Logix 5000 Controllers Sequential Function Charts

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

Rockwell Automation Publication 1756-PM006K-EN-P - March 2022
Supersedes Publication 1756-PM006J-EN-P - September 2020

Rockwell Automation

Programming Manual

Original Instructions

Important User Information

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Global changes

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

Change	Topic
Updated Legal notices.	Legal notices on page 10

Chapter 2

Program a sequential function chart

Force steps

Chapter 3

Index

This manual shows how to design and program Sequential Function Charts (SFCs) for Logix 5000 controllers to execute. This manual is one of a set of related manuals that show common procedures for programming and operating Logix 5000 controllers.

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

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

Studio 5000 environment

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

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

Additional resources

These documents contain additional information concerning related Rockwell Automation products.

Resource	Description
Logix 5000 Controllers Program Parameters Programming Manual, publication 1756-PM021	Describes how to use program parameters when programming Logix 5000 controllers.

Resource	Description
Logix 5000 Controllers General Instructions Reference Manual, publication 1756-RM003	Describes the available instructions for a Logix 5000 controller.
Logix 5000 Controllers Process and Drives Instructions Reference Manual, publication 1756-RM006	Describes how to program a Logix 5000 controller for process or drives applications.
Logix 5000 Controllers Motion Instruction Set Reference Manual, publication MOTION-RM002	Describes how to program a Logix 5000 controller for motion applications.
Product Certifications website, http://ab.rockwellautomation.com	Provides declarations of conformity, certificates, and other certification details.

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

Legal notices

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

End User License Agreement (EULA)

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

The default location of this file is:

C:\Program Files (x86)\Common Files\Rockwell\license.rtf.

Open Source Software Licenses

The software included in this product contains copyrighted software that is licensed under one or more open source licenses.

You can view a full list of all open source software used in this product and their corresponding licenses by opening the oss_license.txt file located in your product's OPENSOURCE folder on your hard drive. This file is divided into these sections:

- Components
  Includes the name of the open source component, its version number, and the type of license.
- Copyright Text
  Includes the name of the open source component, its version number, and the copyright declaration.

- Licenses
  Includes the name of the license, the list of open source components citing the license, and the terms of the license.

The default location of this file is:

C:\Program Files (x86)\Common Files\Rockwell\Help\<*product name*>\Release Notes\OPENSOURCE\oss_licenses.txt.

You may obtain Corresponding Source code for open source packages included in this product from their respective project web site(s). Alternatively, you may obtain complete Corresponding Source code by contacting Rockwell Automation via the **Contact** form on the Rockwell Automation website: http://www.rockwellautomation.com/global/about-us/contact/contact.page. Please include "Open Source" as part of the request text.

Design a sequential function chart

Introduction

A sequential function chart (SFC) is similar to a flowchart of your process. It defines the steps or states through which your system progresses. It helps you do the following:

- Organize the functional specification for your system.
- Program and control your system as a series of steps and transitions.

By using an SFC to specify your process, you gain these advantages.

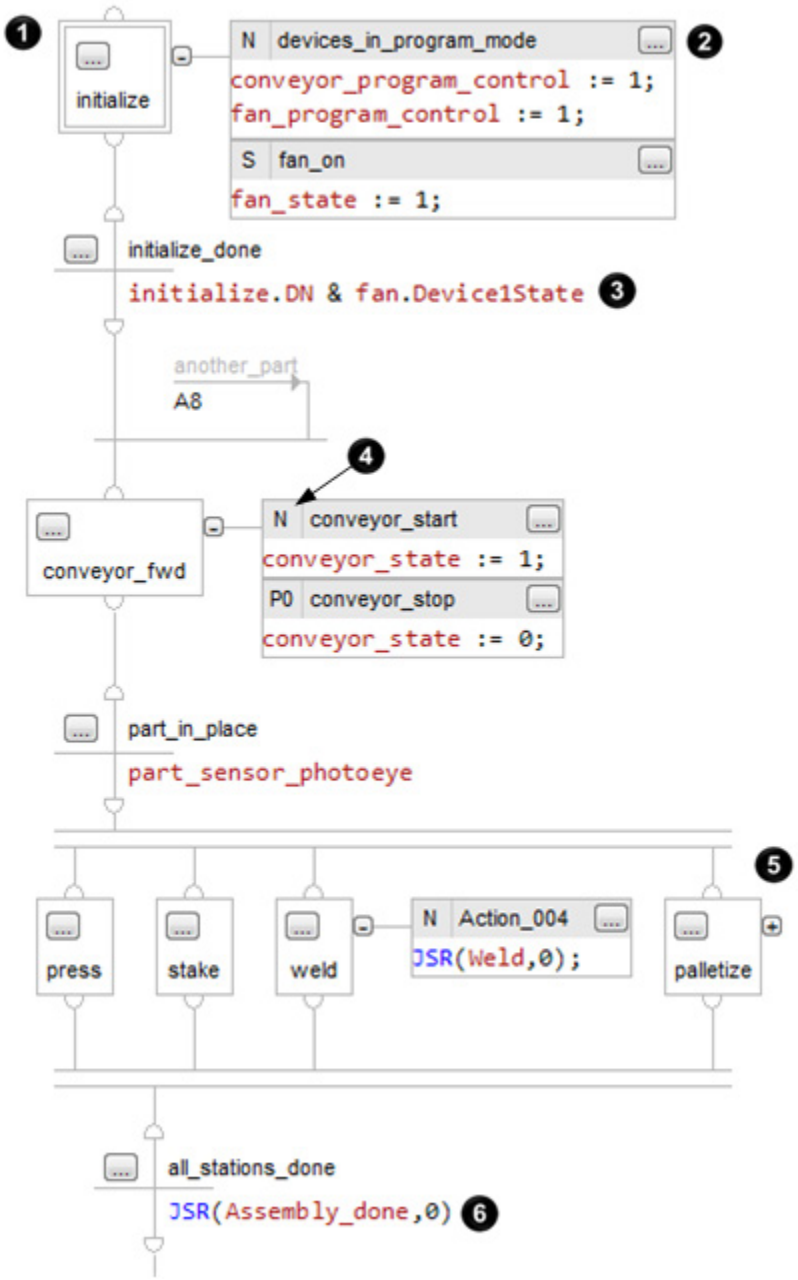
- Since an SFC is a graphical representation of your process, it is easier to organize and read than a textual version.
  - Add notes that clarify steps or capture important information for use later on.
  - Print the SFC to share the information with other individuals.
- Since Logix 5000 controllers support SFCs, you do not have to enter the specification a second time. You are programming your system as you specify it.

By using an SFC to program your process, you gain these advantages.

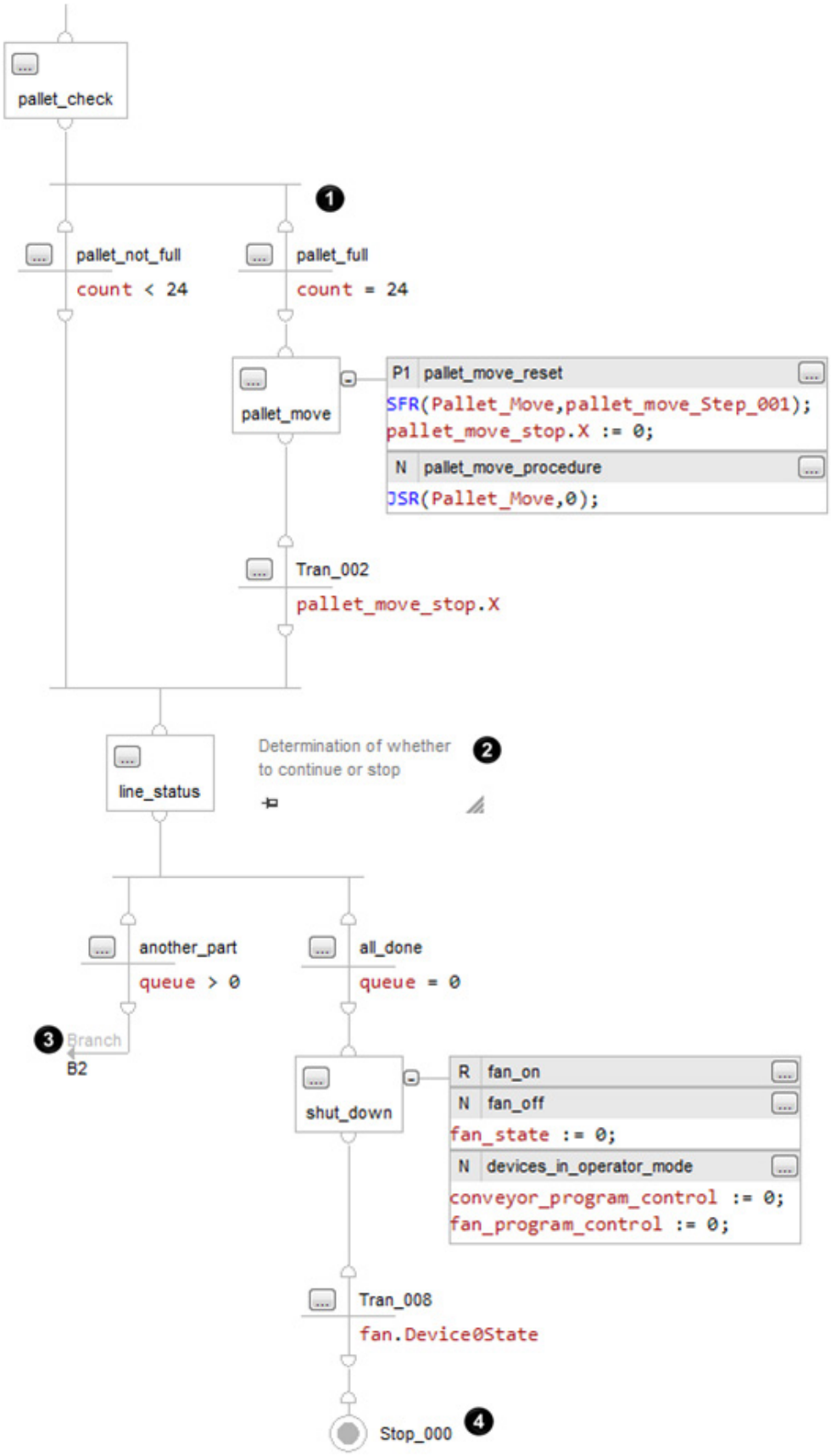
- Graphical division of processes into its major logic pieces (steps)
- Faster repeated execution of individual pieces of your logic
- Simpler screen display
- Reduced time to design and debug your program
- Faster and easier troubleshooting
- Direct access to the point in the logic where a machine faulted
- Easy updates and enhancements

What is a sequential function chart?

A sequential function chart (SFC) is similar to a flowchart. It uses steps and transitions to perform specific operations or actions. This example shows the elements of an SFC. The SFC continues on the following page.



1	A step represents a major function of your process. It contains the actions that occur at a particular time, phase, or station.
2	An action is one of the functions that a step performs.
3	A transition is the TRUE or FALSE condition that tells the SFC when to go to the next step.
4	A qualifier determines when an action starts and stops.
5	A simultaneous branch executes more than 1 step at the same time.
6	JSR instruction calls a subroutine.



1	A **selection branch** chooses between different execution paths.
2	A **text box** lets you add descriptive text or notes to your SFC.

3	A **wire** connects one element to another element anywhere on the chart. This wire takes you to the conveyor step on the first part of this SFC (previous figure).
4	A **stop** lets you stop and wait for a command to restart.

Follow these steps to design a sequential function chart.

1. Define the tasks on page 16
2. Choose how to execute the SFC on page 17
3. Define the steps of the process on page 17
4. Organize the steps on page 21
5. Add actions for each step on page 25
6. Describe each action in pseudocode on page 28
7. Choose a qualifier for an action on page 29
8. Define the transition conditions on page 30
9. Transition after a specified time on page 32
10. Turn off a device at the end of a step on page 35
11. Keep something on from step-to-step on page 41
12. End the SFC on page 44
13. Nest an SFC on page 46
14. Configure when to return to the OS/JSR on page 48
15. Pause or reset an SFC on page 48
16. Execution diagrams on page 48

Define the tasks

The first step in the development of an SFC is to separate the configuration and regulation of devices from the commands to those devices. Logix 5000 controllers let you divide your project into one continuous task and multiple periodic tasks and event tasks.

To define the tasks

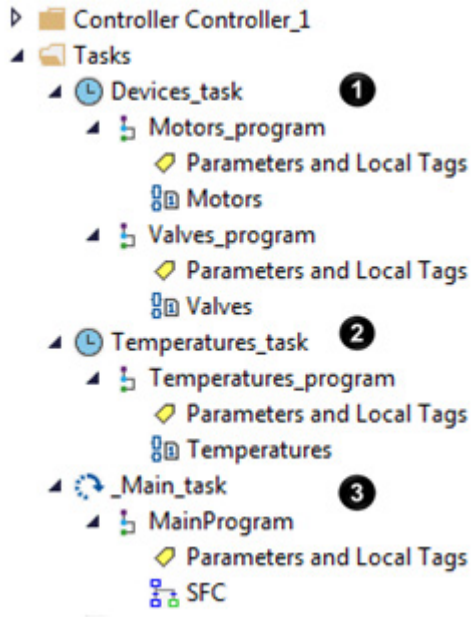
1. Organize your project.

These functions	Go into this type of task
• Configure and regulate devices	Periodic task
• Command a device to a specific state • Sequence the execution of your process	SFC in the continuous task

2. For those functions that go in a periodic task, group the functions according to similar update rates. Create a periodic task for each update rate.

   For example, 2-state devices may require faster updates than PID loops. Use separate periodic tasks for each.
   In this example, a project uses two periodic tasks to regulate motors, valves, and temperature loops. An SFC controls the process.

Define the Tasks:



1. This task (periodic) uses Function Block diagrams to turn on or off motors and open or close valves. The SFC in MainTask commands the state for each device. The Function Block diagrams set and maintain that state.
2. This task (periodic) uses Function Block diagrams to configure and regulate temperature loops. The SFC in MainTask commands the temperatures. The Function Block diagrams set and maintain those temperatures.
3. This task (continuous) executes the sequential function chart (SFC). The SFC commands the specific state or temperature for each device or temperature loop.

Choose how to execute the SFC

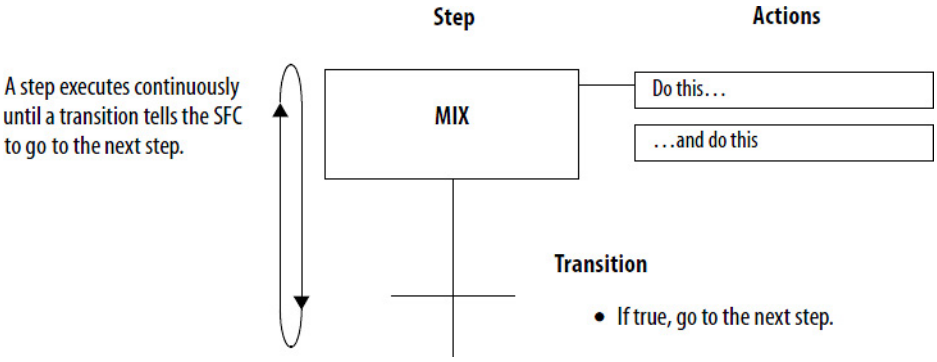
To execute an SFC, either configure it as the main routine for a program or call it as a subroutine.

If	Then
• The SFC is the only routine in the program • The SFC calls *all* the other routines of the program	Configure the SFC as the main routine for the program.
• The program requires other routines to execute independent of the SFC • The SFC uses Boolean actions	1. Configure another routine as the main routine for the program. 2. Use the main routine to call the SFC as a subroutine.

If the SFC uses Boolean actions, then other logic must run independent of the SFC and monitor status bits of the SFC.

Define the steps of the process

A step represents a major function of your process. It contains the actions that occur at a particular time, phase, or station.

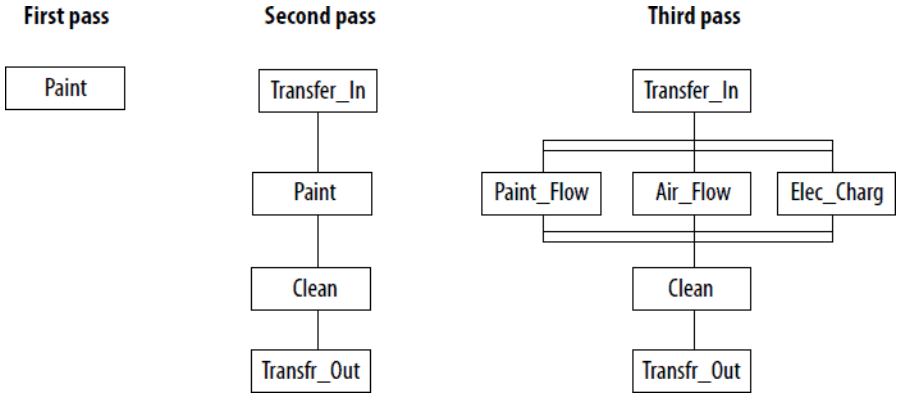


A transition ends a step. The transition defines the physical conditions that must occur or change in order to go to the next step.

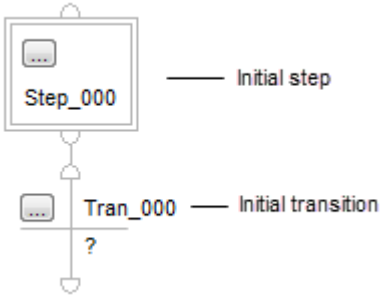
Step guidelines

Follow these guidelines.

- Start with large steps and refine the steps in several passes.



- When you first open an SFC routine, it contains an initial step and transition. Use this step to initialize your process.



The controller executes the initial step in these situations.

- After a project download when the controller goes into Run mode.
- When the controller transitions to Run mode and on power-up (if the SFC is configured for that).
- When the routine containing the chart is modified online and a reset is required, and the controller transitions to or from Test mode.

- To identify a step, look for a physical change in your system, such as new part that is in position, a temperature that is reached, a preset time that is reached, or a recipe selection that occurs. The step is the actions that take place before that change.
- Stop refining the steps when they are in meaningful increments. This is an example.

This organization of steps	Is
produce_solution	Probably too large
set_mode, close_outlet, set_temperature, open_inlet_a, close_inlet_a, set_timer, reset_temperature, open_outlet, reset_mode	Probably too small
preset_tank, add_ingredient_a, cook, drain	Probably about right

SFC_STEP structure

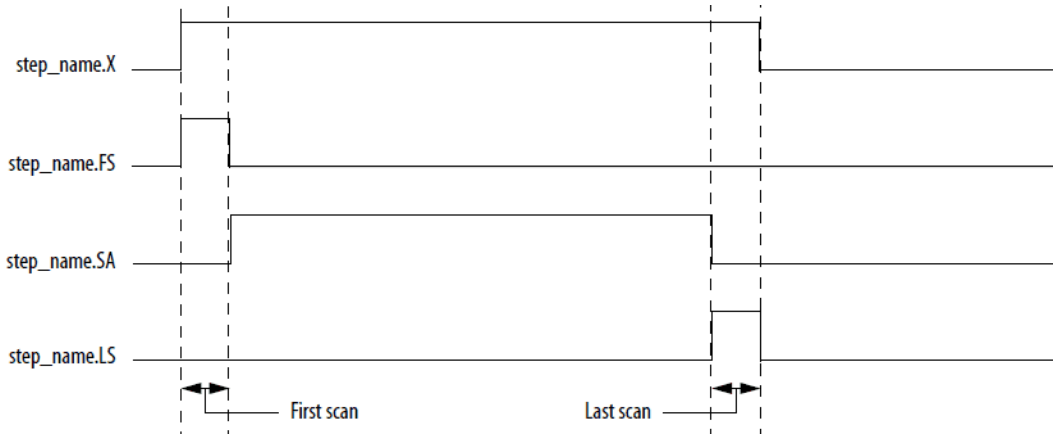
Each step uses a tag to provide information about the step. Access this information with either the **Step Properties** dialog box or the **Monitor Tags** tab of the **Tags** window.

If you want to	Then select or set this member	Data type	Details
Determine how long a step has been active (milliseconds)	T	DINT	When a step becomes active, the **Timer** (T) value resets and then starts to count up in milliseconds. The Timer continues to count up until the step goes inactive, regardless of the **Preset** (PRE) value.
Set a flag when the step has been active for a specific length of time (milliseconds)	PRE	DINT	Enter the time in the **Preset** (PRE) member. When the **Timer** (T) reaches the Preset value, the Done (DN) bit turns on and stays on until the step becomes active again. As an option, select **Use Expression** and click **Define** to enter a numeric expression that calculates the time at runtime.
	DN	BOOL	When the **Timer** (T) reaches the **Preset** (PRE) value, the Done (DN) bit turns on and stays on until the step becomes active again.
Set a flag if a step did not execute long enough	LimitLow	DINT	Enter the time in the **Limit Low** (LimitLow) member (milliseconds). • If the step goes inactive before the **Timer** (T) reaches the LimitLow value, the AlarmLow bit turns on. • The AlarmLow bit stays on until you reset it. • To use this alarm function, turn on (select) the **Alarm Enable** (AlarmEn) bit. As an option, enter a numeric expression that calculates the time at runtime.
	AlarmEn	BOOL	To use the alarm bits, turn on (select) the **Alarm Enable** (AlarmEn) bit.
	AlarmLow	BOOL	If the step goes inactive before the **Timer** (T) reaches the **Limit Low** value, the AlarmLow bit turns on. The bit stays on until you reset it. To use this alarm function, turn on (select) the **Alarm Enable** (AlarmEn) bit.
Set a flag if a step is executing too long	LimitHigh	DINT	Enter the time in the **Limit High** member (milliseconds). • If the **Timer** (T) reaches the LimitHigh value, the AlarmHigh bit turns on. • The AlarmHigh bit stays on until you reset it. • To use this alarm function, turn on (select) the **Alarm Enable** (AlarmEn) bit. As an option, enter a numeric expression that calculates the time at runtime.
	AlarmEn	BOOL	To use the alarm bits, turn on (select) the **Alarm Enable** (AlarmEn) bit.
	AlarmHigh	BOOL	If the **Timer** (T) reaches the **Limit High** value, the AlarmHigh bit turns on. The bit stays on until you reset it. To use this alarm function, turn on (select) the **Alarm Enable** (AlarmEn) bit.
Do something while the step is active (including first and last scan)	X	BOOL	The X bit is on the entire time the step is active (executing). Typically, we recommend that you use an action with a N Non-Stored qualifier to accomplish this.
Do something one time when the step becomes active	FS[1]	BOOL	The FS bit is on during the first scan of the step. Typically, we recommend that you use an action with a P1 Pulse (Rising Edge) qualifier to accomplish this.

If you want to	Then select or set this member	Data type	Details
Do something while the step is active, *except* on the first and last scan	SA	BOOL	The SA bit is on when the step is active except during the first and last scan of the step.
Do something one time on the last scan of the step	LS[1]	BOOL	The LS bit is on during the last scan of the step. Use this bit only if on the **Controller Properties** dialog box, **SFC Execution** tab, you set the **Last Scan of Active Step** to **Don't Scan** or **Programmatic reset**. Typically, we recommend that you use an action with a PO Pulse (Falling Edge) qualifier to accomplish this.
Determine the target of an SFC Reset (SFR) instruction	Reset	BOOL	An SFC Reset (SFR) instruction resets the SFC to a step or stop that the instruction specifies. The Reset bit indicates the step or stop where the SFC goes to begin executing again. Once the SFC executes, the Reset bit clears.
Determine the maximum time that a step has been active during any of its executions	TMax	DINT	Use this for diagnostic purposes. The controller clears this value only when you set the **Restart Position** to **Restart at initial step** and the controller changes modes or experiences a power cycle.
Determine if the Timer (T) value rolls over to a negative value	OV	BOOL	Use this for diagnostic purposes.
Determine how many times a step has become active	Count	DINT	This is not a count of scans of the step. • The count increments each time the step becomes active. • It increments again only after the step goes inactive and then active again. • The count resets only if you configure the SFC to restart at the initial step. With that configuration, it resets when the controller changes from program mode to run mode.
Use one tag for the various status bits of this step	Status	DINT	**For this member** / **Use this bit** Reset: 22 AlarmHigh: 23 AlarmLow: 24 AlarmEn: 25 OV: 26 DN: 27 LS: 28 SA: 29 FS: 30 X: 31

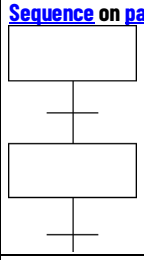
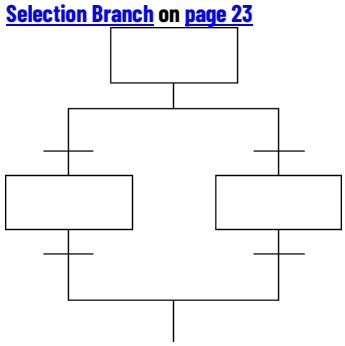
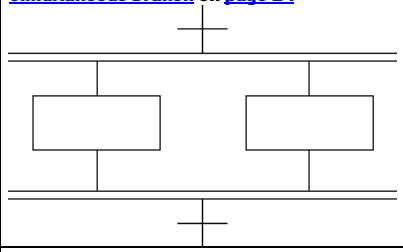
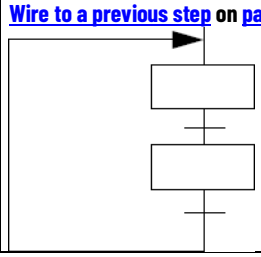
1. The FS and LS bits are only active during a step's execution. Once a step finishes executing the code within its actions, the FS or the LS or both bits are reset. If you reference either of these bits in code outside of the SFC routine in a different part of the project, the bits are always cleared (0).

This diagram shows the relationship of the X, FS, SA, and LS bits.

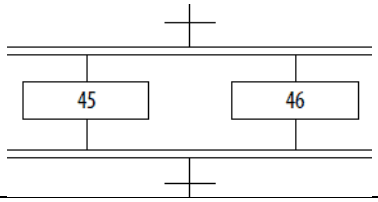
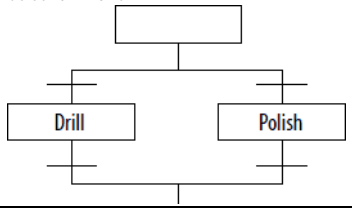
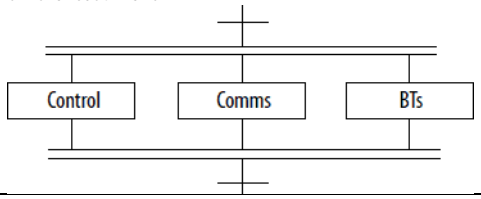
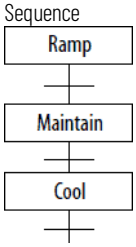
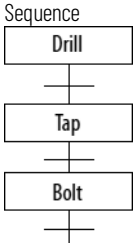
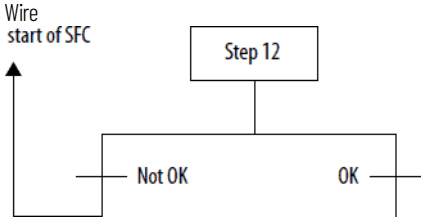


Organize the steps

Once you define the steps of your process, organize them into sequences, simultaneous branches, selection branches, or loops.

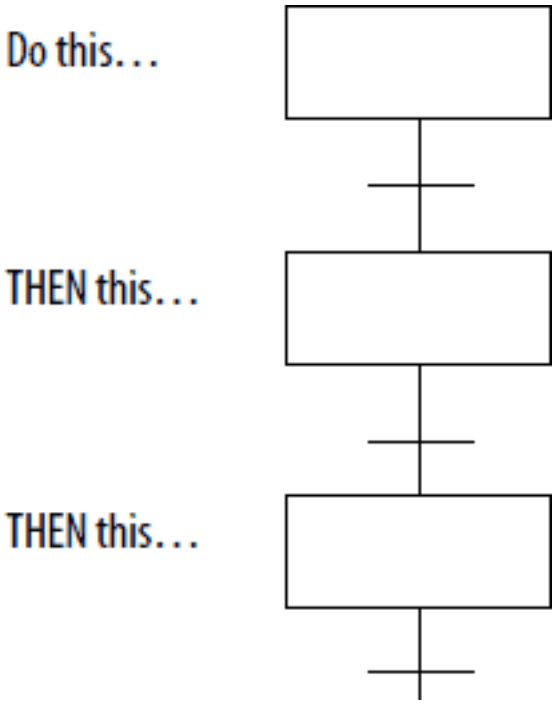
To	Use this structure	With these considerations
Execute 1 or more steps in sequence • One executes repeatedly • Then the next executes repeatedly	Sequence on page 23 	The SFC checks the transition at the end of the step. • If TRUE the SFC goes to the next step. • If FALSE, the SFC repeats the step.
• Choose between alternative steps or groups of steps depending on logic conditions • Execute a step or steps or skip the step or steps depending on logic conditions	Selection Branch on page 23 	• It is OK for a path to have no steps and only a transition. This lets the SFC skip the selection branch. • By default, the SFC checks from left to right the transitions that start each path. It takes the first TRUE path. • If no transitions are TRUE, the SFC repeats the previous step. • The Logix Designer application lets you change the order in which the SFC checks the transitions.
Execute 2 or more steps at the same time. All paths must finish before continuing the SFC	Simultaneous Branch on page 24 	• A single transition ends the branch. • The SFC checks the ending transition after the last step in each path has executed at least once. If the transition is FALSE, the SFC repeats the previous step.
Loop back to a previous step	Wire to a previous step on page 25 	• Connect the wire to the step or simultaneous branch to which you want to go. • Do not wire into, out of, or between a simultaneous branch.

Here are some examples of SFC structures for different situations.

Example situation	Example solution
Station 45 and 46 of an assembly line work on parts simultaneously. When both stations are done, the parts move down 1 station.	Simultaneous Branch 
Depending on the build code, a station either drills or polishes.	Selection Branch 
To simplify my programming, I want to separate communications and block transfers from other control logic. All occur at the same time.	Simultaneous Branch 
In a heat treating area, the temperature ramps up at a specific rate, maintains that temperature for a specific duration, and then cools at a specific rate.	Sequence 
At station 12, the machine drills, taps, and bolts a part. The steps occur one after the other.	Sequence 
Step 12 inspects a process for the correct mix of chemicals. • If OK, then continue with the remaining steps. • If not OK, go to the top of the SFC and purge the system.	Wire 

Sequence

A sequence is a group of steps that execute one after the other.

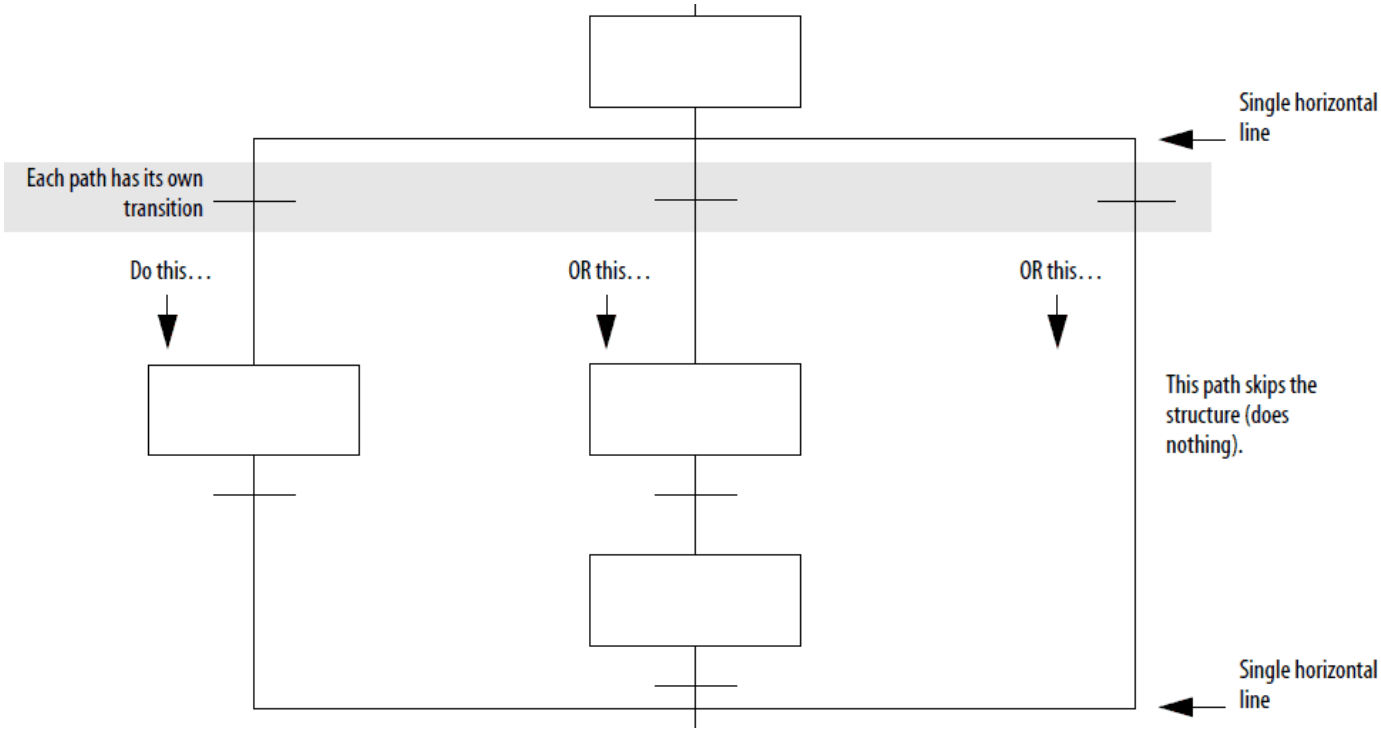


Selection branch

A selection branch represents a choice between one path (step or group of steps) or another path (an OR structure).

- Only one path executes.
- By default the SFC checks the transitions from left to right.
  - The SFC takes the first TRUE path.

- The Logix Designer application lets you change the order in which the SFC checks the transitions (see Chapter 2, *Program a Sequential Function Chart*).



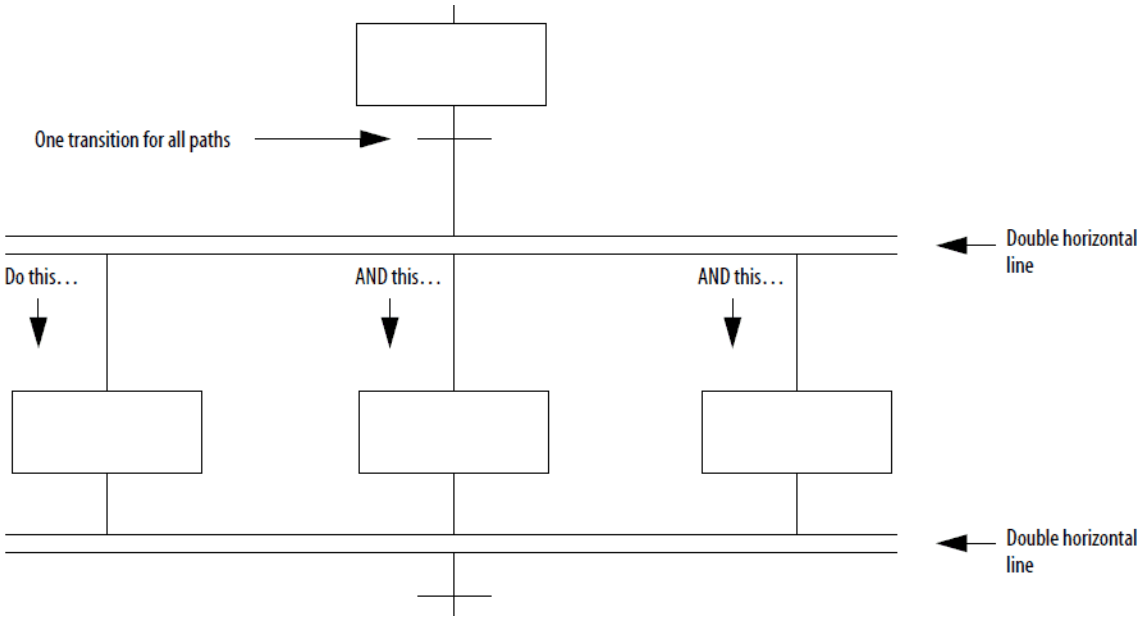
See also

Program a Sequential Function Chart on page 53

Simultaneous branch

A simultaneous branch represents paths (steps or group of steps) that occur at the same time (an AND structure).

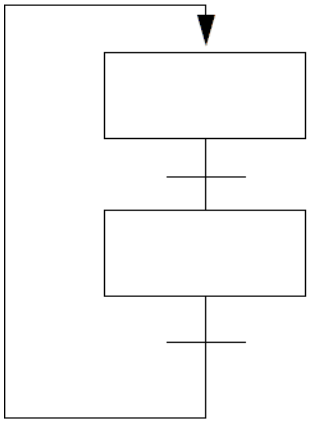
- All paths execute.
- All paths must finish before continuing with the SFC.
- The SFC checks the transition after the last step of each path has executed at least once.



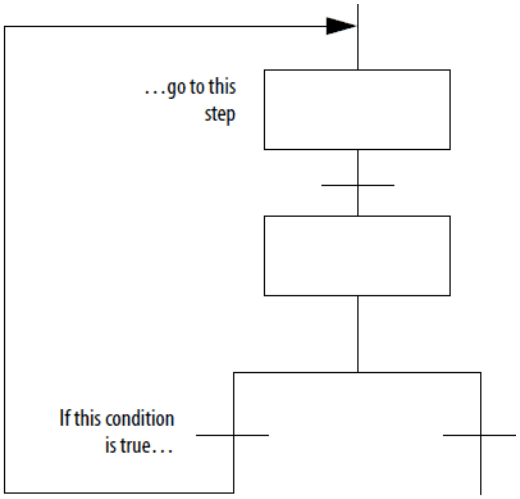
Wire to a previous step

You can also connect a step to a previous point in your SFC.

- Loop back and repeat steps
- Return to the beginning of the SFC and start over



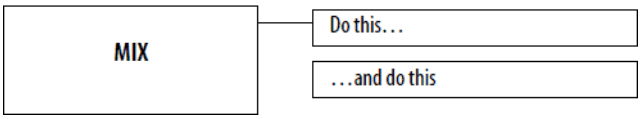
Simple loop that repeats the entire SFC



Path of a selection branch that returns to a previous step

Add actions for each step

Use action**s** to divide a step into the different functions that the step performs, such as commanding a motor, setting the state of a valve, or placing a group of devices in a specific mode.



How do you want to use the action?

There are two types of actions.

If you want to	Then
• Execute structured text directly in the SFC • Call a subroutine • Use the automatic reset option to reset data upon leaving a step	Use a non-Boolean action
• Only set a bit and program other logic to monitor the bit to determine when to execute.	Use a Boolean action

See also

Use a non-Boolean action on page 26

Use a Boolean action on page 27

Use a non-Boolean action

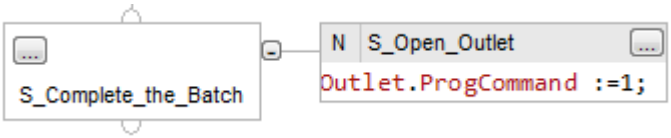
A non-Boolean action contains the logic for the action. It uses structured text to execute assignments and instructions or call a subroutine.

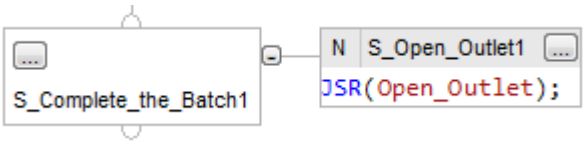
With non-Boolean actions, you also have the option to postscan (automatically reset) the assignments and instructions before leaving a step.

- During postscan the controller executes the assignments and instructions as if all conditions are FALSE.
- The controller postscans both embedded structured text and any subroutine that the action calls.

To automatically reset assignments and instructions, see *Turn off a device at the end of a step*.

To program a non-Boolean action, you have these options.

If you want to	Then
• Execute your logic without additional routines • Use structured text assignments, constructs, and instructions	Embed structured text.  When the S_Complete_the_Batch step is active, the S_Open_Outlet action executes. The action sets the Outlet.ProgCommand tag equal to 1, which opens the outlet valve.

If you want to	Then
• Re-use logic in multiple steps • Use another language to program the action, such as ladder logic • Nest an SFC	Call a subroutine.  When the S_Complete_the_Batch step is active, the S_Open_Outlet action executes. The action calls the Open_Outlet routine. **Open_Outlet Routine**  When the Open_Outlet routine executes, the OTE instruction sets the Outlet.ProgCommand tag equal to 1, which opens the outlet valve.

You cannot reuse a non-Boolean action within the same SFC except to reset a stored action. Only one instance of a specific non-Boolean action is permitted per SFC.

See also

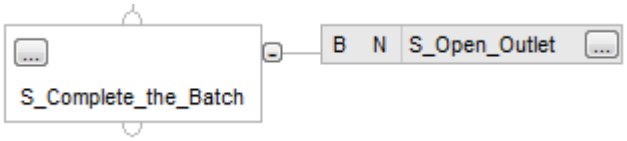
Turn off a device at the end of a step on page 35

Use a Boolean action

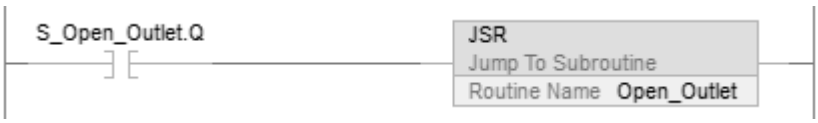
A Boolean action contains no logic for the action. It simply sets a bit in its tag (SFC_ACTION structure). To do the action, other logic must monitor the bit and execute when the bit is on.

With Boolean actions, you have to manually reset the assignments and instructions that are associated with the action. Since there is no link between the action and the logic that performs the action, the automatic reset option does not affect Boolean actions.

Example



When the S_Complete_the_Batch step is active, the S_Open_Outlet action executes. When the action is active, its Q bit turns on.



A ladder Logic routine monitors the Q bit (S_Open_Outlet.Q). When the Q bit is on, the JSR instruction executes and opens the outlet valve.

You can reuse a Boolean action multiple times within the same SFC.

SFC_ACTION structure

Each action (non-Boolean and Boolean) uses a tag to provide information about the action. Access this information via either the **Action Properties** dialog box or the **Monitor Tags** tab of the **Tags** window.

If you want to	Then select or set this member	Data type	Details
Determine when the action is active	Q	BOOL	The status of the Q bit depends on whether the action is a Boolean action or non-Boolean action. **If the action is** / **Then the Q bit is** Boolean: On (1) the entire time the action is active, including the last scan of the action Non-Boolean: On (1) while the action is active but Off (0) at the last scan of the action To use a bit to determine when an action is active, use the Q bit.
	A	BOOL	The A bit is on the entire time the action is active.
Determine how long an action has been active (milliseconds)	T	DINT	When an action becomes active, the **Timer** (T) value resets and then starts to count up in milliseconds. The timer continues to count up until the action goes inactive, regardless of the **Preset** (PRE) value.
Use one of these time-based qualifiers: L, SL, D, DS, SD	PRE	DINT	Enter the time limit or delay in the **Preset** (PRE) member. The action starts or stops when the **Timer** (T) reaches the Preset value. As an option, enter a numeric expression that calculates the time at runtime.
Determine how many times an action has become active	Count	DINT	This is not a count of scans of the action. • The count increments each time the action becomes active. • It increments again only after the action goes inactive and then active again. • The count resets only if you configure the SFC to restart at the initial step. With that configuration, it resets when the controller changes from program mode to run mode.
Use one tag for the various status bits of this action	Status	DINT	**For this member** / **Use this bit** Q: 30 A: 31

Describe each action in pseudocode

To organize the logic for an action, first you describe the action in pseudocode.

- Use a series of short statements that describe what should happen.
- Use terms or symbols, such as: if, then, otherwise, until, and, or, =, >, <.
- Sequence the statements in the order that they should execute.
- If necessary, name the conditions to check first (the "when to act" first) and then the action to take second (the "what to do" second).

Enter the pseudocode into the body of the action.

- Refine the pseudocode so it executes as structured text.
- Use the pseudocode to design your logic and leave the pseudocode as comments. Since all structured text comments download to the

controller, your pseudocode is always available as documentation for the action.

To convert the pseudocode to structured text comments, add these comment symbols.

For a comment	Use one of these formats
On a single line	*// comment*

That spans more than one line	`(*start of comment . . . end of comment*)`
	
	`/*start of comment . . . end of comment*/`

Choose a qualifier for an action

Each action (non-Boolean and Boolean) uses a qualifier to determine when it starts and stops.

The default qualifier is N Non-Stored. The action starts when the step is activated and stops when the step is deactivated.

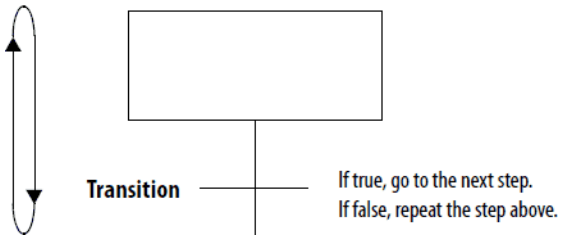
To change when an action starts or stops, assign a different qualifier.

If you want the action to	And	Then assign this qualifier	Which means
Start when the step is activated	Stop when the step is deactivated	N	Non-Stored
	Execute only once	P1	Pulse (Rising Edge)
	Stop before the step is deactivated or when the step is deactivated	L	Time Limited
	Stay active until a Reset action turns off this action	S	Stored
	Stay active until a Reset action turns off this action Or a specific time expires, even if the step is deactivated	SL	Stored and Time Limited
Start a specific time after the step is activated and the step is still active	Stop when the step is deactivated	D	Time Delayed
	Stay active until a Reset action turns off this action	DS	Delayed and Stored
Start a specific time after the step is activated, even if the step is deactivated before this time	Stay active until a Reset action turns off this action	SD	Stored and Time Delayed
Execute once when the step is activated	Execute once when the step is deactivated	P	Pulse
Start when the step is deactivated	Execute only once	P0	Pulse (Falling Edge)
Turn off (reset) a stored action • S Stored • SL Stored and Time Limited • DS Delayed and Stored • SD Stored and Time Delayed	---------->	R	Reset

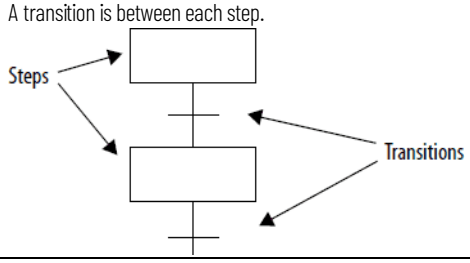
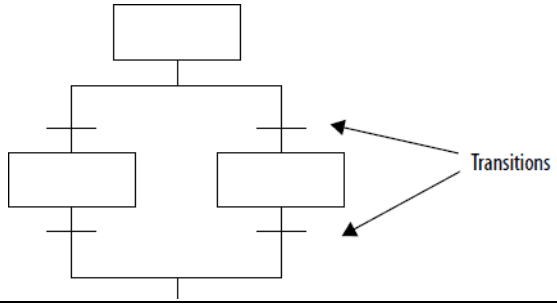
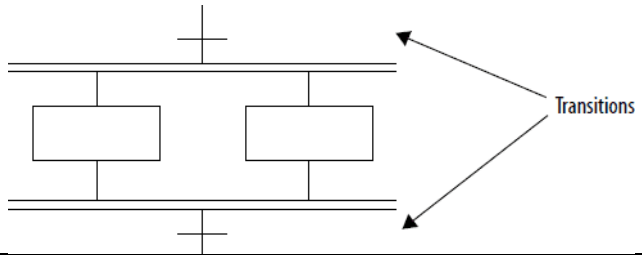
Define the transition conditions

The transition is the physical conditions that must occur or change in order to go to the next step.

The transition tells the SFC when to go to the next step.



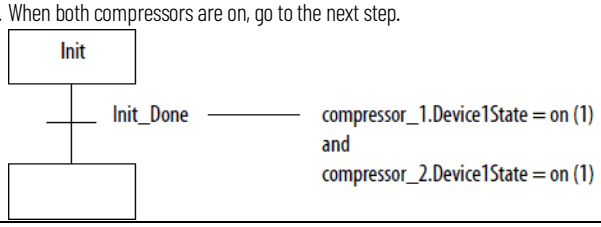
Transitions occur in these structures.

For this structure	Make sure that
Sequence	A transition is between each step. 
Selection branch	Transitions are inside the horizontal lines. 
Simultaneous branch	Transitions are outside the horizontal lines. 

Example

You want to complete these steps.

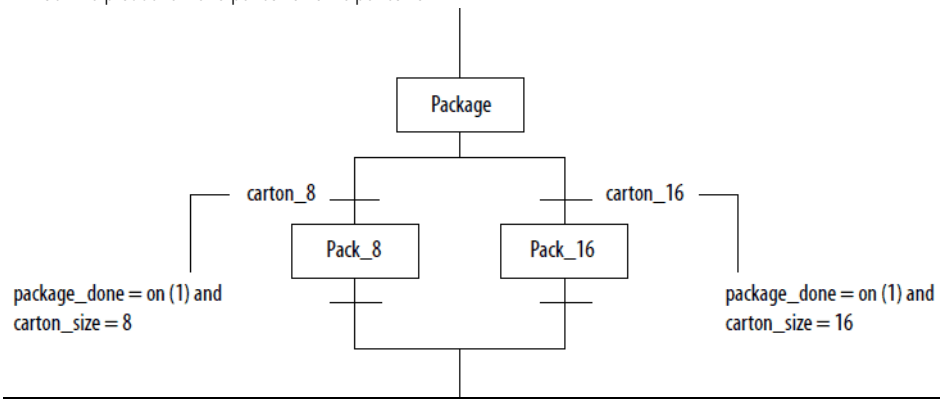
1. Turn on 2 compressors. When a compressor is on, the Device1State bit is on.
2. When both compressors are on, go to the next step.



Example

You want to complete these steps.

1. Package the product. When the product is in the package, the package_done bit turns on.
2. Pack the product either 8 per carton or 16 per carton.



Transition tag

Each transition uses a BOOL tag to represent the TRUE or FALSE state of the transition.

If the transition is	The value is	And
True	1	The SFC goes to the next step.
False	0	The SFC continues to execute the current step.

How do you want to program the transition?

To program the transition, you have the following options.

If you want to	Then
Enter the conditions as an expression in structured text.	Use a BOOL expression
Enter the conditions as instructions in another routine.	Call a subroutine
Use the same logic for multiple transitions.	

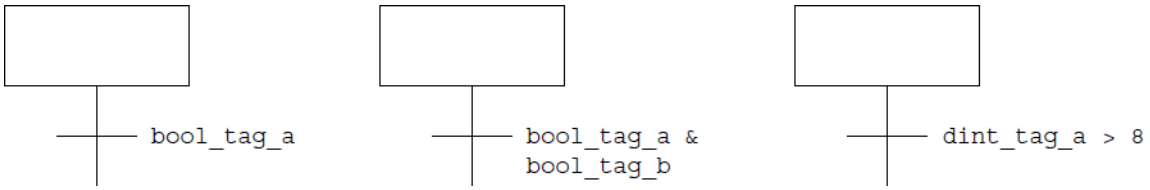
See also

Use a BOOL expression on page 31

Call a subroutine on page 64

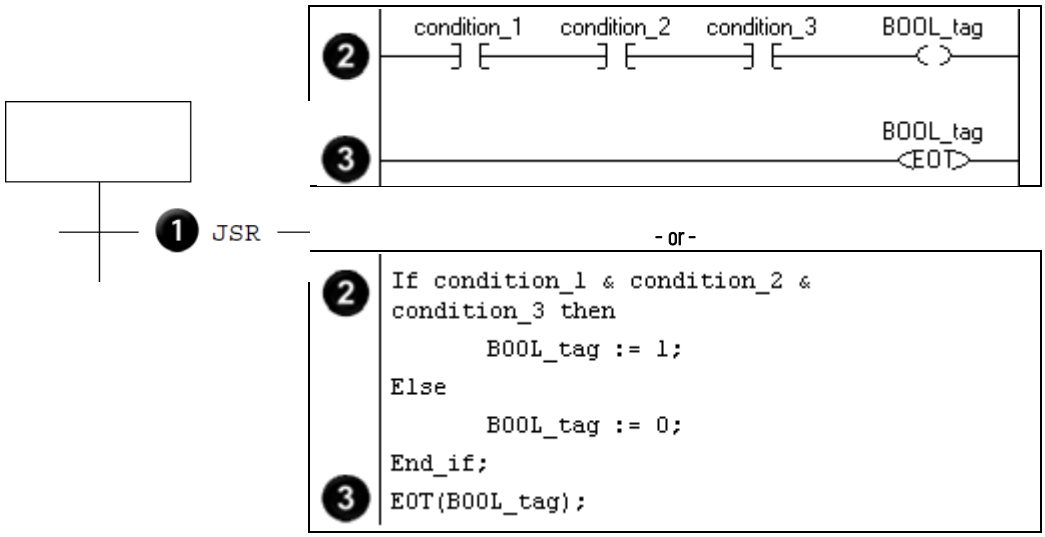
Use a BOOL expression

The simplest way to program the transition is to enter the conditions as a BOOL expression in structured text. A BOOL expression uses BOOL tags, relational operators, and logical operators to compare values or check if conditions are TRUE or FALSE. For example, `tag1>65`.



Call a subroutine in a transition

To use a subroutine to control a transition, include an End Of Transition (EOT) instruction in the subroutine. The EOT instruction returns the state of the conditions to the transition, as shown in this example.



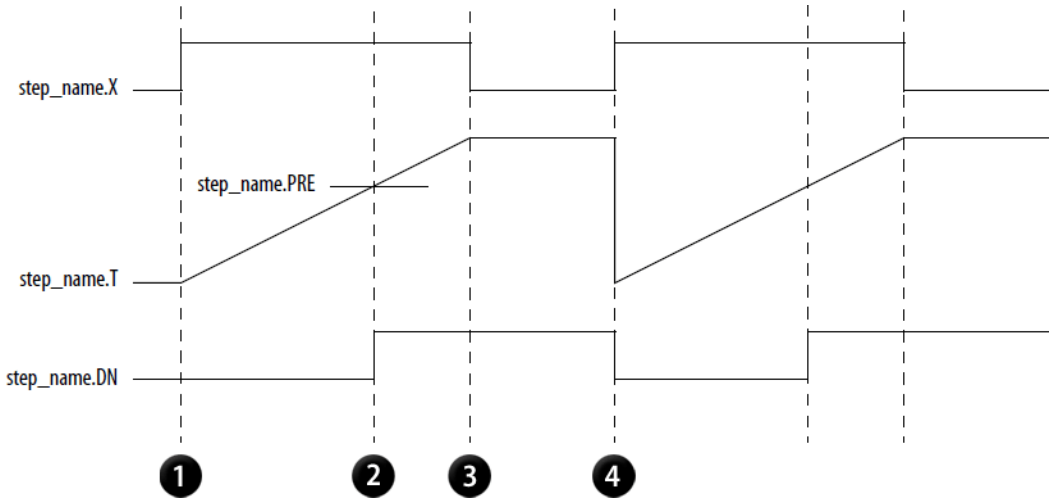
1	Call a subroutine.
2	Check for the required conditions. When those conditions are TRUE, turn on a BOOL tag.
3	Use an EOT instruction to set the state of the transition equal to the value of the BOOL tag. When the BOOL tag is on (TRUE), the transition is TRUE.

Transition after a specified time

Each step of the SFC includes a millisecond timer that runs whenever the step is active. Use the timer to for these situations.

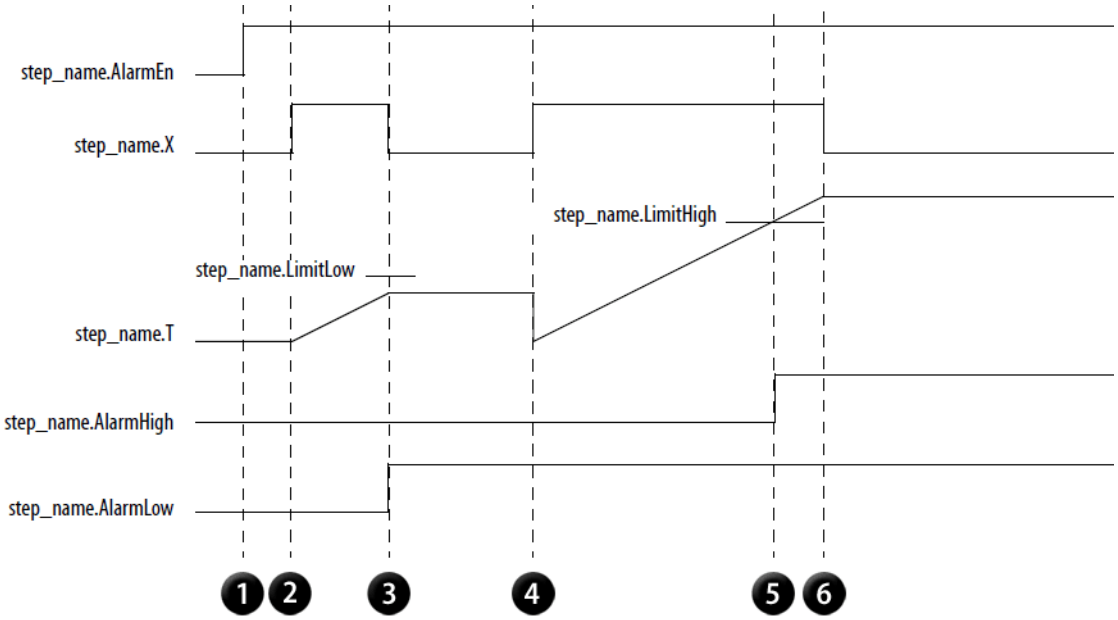
- Signal when the step has run for the required time and the SFC should go to the next step.
- Signal when the step has run too long and the SFC should go to an error step.

The following shows the action of a timer and associated bits of a step.



	Description
1	Step becomes active. X bit turns on. Timer (T) begins to increment.
2	Timer reaches the Preset (PRE) value of the step. DN bit turns on. Timer continues to increment.
3	Step becomes inactive. X bit turns off. Timer retains its value. DN remains on.
4	Step becomes active. X bit turns on. Timer clears and then begins to increment. DN bit turns off.

The following shows the action of the low and high alarms for a step.

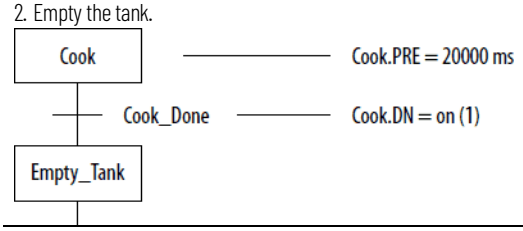


	Description
1	AlarmEn is on. To use the low and high alarms turn this bit on. Turn the bit on via the properties dialog box or the tag for the step.
2	Step becomes active. X bit turns on. Timer (T) begins to increment.
3	Step becomes inactive. X bit turns off. Timer retains its value. Since Timer is less than LimitLow, AlarmLow bit turns on.
4	Step becomes active. X bit turns on. Timer clears and then begins to increment. AlarmLow stays on. (You have to manually turn it off.)
5	Timer reaches the LimitHigh value of the step. AlarmHigh bit turns on. Timer continues to increment.
6	Step becomes inactive. X bit turns off. Timer retains its value. AlarmHigh stays on. (You have to manually turn it off.)

Example

Here is an example of the use of the Preset time of a step. The functional specification has these requirements.

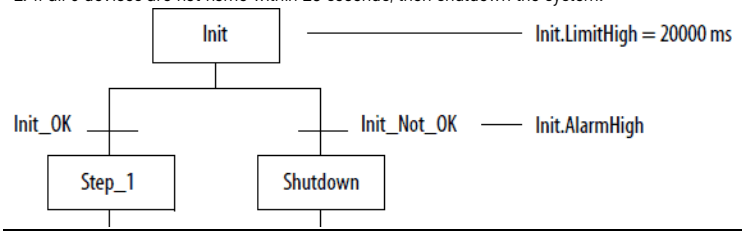
1. Cook the ingredients in the tank for 20 seconds.
2. Empty the tank.



Example

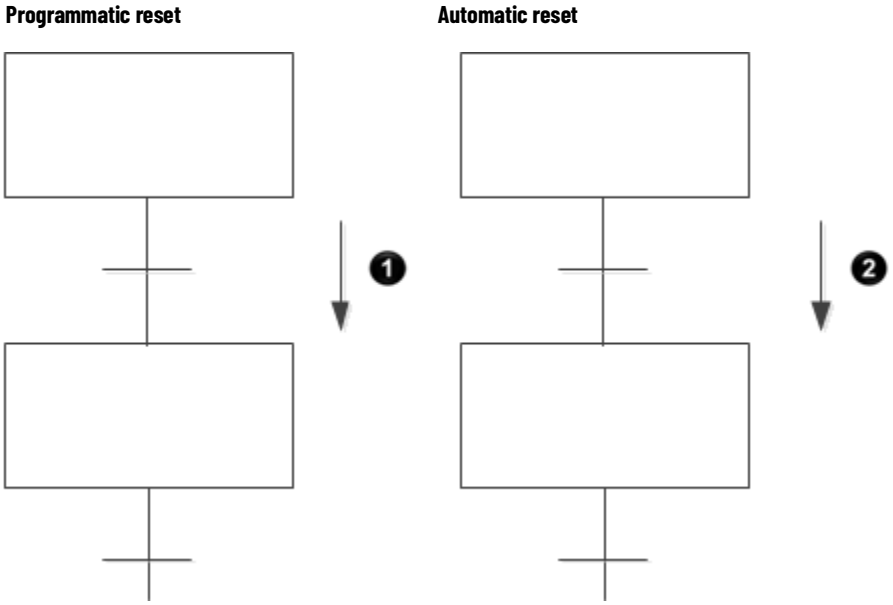
Here is an example of the use of the high alarm of a step. The functional specification has these requirements.

1. Home 8 devices.
2. If all 8 devices are not home within 20 seconds, then shutdown the system.



Turn off a device at the end of a step

When the SFC leaves a step, you have several options on how to turn off devices that the step turned on.



1	Use logic to clear data
2	Let the controller automatically clear data

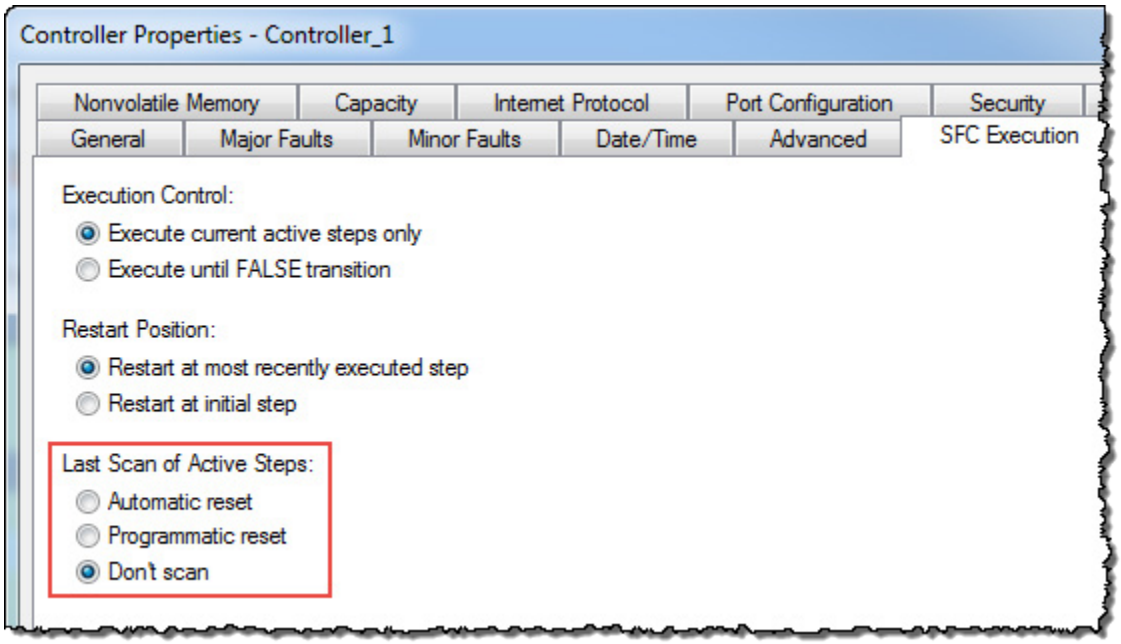
Each option requires you to make these decisions.

- Choose a last scan option.
- Based on the last scan option, develop your logic so that the last scan returns data to the correct values.

Choose a last scan option

On the last scan of each step, you have these options. The option that you choose applies to all the steps in all the SFCs in this controller.

If you want to	And on the last scan of a step	Then see
Control which data to clear	Execute only P and PO actions and use them to clear the required data.	Use the don't scan option
	Execute all actions and use either of these options to clear the required data. • Status bits of the step or action to condition logic • P and PO actions	Use the programmatic reset option
Let the controller clear data	---------->	Use the automatic reset option



The following table compares the different options for handling the last scan of a step.

Characteristic	During the last scan of a step, this option does		
	Don't scan	**Programmatic reset**	**Automatic reset**
Execution actions	Only P and PO actions execute. They execute according to their logic.	All actions execute according to their logic.	• P and PO actions execute according to their logic. • All other actions execute in Postscan mode. • On the next scan of the routine, the P and PO actions execute in Postscan mode.
Retention of data values	All data keeps its current values.	All data keeps its current values.	• Data reverts to its values for postscan. • Tags to the left of [:=] assignments clear to zero.
Method for clearing data	Use P and PO actions.	Use either of these. • Status bits of the step or action to condition logic • P and PO actions	Use either of these. • [:=] assignment (non-retentive assignment) • Instructions that clear their data during postscan
Reset of a nested SFC	A nested SFCs remains at its current step.	A nested SFCs remains at its current step.	For the **Restart Position** property, if you choose the **Restart at initial step option**, then these occur. • A nested SFC resets to its initial step • The X bit of a stop element in a nested SFC clears to zero

See also

Use the don't scan option on page 37

Use the programmatic reset option on page 37

Use the automatic reset option on page 39

Use the Don't Scan option

The default option for handling the last scan of a step is **Don't scan**. With this option, all data keeps its current values when the SFC leaves a step. This requires you to use additional assignments or instructions to clear any data that you want to turn off at the end of a step.

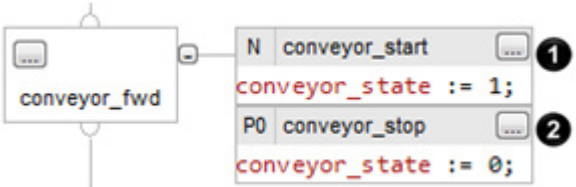
To turn off a device at the end of a step using the Don't Scan option

1. Make sure that the **Last Scan of Active Steps** property is set to the **Don't scan** option (default).
2. Use a PO Pulse (Falling Edge) action to clear the required data. Make sure that the PO action or actions are last in the order of actions for the step.

   During the last scan of the step, the **Don't scan** option executes only P and PO actions. The assignments and instructions of the actions execute according to their logic conditions.

   - The controller **does not** execute a postscan of assignments or instructions.
   - When the SFC leaves the step, all data keeps its current values.

   This example uses an action to turn on a conveyor at the start of a step. A different action turns off the conveyor at the end of the step.



1	This action turns on the conveyor. When conveyor_state turns on, the conveyor turns on.
2	Before the SFC leaves the step, the PO action turns off the conveyor. On the last scan of the step, conveyor_state turns off. This turns off the conveyor.

Use the programmatic reset option

An optional method to programmatically turn off (clear) devices at the end of a step is to execute all actions on the last scan of the step. This lets you execute your normal logic as well as turn off (clear) devices at the end of a step.

1. In the **Last Scan of Active Steps** property, select the **Programmatic reset** option.
2. Clear the required data using any of these methods.

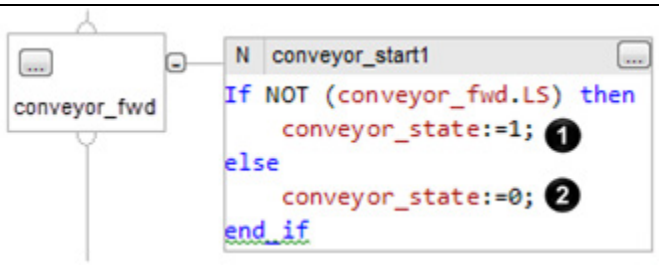
- To your normal logic, add logic that clears the required data. Use the LS bit of the step or the Q bit of the action to condition the execution of the logic.
- Use a P0 Pulse (Falling Edge) action to clear the required data. Make sure that the P0 action or actions are last in the order of actions for the step.

During the last scan of the step, the **Programmatic reset** option executes all assignments and instructions according to logic conditions.

- The controller does not postscan the assignments or instructions.
- When the SFC leaves the step, all data keeps its current value.

This example uses a single action to turn on and off a conveyor and the LS Bit. The LS bit of the step conditions the execution of the logic. See *SFC STEP Structure*.

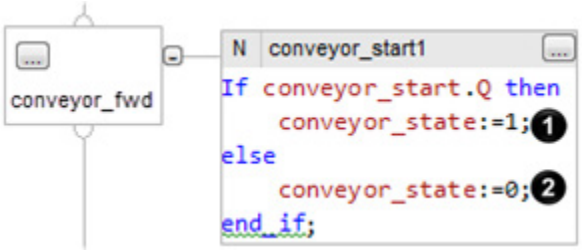
Example



1	When the step is not on its last scan (conveyor_fwd.LS = 0), this statement turns on conveyor_state. When conveyor_state turns on, the conveyor turns on.
2	On the last scan of the step (conveyor_fwd.LS =1), this statement turns off conveyor_state. When conveyor_state turns off, the conveyor turns off.

For an action that uses one of the stored qualifiers, use the Q bit of the action to condition your logic.

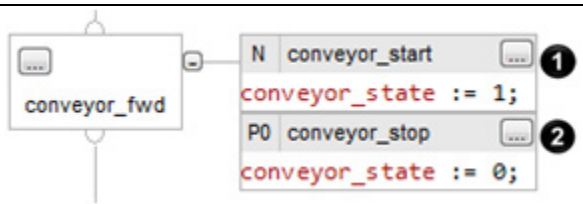
Example



1	When the action is not on its last scan (conveyor_start.Q =1), this statement turns on conveyor_state. When conveyor_state turns on, the conveyor turns on.
2	On the last scan of the action (conveyor_start.Q =0), this statement turns off conveyor_state. When conveyor_state turns off, the conveyor turns off.

You can also use a P0 Pulse (Falling Edge) action to clear data. This example uses an action to turn on a conveyor at the start of a step. A different action turns off the conveyor at the end of the step.

Example



1	This action turns on the conveyor. When conveyor_state turns on, the conveyor turns on.
2	Before the SFC leaves the step, the P0 action turns off the conveyor. On the last scan of the step, conveyor_state turns off. This turns off the conveyor.

See also

SFC_STEP structure on page 19

Use the automatic reset option

Automatic reset provides a system-defined cleanup of actions (known as postscan) when they are shut down when any of the following occur.

- transition out of the associated step
- reset of a stored action
- reset of an SFC routine

Postscan is similar to prescan in that most instructions are executed as if they are FALSE. Some instructions have specific postscan behavior.

- In RLL, OTE instructions are turned off and non-retentive timers are reset.
- In structured text, the destination of a non-retentive assignment "[:=]" is cleared.
- A JSR instruction invokes its subroutine but parameters are not passed and the logic in the subroutine is executed in postscan mode.
- An Add-On Instruction executes its logic routine in postscan mode and then executes its postscan logic in normal mode (if a postscan routine is configured).
- Any nested SFC (SFC that an action calls as a subroutine) is reset.

IMPORTANT The postscan of an action actually occurs when the action goes from active to inactive. Depending on the qualifier of the action, the postscan could occur before or after the last scan of the step.

As a general rule, the postscan executes instructions as if all conditions are FALSE. For example, the Output Energize (OTE) instruction clears its data during postscan.

To automatically turn off, or clear, devices at the end of a step using the automatic reset option

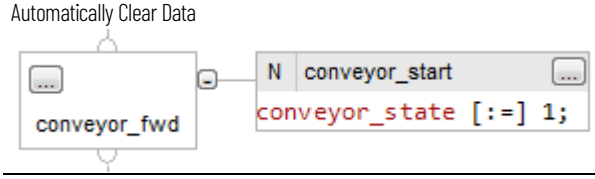
1. In the **Last Scan of Active Steps** property, select the **Automatic reset** option.
2. To turn off a device at the end of the step, control the state of the device with an assignment or instruction.
   - [:=] assignment (non-retentive assignment)
   - Output Energize (OTE) instruction in a subroutine

   Some instructions do not follow the general rule during postscan. For a description of how a specific instruction executes during postscan, see these publications.

   - *Logix 5000 Controllers General Instructions Reference Manual*, publication *1756-RM003*
   - *Advanced Process Control and Drives and Phase Sequence Instruction Reference Manual*, publication *1756-RM006*
   - *Logix 5000 Motion Controllers Instructions Reference Manual*, *MOTION-RM002*

   Here is an example that uses a non-retentive assignment to control a conveyor. It turns on a conveyor at the start of a step and automatically turns off the conveyor when the step is done.

Automatically Clear Data



- This action turns on the conveyor. When conveyor_state turns on, the conveyor turns on.
- When the SFC leaves the step, conveyor_state turns off. This turns off the conveyor.

See also

Logix 5000 Controllers General Instructions Reference Manual, publication 1756-RM003

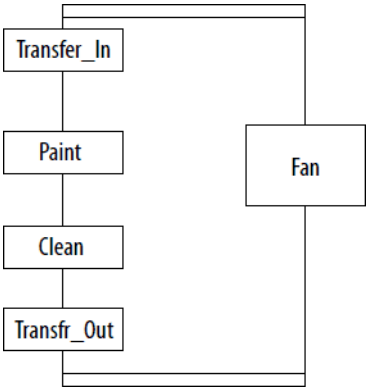
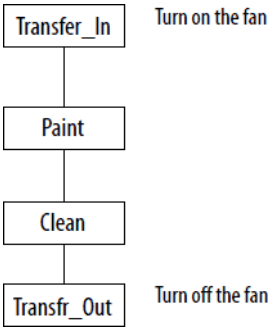
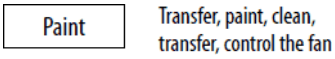
Advanced Process Control and Drives and Phase and Sequence Instruction Reference Manual, publication 1756-RM006

Logix 5000 Motion Controllers Instructions Reference Manual, publication MOTION-RM002

Keep something on from step-to-step

How do you want to control the device?

To provide bumpless control of a device during more than one time or phase (step), do one of the following options.

Option	Example
Use a simultaneous branch on page 41 Make a separate step that controls the device.	
Store and reset an action on page 42 Note the step that turns on the device and the step that turns off the device. Later, define a Stored and Reset Action pair to control the device.	
Use one large step on page 43 Use one large step that contains all the actions that occur while the device is on.	

Use a simultaneous branch

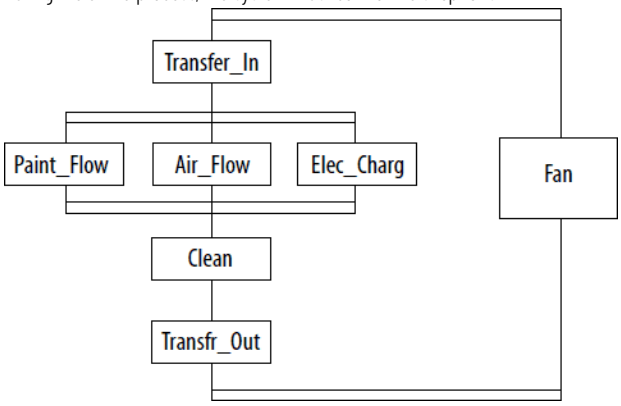
A simple way to control a device or devices during one or more steps is to create a separate step for the devices. Then use a simultaneous branch to execute the step during the rest of the process.

Example

A paint operation completes these actions.

1. Transfers the product into the paint shop.
2. Paints the product using 3 separate paint guns.
3. Cleans the guns.
4. Transfers the product to the paint ovens.

During the entire process, the system must control the shop fans.



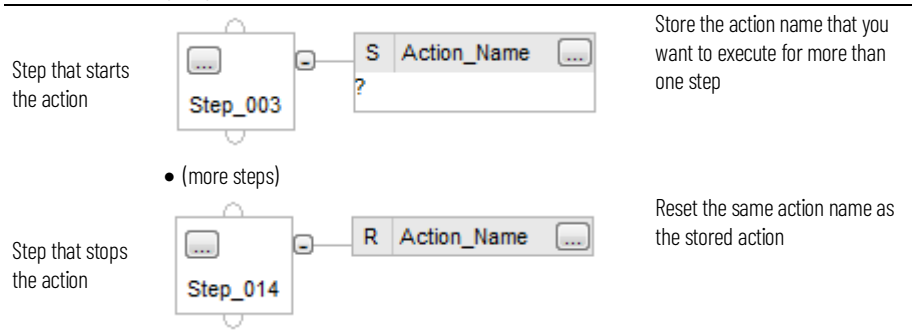
Store and reset an action

Typically, an action turns off (stops executing) when the SFC goes to the next step. To keep a device on from step to step without a bump, store the action that controls the device.

To store and reset an action

1. In the step that turns on the device, assign a stored qualifier to the action that controls the device.
2. In the step that turns off the device, use a Reset action.

The following figure shows the use of a stored action.



When the SFC leaves the step that stores the action, the Logix Designer application continues to show the stored action as active. By default, a green border displays around the action. This lets you know that the SFC is executing the logic of that action.

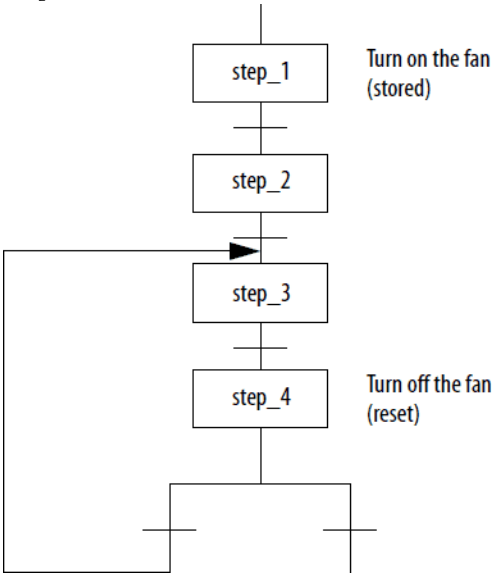
When using a stored action, follow these guidelines.

- The Reset action only turns off the stored action. It does not automatically turn off the devices of the action. To turn off the

device, follow the Reset action with another action that turns off the device. Or use the **Automatic reset** option described in *Use the automatic reset option*.

- Before the SFC reaches a stop element, reset any stored actions that you do not want to execute at the stop. An active stored action remains active even if the SFC reaches a stop.
- Use caution when you jump in between a step that stores an action and a step that resets the action. Once you reset an action, it only starts when you execute the step that stores the action.

In this example, step_1 – step_4 require a fan to be on. At the end of step_4, the fan is reset (turned off). When the SFC jumps back to step_3, the fan remains off.



To turn the fan back on, the SFC has to jump back to step_1.

See also

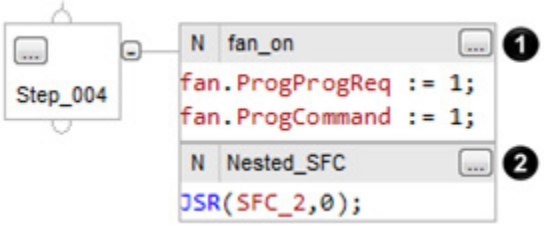
Use the automatic reset option on page 39

Use one large step

If you use one large step for multiple functions, then use additional logic to sequence the functions. One option is to nest an SFC within the large step.

In this example, a step turns on a fan and then calls another SFC. The nested SFC sequences the remaining functions of the step. The fan stays on throughout the steps of the nested SFC.

Use a Large Step



1	This action turns on a fan. • fan.ProgProgReq lets the SFC command the state of the fan. • fan.ProgCommand turns on the fan.
2	This action calls another SFC. The SFC sequences the remaining functions of the step.

End the SFC

Once an SFC completes its last step, it does not automatically restart at the first step. You must tell the SFC what to do when it finishes the last step.

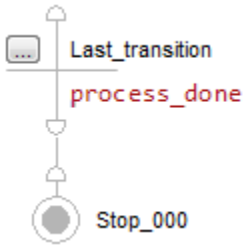
To	Do this
Automatically loop back to an earlier step	Wire the last transition to the top of the step to which you want to go.
Stop and wait for a command to restart	Use a stop element.

See also

Use a stop element on page 44

Use a stop element

Use the stop element to stop the execution of an entire SFC or of a path of a simultaneous branch and wait to restart.

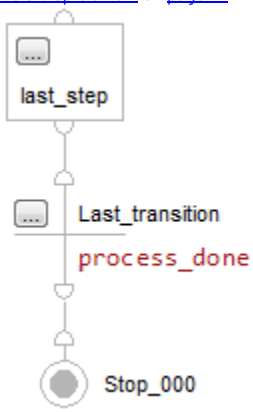


When an SFC reaches a stop element, the following actions occur.

- The X bit of the stop element turns on. This signals that the SFC is at the stop element.
- Stored actions remain active.
- Execution stops for part or all of the SFC.

If the stop element is at the end of a	Then
• Sequence • Selection branch	The entire SFC stops
• Path within a simultaneous branch	Only that path stops while the rest of the SFC continues to execute.

Example

Use a stop element on page 44 	When the SFC reaches last_step and process_done is TRUE, the execution of the SFC stops.

Restart (reset) the SFC

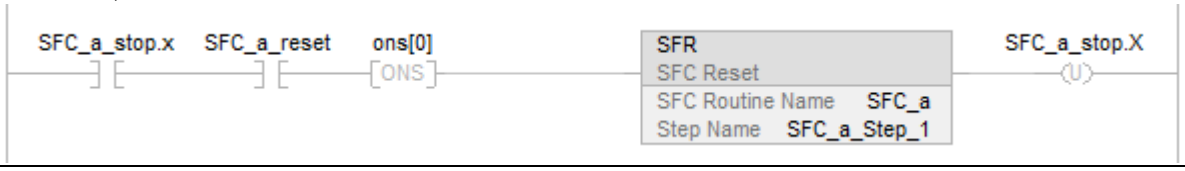
Once at the stop element, you have several options to restart the SFC.

If the SFC is	And the Last Scan of the Active Steps option is	Then
Nested (i.e., another SFC calls this SFC as a subroutine)	• Automatic reset	At the end of the step that calls the nested SFC, the nested SFC automatically resets. • The nested SFC resets to the initial step. • The X bit of the stop element in the nested SFC clears to zero.
	• Programmatic reset • Don't scan	1. Use an SFC Reset (SFR) instruction to restart the SFC at the required step. 2. Use logic to clear the X bit of the stop element.
Not nested (i.e., no SFC calls this SFC as a subroutine)	---------->	1. Use an SFC Reset (SFR) instruction to restart the SFC at the required step. 2. Use logic to clear the X bit of the stop element.

Example

This example shows the use of the SFC Reset (SFR) instruction to restart the SFC and clear the X bit of the stop element (see *Restart (reset) the SFC*).

If SFC_a_stop.X = on (SFC_a is at the stop) and SFC_a_reset = on (time to reset the SFC) then for one scan (ons[0] = on):
Reset SFC_a to SFC_a_Step_1
SFC_a_stop.X = 0



SFC_STOP structure

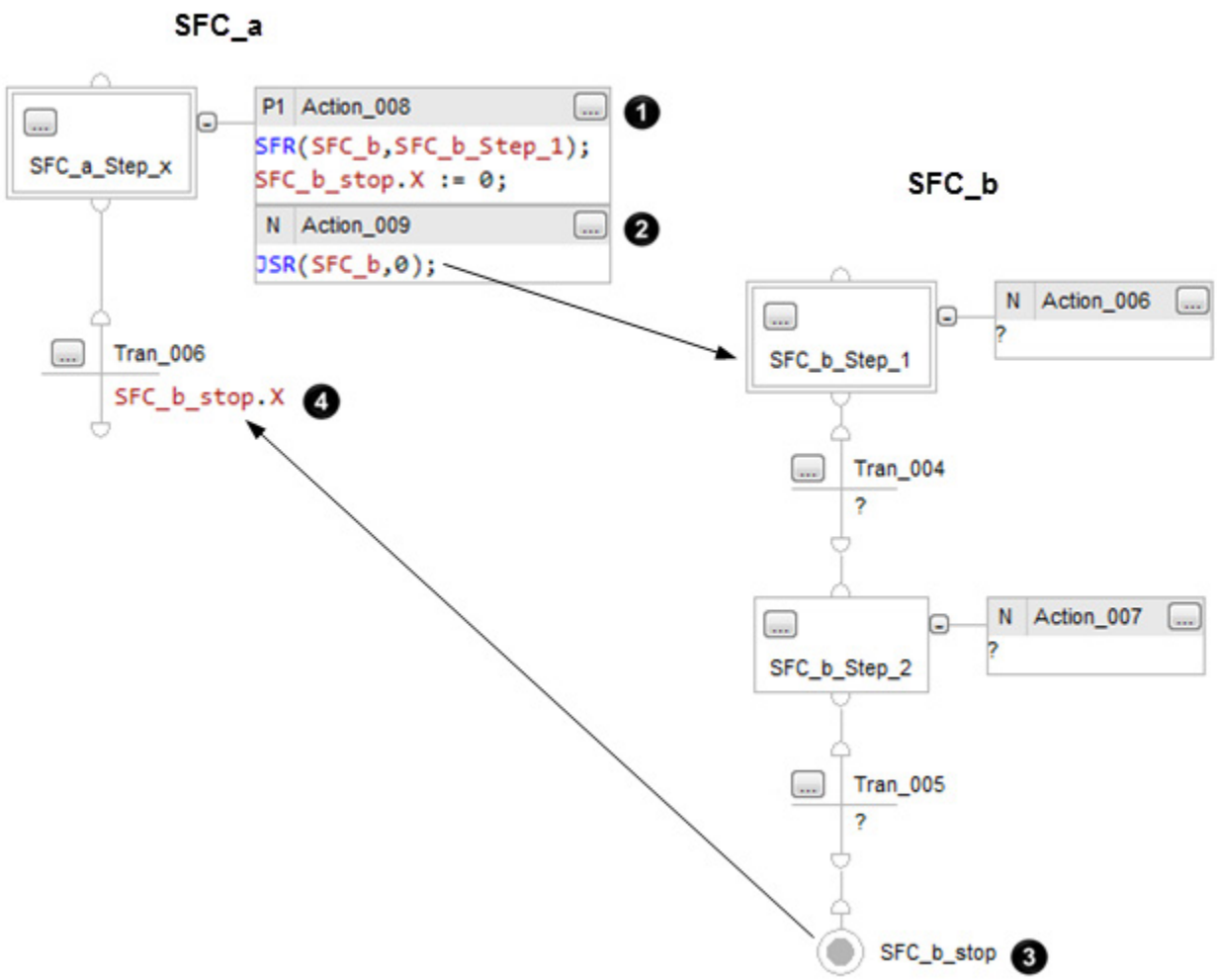
Each stop uses a tag to provide information about the stop element.

If you want to	Then check or set this member	Data type	Details
Determine when the SFC is at the stop	X	BOOL	• When the SFC reaches the stop, the X bit turns on. • The X bit clears if you configure the SFCs to restart at the initial step and the controller changes from program to run mode. • In a nested SFC, the X bit also clears if you configure the SFCs for automatic reset and the SFC leaves the step that calls the nested SFC.
Determine the target of an SFC Reset (SFR) instruction	Reset	BOOL	An SFC Reset (SFR) instruction resets the SFC to a step or stop that the instruction specifies. • The Reset bit indicates to which step or stop the SFC will go to begin executing again. • Once the SFC executes, the Reset bit clears.
Determine how many times a stop has become active	Count	DINT	This is not a count of scans of the stop. • The count increments each time the stop becomes active. • It increments again only after the stop goes inactive and then active again. • The count resets only if you configure the SFC to restart at the initial step. With that configuration, it resets when the controller changes from program mode to run mode.
Use one tag for the various status bits of this stop	Status	DINT	For this member: Use this bit Reset: 22 X: 31

Nest an SFC

One method for organizing your project is to create one SFC that provides a high-level view of your process. Each step of that SFC calls another SFC that performs the detailed procedures of the step (nested SFC).

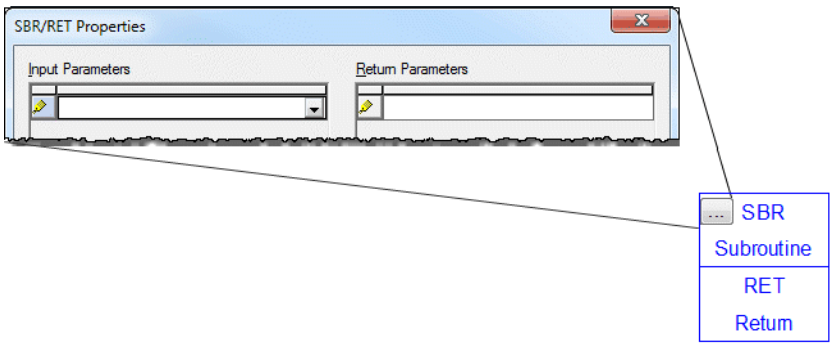
This figure shows one way to nest an SFC. In this method, the last scan option of the SFC is configured for either Programmatic reset or Don't Scan. If you configure the SFC for Automatic reset, then step 1 in is unnecessary.

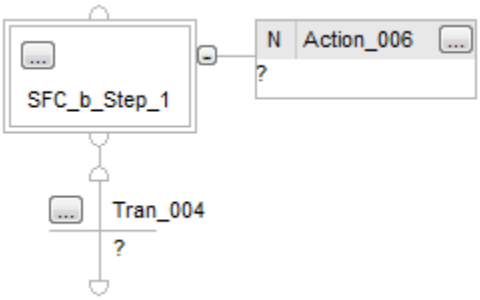


1	Reset the nested SFC. • The SFR instruction restarts SFC_b at SFC_b_Step_1. Each time SFC_a leaves this step and then returns, you have to reset SFC_b. • The action also clears the X bit of the stop element.
2	Call SFC_b.
3	Stop SFC_b. This sets the X bit of the stop element.
4	Use the X bit of the stop element to signal that SFC_b is done and it is time to go to the next step.

Pass parameters

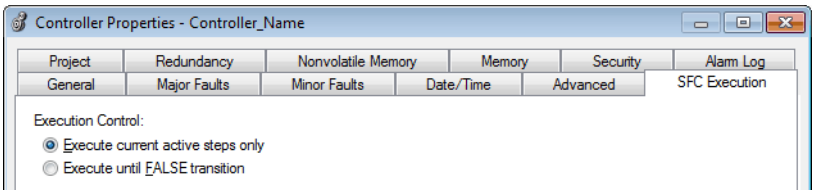
To pass parameters to or from an SFC, place a Subroutine/Return element in the SFC.





Configure when to return to the OS/JSR

By default, an SFC executes a step or group of simultaneous steps and then returns to the operating system (OS) or the calling routine (JSR).



You have the option of letting the SFC execute until it reaches a false transition. If several transitions are TRUE at the same time, this option reduces the time to get to the correct step.

Select the **Execute until FALSE** transition option only when either of these are true:

- You do not have to update JSR parameters before each step. Parameters update only when the SFC returns to the JSR.
- A FALSE transition occurs within the watchdog timer for the task. If the time that it takes to return to a JSR and complete the rest of the task is greater than the watchdog timer, a major fault occurs.

Pause or reset an SFC

Two optional instructions are available that give you further control over the execution of your SFC.

If you want to	Then use this instruction
Pause an SFC	Pause SFC (SFP)
Reset an SFC to a specific step or stop	Reset SFC (SFR)

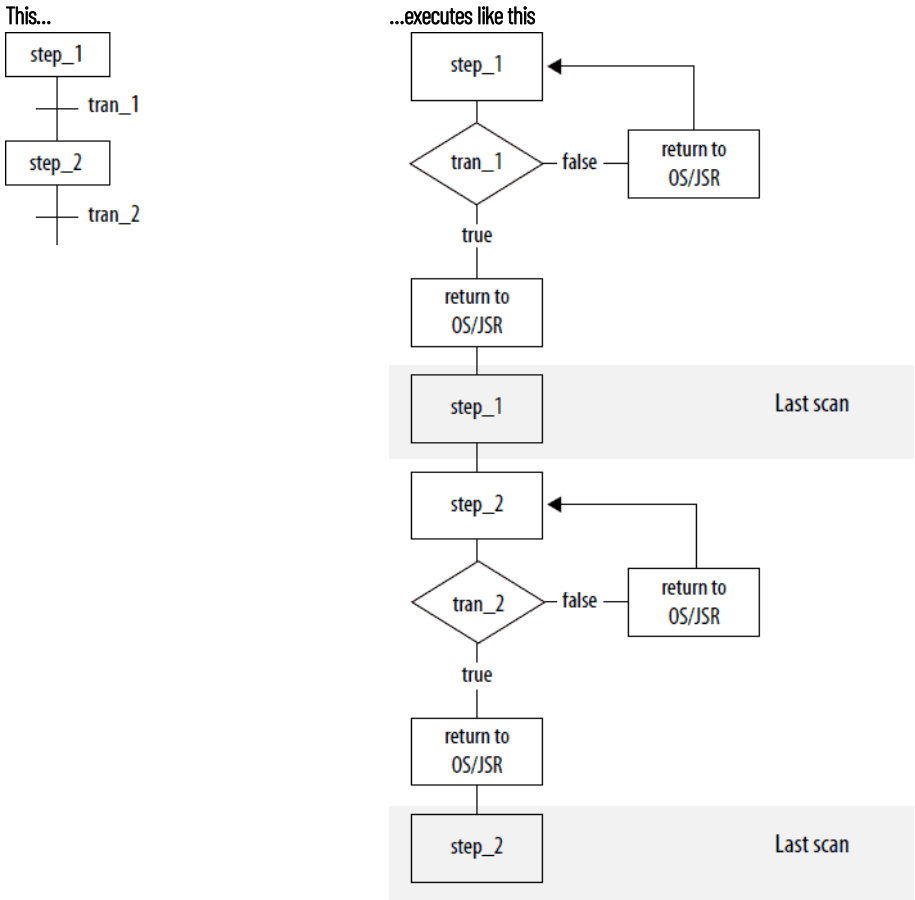
Both instructions are available in the ladder logic and structured text programming languages.

Execution diagrams

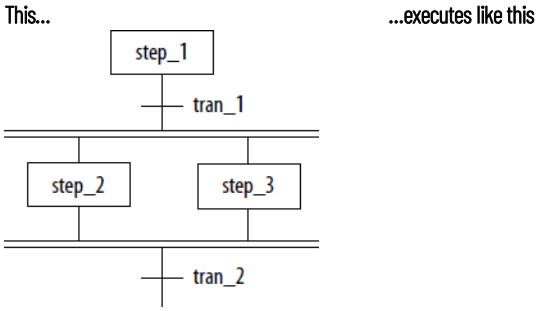
The following diagrams show the execution of an SFC with different organizations of steps or different selections of execution options.

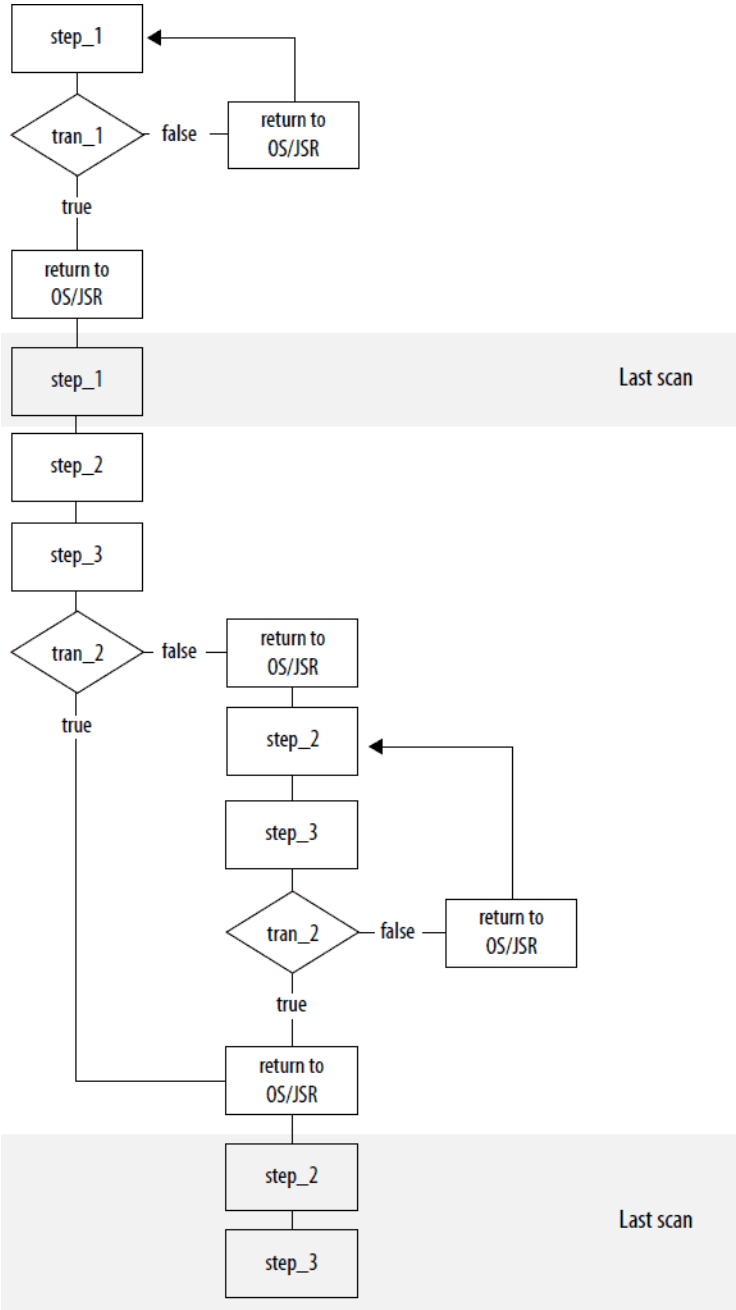
- Execution of a sequence
- Execution of a simultaneous branch
- Execution of a selection branch
- When parameters enter and exit an SFC
- Options for execution control

The following diagram shows the execution of a sequence.

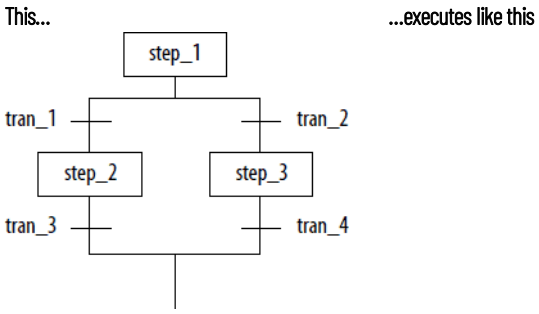


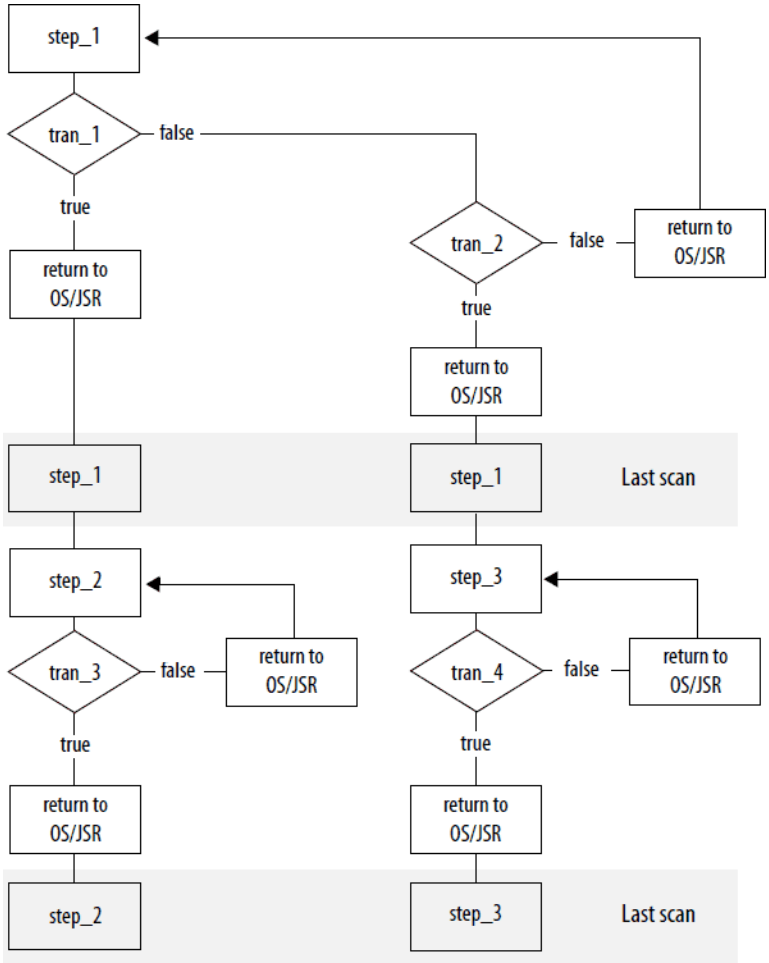
The following diagram shows the execution of a simultaneous branch.



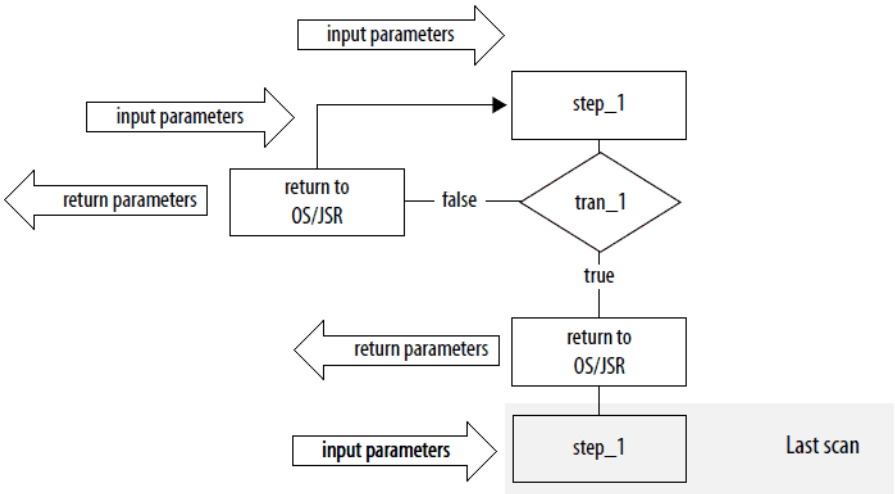


The following diagram shows the execution of a selection branch.

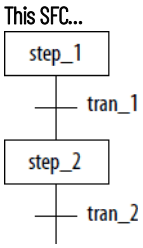


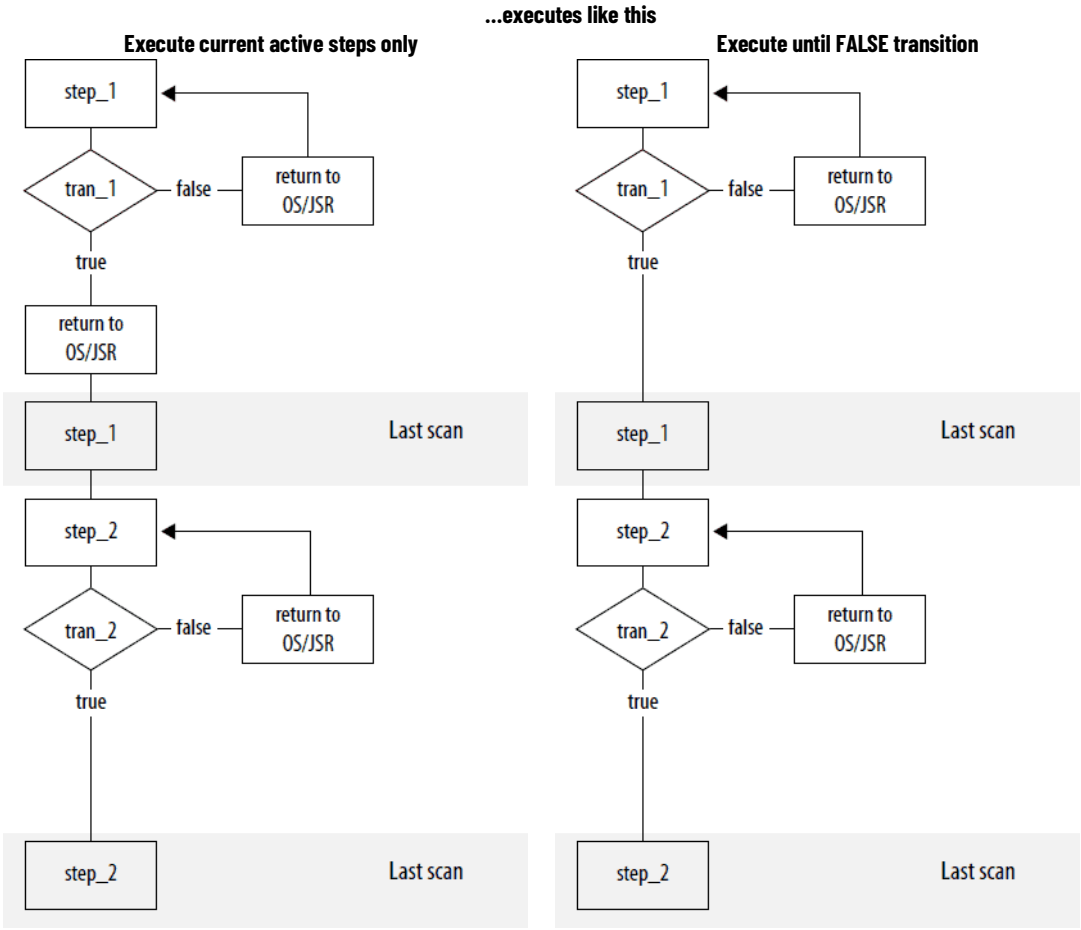


The following diagram shows when parameters enter and exit an SFC



The following diagram shows options for execution control.

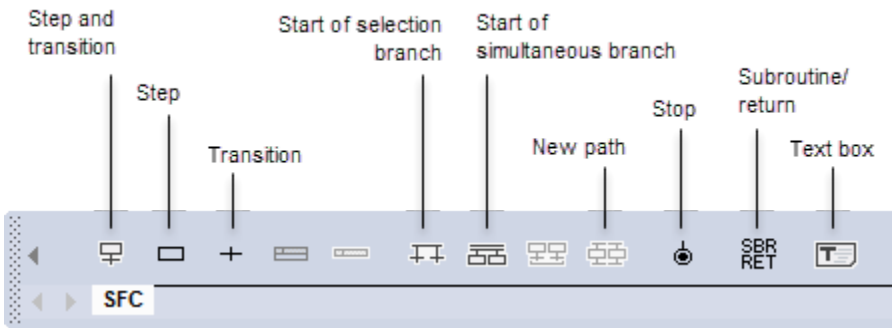



...executes like this
Execute current active steps only
step_1
tran_1
false
return to OS/JSR
true
return to OS/JSR
step_1
Last scan
step_2
tran_2
false
return to OS/JSR
true
step_2
Last scan
Execute until FALSE transition
step_1
tran_1
false
return to OS/JSR
true
step_1
Last scan
step_2
tran_2
false
return to OS/JSR
true
step_2
Last scan

Program a sequential function chart

Introduction

To add SFC elements, use the SFC toolbar.

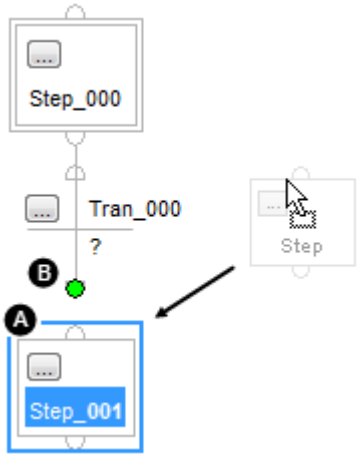


Add and manually connect elements

Use these steps to add and manually connect element.

To add and manually connect elements

1. On the SFC toolbar, click the button for the item that you want to add.
2. Drag the element to the required location on the SFC.



3. To wire (connect) two elements together, click a pin on one of the elements A and then click the pin on the other element B. A green dot shows a valid connection point.

IMPORTANT Use caution when copying and pasting components between different versions of the Logix Designer application. The application only supports pasting to the same version or newer. Pasting to a prior version of the application is not supported. When pasting to a prior version, the paste action may succeed but the results may not be as intended.

Add and automatically connect elements

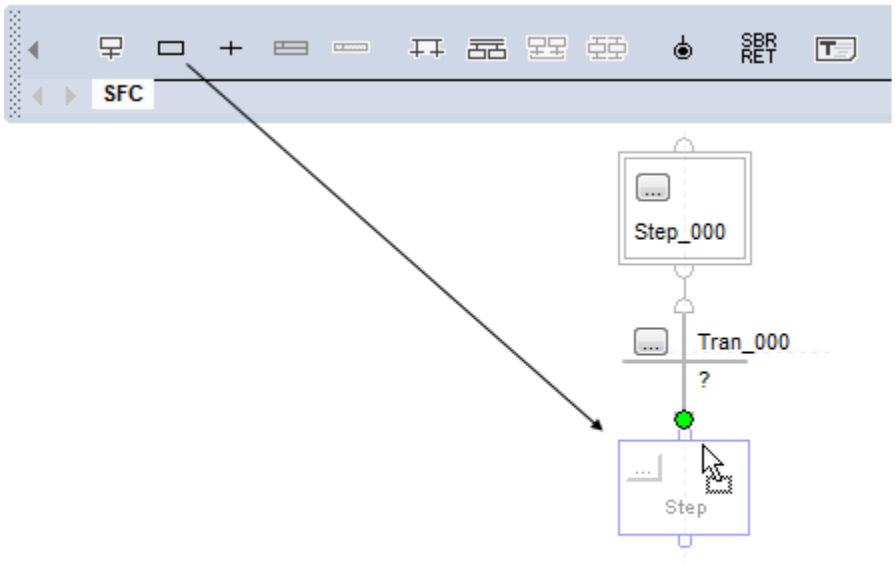
Select an existing element and add a new element to easily join those elements.

To add and automatically connect elements

1. Select the element to which you want to connect a new element.
2. With the element still selected, select the toolbar button for the next, new element.

Drag elements

From the SFC toolbar, drag the button for the required element to the correct connection point on the SFC. A green dot shows a valid connection point.



Create a simultaneous branch

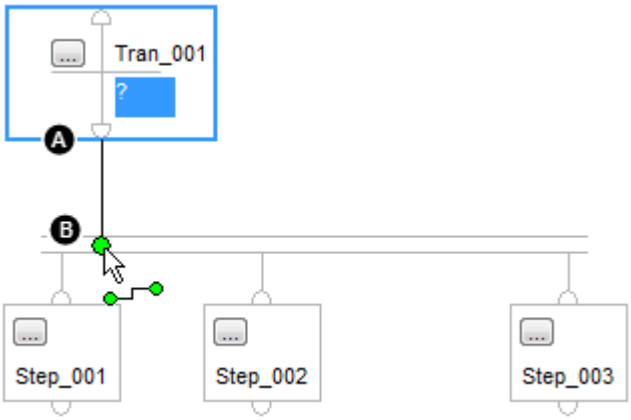
Use the instructions in this section to create a simultaneous branch.

Start a simultaneous branch

Follow these instructions to start a simultaneous branch.

1. On the SFC toolbar, click the button. Drag the new branch to the correct location.

2. To add a path to the branch, click the first step of the path that is to the left of where you want to add the new path. Click .

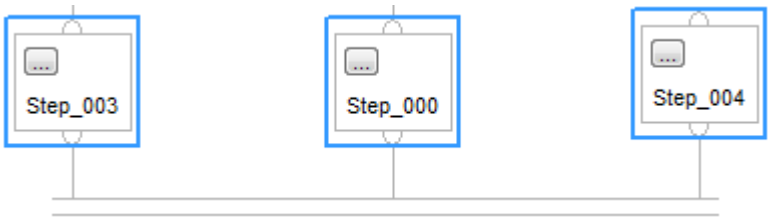


3. To wire the simultaneous branch to the preceding transition, click the bottom pin of the transition A and then click the horizontal line of the branch B. A green dot shows a valid connection point.

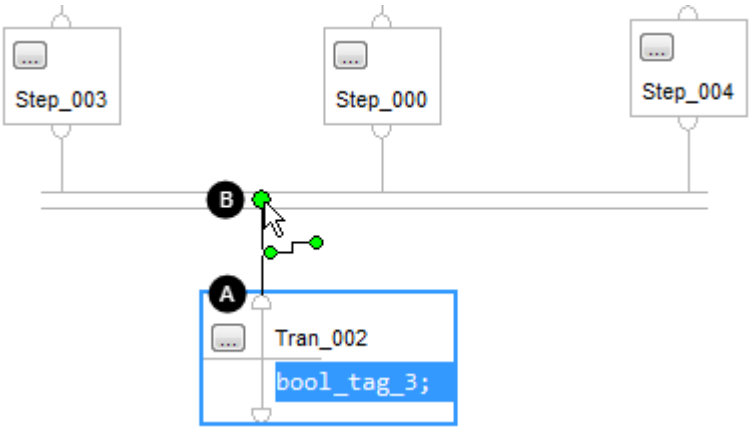
End a simultaneous branch

To end a simultaneous branch

1. Select the last step of each path in the branch. To select the steps, do either of these actions.
   - Drag the pointer around the steps that you want to select.
   - Click the first step. Hold down **Shift** while clicking the rest of the steps that you want to select.



2. On the SFC toolbar, click .
3. Add the transition that follows the simultaneous branch.



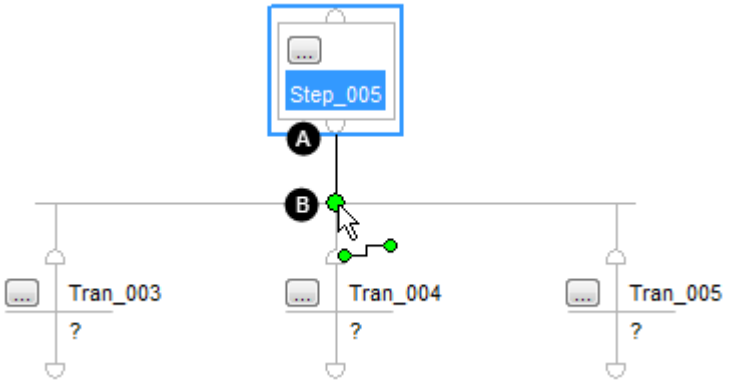
4. To wire the simultaneous branch to the transition, click the top pin of the transition Ⓐ and then click the horizontal line of the branch Ⓑ. A green dot shows a valid connection point.

Create a selection branch

Follow the instructions in this section to create a selection branch.

Start a selection branch

1. On the SFC toolbar, click the button. Then drag the new branch to the correct location.
2. To add a path to the branch, click the first transition of the path that is to the left of where you want to add the new path. Click .

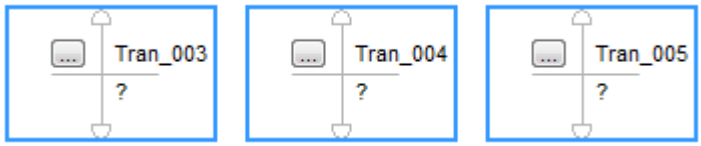


3. To wire the selection branch to the preceding step, click the bottom pin of the step Ⓐ and then click the horizontal line of the branch Ⓑ. A green dot shows a valid connection point.

End a selection branch

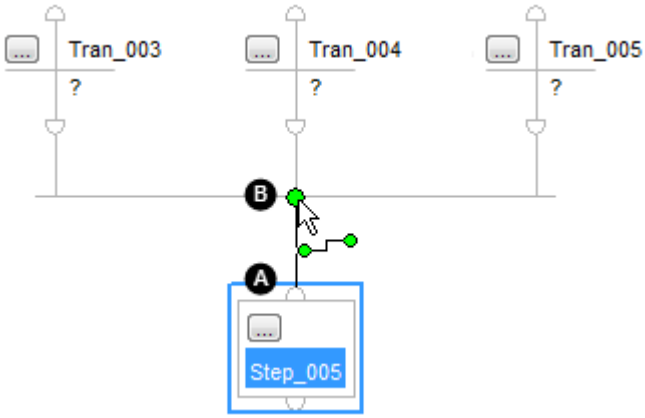
To end a selection branch

1. Select the last transition of each path in the branch. To select the transitions, do either of these actions.
   - Drag the pointer around the transitions that you want to select.
   - Click the first transition. Hold down **Shift** while clicking the rest of the transitions that you want to select.



2. On the SFC toolbar, click .

3. Add the step that follows the selection branch.



4. To wire the selection branch to the step:
   a. Click the top pin of the step A.
   b. Click the horizontal line of the branch B.

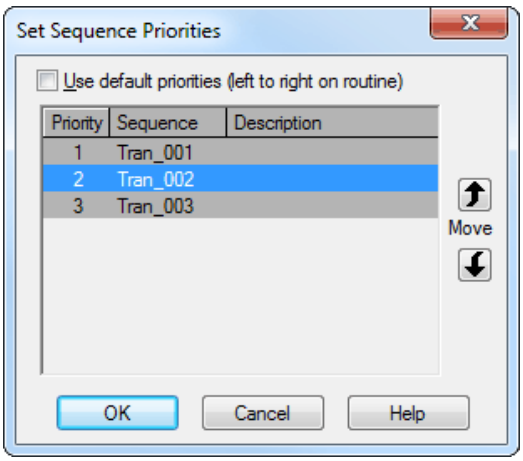
   A green dot shows a valid connection point.

Set the priorities of a selection branch

By default, the SFC checks the transitions that start a selection branch from left to right. If you want to check a different transition first, assign a priority to each path of the selection branch. For example, it is a good practice to check for error conditions first. Then check for normal conditions.

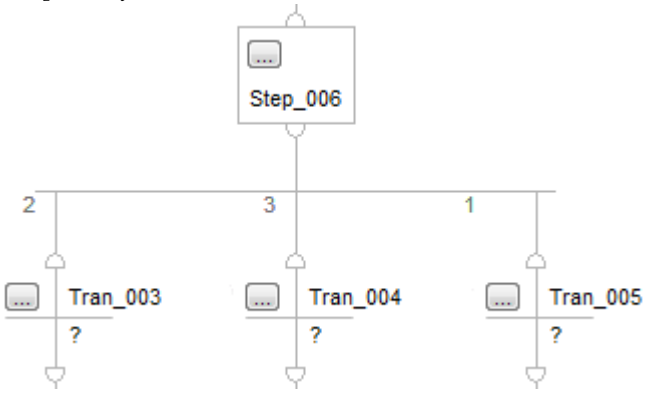
To set the priorities of a selection branch

1. Right-click the horizontal line that starts the branch and then click **Set Sequence Priorities**.
2. Clear the **Use default priorities** check box and select a transition.



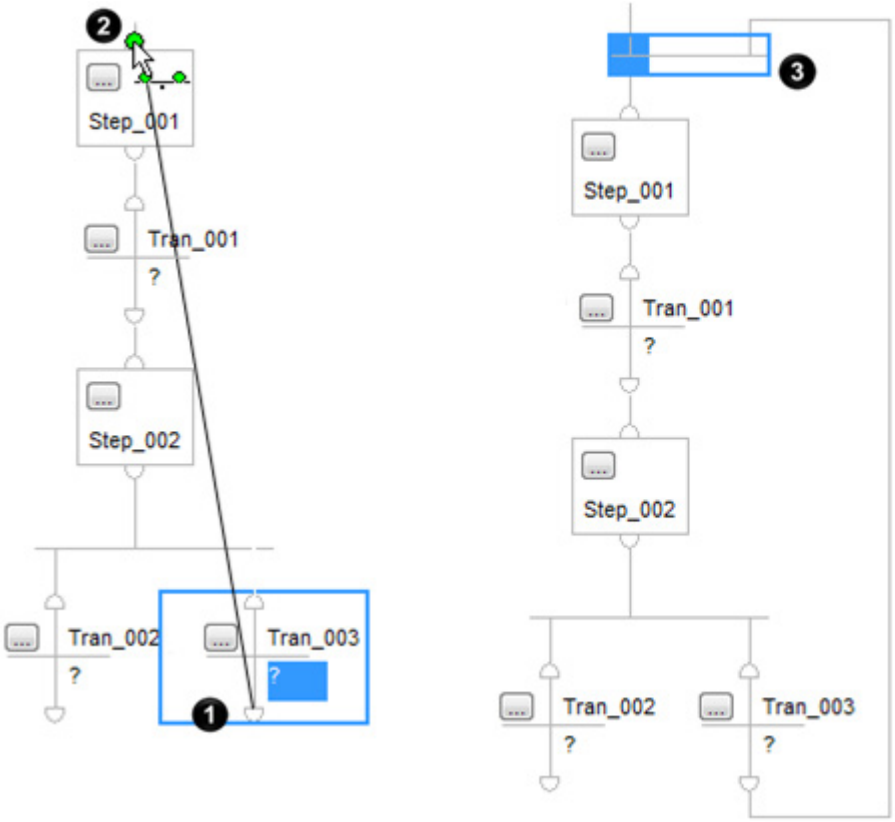
   Use the **Move** buttons to raise or lower the priority of the transition.
3. When all the transitions have the correct priority, click **OK**.

When you clear the **Use default priorities** check box, numbers show the priority of each transition.



Connect a wire to the step

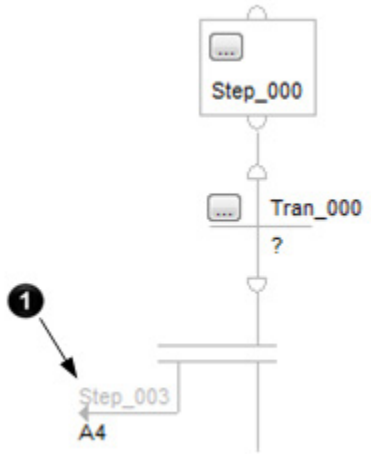
You may have to reposition a wire after you connect it to a step. This example shows how to go to Step_001 from Tran_003.



1. Click the lower pin of the transition that signals the jump.
2. Then click the top pin of the step to which you want to go. A green dot shows a valid connection point.
3. To make the jump easier to read, drag its horizontal bar above the step to which the jump goes. You may also have to reposition some of the SFC elements.

Hide a wire

If a wire gets in the way of other parts of your SFC, hide the wire to make the SFC easier to read. To hide a wire, right-click the wire and select **Hide Wire**.



To see the SFC element to which the wire goes, click the grid location on the wire.

1 Location to which the wire goes

Configure a step

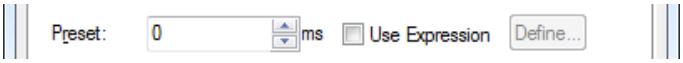
Follow the instructions in this section to configure a step.

Assign the preset time for a step

Use these steps to assign the present time for a step.

To assign the present time for a step

1. Click the button of the step.



2. In the **Step Properties** dialog box, on the **General** tab, in the **Preset** box, enter the time for the step, in milliseconds.
3. Click **OK**.

When the step is active for the preset time (Timer = Preset), the DN bit of the step turns on.

To calculate the preset time for a step at runtime, see *Use an expression to calculate a time.*

See also

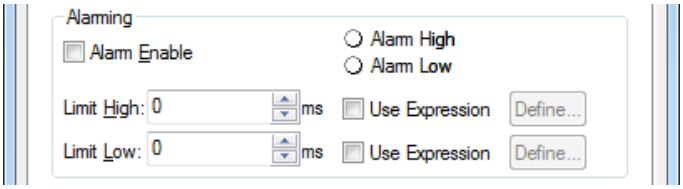
Use an expression to calculate a time on page 60

Configure alarms for a step

Follow these steps to turn on an alarm if a step executes too long or not long enough.

To configure alarms for a step

1. Click the button of the step.
2. In the **Step Properties** dialog box, on the **General** tab, select the **Alarm Enable** check box.



3. Enter the time for the high alarm (**Limit High**) and low alarm (**Limit Low**), in milliseconds.
4. Click **OK**.

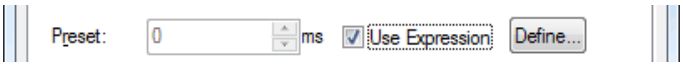
Use an expression to calculate a time

To calculate a time based on tags in your project, enter the time as a numeric expression. You can use an expression to calculate these values.

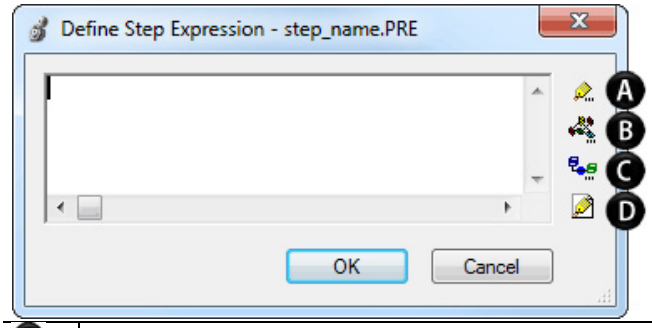
- Preset
- LimitHigh
- LimitLow

To use an expression to calculate a time

1. Click the button of the step.
2. In the **Step Properties** dialog box, on the **General** tab, select the **Use Expression** check box.



3. Click **Define** and enter an expression.



A	Browse for a tag
B	Choose a function
C	Choose an operator
D	Create a tag

4. Enter a numeric expression that defines the time. Use the buttons on the right side of the dialog box to help you complete the expression.
5. Click **OK**.
6. To close the **Step Properties** dialog box, click **OK**.

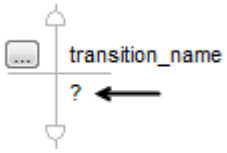
Program a transition

Enter a BOOL expression

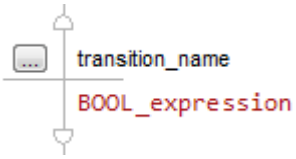
The simplest way to program the transition is to enter the conditions as a BOOL expression in structured text.

To enter a BOOL expression

1. Double-click the text area of the transition.



2. Type the BOOL expression that determines when the transition is TRUE or FALSE.
3. To close the text entry window, press **Ctrl+Enter**.



This example shows three transitions that use a BOOL expression (see *Enter a BOOL expression*).

1	Tag name of the transition.
2	BOOL expression that controls when the transition is TRUE or FALSE

See also

Enter a BOOL expression on page 61

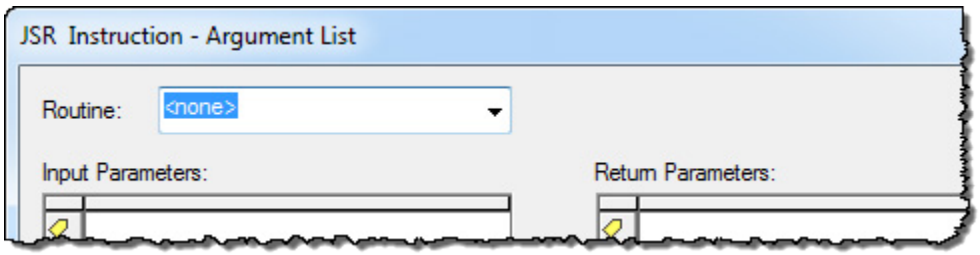
Call a subroutine when programming a transition

Use the **Set JSR** option to call a subroutine when you program a transition.

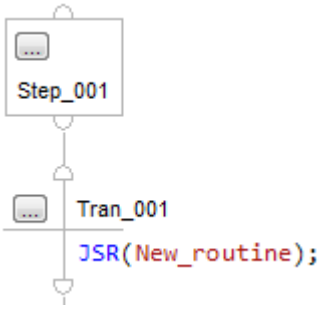
To call a subroutine when programming a transition

1. In the SFC, right-click the transition and then click **Set JSR**.

2. In the **Routine** box, select the routine to call.

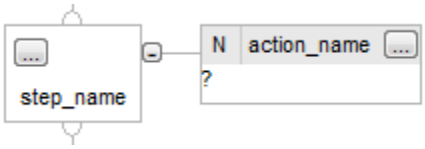


3. Click **OK**.



Add an action

To add an action to a step, right-click the step in which the action executes and then click **Add Action**.



Configure an action

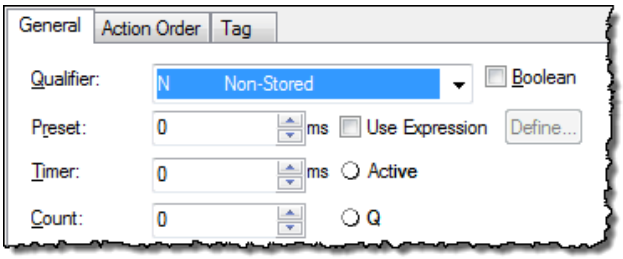
Change the qualifier of an action

Follow the instructions in this section to configure an action.

A qualifier determines when an action starts and stops. The default qualifier is N Non-Stored. The action starts when the step is activated and stops when the step is deactivated.

To change the qualifier of an action

1. Click the button in the action.
2. In the **Action Properties** dialog box, on the **General** tab, select the qualifier for the action.



If you chose a timed qualifier, type the time limit or delay for the action, in milliseconds. These are the timed qualifiers.

- **L Time Limited**

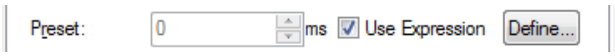
- **SL Stored and Time Limited**
- **D Time Delayed**
- **DS Delayed and Stored**
- **SD Stored and Time Delayed**

3. Click **OK**.

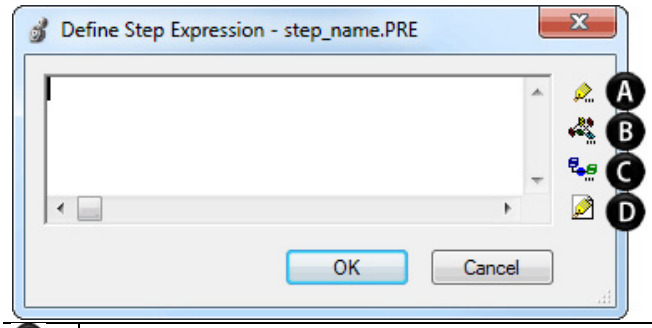
Calculate a preset time at runtime

To calculate a preset value based on tags in your project, enter the value as a numeric expression.

1. Click the button of the action.
2. Check the **Use Expression** check box.



3. Click **Define** and enter an expression.



A	Browse for a tag
B	Choose a function
C	Choose an operator
D	Create a tag

4. Enter a numeric expression that defines the preset time. Use the buttons on the right side of the dialog box to help you complete the expression.
5. Click **OK**.
6. To close the **Action Properties** dialog box, click **OK**.

Mark an action as a BOOLean action

Use a Boolean action to only set a bit when the action executes.

Mark an action as a Boolean action

1. Click the button in the action.
2. In the **Action Properties** dialog box, select the **Boolean** check box.

3. Click **OK**.

Program an action

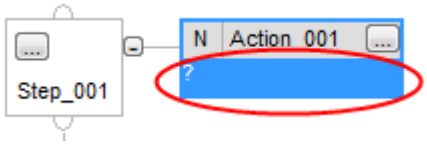
You can use structured text or a subroutine to program an action.

Enter structured text

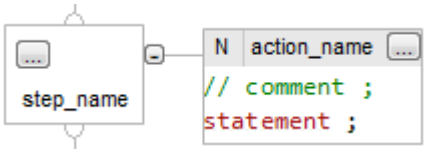
The easiest way to program an action is to write the logic as structured text within the body of the action. When the action turns on, the controller executes the structured text.

To enter structured text

1. Double-click the text area of the action.



2. Type the required structured text.
3. To close the text entry window, press **Ctrl+Enter**.

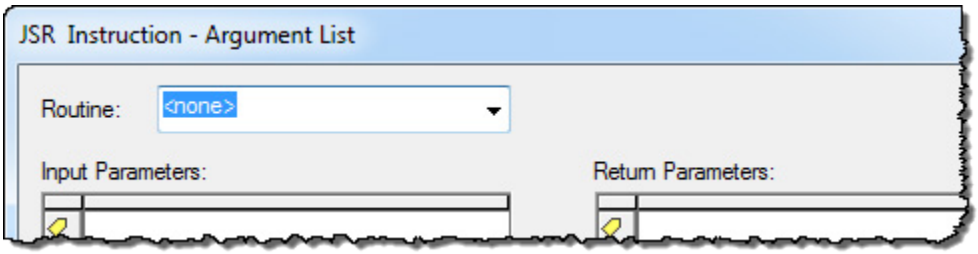


Call a subroutine in an action

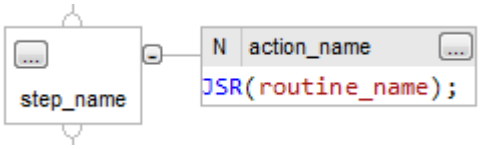
Use a Jump to Subroutine (JSR) instruction to execute a subroutine when the action is active.

To call a subroutine in an action

1. In the SFC, right-click the action and then click **Set JSR**.
2. In the **Routine** box, select the routine to call.

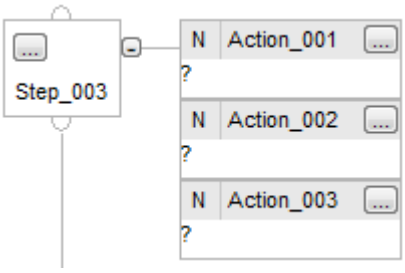


3. To pass a parameter to the routine, click an empty **Input Parameters** box. Click the down arrow and then click the tag that contains the parameter.
4. To receive a parameter from the routine, click an empty **Return Parameters** box. Click the down arrow and then click the tag in which to store the parameter from the routine.
5. Click **OK**.



Assign the execution order of actions

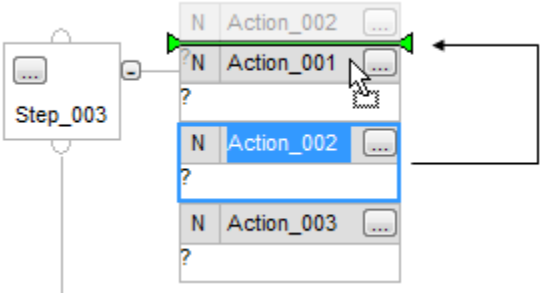
Actions execute in the order in which they appear.



When Step_003 is active, its actions execute in this order.

1. Action_000
2. Action_001
3. Action_002

To change the order in which an action executes, drag the action to the correct location in the sequence. A green bar shows a valid placement location. The following shows dragging Action_002 from after Action_001 to before Action_001.



Document an SFC

You can document these SFC components.

To document this	And you want to	Do this
General information about the SFC	---------->	Add a text box on page 67
Step	---------->	Add a text box on page 67 -or- Add a tag description on page 67
Transition	• Download the documentation to the controller	Add structured text comments on page 66
	• Have the option of showing or hiding the documentation • Position the documentation anywhere in the SFC	Add a text box on page 67 -or- Add a tag description on page 67
Action	Download the documentation to the controller	Add structured text comments on page 66
Stop	---------->	Add a text box on page 67 -or- Add a tag description on page 67
Other element (such as a selection branch)	---------->	

Language switching

With version 17 and later of the application, 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 supports and set the current, default, and optional custom localized language. The default language is used if the current language's content is blank for a particular component of the project. However, you can use a custom language to tailor documentation to a specific type of project file user. Enter the localized descriptions in your project, either when programming in that language or by using the import/export utility to translate the documentation off-line and then import it back into the project. Once you enable language switching, you can dynamically switch between languages as you use the software.

Project documentation that supports multiple translations includes these variables:

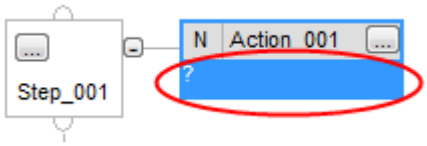
- Component descriptions in tags, routines, programs, Equipment Phases, Equipment Sequences, user-defined data types, and Add-On Instructions
- Engineering units and state identifiers added to tags, user-defined data types, or Add-On Instructions
- Trends
- Controllers
- Alarm Messages (in configuration of ALARM_ANALOG and ALARM_DIGITAL tags)
- Tasks
- Property descriptions for module in the Controller Organizer
- Rung comments, SFC text boxes, and FBD text boxes

For more information on enabling a project to support multiple translations of project documentation, see the online help.

Add structured text comments

Comments embedded in the structured text section of an action are downloaded into controller memory and are available for upload.

1. Double-click the text area of the action.



2. Type the comments.

To add a comment	Use one of these formats
On a single line	*// comment* (** comment**) */* comment* /*
At the end of a line of structured text	

To add a comment	Use one of these formats
Within a line of structured text	(* *comment**) /* *comment** /
That spans more than one line	(* *start of comment . . . end of comment**) /* *start of comment . . . end of comment** /

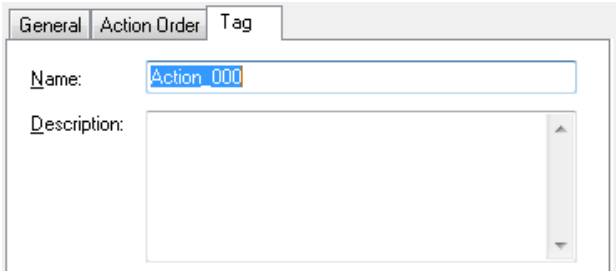
3. To close the text entry window, press **Ctrl+Enter**.

Add a tag description

Add tag descriptions to help identify each tag's purpose.

To add a tag description

1. Select the button of the element.
2. In the element **Properties** dialog box, Select the **Tag** tab and type the description for the element.

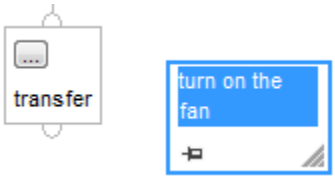


3. Select **OK**.
4. Drag the description box to the correct location on the SFC.

Add a text box

A text box lets you add notes that clarify the function of an SFC element (step, transition, or stop). Text boxes are only stored in the offline, ACD project file. Text boxes are not downloaded into controller memory.

You can also use a text box to capture information that you can refer to later.



To add a text box

1. Click the Text Box icon .

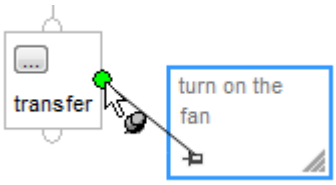
   A text box appears.

2. Drag the text box to a location near the element to which it applies.
3. Double-click the text box and type the note. Then press **Ctrl+Enter**.

4. As you move the element on the SFC, what do you want the text box to do?

If you the text box to	Then
Stay in the same spot	Stop. You are done.
Move with the element to which it applies	Go to step 5.

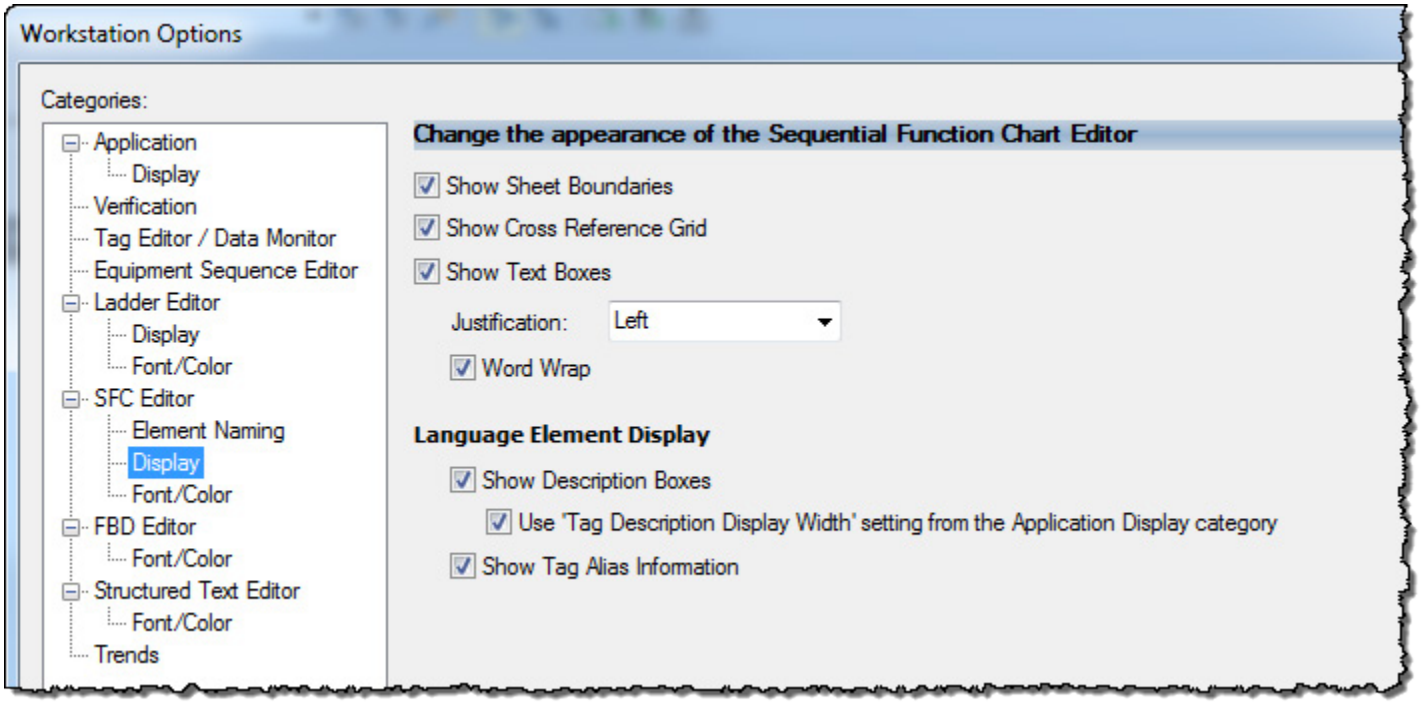
5. Click the pin symbol in the text box and then click the SFC element to which you want to attach the text box. A green dot shows a valid connection point.



Show or hide text boxes or tag descriptions

You have the option of showing or hiding both text boxes and tag descriptions. If you select to show descriptions, the SFC window only shows the descriptions for steps, transitions, and stops (not actions).

1. From the **Tools** menu, select **Options**.
2. Under **SFC Editor**, select the **Display** category.



3. Select the check boxes for the features you want to appear on SFC windows.

Hide an individual tag description

Follow these steps to hide the description of a specific element while showing other descriptions.

To hide an individual tag description

1. Click the button of the element whose description you want to hide.
2. Select the **Never display description in routine** check box.

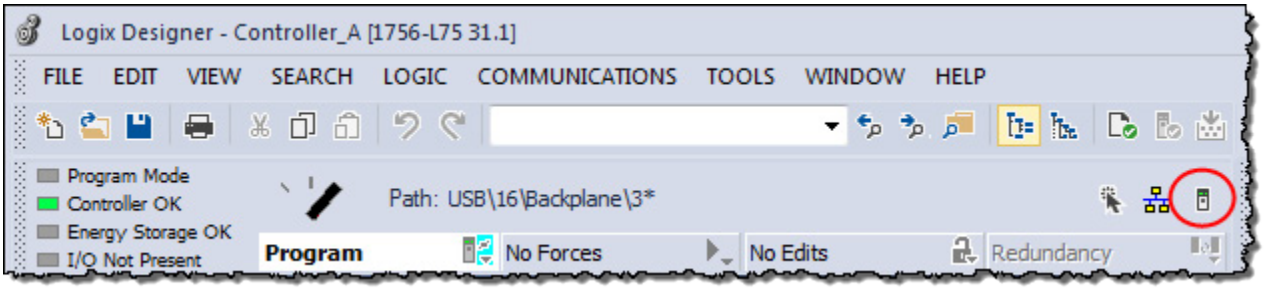
3. Click **OK**.

Configure the execution of the SFC

The **SFC Execution** tab of the controller properties lets you configure these decisions.

- What to do when a transition is TRUE.
- Where to start after a transition to the Run mode or recovery from a power loss.
- What to do on the last scan of a step.

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



2. Select the **SFC Execution** tab.
3. Choose one of the following.
   - Whether or not to return to the OS/JSR if a transition is TRUE (**Execution Control**).
   - Where to restart the SFC (**Restart Position**).

     The restart position applies when the controller loses power or leaves the Run or Remote Run mode.

If you want to restart at the	Select
Last step that was running	Restart at most recently executed step
Initial step	Restart at initial step

   The restart position does not apply for major faults. After you clear a major fault, the SFC **always** restarts at the initial step.

   - What to do on the last scan of a step (**Last Scan of Active Steps**).
4. Click **OK**.

Verify the routine

As you program your routine, periodically verify your work.

To verify the routine

1. In the main toolbar of the application window, click .

2. Follow these steps if any errors are listed in the **Output** window.
   c. To go to the first error or warning, press **F4**.
   d. Correct the error according to the description in the in the **Output** window on the **Search Results** tab.
   e. Repeat step 1.
3. To close the **Results** window, press **Alt+1**.

To check your SFC, you can use either of these options.

- Force transitions
- Step through the SFC

See the chapter on *Force steps*.

See also

Force steps on page 71

Edit an SFC online

Firmware revision 13 added support for editing SFCs online. When you transition the controller to test or un-test edits, the controller resets the SFC and starts execution at the initial step. Keep these guidelines in mind if you edit an SFC online.

- Time when you test or un-test edits to coincide with the SFC execution of the initial step.
- Place structured text logic in subroutines to minimize the impact of online edits.
- Use an SFR instruction to programmatically shift SFC execution to the correct step.

Maintain active SFC step

As of firmware revision 18, the following online edits to an SFC no longer reset the SFC to the initial step.

- Modify structured text in actions and transitions
- Physically move steps, actions, and transitions on SFC sheets without changing the wiring
- Add, delete, or modify text and description boxes
- Modify indicator tags
- Add, delete or modify an SBR/RET
- Add, delete or modify any step or action expression

Force steps

Introduction

Use a force to override data that your logic either uses or produces.

- Test and debug your logic.
- Temporarily keep your process functioning when an input device has failed.

Use forces only as a temporary measure. They are not intended to be a permanent part of your application.

Precautions

Make sure you understand the following before using forces.

ATTENTION: Forcing can cause unexpected machine motion that could injure personnel. Before you use a force, determine how the force will affect your machine or process and keep personnel away from the machine area.

- Enabling SFC forces causes your machine or process to go to a different state or phase.
- Removing forces may still leave forces in the enabled state.
- If forces are enabled and you install a force, the new force immediately takes effect.

Enable forces

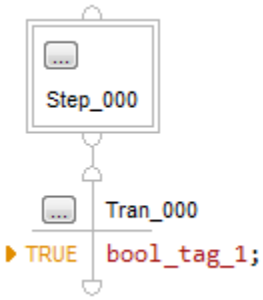
For a force to take effect, you enable forces. You can only enable and disable forces at the controller level.

To enable forces

- You can enable I/O forces and SFC forces separately or at the same time.
- You cannot enable or disable forces for a specific module, tag collection, or tag element.

IMPORTANT If you download a project that has forces enabled, the programming software prompts you to enable or disable forces after the download completes.

When forces are in effect (enabled), a ▶ and TRUE or FALSE appears next to the forced element.



Disable or remove a force

To stop the effect of a force and let your project execute as programmed, disable or remove the force.

- You can disable or remove I/O and SFC forces at the same time or separately.
- Removing a force on an alias tag also removes the force on the base tag.

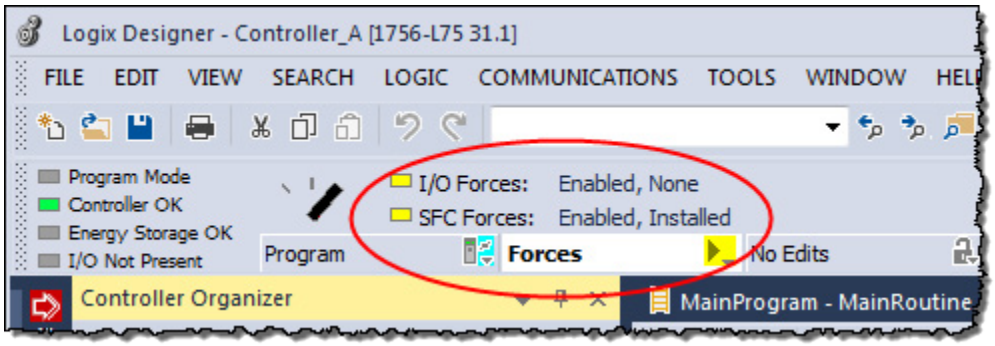
ATTENTION: Changes to forces can cause unexpected machine motion that could injure personnel. Before you disable or remove forces, determine how the change will affect your machine or process and keep personnel away from the machine area.

Check force status

Before you use a force, determine the status of forces for the controller.

To determine the status of	Use any of the following
I/O forces	• Online toolbar • FORCE LED • GSV instruction
SFC forces	Online Toolbar

The Online toolbar shows the status of forces. It shows the status of I/O forces and SFC forces separately.



Forces tab status	Means
Enabled	• If the project contains any forces of this type, they are overriding your logic. • If you add a force of this type, the new force immediately takes effect
Disabled	Forces of this type are inactive. If the project contains any forces of this type, they are not overriding your logic.
Installed	At least one force of this type exists in the project.
None Installed	No forces of this type exist in the project.

Force LED

If your controller has a FORCE LED, use the LED to determine the status of any I/O forces.

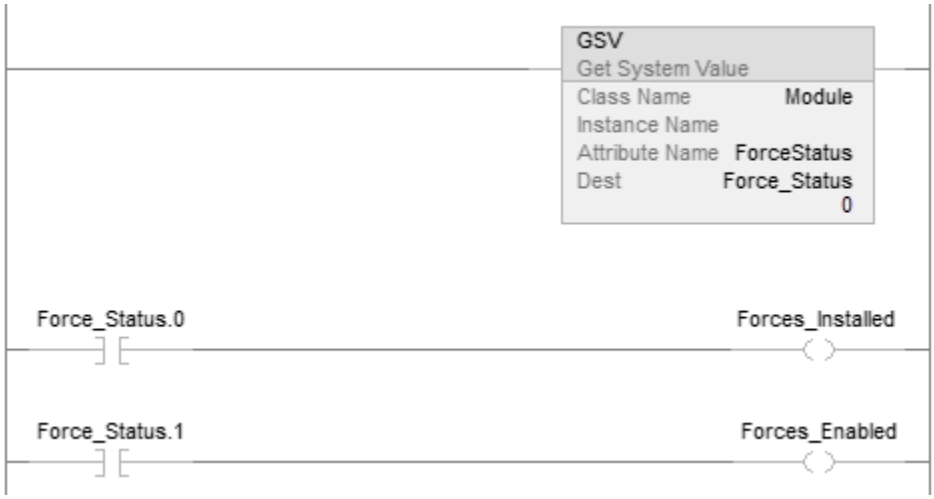
IMPORTANT The FORCE LED shows only the status of I/O forces. It does not show the status of SFC forces.

If the FORCE LED is:	Then:
Off	• No tags contain force values. • I/O forces are inactive (disabled).
Flashing	• At least one tag contains a force value. • I/O forces are inactive (disabled).
Solid	• I/O forces are active (enabled). • Force values may or may not exist.

GSV instruction

This example shows how to use a GSV instruction to get the status of forces.

IMPORTANT The ForceStatus attribute shows only the status of I/O forces. It does not show the status of SFC forces.



where:
Force_Status is a DINT tag.

To determine if	Examine this bit	For this value
forces are installed	0	1
no forces are installed	0	0
forces are enabled	1	1
forces are disabled	1	0

Step through a transition or a force of a path

To override a false transition one time and go from an active step to the next step, use the **Step Through** option.

- You do not have to add, enable, disable, or remove forces.

- The next time the SFC reaches the transition, it executes according to the conditions of the transition.

This option also lets you override one time the false force of a simultaneous path. When you step through the force, the SFC executes the steps of the path.

To step through the transition of an active step or a force of a simultaneous path

1. Open the SFC routine.
2. Right-click the transition or the path that is forced and then select **Step Through**.

When to use an SFC force

To override the logic of an SFC, you have these options.

If you want to	Then
Override the conditions of a transition each time the SFC reaches the transition	Force a transition.
Prevent the execution of one or more paths of a simultaneous branch	Force a simultaneous path.

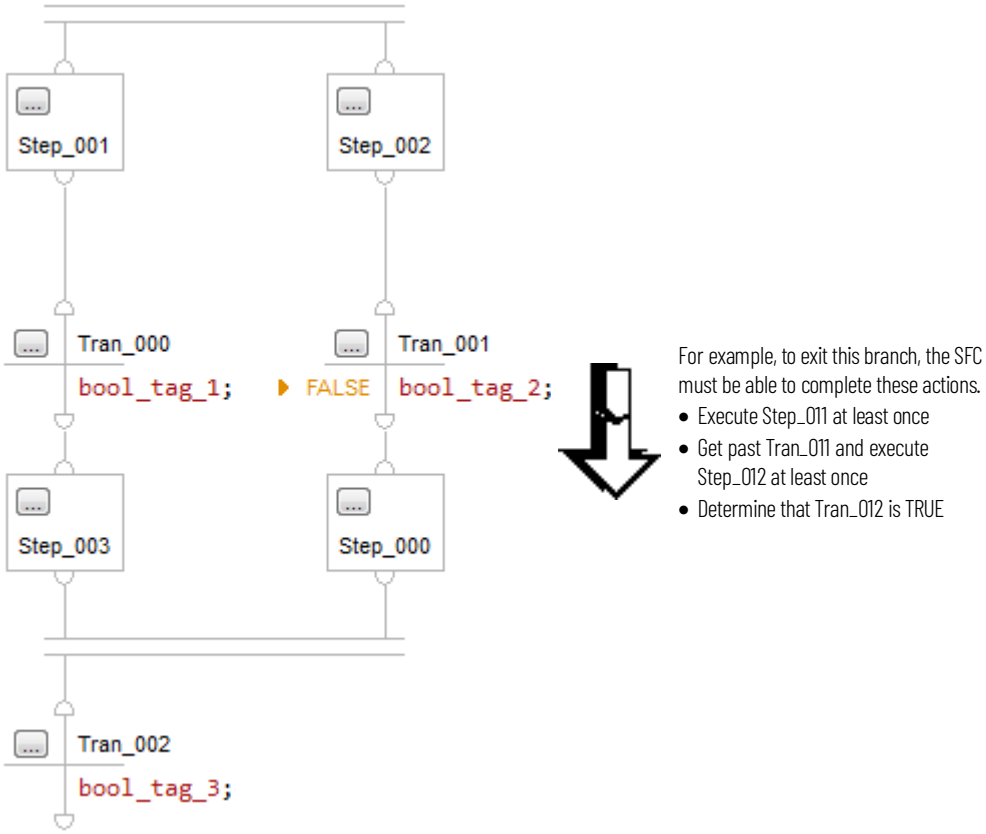
Force a transition

To override the conditions of a transition through repeated executions of an SFC, force the transition. The force remains until you remove it or disable forces.

If you want to	Then
Prevent the SFC from going to the next step	Force the transition FALSE.
Cause the SFC go to the next step regardless of transition conditions	Force the transition FALSE.

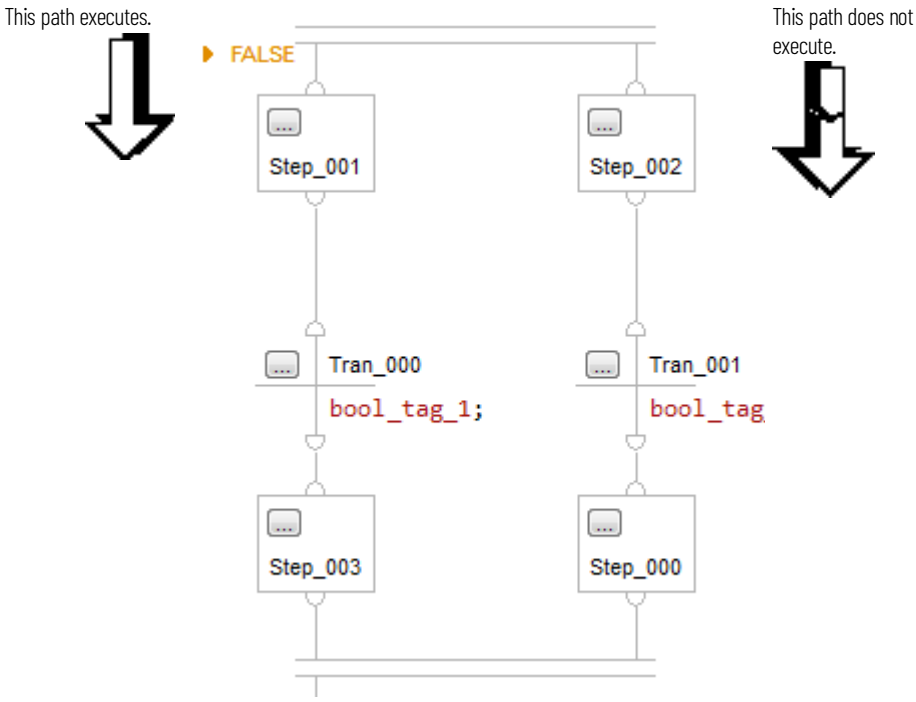
If you force a transition within a simultaneous branch to be FALSE, the SFC stays in the simultaneous branch as long as the force is active (installed and enabled).

- To leave a simultaneous branch, the last step of each path must execute at least one time and the transition below the branch must be TRUE.
- Forcing a transition FALSE prevents the SFC from reaching the last step of a path.
- When you remove or disable the force, the SFC can execute the rest of the steps in the path.



Force a simultaneous path

To prevent the execution of a path of a simultaneous branch, force the path FALSE. When the SFC reaches the branch, it executes only the un-forced paths.



If you force a path of a simultaneous branch to be FALSE, the SFC stays in the simultaneous branch as long as the force is active (installed and enabled).

- To leave a simultaneous branch, the last step of each path must execute at least one time and the transition below the branch must be TRUE.
- Forcing a path FALSE prevents the SFC from entering a path and executing its steps.
- When you remove or disable the force, the SFC can execute the steps in the path.

Add an SFC force

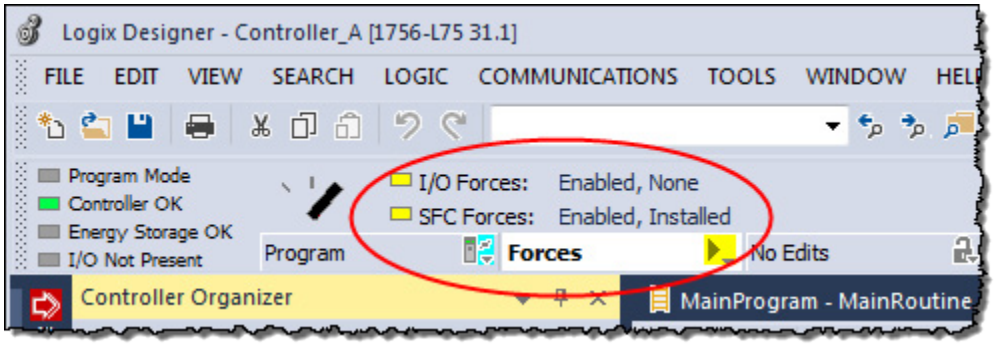
To override the logic of an SFC, use an SFC force.

ATTENTION: Forcing can cause unexpected machine motion that could injure personnel. Before you use a force, determine how the force will affect your machine or process and keep personnel away from the machine area.

- Enabling SFC forces causes your machine or process to go to a different state or phase.
- If forces are enabled and you install a force, the new force immediately takes effect.

To add an SFC force

1. What is the state of the SFC Forces indicator?



If	Then
Off	No SFC forces currently exist.
Flashing	No SFC forces are active. But at least one force already exists in your project. When you enable SFC forces, all existing SFC forces will also take effect.
Solid	SFC forces are enabled (active). When you install (add) a force, it immediately takes effect.

2. Open the SFC routine.
3. Right-click the transition or start of a simultaneous path that you want to force, and then click either **Force TRUE** (only for a transition) or **Force FALSE**.
4. Are SFC forces enabled (See step 1)?

If	Then
No	From the Logic menu, choose SFC Forcing > Enable All SFC Forces. Then choose Yes to confirm.
Yes	Stop.

Remove or disable forces

Make sure you understand the following before using forces.

ATTENTION: Changes to forces can cause unexpected machine motion that could injure personnel. Before you disable or remove forces, determine how the change will affect your machine or process and keep personnel away from the machine area.

Disable all SFC forces

From the **Logic** menu, click **SFC Forcing** and then click **Disable All SFC Forces**. Then click **Yes** to confirm.

Remove all SFC forces

From the **Logic** menu, click **SFC Forcing** and then click **Remove All SFC Forces**. Then click **Yes** to confirm.

Index

R

S

T

V

W

Rockwell Automation support

Use these resources to access support information.

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

Documentation feedback

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

Waste Electrical and Electronic Equipment (WEEE)

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

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

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

Connect with us.

rockwellautomation.com — expanding **human possibility**™

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

Rockwell Automation Publication 1756-PM006K-EN-P - March 2022
Supersedes Publication 1756-PM006J-EN-P - September 2020