Logix5000 Controllers Tasks, Programs, and Routines

Allen-Bradley

Catalog Numbers 1756 ControlLogix, 1756 GuardLogix,
1768 Compact GuardLogix, 1768 CompactLogix,
1769 CompactLogix, 1789 SoftLogix, PowerFlex with DriveLogix

Programming Manual

Allen-Bradley • Rockwell Software

Rockwell Automation

Important User Information

Solid state equipment has operational characteristics differing from those of electromechanical equipment. Safety Guidelines for the Application, Installation and Maintenance of Solid State Controls (publication SGI-1.1 available from your local Rockwell Automation sales office or online at http://www.rockwellautomation.com/literature/) describes some important differences between solid state equipment and hard-wired electromechanical devices. Because of this difference, and also because of the wide variety of uses for solid state equipment, all persons responsible for applying this equipment must satisfy themselves that each intended application of this equipment is acceptable.

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

Introduction

The release of this document contains new information.

New Information

New information is marked by change bars in the side column, as shown to the right.

Section	Changes
Chapter 1	For RSLogix 5000 software version 16 and later, the system-overhead time slice requires at least 1 ms of execution time for a continuous task.
Chapter 2	Descriptions and procedures for event tasks have been consolidated in this new chapter.

Notes:

Table of Contents

Index

Purpose of This Manual

This manual details how to set up controller tasks along with the programs and routines for the proper execution of these tasks. This manual is one of a set of related manuals that show common procedures for programming and operating Logix5000 controllers.

For a complete list of common procedures manuals, see the Logix5000 Controllers Common Procedures Programming Manual, publication 1756-PM001.

The term Logix5000 controller refers to any controller that is based on the Logix5000 operating system, such as:

- CompactLogix controllers.
- ControlLogix controllers.
- GuardLogix controllers.
- DriveLogix controllers.
- FlexLogix controllers.
- SoftLogix5800 controllers.

Notes:

Manage Tasks

Introduction

The default RSLogix 5000 project provides a single task for all your logic. While this is sufficient for many applications, some situations may require more than one task.

Select Controller Tasks

A Logix5000 controller supports multiple tasks to schedule and prioritize the execution of your programs based on specific criteria. This balances the processing time of the controller.

- The controller executes only one task at one time.
- A different task can interrupt a task that is executing and take control.
- In any given task, only one program executes at one time.

A Logix5000 controller supports three types of tasks.

If you want to execute a section of your logic	Then use this type of task	Description
All of 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 execute 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 (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. Whenever the time for the periodic task expires, the periodic task: • interrupts any lower priority tasks. • executes 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.

If you want to execute a section of your logic	Then use this type of task	Description
Immediately when an event occurs	Event Task	An event task performs a function only when a specific event (trigger) occurs. Whenever the trigger for the event task occurs, the event task: • interrupts any lower priority tasks. • executes 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 Logix5000 controllers do not support all triggers.

Here are some 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 execute 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 executes, 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 SoftLogix5800 GuardLogix	32	Only one task can be continuous.
CompactLogix		
1769-L2*x* 1769-L31 1769-L32*x* 1769-L35*x*	3 4 6 8	
1768-L43 1768-L45	16 30	
DriveLogix	8	

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.
- other tasks may experience overlaps. If a task is interrupted too frequently or too long, it may not complete its execution before it is triggered again.

Prioritize Periodic and Event Tasks

Although a project can contain multiple tasks, the controller executes only one task at a time. If a periodic or event task is triggered while another task is currently executing, the priority of each task tells the controller what to do.

The number of priority levels depends on the controller.

This Logix5000 controller	Has this many priority levels
CompactLogix	15
ControlLogix	15
DriveLogix	15
FlexLogix	15
SoftLogix5800	3

To assign a priority to a task, use these 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 executes each one for 1 ms.

Additional Considerations

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

Consideration	Description
Motion planner	The motion planner interrupts all user tasks, regardless of their priority. • The number of axes and coarse update period for the motion group effect 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.
I/O task	CompactLogix, 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. • executes at the fastest RPI you have scheduled for the system. • executes 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.

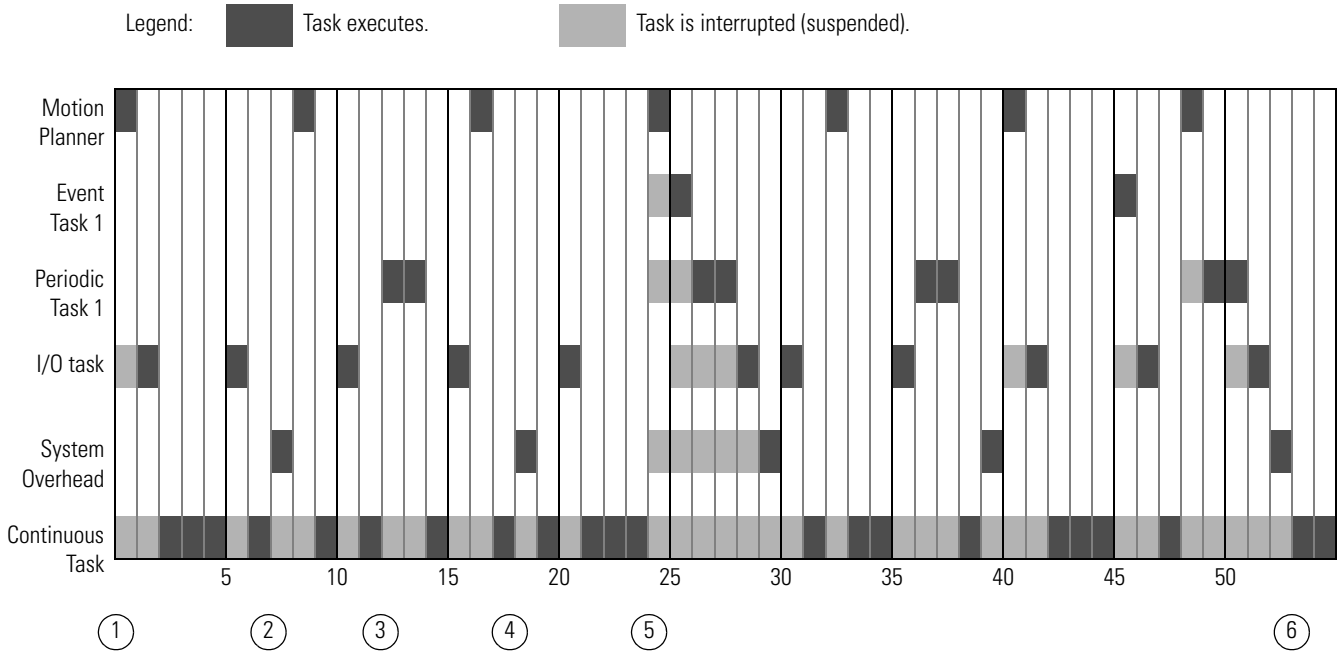
If you want a task to	Then assign one of these priorities
Interrupt or delay I/O processing	1…5
Share controller time with I/O processing	6
Let I/O processing interrupt or delay the task	7…15

Consideration	Description
System overhead	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.
Continuous task	You do not assign a priority to the continuous task. It always runs at the lowest priority. All other tasks interrupt the continuous task.

EXAMPLE

This example depicts the execution of a project with three user tasks.

Task	Priority	Period	Execution time	Duration
Motion planner	N/A	8 ms (course 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 ControlLogix and SoftLogix controllers. See page 12.	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 executes the motion planner and the I/O task (if one exists).
2	After executing 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 executing the continuous task again for 4 ms, the controller triggers the system overhead.
5	The triggers occurs for event task 1. Event task 1 waits until the motion planner is done. Lower priority tasks experience longer delays.
6	The continuous task automatically restarts.

RSLogix 5000 software includes a task monitor tool on the distribution CD. You can use this tool to analyze how tasks are executing.

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 these 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).

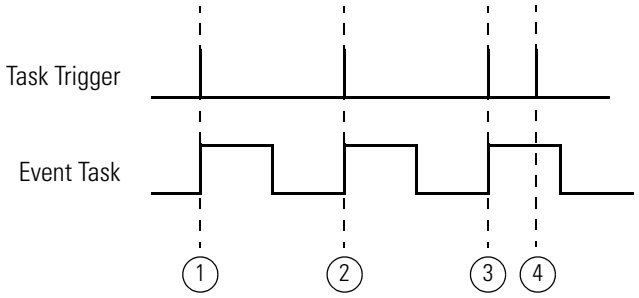
This generally leaves enough time for unscheduled communication.

- Adjust the period of the tasks as needed to get the best trade-off between executing 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 executing 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 executes.
2	Task trigger occurs. Task executes.
3	Task trigger occurs. Task executes.
4	Overlap occurs. Task is triggered while it is still executing. The trigger does not restart the task. The trigger is ignored.

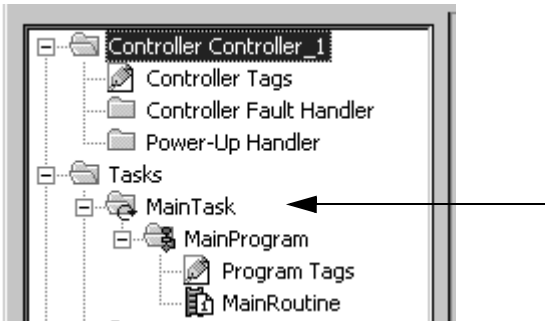
Each task requires enough time to finish before it is triggered again. Make sure that the scan time of the task is significantly less than the rate at which the trigger occurs. If an overlap occurs, reduce the frequency at which you trigger the task:

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

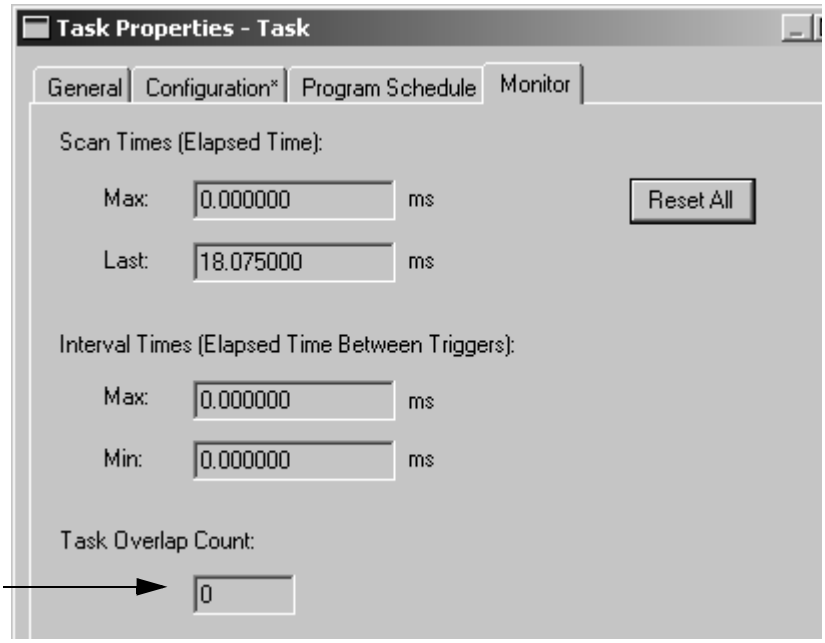
Follow these steps to manually see if overlaps are occurring for a task.

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



   The Task Properties dialog box appears.

2. Click the Monitor tab.



Number of overlaps since the counter was last reset.

3. Click OK.

Programmatically Check for Overlaps

When an overlap occurs, the controller:

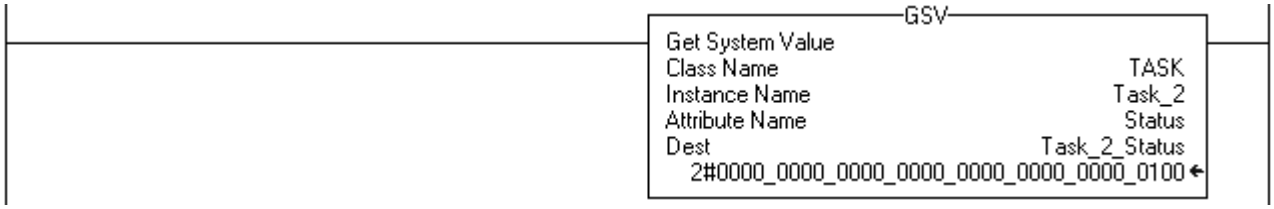
- logs a minor fault to the FAULTLOG object.
- stores overlap information in the TASK object for the task.

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 is not present or needs replacement. / 10
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.

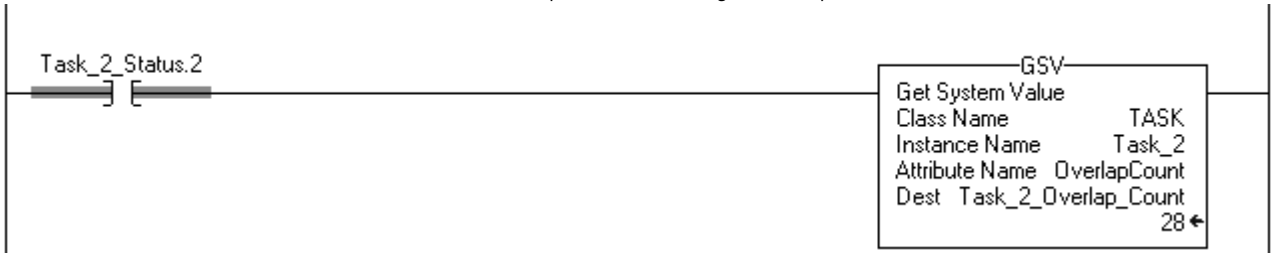
EXAMPLE Programmatically Check for Overlaps

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:

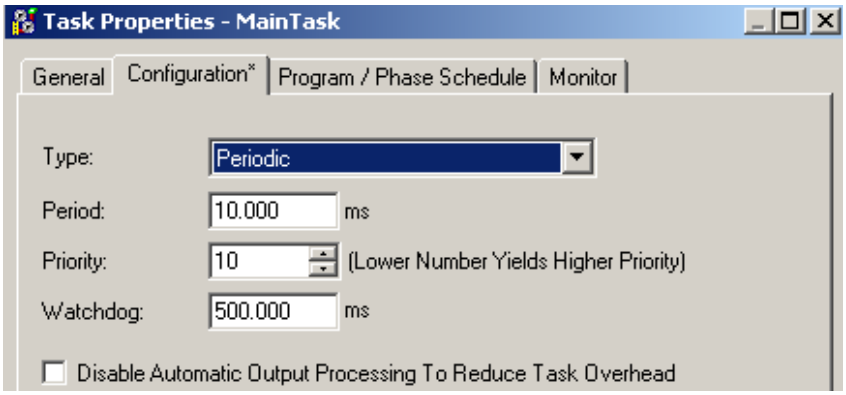
   The SSV instruction sets the Status attribute of Task_2 = Zero. Zero is a DINT tag with a value of 0.

Configure Output Processing for a Task

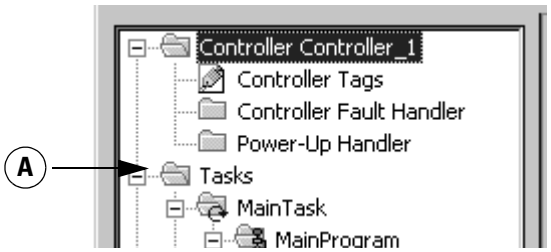
At the end of a task, the controller performs overhead operations (output processing) for the I/O modules in your system. While not the same as updating the modules, this output processing may effect the update of the I/O modules in your system.

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

Enable or disable the processing of outputs at the end of the task.

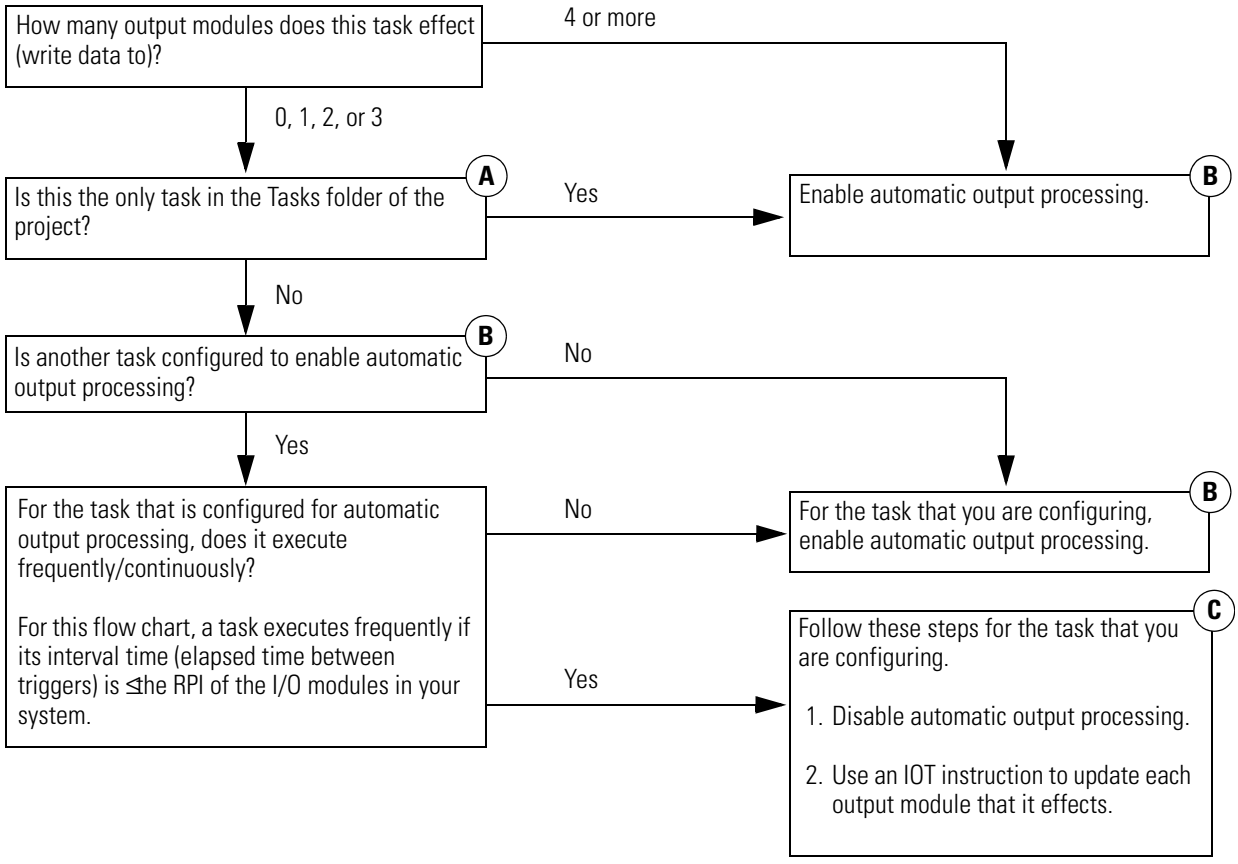


Choose how to configure output processing for a task.



B ☐ Disable Automatic Output Processing To Reduce Task Overhead

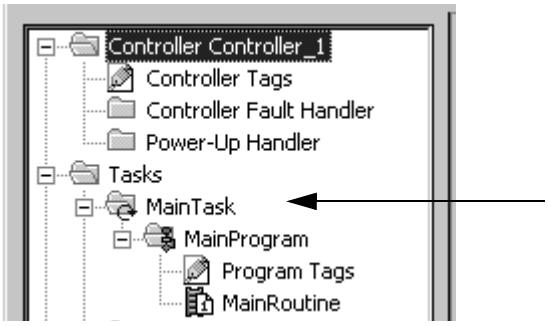
C ☑ Disable Automatic Output Processing To Reduce Task Overhead



Manually Configure Output Processing

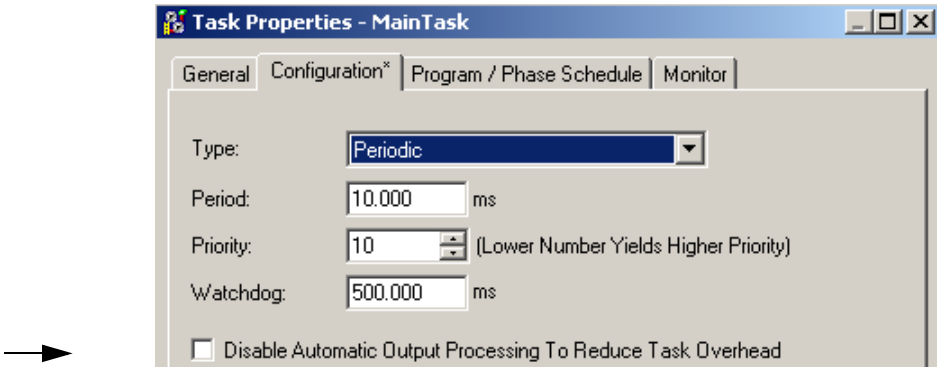
Follow these steps to manually configure output processing.

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



   The Task Properties dialog box appears.

2. 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

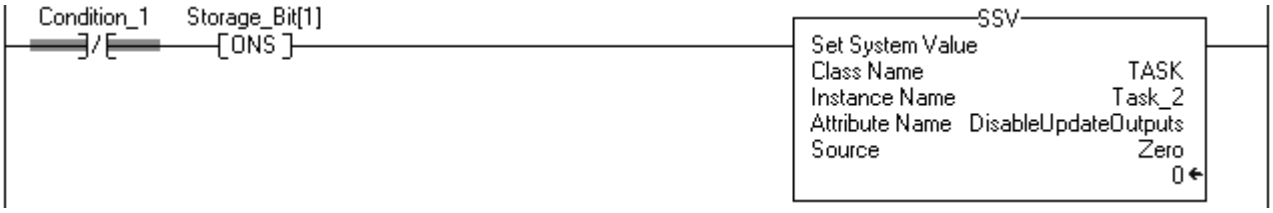
To write logic to configure output processing for a task, use a Set System Value (SSV) instruction. Access the attribute of the TASK object for the task.

If You Want to	Access This Attribute	Data Type	Instruction	Description	
Enable or disable the processing of outputs at the end of a task	DisableUpdateOutputs	DINT	GSV SSV	**To**	**Set the attribute to**
				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 Programmatically Configure Output Processing

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

1. The ONS instruction limits the true execution 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 execution.



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

1. The ONS instruction limits the true execution 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 execution.

Inhibit a Task

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

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

EXAMPLE

Inhibit a Task

During the commissioning of a system that uses several task, 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.

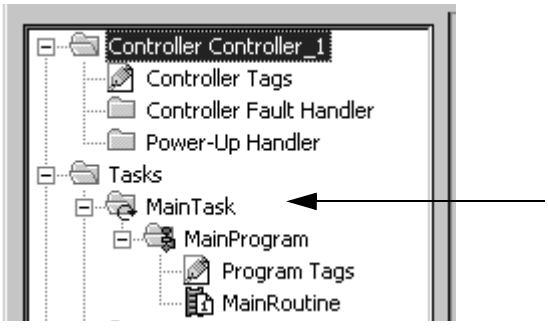
Continue this process until you have tested all your tasks.

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

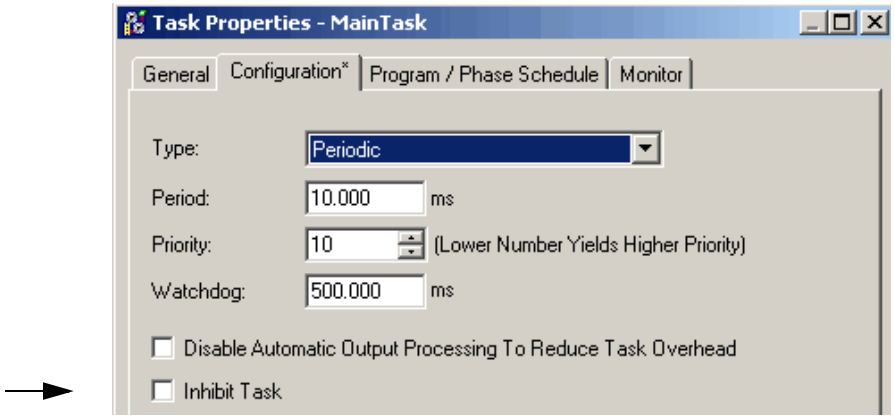
Follow these steps to manually inhibit or uninhibit the execution of a task.

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



The Task Properties dialog box appears.

2. Click the Configuration tab.



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

If You Want to	Then
Let the task execute when its trigger occurs	Clear 'Inhibit Task' (default).
Prevent the task from executing 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.

Attribute	Data Type	Instruction	Description	
InhibitTask	DINT	GSV SSV	Prevents the task from executing.	
			To	**Set the attribute to**
			Enable the task	0 (default)
			Inhibit (disable) the task	1 (or any non-zero value)

EXAMPLE Programmatically Inhibit or Uninhibit a Task

If Condition_1 = 0 then let Task_2 execute.

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

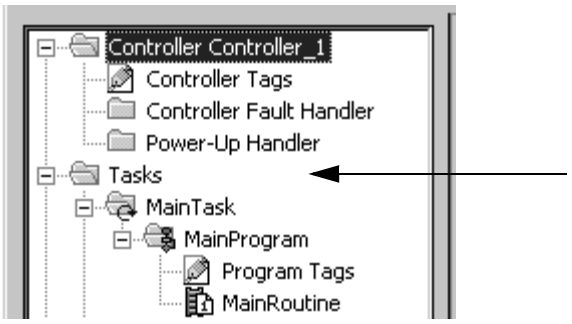
If Condition_1 = 1 then do not let Task_2 execute.

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

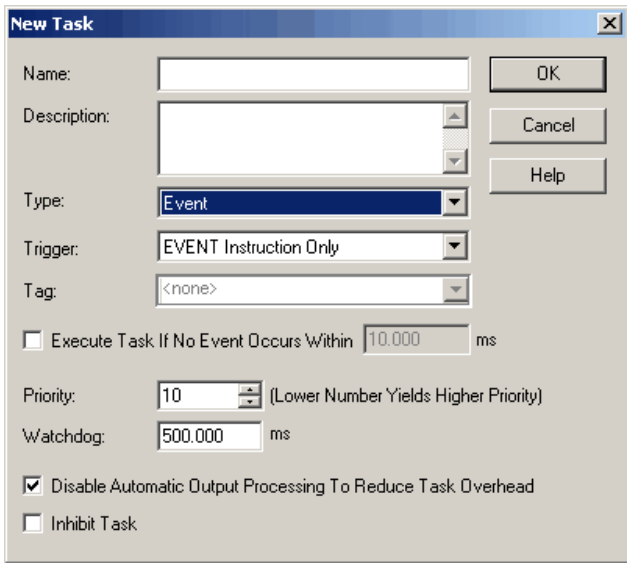
Create a Task

Follow these steps to create an event task.

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



The New Task dialog box appears.



2. Enter 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.

Topic	Description
Execute Task If No Event Occurs Within	Check the box and type a value that must elapse prior to a task occurring.
Priority	Enter the task priority value.
Watchdog	Type the watchdog time for the task.

3. Click OK.

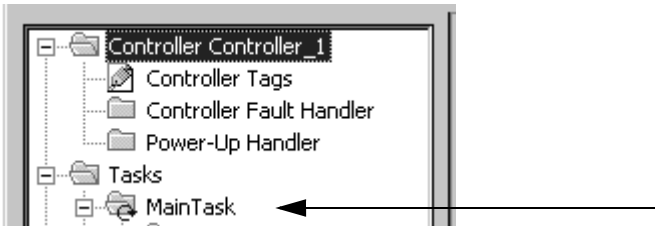
Create a Periodic Task

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

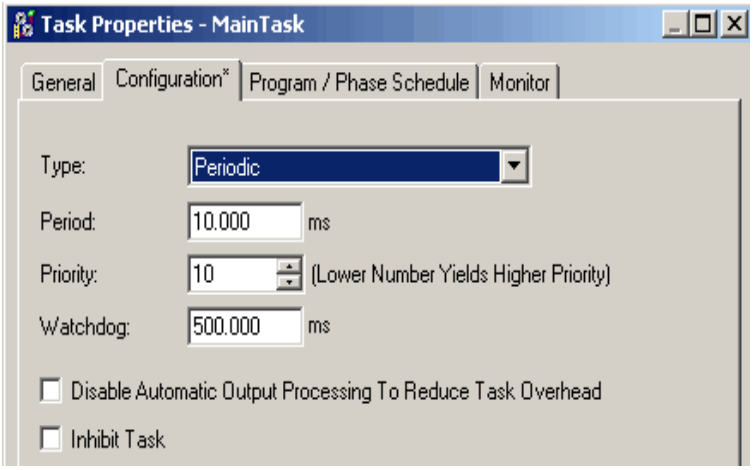
IMPORTANT Be sure that the time period is longer than the sum of the execution times of all the programs assigned to the task.

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

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



The Task Properties dialog box appears.



2. Click the Configuration tab.

3. Enter information in the Task Properties dialog box.

Topic	Description
Type	Choose Periodic (default) for the type of task.
Period	Type a value for when you want the task to execute.
Priority	Enter the task priority value.
Watchdog	Type the watchdog time for the task.

4. Click OK.

Language Switching

With RSLogix 5000 software, version 17, you have the option to 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 RSLogix 5000 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. Once you enable language switching in RSLogix 5000 software, you can dynamically switch between languages as you use the software.

Project documentation that supports multiple translations within a project includes:

- 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 Logix5000 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, RSLogix 5000 software)	Service Communication
Communicate with HMI devices	
Execute Message (MSG) instructions, including block-transfers	
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 execution 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 affect. If you have both a periodic and continuous task, the value selected on the Advanced tab of the Controller Properties dialog box (see page 32) will determine the ratio of running the continuous task and service communication.

The 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

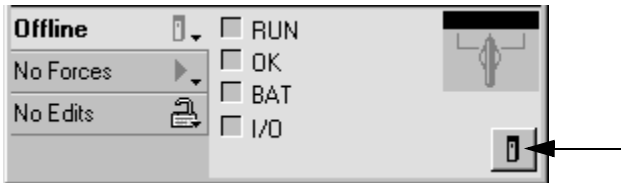
As shown in the table, for RSLogix 5000 version 16 and later, the system-overhead time slice at 50% will stay fixed at 1 ms.

The same applies for 66% and higher, except 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

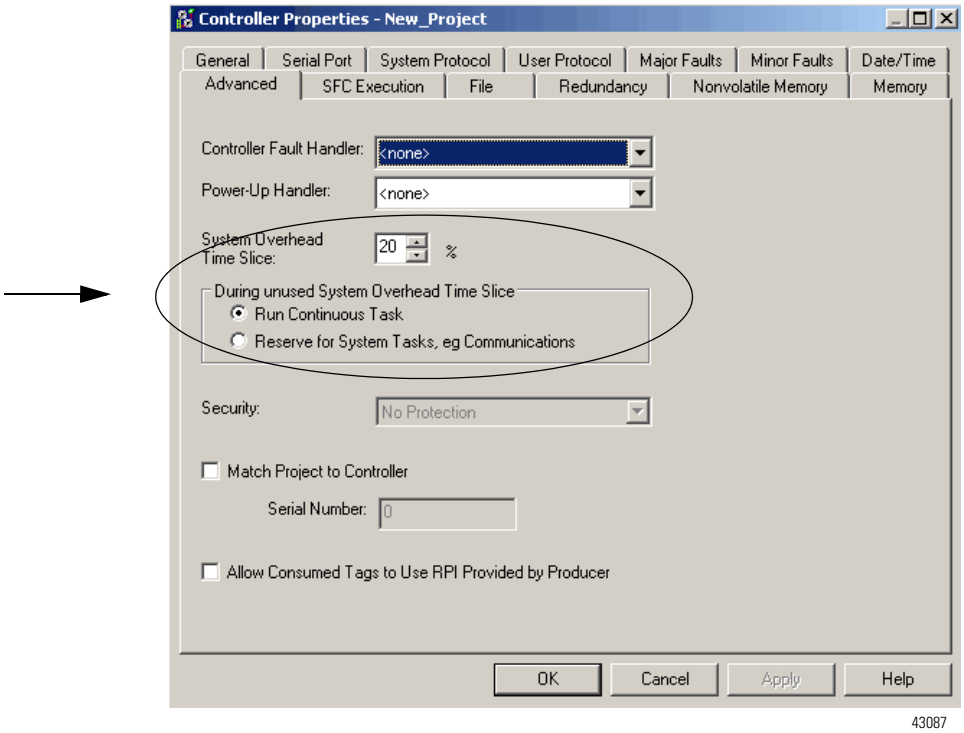
Follow these steps to configure the system-overhead time slice.

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



The Controller Properties dialog box appears.

2. Click the Advanced tab.



3. Enter a numeric value in the System Overhead Time Slice box.

4. Use either Run Continuous Task (default) or Reserve for System Tasks.
   - The Run Continue Task radio button is used when there is no communication or background tasks to process; controller immediately returns to the continuous task.
   - The Reserve for System Task radio button allocates the entire 1 ms of the system-overhead time slice whether the controller has communication or background tasks to perform before returning back 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. This setting is to be used for testing purposes only.

5. Click 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

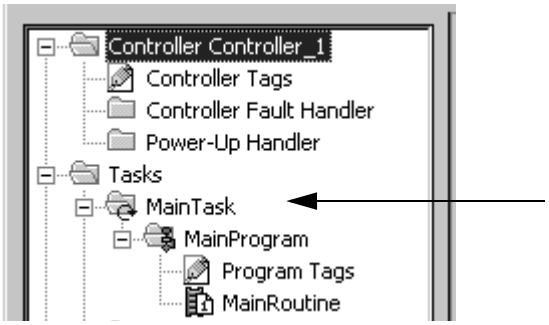
If the watchdog timer reaches a configurable preset, a major fault occurs. Depending on the controller fault handler, the controller might shut down.

- A watchdog time can range from 1…2,000,000 ms (2000 seconds). The default is 500 ms.
- The watchdog timer begins to time when the task is initiated and stops when all the programs within the task have executed.
- 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.

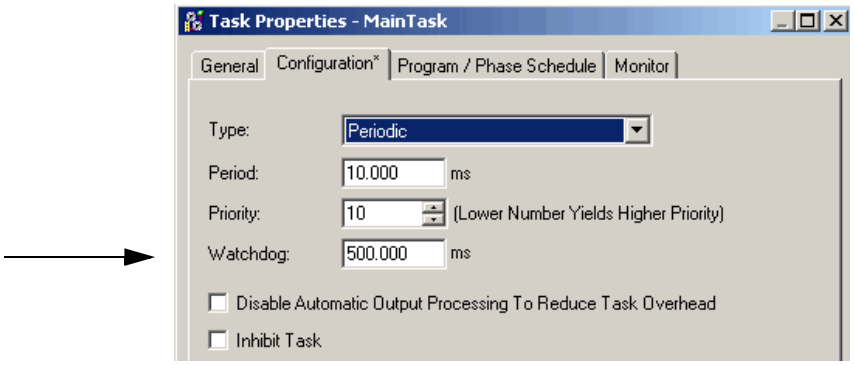
Adjust the Watchdog Timer for a Task

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

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



The Task Properties dialog box appears.



2. Click the Configuration tab.
3. Type a numeric value for the watchdog timeout for the task.
4. Click OK.

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 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 execute. The following table reviews some of these triggers.

Event Task 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 executes.
Registration input for an axis turns On (or Off)	Axis Registration 1 or 2	• In order for the registration input to trigger the event task, first execute 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, execute 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.

Event Task Triggers

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	• In order for the registration input to trigger the event task, first execute 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, execute 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 coarse update period for the motion group triggers the execution of both the motion planner and the event task. • Because the motion planner interrupts all other tasks, it executes first. If you assign the event task as the highest priority task, it executes 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 execute a task from different programs.

Here are 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 execute 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 doesn't 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 executes, 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 varies depending on your type of Logix5000 controller.

IMPORTANT RSLogix 5000 programming software may let you configure a trigger for an event task that your controller does not support. The project will verify and successfully download, but the event task will not execute.

Event Task Triggers

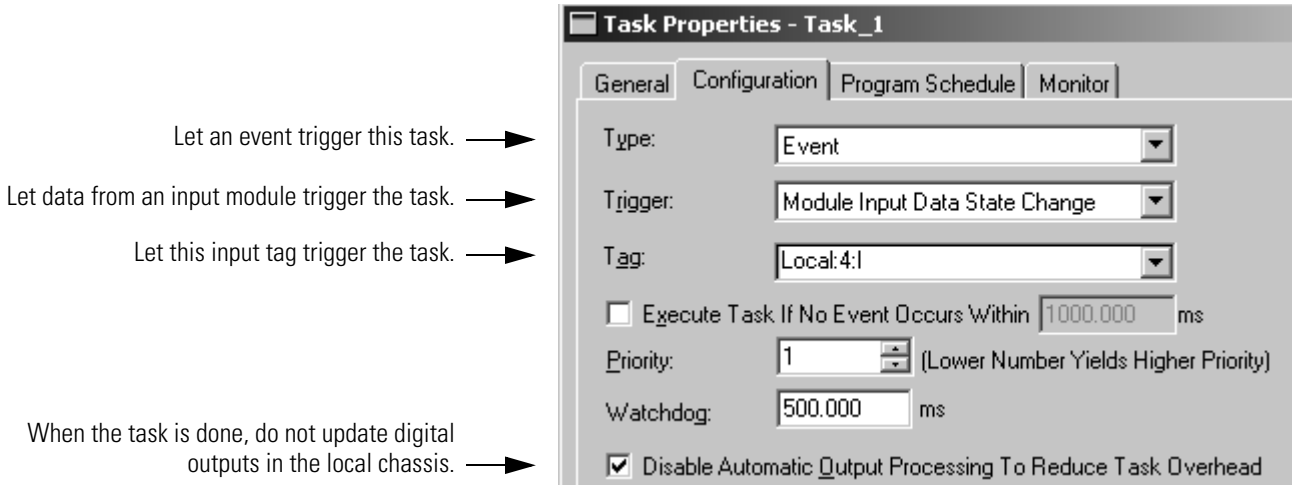
If You Have This Controller	Then You Can Use These Event Task Triggers					
	Module Input Data State Change	Consumed Tag	Axis Registration 1 or 2	Axis Watch	Motion Group Execution	EVENT instruction
CompactLogix		X				X
FlexLogix		X				X
ControlLogix	X	X	X	X	X	X
DriveLogix		X	X	X	X	X
SoftLogix5800	X(1)	X(2)	X	X	X	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.

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.



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 that one listener (device). Describes the feature of the Logix5000 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.

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

Input Module Multicasts Data

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

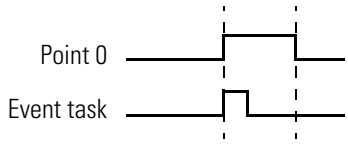
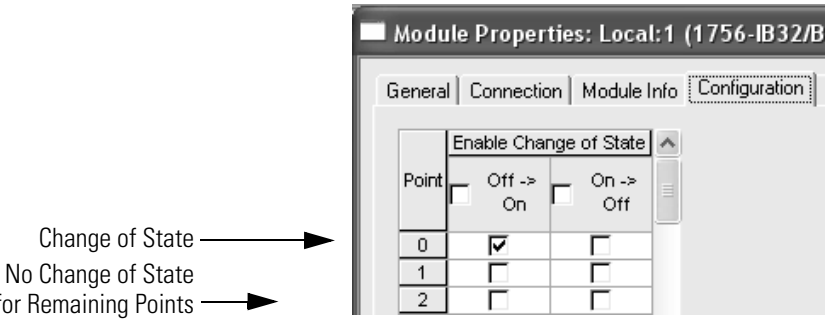
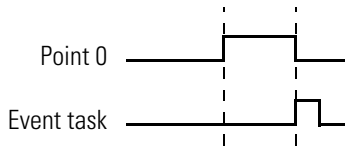
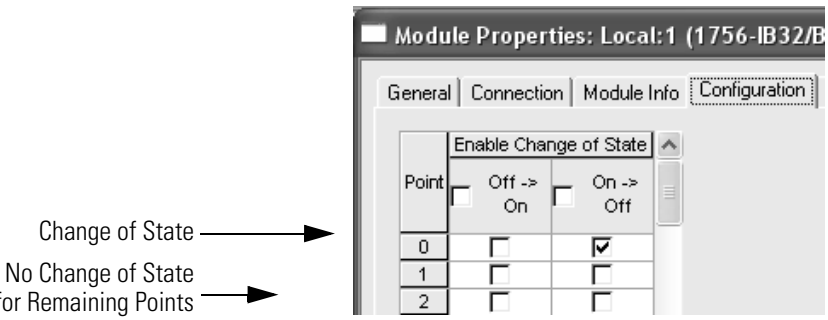
If the module is in a remote chassis, only the RPI determines when the controller receives the data and event trigger over the network.

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

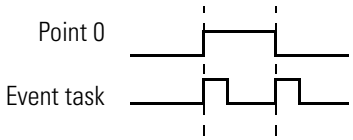
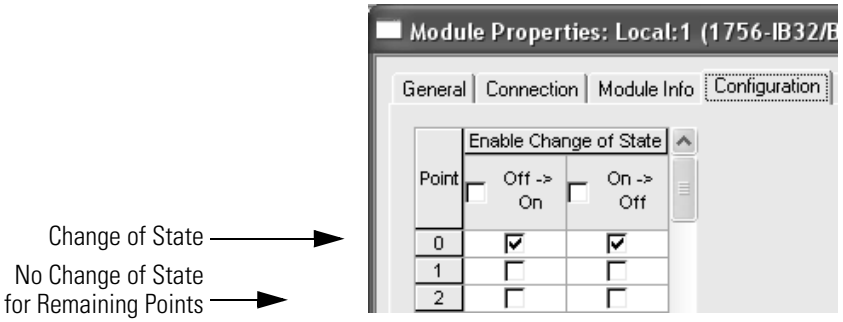
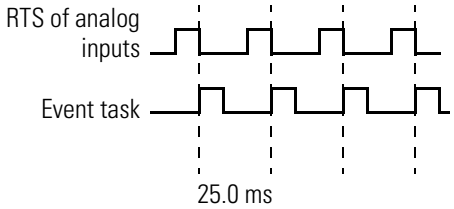
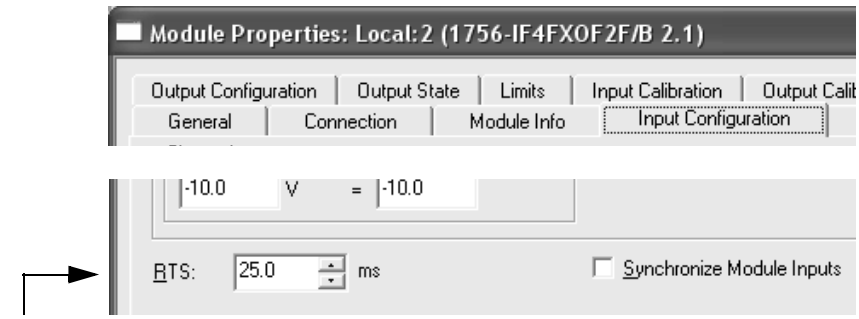
Here are some examples that show COS and RTS configurations.

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

If you want this	Then configure the input module like this (Point 0 is an example)
	
	

COS and RTS Configuration Examples

If you want this	Then configure the input module like this (Point 0 is an example)
	
	

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 we have 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	

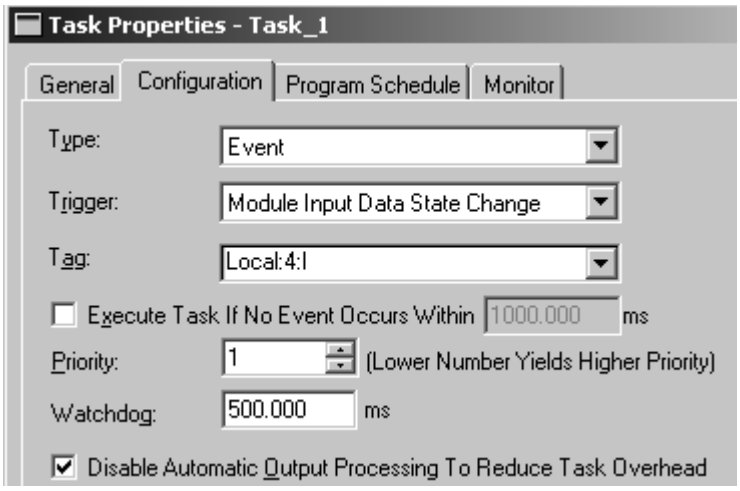
Checklist for an Input Event Task

	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 effect 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.

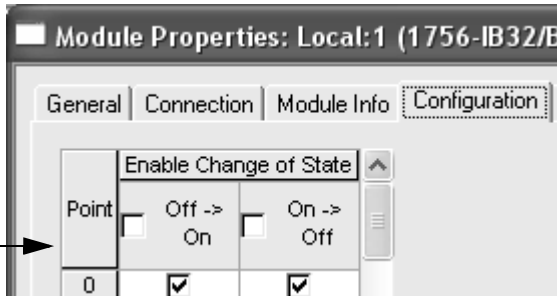
EXAMPLE

As parts move past a diverter location, the controller must decide 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. The input is wired to the module in slot 4 of the local chassis.



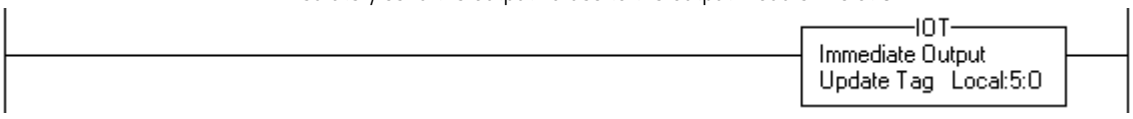
The diverter photoeye (point 0) is configured for change of state for both Off and On. This lets the photoeye trigger the event task when it turns on and when it turns off.



The event task uses this 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 this worksheet.

Consideration	Value
1. 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).	µs
2. 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 on page 45.	µs
3. What is the backplane communication time?	µs
4. What is the total execution time of the programs of the event task?	µs
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 step 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

If chassis size is	Use this value (worst case)
4 slot	13 µs
7 slot	22 µs
10 slot	32 µs
13 slot	42 µs
17 slot	54 µs

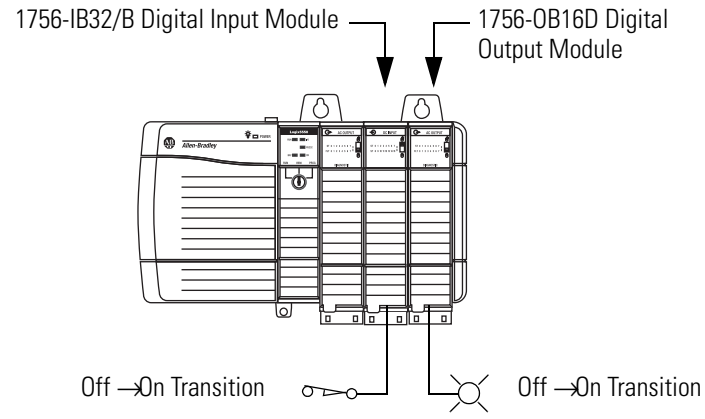
Nominal Hardware Response Times for 1756 I/O Modules With Event Tasks

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

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



Consideration	Value
1. 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).	0 µs
2. 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 on page 45.	330 µs
3. What is the backplane communication time?	13 µs
4. What is the total execution time of the programs of the event task?	400 µs
5. What is the backplane communication time? (Same value as step 3.)	13 µs
6. What is the hardware response time of the output module.	51 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.	807 µs
8. What is the scan time of the motion group?	1130 µs
9. What is the total scan time of the tasks that have a higher priority than this event task (if any)?	0 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.	1937 µs

If Chassis Size is	Use This Value (Worst Case)
4 slot	13 µs
7 slot	22 µs
10 slot	32 µs
13 slot	42 µs
17 slot	54 µs

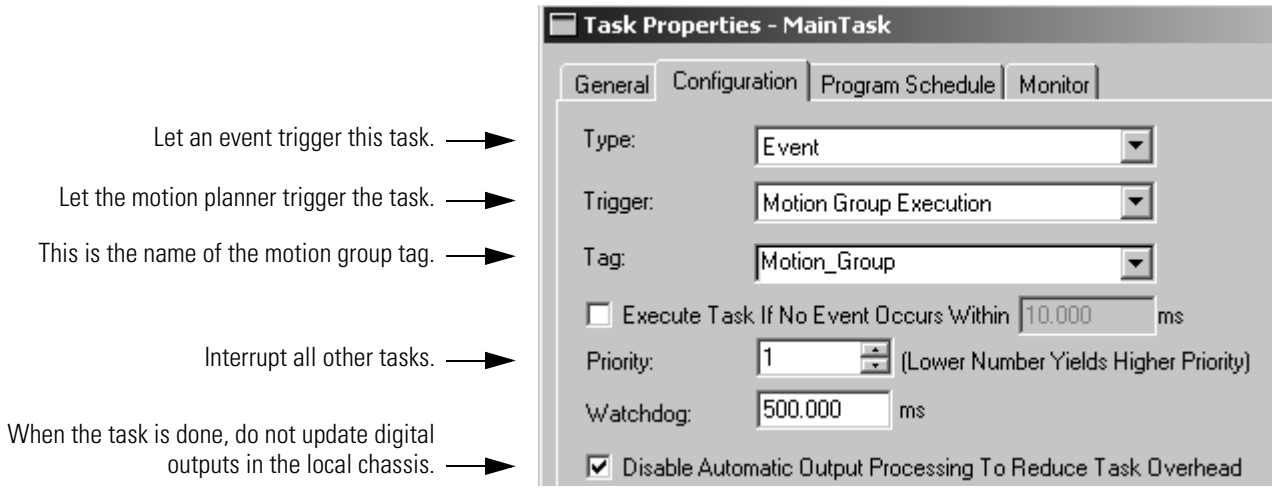
Additional Considerations

These considerations effect the scan time of the event task, which effects 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 executing task. (The task with the CPS or UID.)

Motion Group Trigger

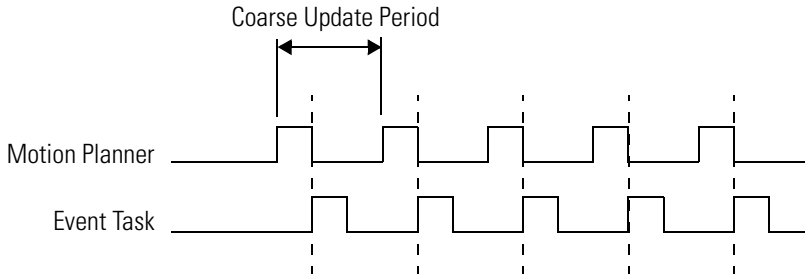
To couple the execution of an event task with the execution of the motion planner, use the Motion Group Execution trigger.



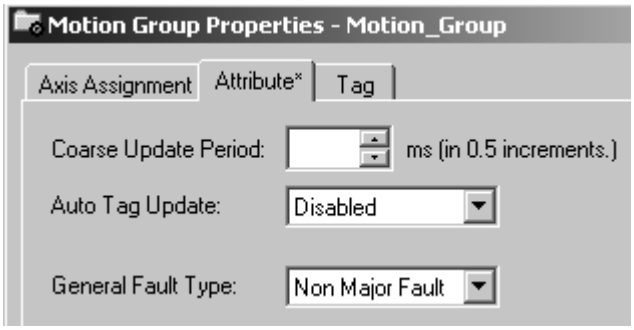
The Motion Group Execution trigger works as follows:

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

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



The coarse update period for the motion group triggers both the motion planner and the event task.

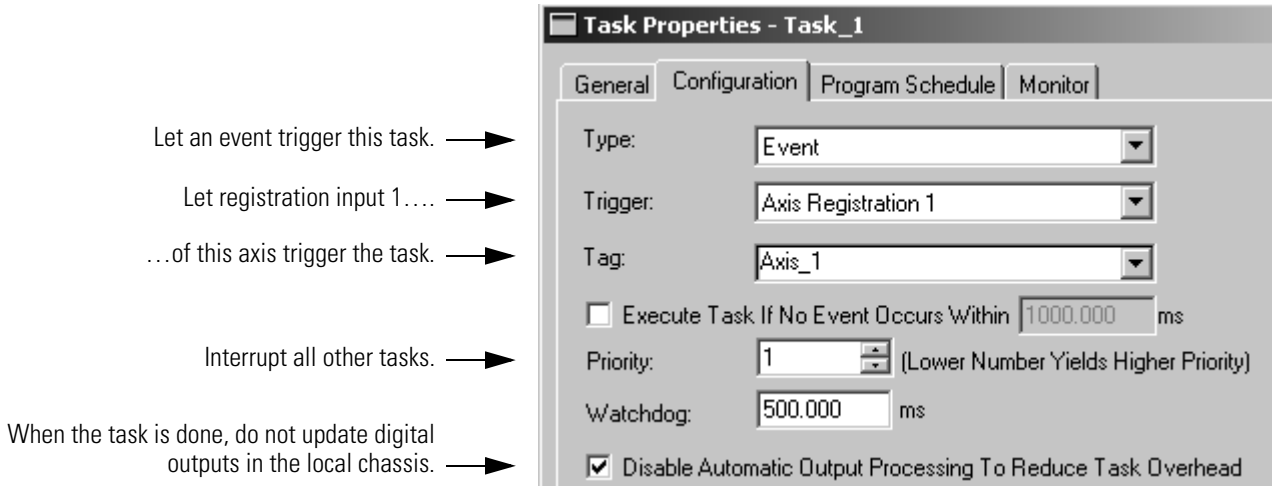


Checklist for a Motion Group Task

For This		Make Sure You
☐	1. Scan time	Make sure the scan time of the event task is significantly less than the course 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

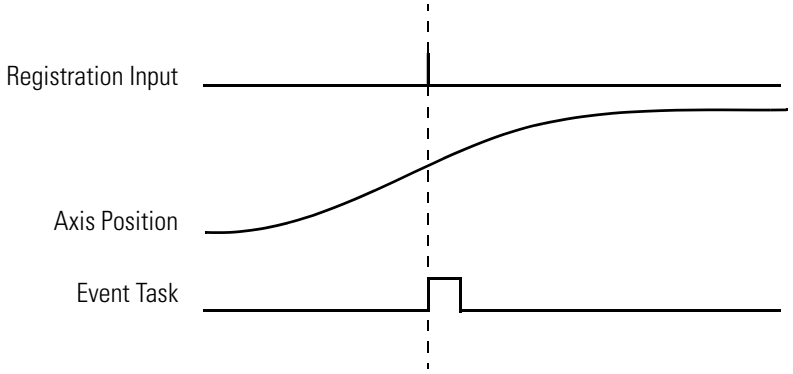
To let the registration input of an axis trigger an event task, use the Axis Registration (1 or 2) trigger.



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

<table>
<tr><th colspan="2">For this</th><th colspan="2">Make sure you</th></tr>
<tr><td>☐</td><td>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></td><td></td><td>If your normal logic is</td><td>Then</td></tr>
<tr><td></td><td></td><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></td><td></td><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

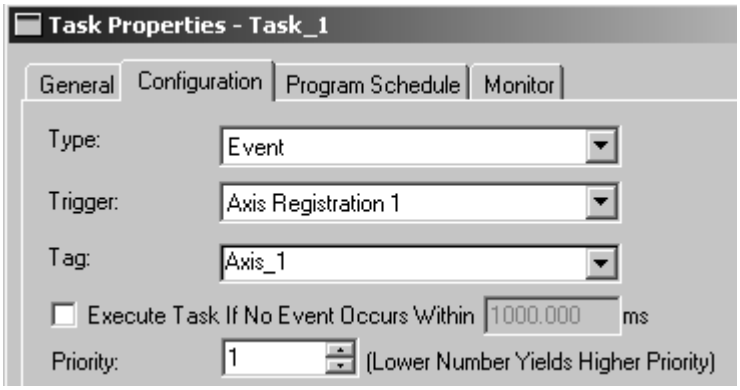
In a line that packages candy bars, you have to make sure that the perforation occurs in the correct location on each bar.

- 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.

A registration sensor is wired as registration input 1…

…for the axis named *Axis_1*.

This event task interrupts all other tasks.



This 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 execution of the EVENT instruction to one scan.

the EVENT instruction triggers an execution of Task_1 (event task).

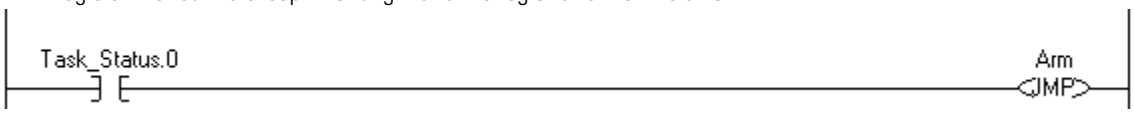
Task_1 (event task)

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).

Continued on next page.

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

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



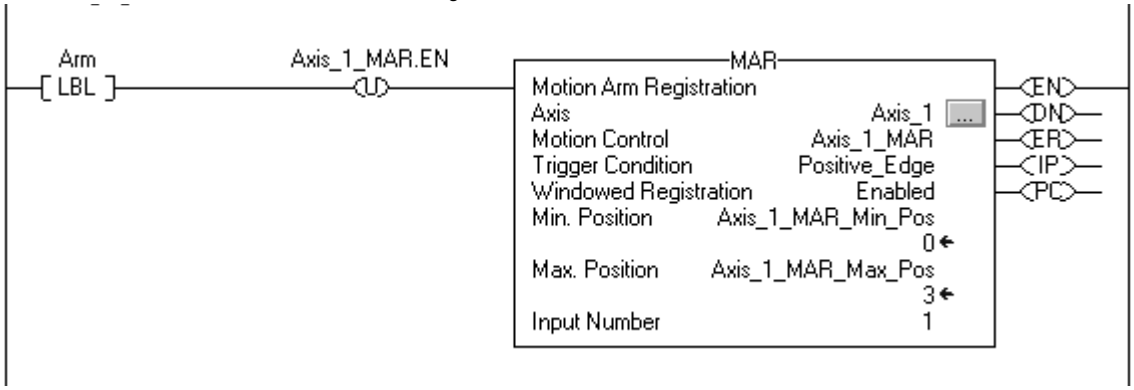
-
- Other logic
-

The MAR instruction executes each time the task executes 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.
- To execute the MAR 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 MAR instruction arms the axis for registration.

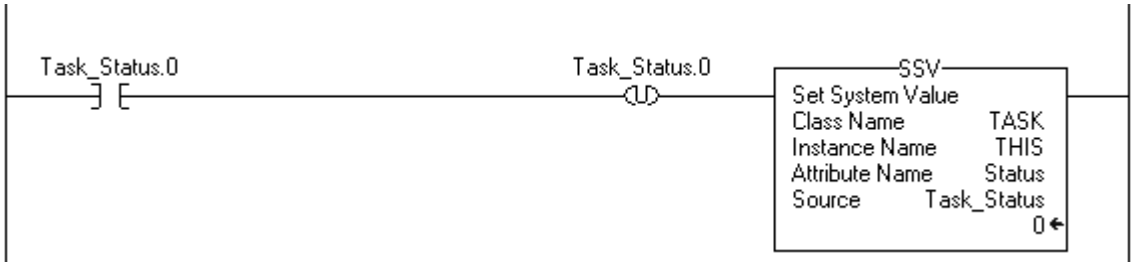


The controller does not clear the bits of the Status attribute once they are set. To use a bit for new status information, you must manually clear the bit.

If Task_Status.0 = 1 then clear that bit.

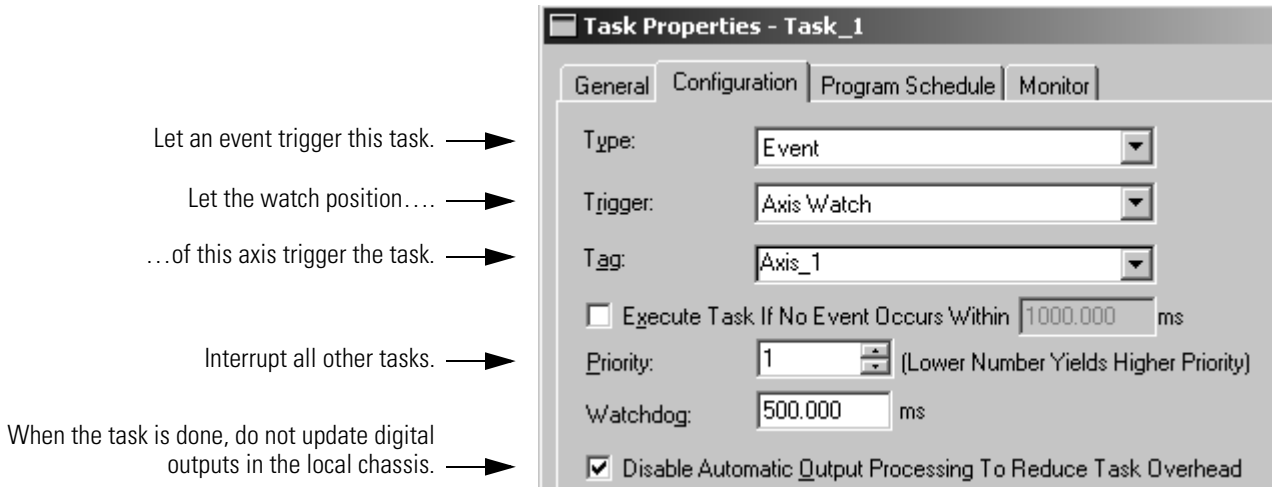
The OTU instruction sets Task_Status.0 = 0.

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



Axis Watch Trigger

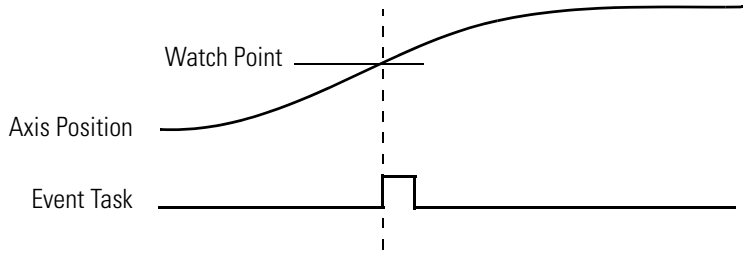
To let the watch position of an axis trigger an event task, use the Axis Watch trigger.



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

<table>
<tr><th colspan="2">For this</th><th colspan="2">Make sure you</th></tr>
<tr><td>☐</td><td>1. 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.
• Once 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></td><td></td><td>If your normal logic is</td><td>Then</td></tr>
<tr><td></td><td></td><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></td><td></td><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. 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

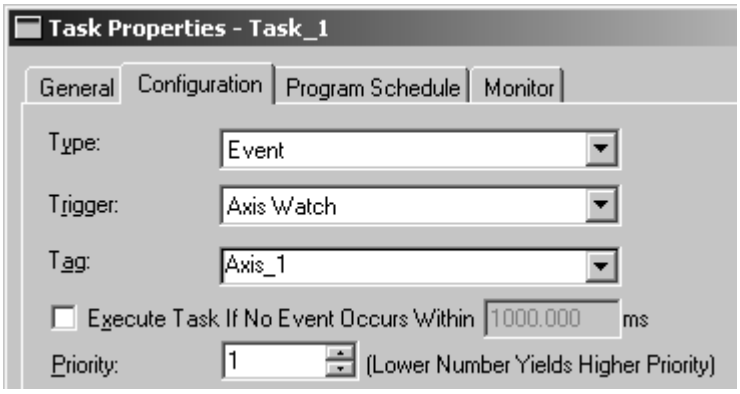
At the labeling station of a bottling line, you want to check the position of the label on the bottle.

- 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.

Let the watch position…

…for the axis named *Axis_1* trigger the event task.

This event task interrupts all other tasks.



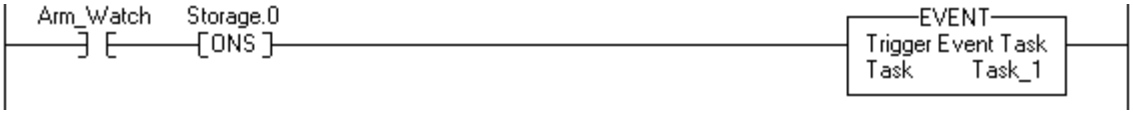
This logic arms and re-arms the axis for the watch position.

Continuous task

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

the ONS instruction limits the execution of the EVENT instruction to one scan.

the EVENT instruction triggers an execution of Task_1 (event task).



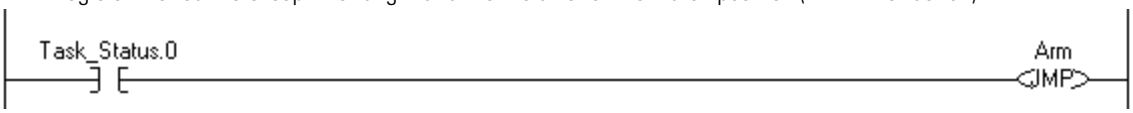
Task_1 (event task)

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).

Continued on next page.

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

The JMP instruction causes the controller to jump its execution 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).



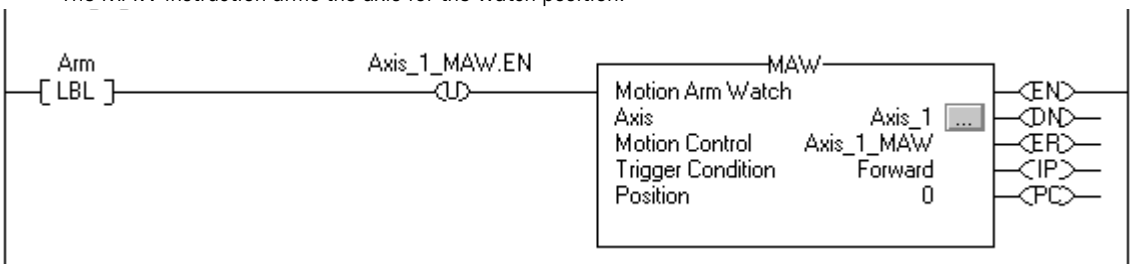
-
- Other logic
-

The MAW instruction executes each time the task executes 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 execute 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.

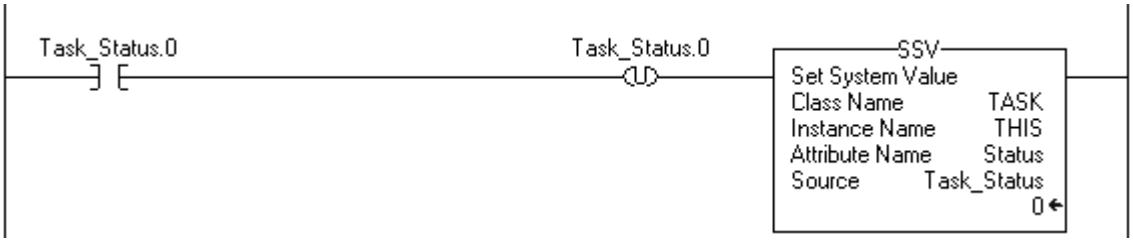


The controller does not clear the bits of the Status attribute once they are set. To use a bit for new status information, you must manually clear the bit.

If Task_Status.0 = 1 then clear that bit.

The OTU instruction sets Task_Status.0 = 0.

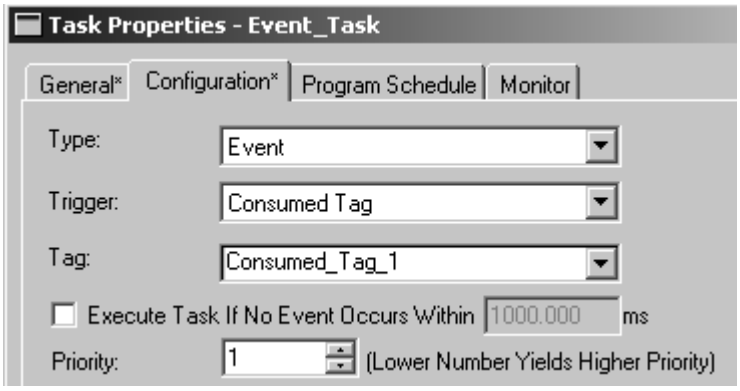
The SSV instruction sets the Status attribute of THIS task (Task_1) = Task_Status. This includes the cleared bit.



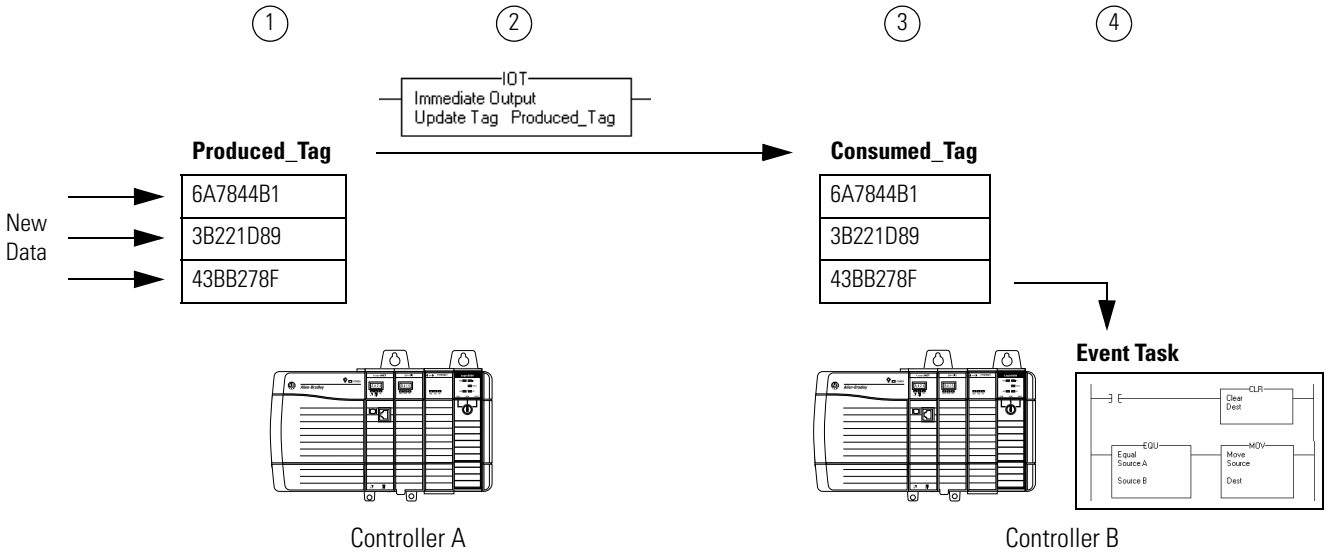
Consumed Tag Trigger

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

Let an event trigger this task. →
Let a consumed tag trigger the task. →
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 executes 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 executes the event task.

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

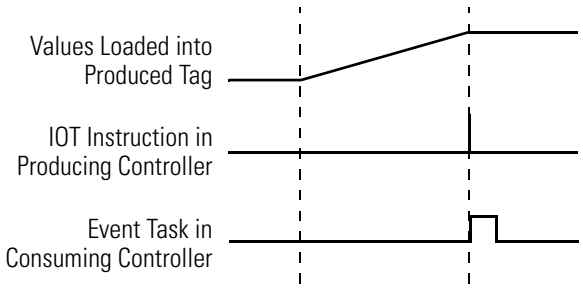
When the Consuming Device Receives the New Data and Event Trigger

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)

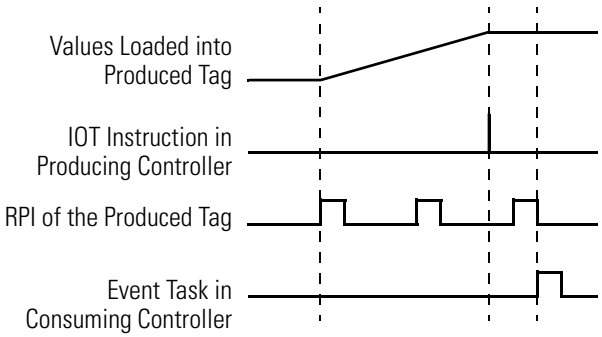
These diagrams compare the receipt of data via an IOT instruction over EtherNet/IP and ControlNet networks.

Compare the Receipt of Data via an IOT Instruction

EtherNet/IP Network (ControlLogix controller)

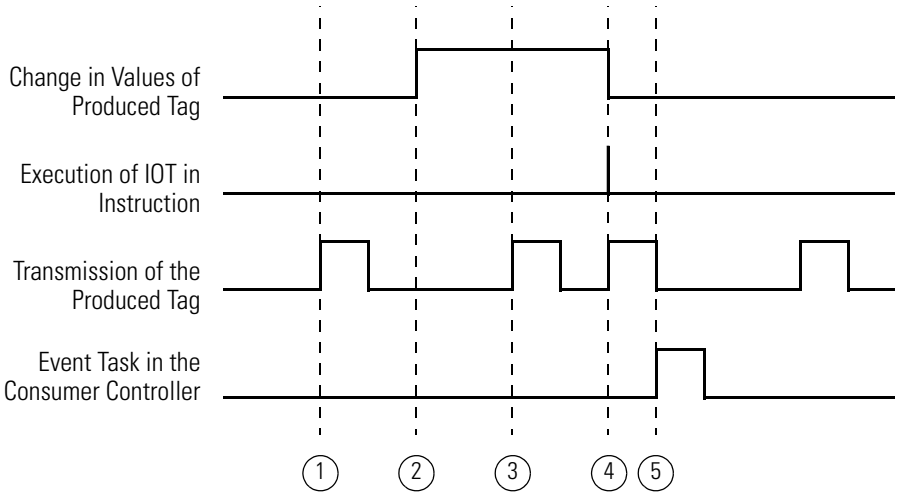


ControlNet Network



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.

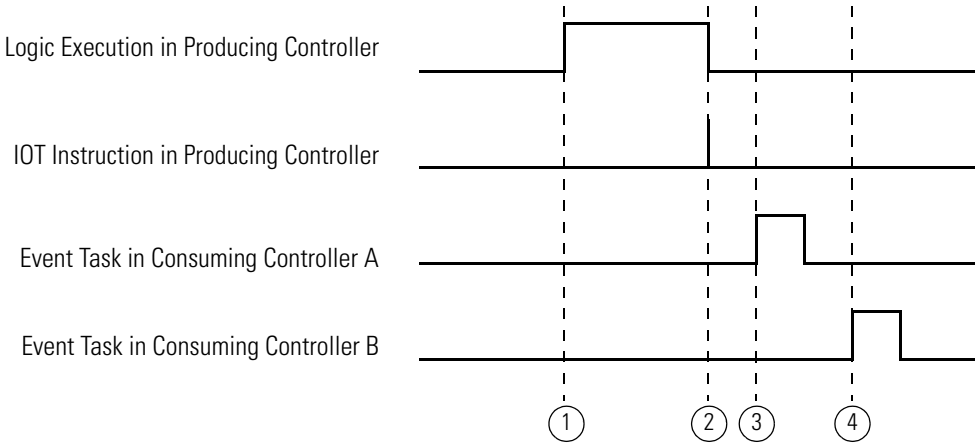


	Description
1	RPI occurs for the produced tag. The produced tag transfers old data to the consuming controller.
2	The producer controller starts to update the values of the produced tag.
3	RPI occurs again for the produced tag. The produced tag transfers a mix of old and new data to the consuming controller.
4	The producer controller finishes updating the values of the produced tag. The producer controller executes 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 executes its event task.

Although the producing controller executes 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

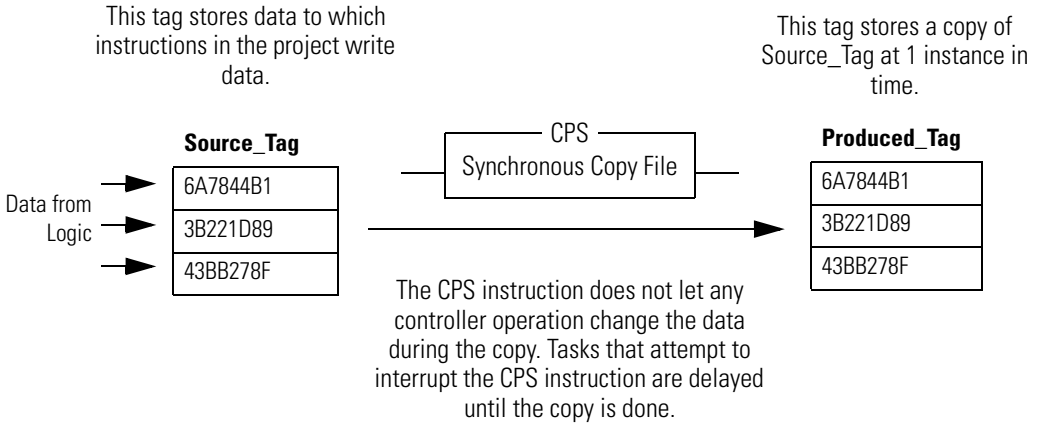
You can also use the produced/consumed tag relationship to synchronize controllers. In this case, the produced/consumed tag serves only as a triggering mechanism.



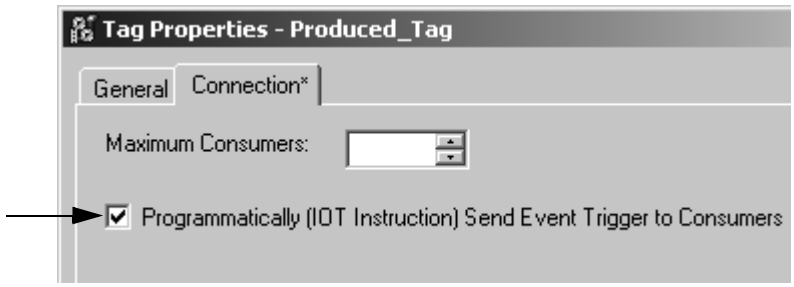
	Description
1	The first controller executes an action with which other controllers need to stay synchronized.
2	When the action is done, the controller executes an IOT instruction. The IOT instruction uses a produced tag as its target.
3	When controller A receives the produced tag, it executes its event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

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 below.



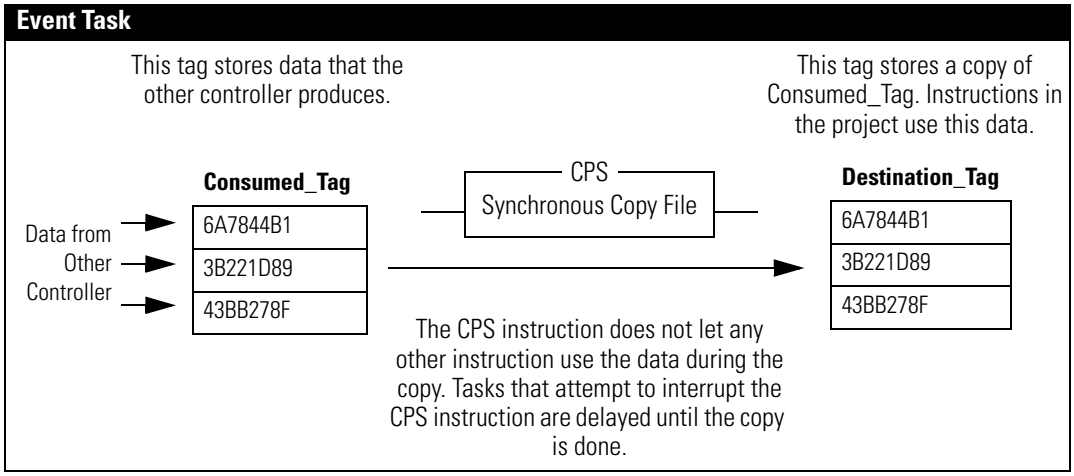
☐	2. Produced tag properties	In the Connection properties of the produced tag, check 'Programmatically (IOT Instruction) Send Event Trigger to Consumers'.



		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 will trigger 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

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 below.



☐	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

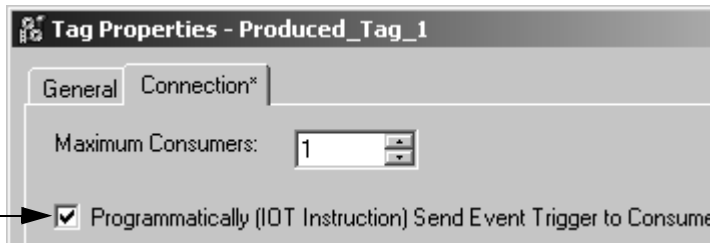
As parts move along a production line, each station requires production specifications for the part at its station. To make sure that a station doesn't act on old data, an event task signals the arrival of new data for the next part.

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 these elements:

Produced Tag Properties

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

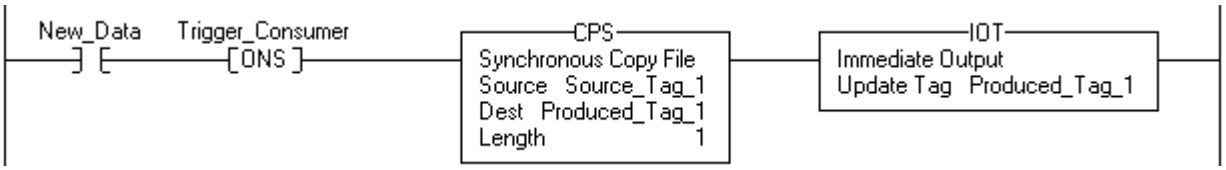


Ladder Logic

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.



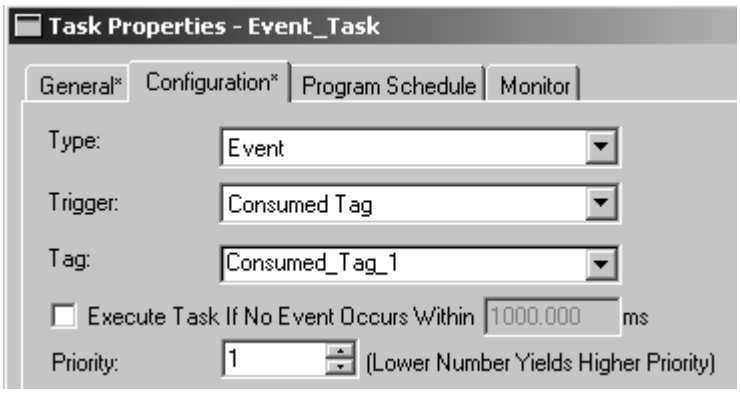
Consumer Controller The controller at station 25 uses the data produced by station 24. To determine when new data has arrived, the controller uses an event task.

Event Task Properties

Let an event trigger this task. →

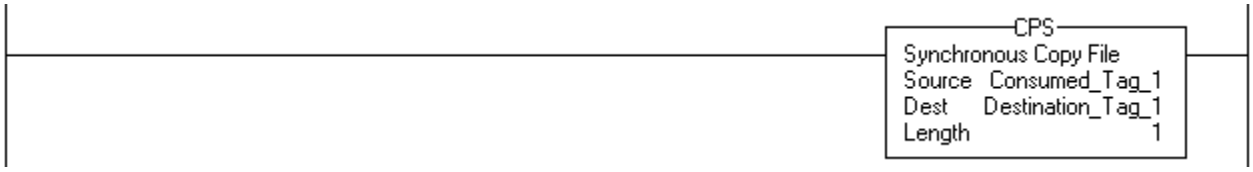
Let a consumed tag trigger the task. →

Let this consumed tag trigger the task. →



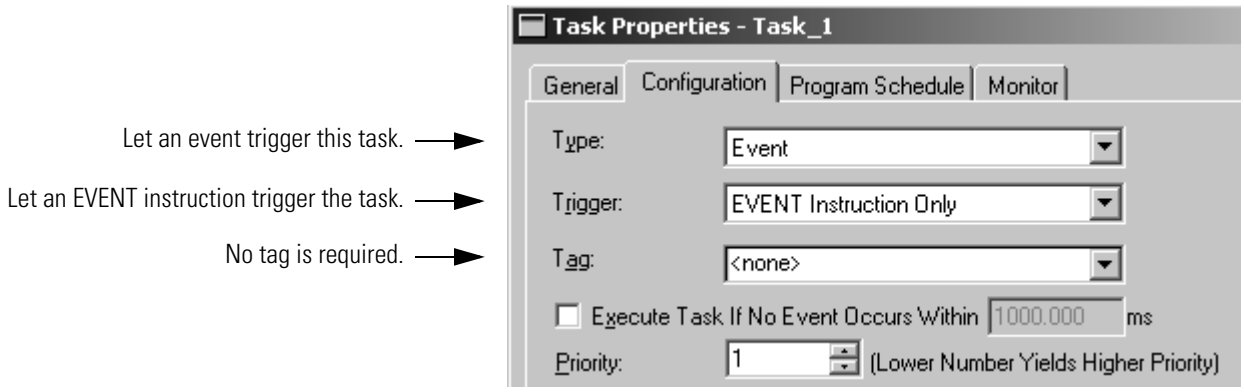
Ladder Diagram in the Event Task

When the event task executes, 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.

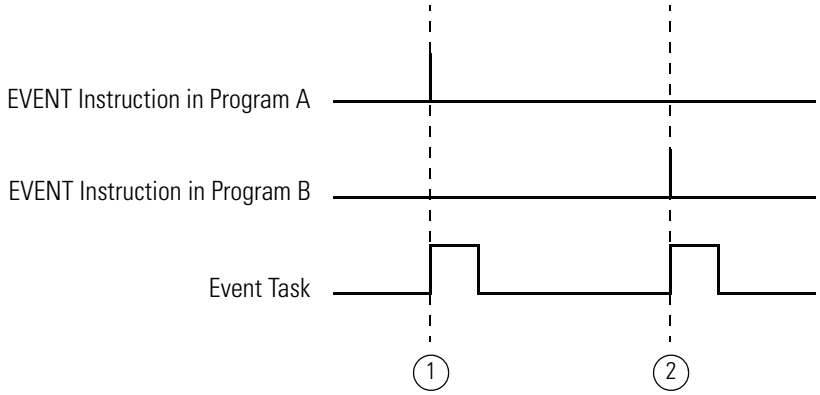


EVENT Instruction Trigger

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



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 executes, it triggers the specified event task.



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

Status Attribute of 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**
			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

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

For this		Make sure you
☐	1. EVENT instruction	Use a Trigger Event Task (EVNT) 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

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.

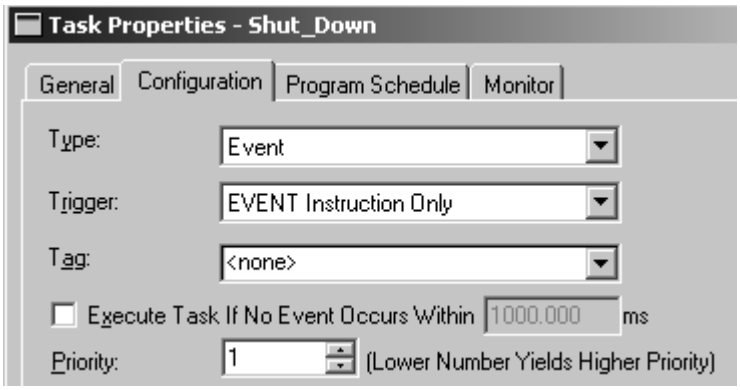
Event Task Properties

Let an event trigger this task. →

Let an EVENT instruction trigger the task. →

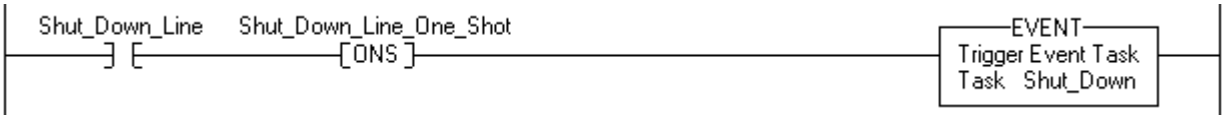
No tag is required. →

Interrupt all other tasks. →



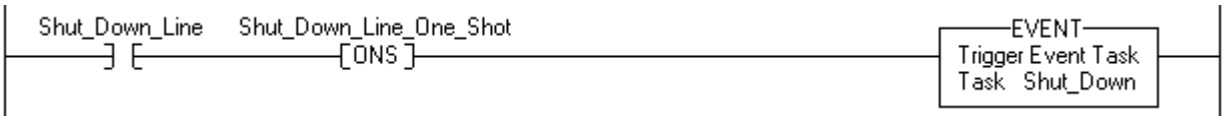
Ladder Diagram in Program_A

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



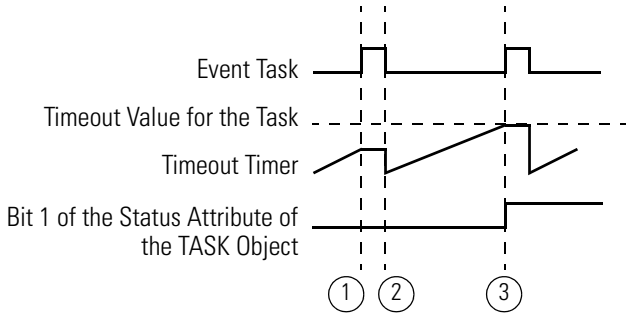
Ladder Diagram in Program_B

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



Define a Timeout Value for an Event Task

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

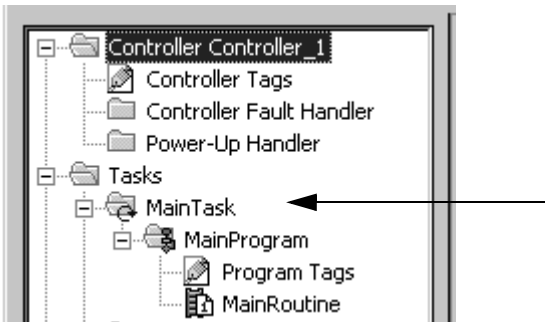


	Description
1	Event task executes. Timeout time stops incrementing.
2	Event task is done. Timeout timer resets and begins incrementing.
3	Timeout timer reaches the timeout value. Event task automatically executes. In the Status attribute of the TASK object, bit 1 turns on.

Assign a Timeout Value to an Event Task

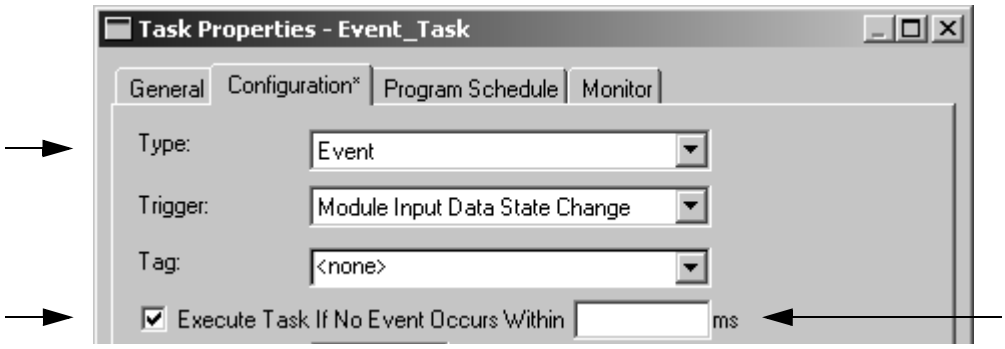
Follow these steps to assign a timeout value to an event task.

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



The Task Properties dialog box appears.

2. Click the Configuration tab.



3. From the Type pull-down menu, choose Event.
4. Check 'Execute Task If No Event Occurs Within'.
5. Type the timeout value.
6. Click OK.

Programmatically Configure a Timeout

To programmatically configure a timeout, use a Get System Value (GSV) instruction to access the attributes of the task.

Status Attribute of 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)

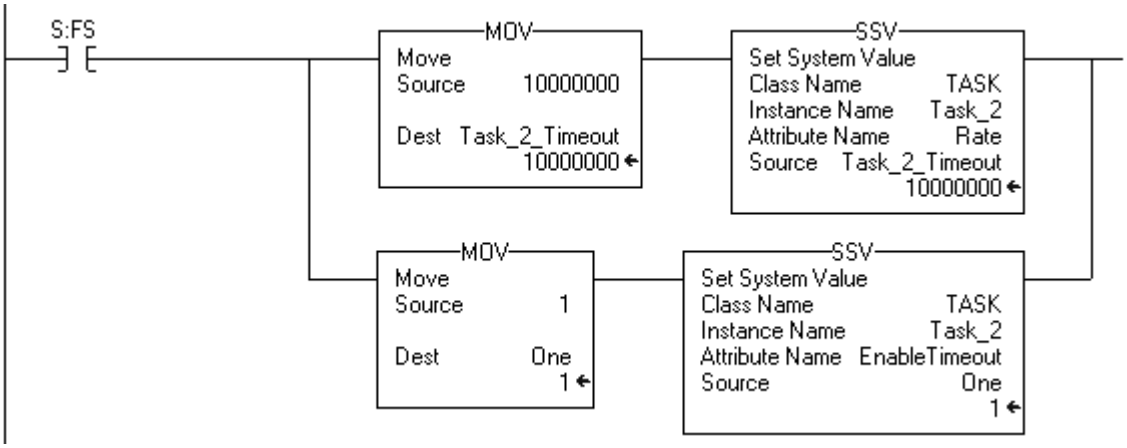
EXAMPLE

Programmatically Configure a Timeout

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 executed due to a timeout, use a Get System Value (GSV) instruction to monitor the Status attribute of the task.

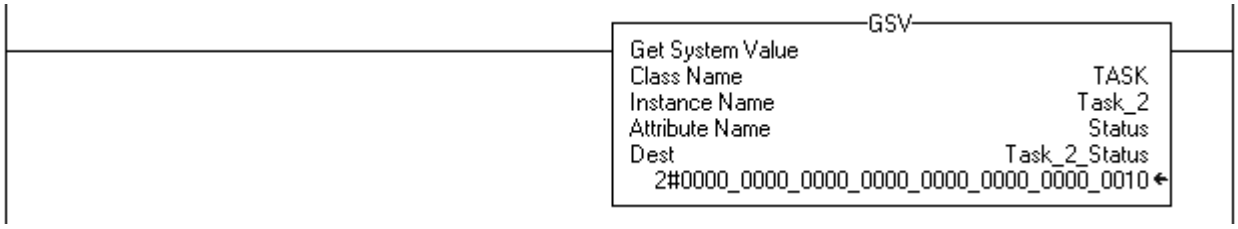
Status Attribute of the TASK Object

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**
			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 Define a Timeout Value for an Event Task

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).

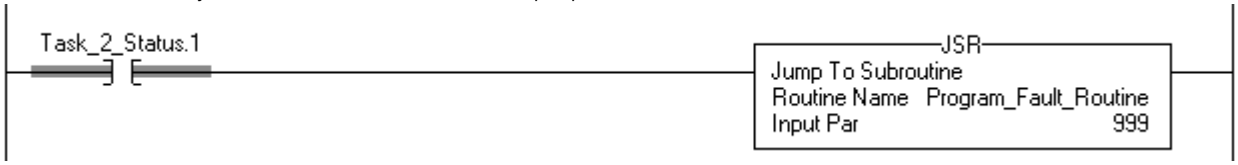
1. The GSV instruction sets Task_2_Status = Status attribute for Task_2 (DINT value).



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.

Notes:

T

W

Rockwell Automation Support

Rockwell Automation provides technical information on the Web to assist you in using its products. At http://www.rockwellautomation.com/support/, you can find technical manuals, a knowledge base of FAQs, technical and application notes, sample code and links to software service packs, and a MySupport feature that you can customize to make the best use of these tools.

For an additional level of technical phone support for installation, configuration, and troubleshooting, we offer TechConnect support programs. For more information, contact your local distributor or Rockwell Automation representative, or visit http://www.rockwellautomation.com/support/.

Installation Assistance

If you experience an anomoly within the first 24 hours of installation, review the information that is contained in this manual. You can contact Customer Support for initial help in getting your product up and running.

United States or Canada	1.440.646.3434
Outside United States or Canada	Use the Worldwide Locator at http://www.rockwellautomation.com/support/americas/phone_en.html, or contact your local Rockwell Automation representative.

New Product Satisfaction Return

Rockwell Automation tests all of its products to ensure that they are fully operational when shipped from the manufacturing facility. However, if your product is not functioning and needs to be returned, follow these procedures.

United States	Contact your distributor. You must provide a Customer Support case number (call the phone number above to obtain one) to your distributor to complete the return process.
Outside United States	Please contact your local Rockwell Automation representative for the return procedure.

Documentation Feedback

Your comments will help us serve your documentation needs better. If you have any suggestions on how to improve this document, complete this form, publication RA-DU002, available at http://www.rockwellautomation.com/literature/.

www.rockwellautomation.com

Power, Control and Information Solutions Headquarters

Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
Europe/Middle East/Africa: Rockwell Automation, Vorstlaan/Boulevard du Souverain 36, 1170 Brussels, 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

Publication 1756-PM005C-EN-P - October 2009
Supersedes Publication 1756-PM005B-EN-P - July 2008