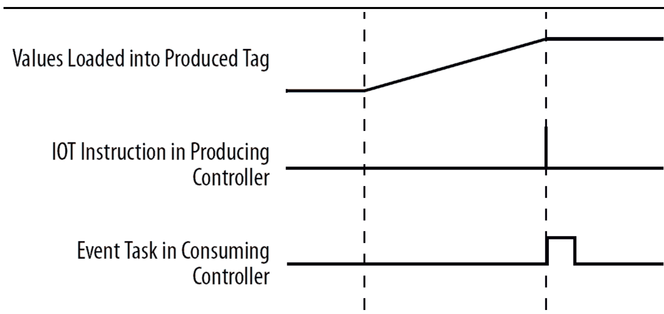
The type of network between the controllers determines when the consuming controller receives the new data and event trigger through the IOT instruction.

The following table lists the times when the consuming device receives the new data and event trigger.

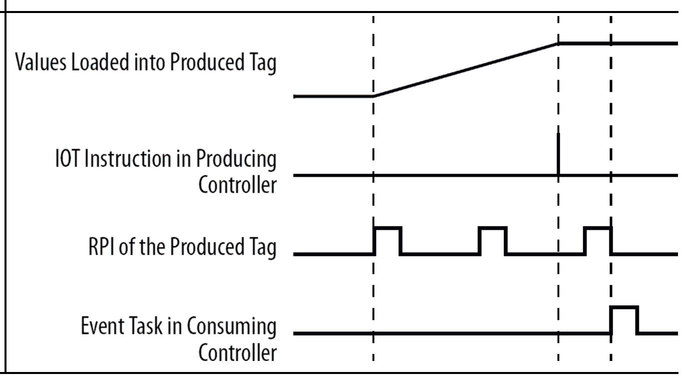
With this controller	Over this network	The consuming device receives the data and event trigger
ControlLogix	Backplane	Immediately
	EtherNet/IP network	Immediately
	ControlNet network	Within the actual packet interval (API) of the consumed tag (connection)
SoftLogix5800	You can produce and consume tags only over a ControlNet network	Within the actual packet interval (API) of the consumed tag (connection)

The following diagrams compare the receipt of data via an IOT instruction over EtherNet/IP and ControlNet networks.

EtherNet/IP Network (ControlLogix controller)

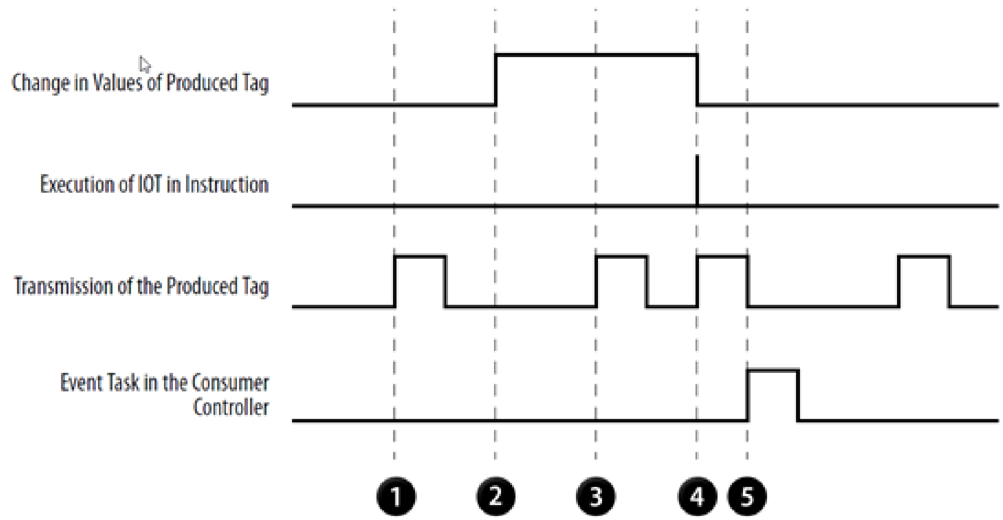


ControlNet Network



Maintain the Integrity of Data

An event task with a consumed tag trigger provides a simple mechanism to pass data to a controller and make sure that the controller doesn't use the data while the data is changing.

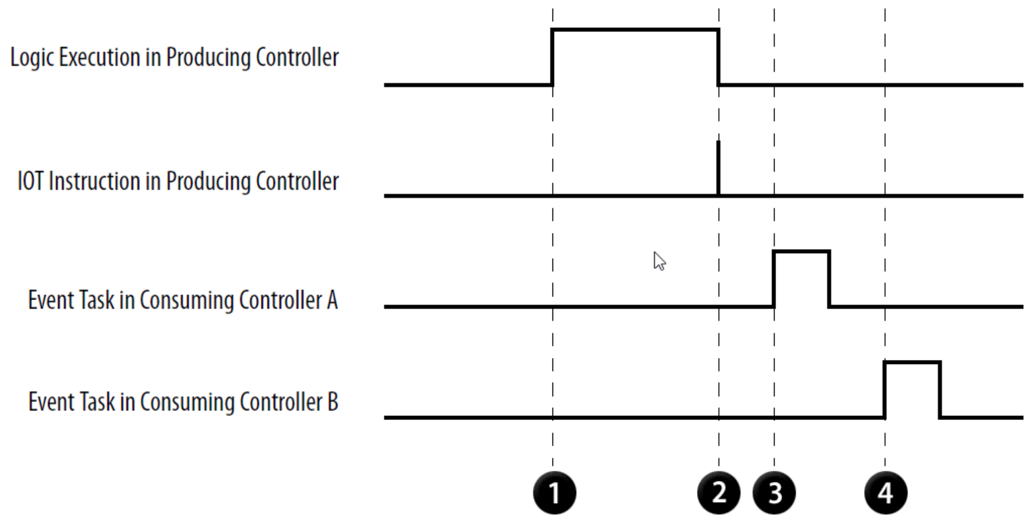


Description	
1	RPI occurs for the produced tag. The produced tag transfers old data to the consuming controller.
2	The producer controller starts to update the values of the produced tag.
3	RPI occurs again for the produced tag. The produced tag transfers a mix of old and new data to the consuming controller.
4	The producer controller finishes updating the values of the produced tag. The producer controller runs an Immediate Output (IOT) instruction. The produced tag immediately transfers all the new data to the consuming controller.
5	When the consumer controller receives all the data, it runs its event task.

Although the producing controller runs the IOT instruction immediately after it loads new data, the event task is not triggered (in the consuming controller) until the consuming controller has received all the new data. This verifies that the controller operates on a complete packet of new data.

Synchronize multiple controllers

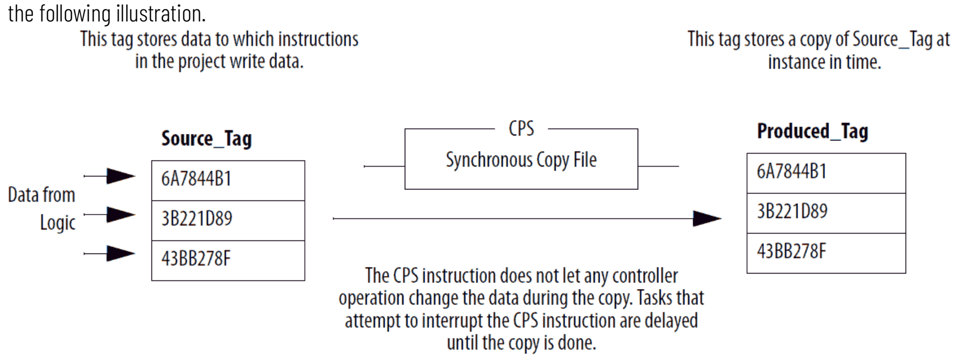
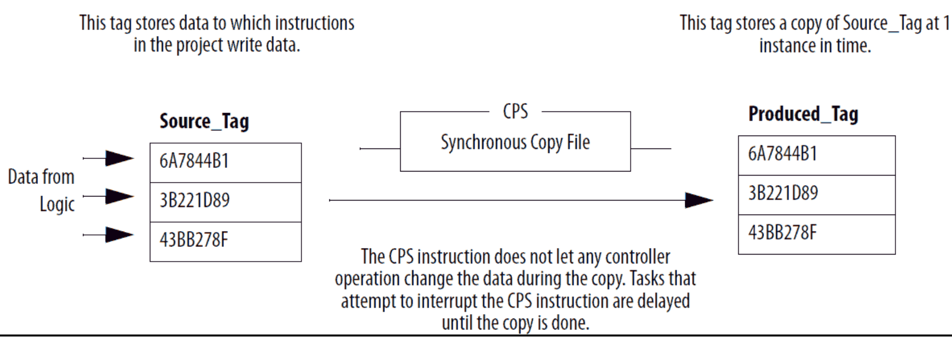
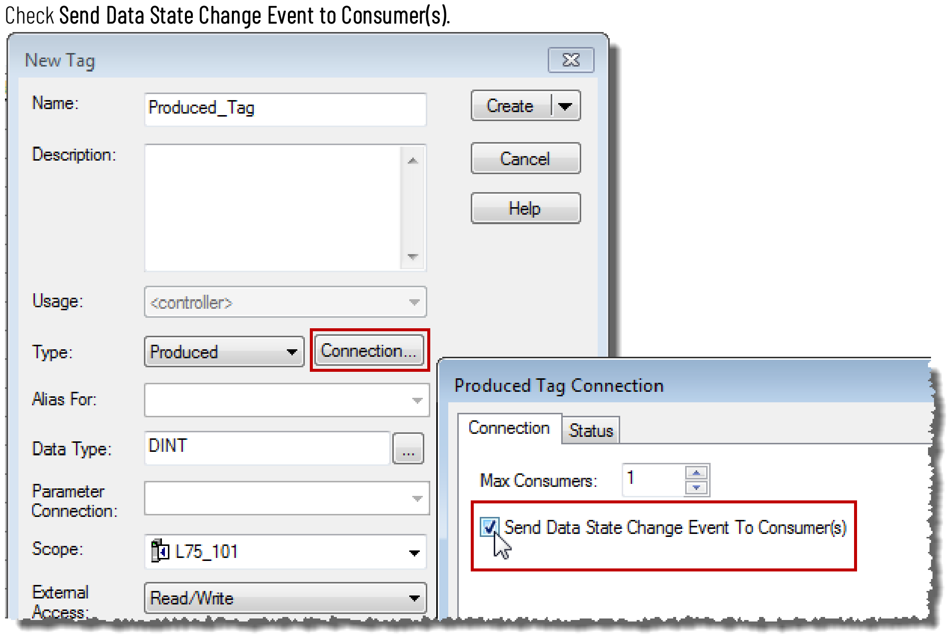
Use the produced/consumed tag relationship to synchronize controllers. In this case, the produced/consumed tag serves only as a triggering mechanism..



	Description
1	The first controller runs an action with which other controllers need to stay synchronized.
2	When the action is done, the controller runs an IOT instruction. The IOT instruction uses a produced tag as its target.
3	When controller A receives the produced tag, it runs its event task.
4	When controller B receives the produced tag, it runs its event task.

Checklist for the Producer Controller

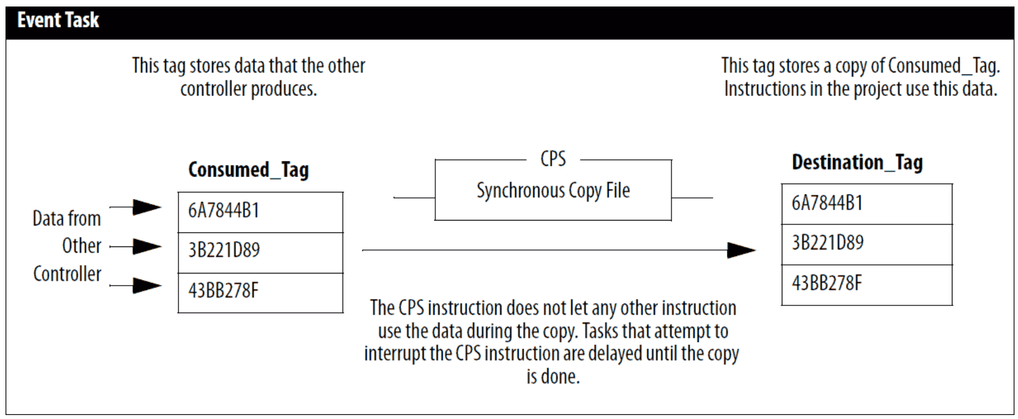
The following is a checklist for the producer controller:

	For This	Make Sure You
☐	1. Buffer of data	If you want to send a complete image of data at one instance in time, then produce a copy of the data, as shown in the following illustration.  
☐	2. Produced tag properties	On the **New Tag** dialog box for the produced tag, click **Connection** to open the Produced Tag Connection dialog box. Check **Send Data State Change Event to Consumer(s)**.  If you leave this checkbox cleared (unchecked), the producing controller triggers the event task at the end of any task that automatically updates local outputs. In other words, the task scan triggers the event in addition to the IOT instruction.
☐	3. IOT instruction	Use an IOT instruction at the point in your logic where you want to trigger the event task. The IOT instruction overrides the RPI for the tag and immediately sends the event trigger and the data of the tag.

Checklist for the Consumer Controller

The following is a checklist for the consumer controller:

For This		Make Sure You
☐	1. Buffer of data	If you want to make sure that the controller does not use data from the consumed tag while the data is changing, use a copy of the consumed tag. Use the event task to copy the data, as shown in the Event Task diagram.
☐	2. Task priority	Configure the event task as the highest priority task. If a periodic task has a higher priority, the event task may have to wait until the periodic task is finished.
☐	3. Number of event tasks	Limit the number of event tasks. Each additional task reduces the processing time that is available for other tasks. This could cause an overlap.
☐	4. Automatic output processing	For an event task, you can typically disable automatic output processing (default). This reduces the elapsed time of the task.



Example - Producer Controller and Consumer Controller

As parts move along a production line, each station requires production specifications for the part at its station. To make sure that a station doesn't act on old data, an event task signals the arrival of new data for the next part.

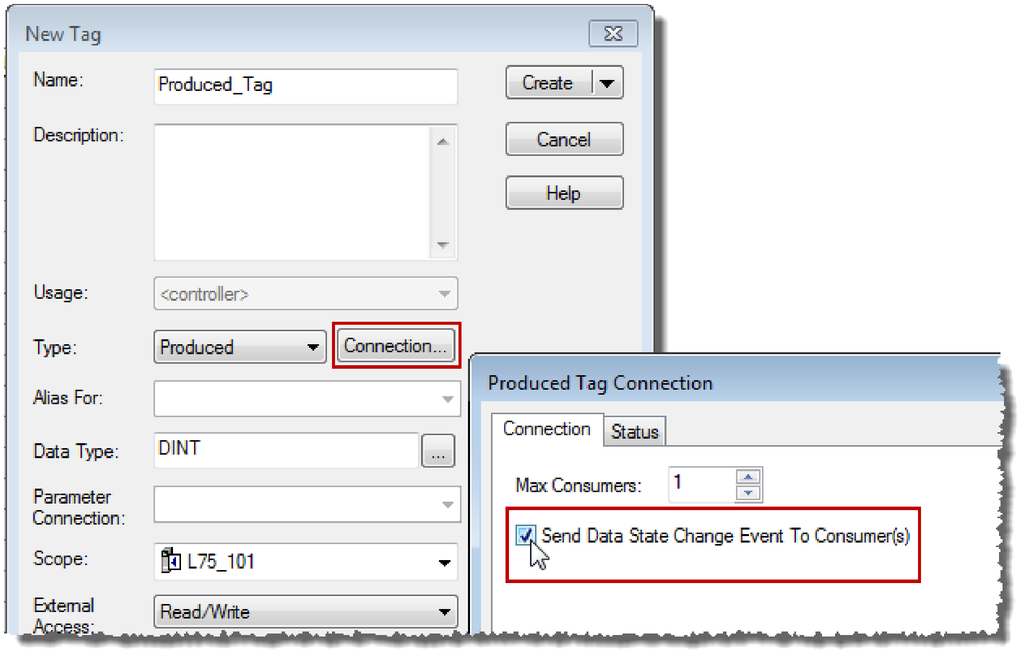
Producer Controller

This controller controls station 24 and produces data for the next station (station 25). To signal the transmission of new data, the controller uses the following elements:

- Produced_Tag
- Ladder logic

Produced Tag Properties

Produced_Tag is configured to update its event trigger through an IOT instruction.

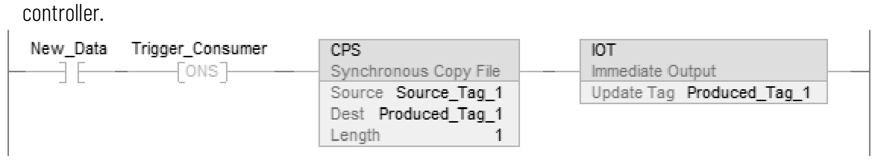


Ladder Logic

If New_Data = on, then this occurs for one scan.

The CPS instruction sets Produced_Tag_1 = Source_Tag_1.

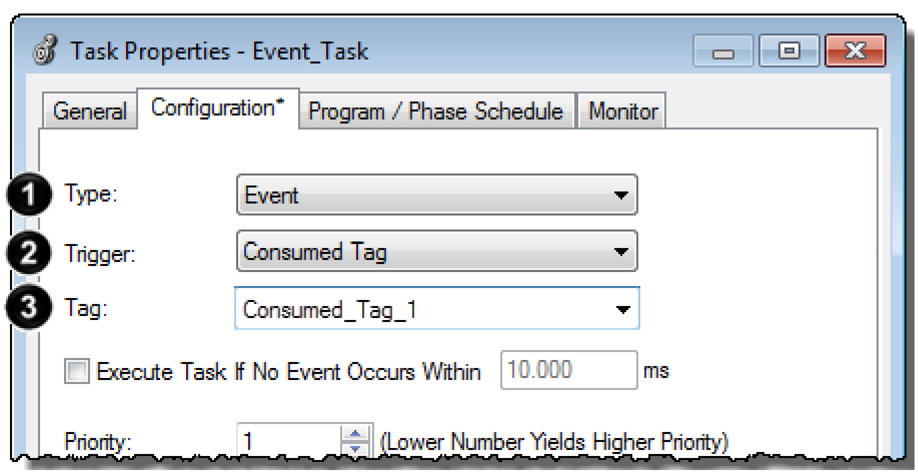
The IOT instruction updates Produced_Tag_1 and sends this update to the consuming controller (station 25). When the consuming controller receives this update, it triggers the associated event task in that controller.



Consumer Controller

The controller at station 25 uses the data produced by station 24. To determine when new data has arrived, the controller uses an event task.

Event Task Properties



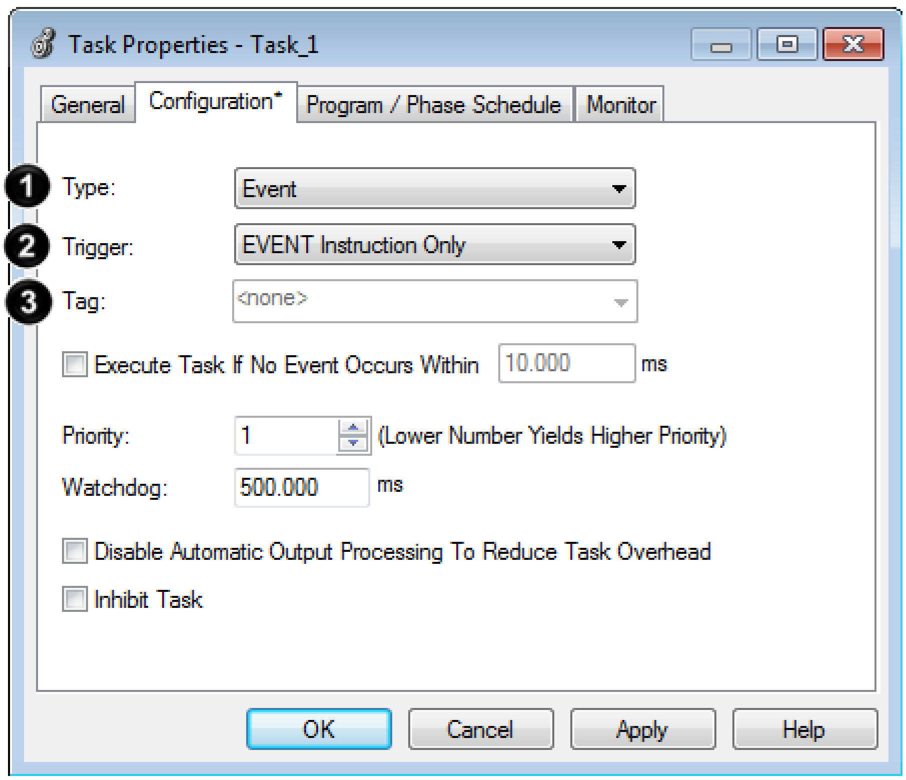
1	Let an event trigger this task.
2	Let a consumed tag trigger the task.
3	Let this consumed tag trigger the task.

Ladder Diagram in the Event Task

When the event task runs, the CPS instruction sets Destination_Tag_1 = Consumed_Tag_1 (the values from the producing controller). The remaining logic in this controller uses the values from Destination_Tag_1.

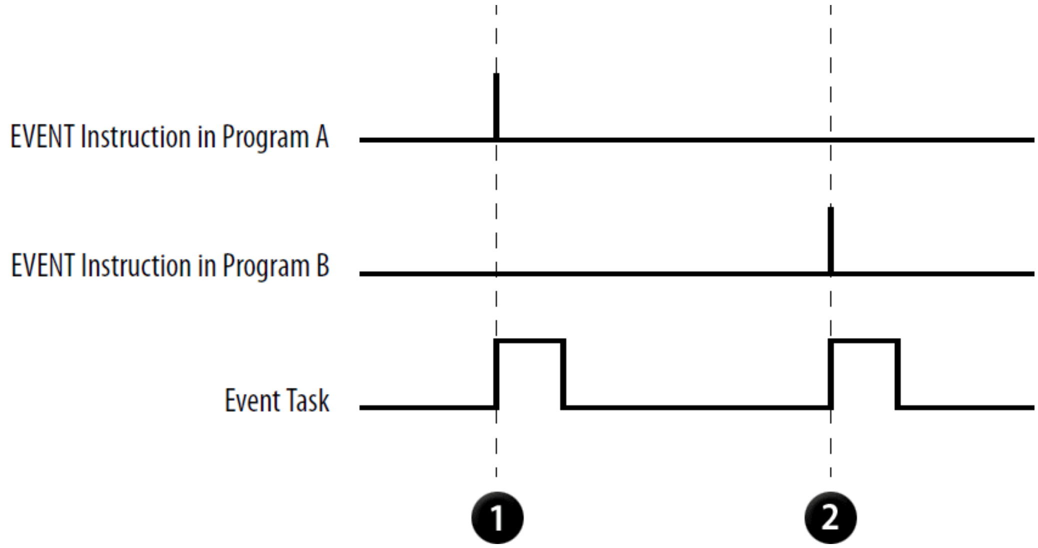
EVENT Instruction Trigger

To trigger an event task based on conditions in your logic, use the **EVENT Instruction Only** trigger.



1	Let an event trigger this task.
2	Let an EVENT instruction trigger the task.
3	No tag is required.

The EVENT Instruction Only trigger requires that you use a Trigger Event Task (EVENT) instruction to trigger the task. You can use an EVENT instruction from multiple points in your project. Each time the instruction runs, it triggers the specified event task.



	Description
1	Program A runs an EVENT instruction. The event task that is specified by the EVENT instruction runs one time.
2	Program B runs an EVENT instruction. The event task that is specified by the EVENT instruction runs one time.

Programmatically Determine if EVENT Instruction Triggered Task

To determine if an EVENT instruction triggered an event task, use a Get System Value (GSV) instruction to monitor the Status attribute of the task.

Table 8 - Status Attribute of the TASK Object

<table>
<tr><th>Attribute</th><th>Data Type</th><th>Instruction</th><th colspan="2">Description</th></tr>
<tr><td rowspan="5">Status</td><td rowspan="5">DINT</td><td rowspan="5">GSV
SSV</td><td colspan="2">Provides status information about the task. Once the controller sets a bit, you must manually clear the bit to determine if another fault of that type occurred.</td></tr>
<tr><td>To determine if</td><td>Examine this bit</td></tr>
<tr><td>An EVENT instruction triggered the task (event task only).</td><td>0</td></tr>
<tr><td>A timeout triggered the task (event task only).</td><td>1</td></tr>
<tr><td>An overlap occurred for this task.</td><td>2</td></tr>
</table>

The controller does not clear the bits of the Status attribute once they are set.

- To use a bit for new status information, you must manually clear the bit.
- Use a Set System Value (SSV) instruction to set the attribute to a different value.

Checklist for an EVENT Instruction Task

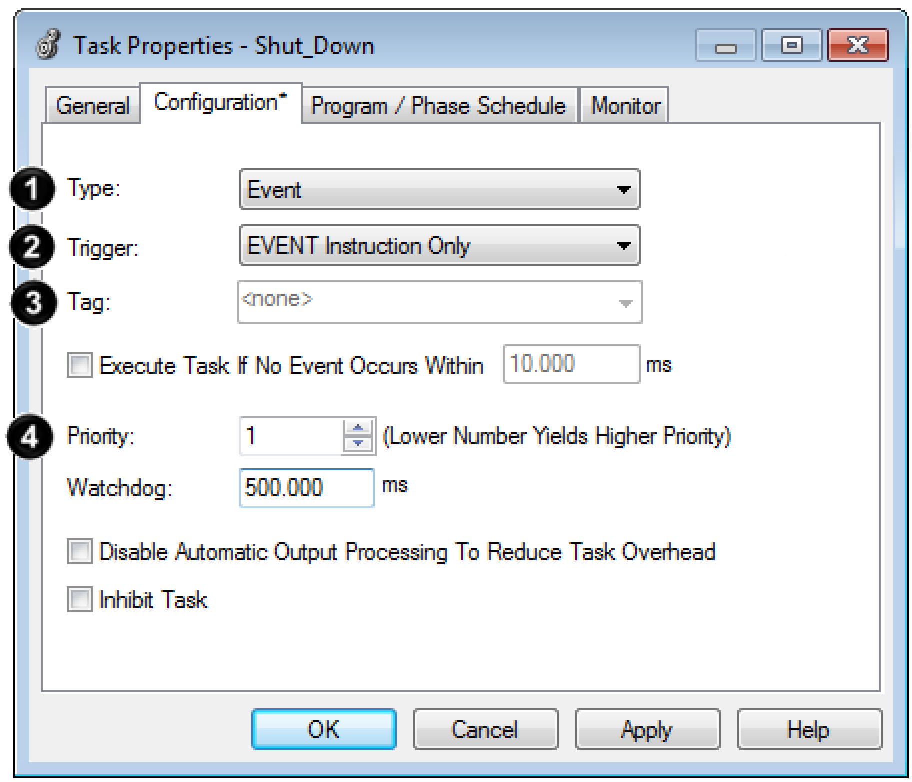
The following is checklist for an EVENT instruction task:

For This		Make Sure You
☐	1. EVENT instruction	Use a **Trigger Event Task (EVENT)** instruction at each point in your logic that you want to trigger the event task.
☐	2. Task priority	Configure the event task as the highest priority task. If a periodic task has a higher priority, the event task may have to wait until the periodic task is finished.
☐	3. Number of event tasks	Limit the number of event tasks. Each additional task reduces the processing time that is available for other tasks. This could cause an overlap.
☐	4. Automatic output processing	For an event task, you can typically disable automatic output processing (default). This reduces the elapsed time of the task.

Example - EVENT Instruction Trigger

A controller uses multiple programs except for a common shut down procedure. Each program uses a program-scoped tag named Shut_Down_Line that turns on if the program detects a condition that requires a shut down.

Event Task Properties



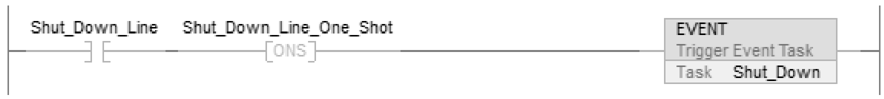
1	Let an event trigger this task.
2	Let an EVENT instruction trigger the task.
3	No tag is required.
4	Interrupt all other tasks.

Ladder Diagram in Program_A

If Shut_Down_Line = on (conditions require a shut down) then
run the Shut_Down task one time.

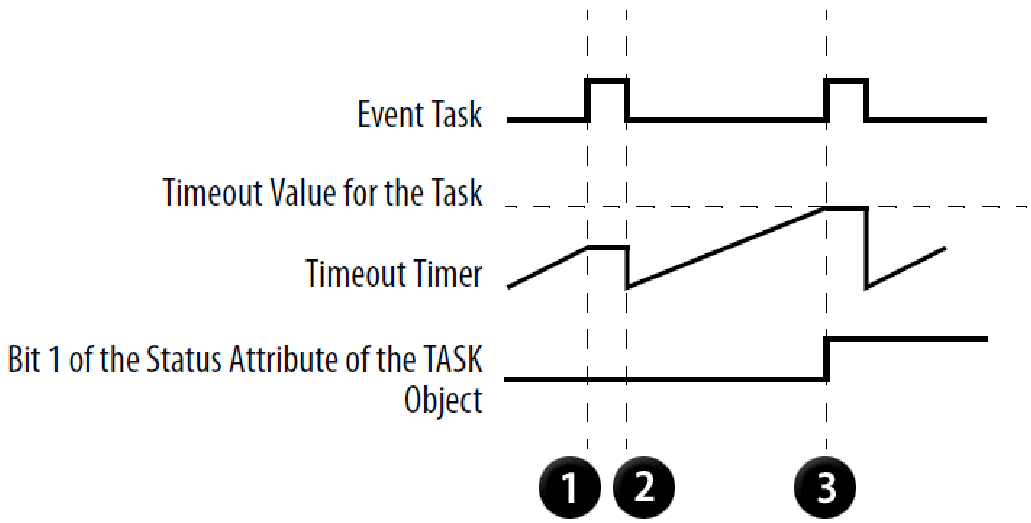
Ladder Diagram in Program_B

If Shut_Down_Line = on (conditions require a shut down) then
run the Shut_Down task one time.



Define a Timeout Value for an Event Task

If you want your event task to run automatically, if the trigger fails to occur within a certain time, assign a timeout value to the task. When the event task is finished, the timeout timer begins to increment. If the timer reaches its preset value before the event task is triggered, the event task runs automatically.



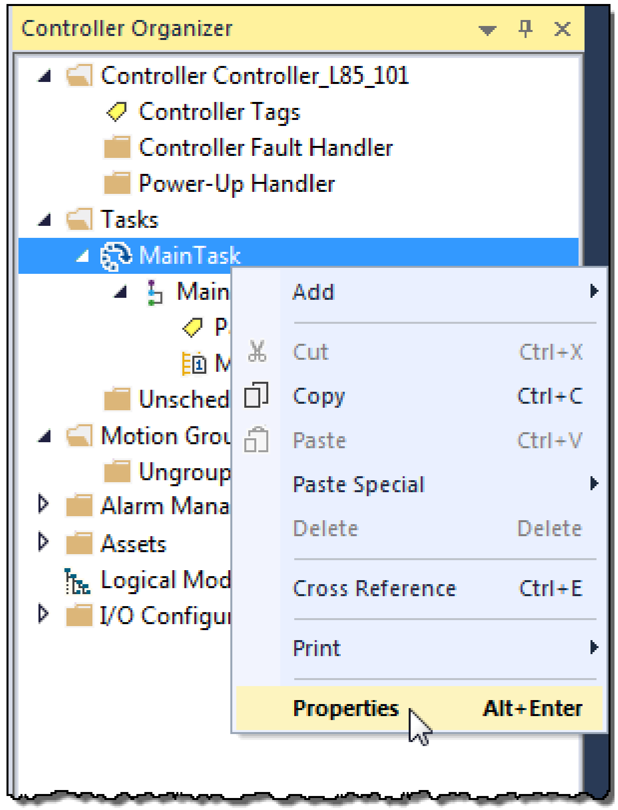
Description	
1	Event task runs. Timeout time stops incrementing.
2	Event task is done. Timeout timer resets and begins incrementing.
3	Timeout timer reaches the timeout value. Event task automatically runs. In the Status attribute of the TASK object, bit 1 turns on.

Assign a Timeout Value to an Event Task

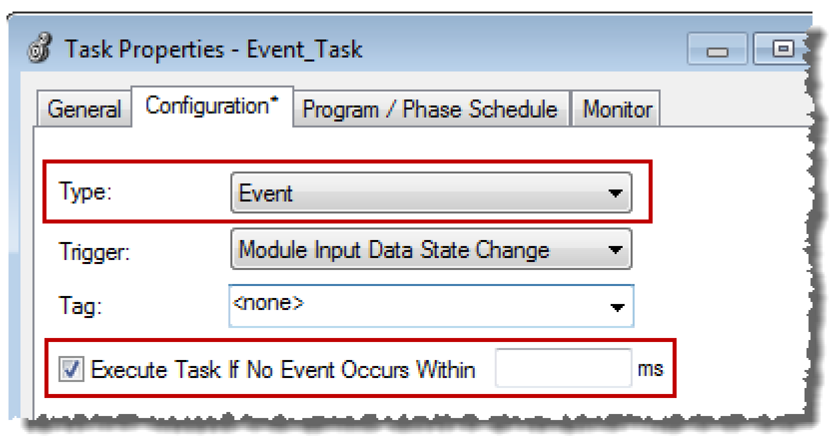
Follow these steps to assign a timeout value to an event task.

To assign a timeout value to an event task

1. In the **Controller Organizer**, right-click **Main Task** and choose **Properties**.



2. On the **Task Properties** dialog box, select the **Configuration** tab.



3. From the **Type** menu, choose **Event**.
4. Select **Execute Task If No Event Occurs Within**.
5. Type the timeout value.
6. Select **OK**.

Programmatically Configure a Timeout

To programmatically configure a timeout, use a Get System Value (GSV) instruction to access the attributes of the task. The following table lists the status attribute for the TASK object.

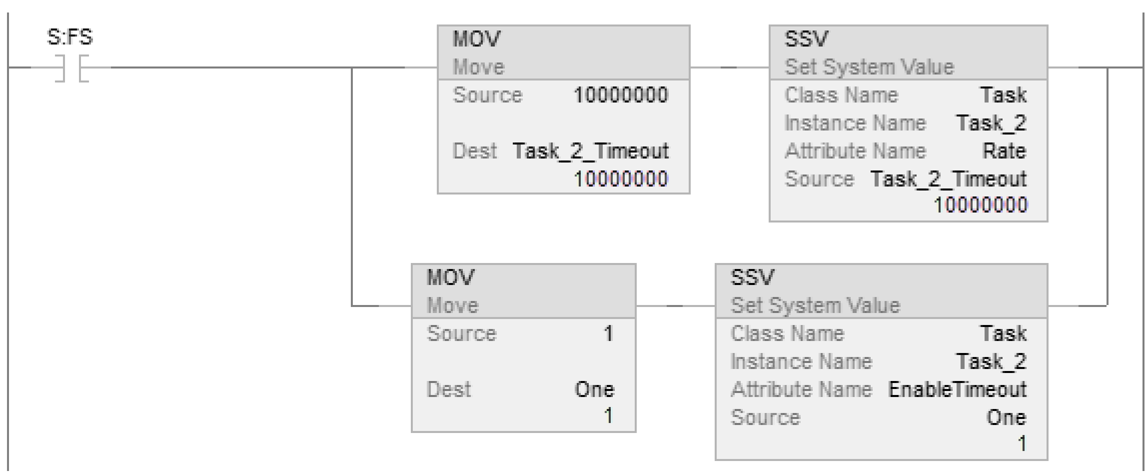
Attribute	Data Type	Instruction	Description	
Rate	DINT	GSV SSV	**If the task type is**	**Then the Rate attribute specifies the**
			Periodic	Period for the task. Time is in microseconds.
			Event	The timeout value for the task. Time is in microseconds.
EnableTimeOut	DINT	GSV SSV	Enables or disables the timeout Function of an event task.	
			To	**Set the attribute to**
			Disable the timeout function	0 (default)
			Enable the timeout function	1 (or any non-zero value)

Example

To make sure that a timeout value is always defined and enabled for an event task, the logic configures the timeout when the controller enters Run mode.

If S:FS = 1 (first scan) then set the timeout value for Task_2 and enable the timeout function.

1. The first MOV instruction sets Task_2_Timeout = 10000000 µs (DINT value). Then the SSV instruction sets the Rate attribute for Task_2 = Task_2_Timeout. This configures the timeout value for the task.
2. The second MOV instruction sets One = 1 (DINT value). Then the SSV instruction sets the EnableTimeout attribute for Task_2 = One. This enables the timeout function for the task.



Programmatically determine if a timeout occurs

To determine if an event task ran due to a timeout, use a Get System Value (GSV) instruction to monitor the Status attribute of the task. The following table lists the Status attribute for the TASK object.

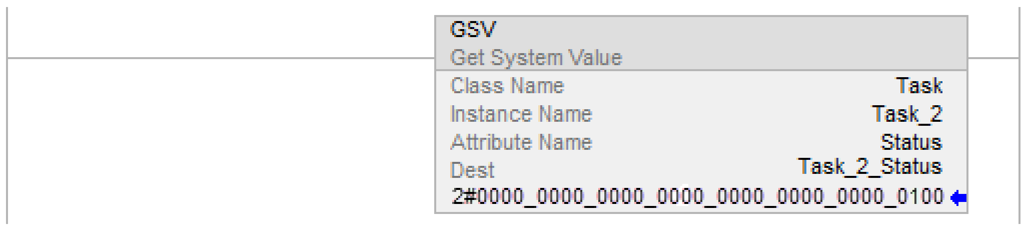
Attribute	Data Type	Instruction	Description	
Status	DINT	GSV SSV	Provides status information about the task. Once the controller sets a bit, you must manually clear the bit to determine if another fault of that type occurred.	
			To determine if	**Examine this bit**

Attribute	Data Type	Instruction	Description	
			An EVENT instruction triggered the task (event task only).	0
			A timeout triggered the task (event task only).	1
			An overlap occurred for this task.	2

Example

If a timeout occurs for the event task, communication with the triggering device might have failed. This requires the process to shut down. To shut down the controller, the event task calls the fault routine for the program and supplies a user-defined fault code (999 in this example).

1. The GSV instruction sets Task_2_Status = Status attribute for Task_2 (DINT value).



2. If Task_2_Status.1 = 1, then a timeout occurred so shut down the controller and set the major fault code to 999.

   The JSR instruction calls the fault routine for the program. This produces a major fault.
   The major fault code = 999 (value of the input parameter of 999).

3. If Condition_1 = 1, then clear the bits of the Status attribute for Task_2.

   The SSV instruction sets the Status attribute of Task_2 = Zero. Zero is a DINT tag with a value of 0.

Index

Rockwell Automation support

Use these resources to access support information.

Technical Support Center	Find help with how-to videos, FAQs, chat, user forums, and product notification updates.	rok.auto/support
Knowledgebase	Access Knowledgebase articles.	rok.auto/knowledgebase
Local Technical Support Phone Numbers	Locate the telephone number for your country.	rok.auto/phonesupport
Literature Library	Find installation instructions, manuals, brochures, and technical data publications.	rok.auto/literature
Product Compatibility and Download Center (PCDC)	Get help determining how products interact, check features and capabilities, and find associated firmware.	rok.auto/pcdc

Documentation feedback

Your comments help us serve your documentation needs better. If you have any suggestions on how to improve our content, complete the form at rok.auto/docfeedback.

Waste Electrical and Electronic Equipment (WEEE)

At the end of life, this equipment should be collected separately from any unsorted municipal waste.

Rockwell Automation maintains current product environmental information on its website at rok.auto/pec.

Rockwell Otomayson Ticaret A.Ş. Kar Plaza İş Merkezi E Blok Kat:6 34752, İçerenkÖy, İstanbul, Tel: +90 (216) 5698400 EEE YÖnetmeliğine Uygundur

AMERICAS: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
EUROPE/MIDDLE EAST/AFRICA: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640
ASIA PACIFIC: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Rockwell Automation Publication 1756-PM005K-EN-P - November 2022
Supercedes Publication 1756-PM005J-EN-P - March 2022