Logix5000 Controllers Sequential Function Charts

Allen-Bradley

Catalog Numbers 1756 ControlLogix, 1769 CompactLogix, 1789 SoftLogix, 1794 FlexLogix, PowerFlex 700S with DriveLogix

Programming Manual

Allen-Bradley • Rockwell Software

Rockwell Automation

Important User Information

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

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

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

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

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

WARNING	Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss.
IMPORTANT	Identifies information that is critical for successful application and understanding of the product.
ATTENTION	Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you identify a hazard, avoid a hazard, and recognize the consequence
SHOCK HAZARD	Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present.
BURN HAZARD	Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.

Allen-Bradley, Rockwell Automation, CompactLogix, ControlLogix, DriveLogix, FlexLogix, Logix5000, PowerFlex 700S, Rockwell Software, RSLogix 5000, SoftLogix, TechConnect are trademarks of Rockwell Automation, Inc.

Trademarks not belonging to Rockwell Automation are property of their respective companies.

Summary of Changes

Summary of Changes

This manual contains new and updated information. Changes throughout this revision are marked by change bars, as shown to the right of this paragraph.

Change	Page
Description of postscan of SFC logic	42

Notes:

Preface

Purpose of This Manual. 9
How to Use This Manual. 9

Chapter 1

Design a Sequential Function Chart

Introduction 11
What is a Sequential Function Chart? 12
Define the Tasks. 15
Choose How to Execute the SFC 16
Define the Steps of the Process. 16
  Step Guidelines 17
  SFC_STEP Structure. 18
Organize the Steps 21
  Sequence. 23
  Selection Branch 23
  Simultaneous Branch 24
  Wire to a Previous Step 25
Add Actions for Each Step 25
  How Do You Want to Use the Action? 26
  Use a Non-Boolean Action 26
  Use a Boolean Action 28
  SFC_ACTION Structure 29
Describe Each Action in Pseudocode. 30
Choose a Qualifier for an Action 30
Define the Transition Conditions 31
  Transition Tag 33
  How Do You Want to Program the Transition? 33
  Use a BOOL Expression. 34
  Call a Subroutine 34
Transition After a Specified Time 35
Turn Off a Device at the End of a Step 38
  Choose a Last Scan Option. 38
  Use the Don't Scan Option. 40
  Use the Programmatic Reset Option 41
  Use the Automatic Reset Option 42
Keep Something On From Step-to-Step 44
  How Do You Want to Control the Device? 44
  Use a Simultaneous Branch. 45
  Store and Reset an Action 45
  Use One Large Step. 47
End the SFC. 48
  Use a Stop Element 48
  Restart (Reset) the SFC 49
  SFC_STOP Structure 50
Nest an SFC 51
  Pass Parameters 52
Configure When to Return to the OS/JSR. 52

Pause or Reset an SFC 53
Execution Diagrams 53

Chapter 2

Program a Sequential Function Chart

Introduction 59
    Add and Manually Connect Elements 59
    Add and Automatically Connect Elements 60
    Drag and Drop Elements 60
Create a Simultaneous Branch 61
    Start a Simultaneous Branch 61
    End a Simultaneous Branch 61
Create a Selection Branch 62
    Start a Selection Branch 62
    End a Selection Branch 63
Set the Priorities of a Selection Branch 63
Connect a Wire to the Step 64
Hide a Wire 65
Configure a Step 66
    Assign the Preset Time for a Step 66
    Configure Alarms for a Step 66
    Use an Expression to Calculate a Time 67
Program a Transition 68
    Enter a BOOL Expression 68
    Call a Subroutine 68
Add an Action 69
Configure an Action 69
    Change the Qualifier of an Action 69
    Calculate a Preset Time at Runtime 70
    Mark an Action as a Boolean Action 71
Program an Action 71
    Enter Structured Text 71
    Call a Subroutine 72
Assign the Execution Order of Actions 73
Document an SFC 73
    Language Switching 74
    Add Structured Text Comments 74
    Add a Tag Description 75
    Add a Text Box 76
    Show or Hide Text Boxes or Tag Descriptions 77
    Hide an Individual Tag Description 77
Configure the Execution of the SFC 77
Verify the Routine 79
Edit an SFC Online 79
    Maintain Active SFC Step 79

Chapter 3

Force Steps

Introduction 81
Precautions 81
  Enable Forces 81
  Disable or Remove a Force. 82
Check Force Status. 82
  FORCE LED. 83
  GSV Instruction. 84
When to Use Step Through. 84
Step Through a Transition or a Force of a Path. 84
When to Use an SFC Force 85
  Force a Transition 85
  Force a Simultaneous Path 86
Add an SFC Force 87
Remove or Disable Forces. 88
  Disable All SFC Forces 88
  Remove All SFC Forces 88

Notes:

Purpose of This Manual

This manual shows how to design and program Sequential Function Charts (SFCs) for Logix5000 controllers to execute. This manual is one of a set of related manuals that show common procedures for programming and operating Logix5000 controllers. For a complete list of common procedures manuals, see the Logix5000 Controllers Common Procedures Programming Manual, publication 1756-PM001.

The term Logix5000 controller refers to any controller that is based on the Logix5000 operating system.

- CompactLogix controllers
- ControlLogix controllers
- DriveLogix controllers
- FlexLogix controllers
- SoftLogix5800 controllers

How to Use This Manual

Some text is formatted differently from the rest of the text.

Text that is	Identifies	For example	Means
Italic	the actual name of an item that you see on your screen or in an example	Right-click *User-Defined* …	Right-click the item that is named User-Defined.
`courier`	information that you must supply based on your application (a variable)	Right-click `name_of_program` …	You must identify the specific program in your application. Typically, it is a name or variable that you have defined.
enclosed in brackets	a keyboard key	Press [Enter].	Press the Enter key.

Notes:

Design a Sequential Function Chart

Introduction

An SFC is similar to a flowchart of your process. It defines the steps or states through which your system progresses.

- 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 Logix5000 controllers support SFCs, you do not have to enter the specification a second time. You are programming your system as you specify it.

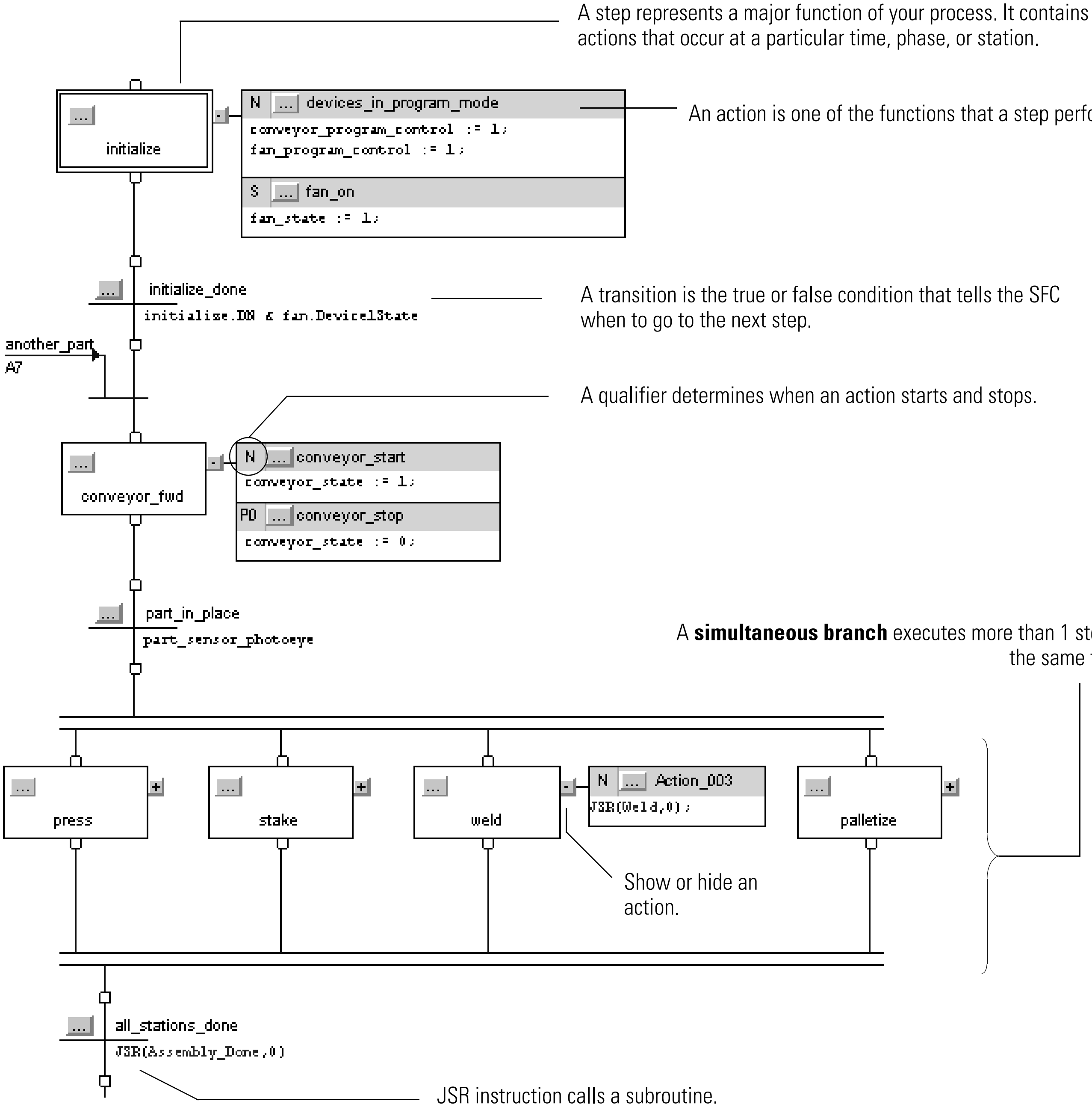
By using an SFC to program your process, you gain these advantages.

- 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.

Figure 1.1 SFC Example


initialize
N
devices_in_program_mode
S
fan_on
initialize_done
another_part
A7
conveyor_fwd
N
conveyor_start
P0
conveyor_stop
part_in_place
press
stake
weld
N
Action_003
palletize
all_stations_done

```
conveyor_program_control := 1;
fan_program_control := 1;
fan_state := 1;
initialize.DN & fan.Device1State
conveyor_state := 1;
conveyor_state := 0;
part_sensor_photoeye
JSR(Weld,0);
JSR(Assembly_Done,0)
```

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

An action is one of the functions that a step performs.

A transition is the true or false condition that tells the SFC when to go to the next step.

A qualifier determines when an action starts and stops.

A **simultaneous branch** executes more than 1 step at the same time.

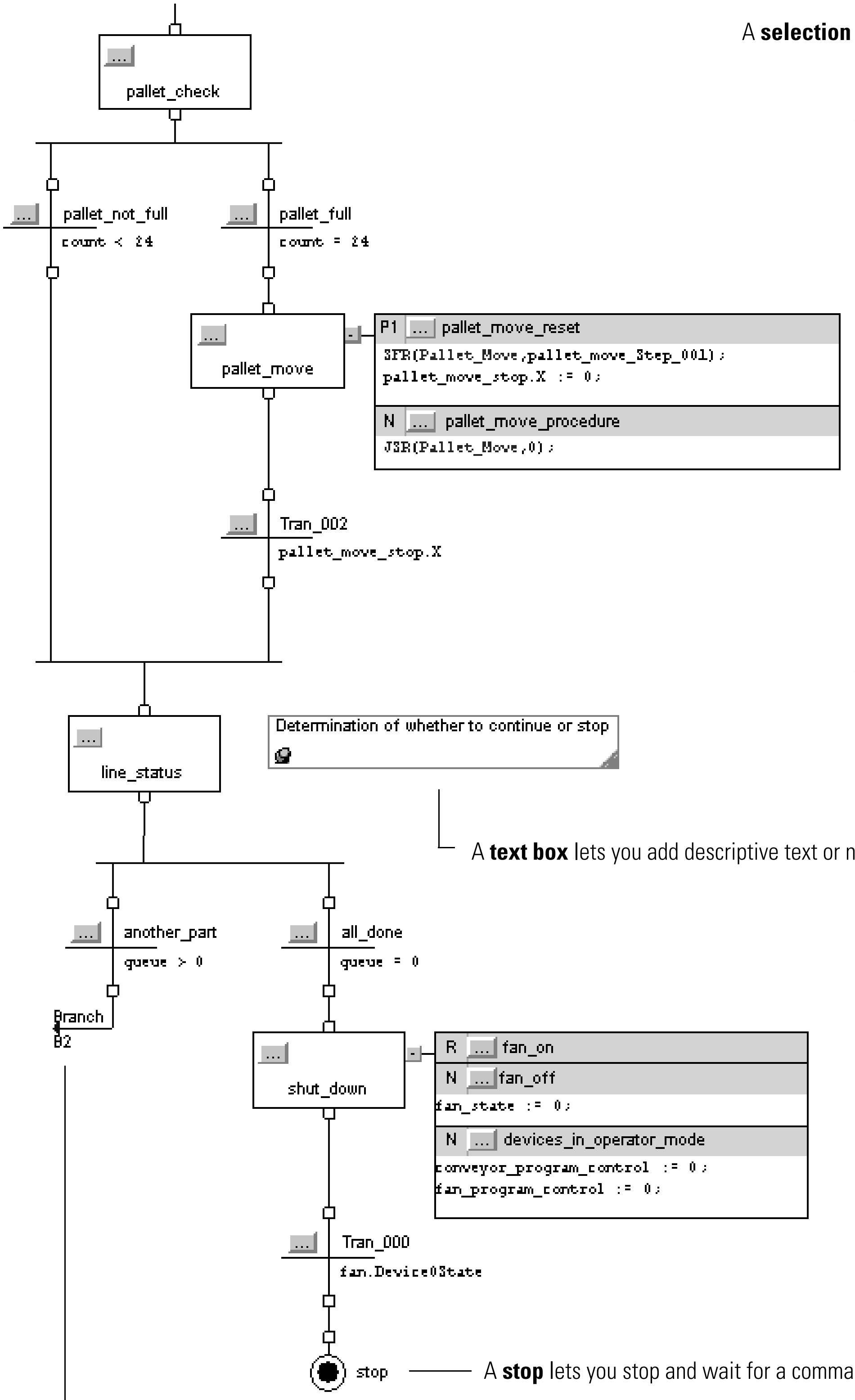
Show or hide an action.

JSR instruction calls a subroutine.

(continued on next page)

Figure 1.2 SFC Example (continued from previous page)

A **selection branch** chooses between different execution paths.


pallet_check
pallet_not_full
pallet_full
pallet_move
P1
pallet_move_reset
N
pallet_move_procedure
Tran_002
line_status
Determination of whether to continue or stop
another_part
all_done
Branch
B2
shut_down
R
fan_on
N
fan_off
N
devices_in_operator_mode
Tran_000
stop

```
count < 24
```

```
count = 24
```

```
SFR(Pallet_Move,pallet_move_Step_001);
pallet_move_stop.X := 0;
```

```
JSR(Pallet_Move,0);
```

```
pallet_move_stop.X
```

```
queue > 0
```

```
queue = 0
```

```
fan_state := 0;
```

```
conveyor_program_control := 0;
fan_program_control := 0;
```

```
fan.Device0State
```

A **text box** lets you add descriptive text or notes to your SFC.

A **stop** lets you stop and wait for a command to restart.

A **wire** connects one element to another element anywhere on the chart. This wire takes you to the *conveyor* step on Figure 1.1 on the previous page.

Follow these steps to design a sequential function chart.

Topic	Page
Define the Tasks	15
Choose How to Execute the SFC	16
Define the Steps of the Process	16
Organize the Steps	21
Add Actions for Each Step	25
Describe Each Action in Pseudocode	30
Choose a Qualifier for an Action	30
Define the Transition Conditions	31
Transition After a Specified Time	35
Turn Off a Device at the End of a Step	38
Keep Something On From Step-to-Step	44
End the SFC	48
Nest an SFC	51
Configure When to Return to the OS/JSR	52
Pause or Reset an SFC	53
Execution Diagrams	53

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. Logix5000 controllers let you divide your project into one continuous task and multiple periodic tasks and event tasks.

1. Organize your project.

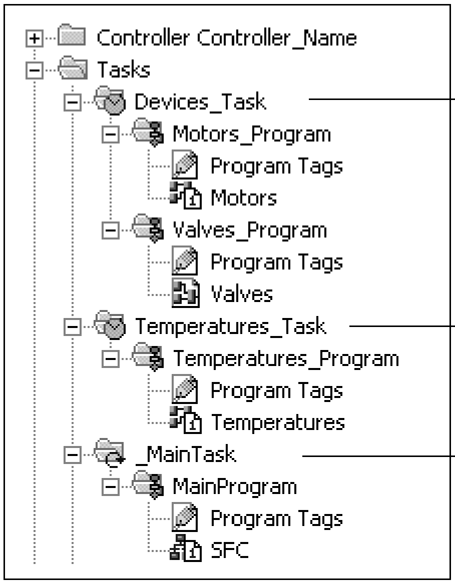
These functions	Go here
Configure and regulate devices	Periodic task
Command a device to a specific state	SFC in the continuous task
Sequence the execution of your process	

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 to control the process.

EXAMPLE Define the Tasks


Controller Controller_Name
Tasks
Devices_Task
Motors_Program
Program Tags
Motors
Valves_Program
Program Tags
Valves
Temperatures_Task
Temperatures_Program
Program Tags
Temperatures
_MainTask
MainProgram
Program Tags
SFC

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.

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.

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

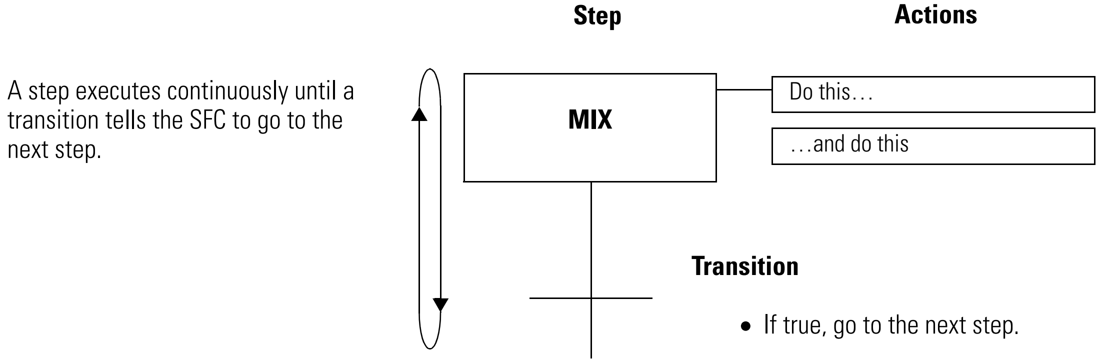
To execute an SFC, either configure it as the main routine for a program or call it as a subroutine.

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

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.

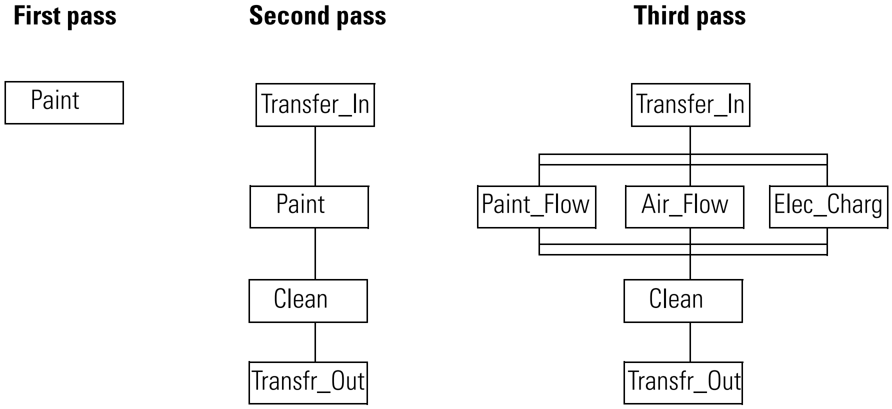

Step
Actions
A step executes continuously until a transition tells the SFC to go to the next step.
MIX
Do this…
…and do this
Transition
• If true, go to the next step.

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

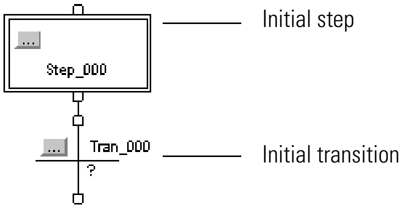
Step Guidelines

Follows these guidelines.

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


First pass
Second pass
Third pass
Paint
Transfer_In
Paint
Clean
Transfr_Out
Transfer_In
Paint_Flow
Air_Flow
Elec_Charg
Clean
Transfr_Out

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


Step_000
Tran_000
?
Initial step
Initial transition

The controller executes the initial step in these situations.

- After a project download when the controller goes into Run mode
- If the SFC is configured to, when the controller transitions to Run mode and on power-up
- 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 when your steps 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 via either the Step Properties dialog box or the Monitor Tags tab of the Tags window.

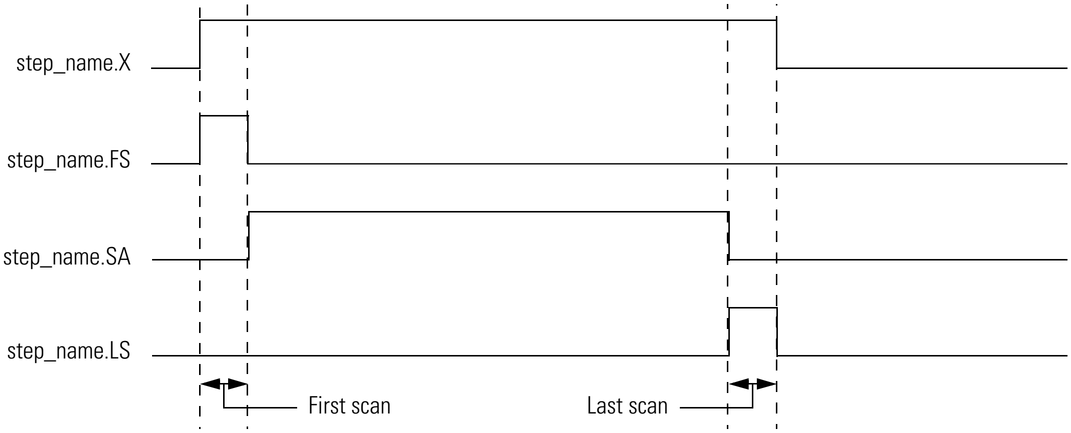
If you want to	Then check or set this member	Data type	Details
Determine how long a step has been active (milliseconds)	T	DINT	When a step becomes active, the Timer (T) value resets and then starts to count up in milliseconds. The timer continues to count up until the step goes inactive, regardless of the Preset (PRE) value.
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, 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.
Flag if a step did not execute long enough	LimitLow	DINT	Enter the time in the 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 (check) the AlarmEnable (AlarmEn) bit. As an option, enter a numeric expression that calculates the time at runtime.
	AlarmEn	BOOL	To use the alarm bits, turn on (check) the AlarmEnable (AlarmEn) bit.
	AlarmLow	BOOL	If the step goes inactive before the Timer (T) reaches the LimitLow value, the AlarmLow bit turns on. • The bit stays on until you reset it. • To use this alarm function, turn on (check) the AlarmEnable (AlarmEn) bit.

If you want to	Then check or set this member	Data type	Details
Flag if a step is executing too long	LimitHigh	DINT	Enter the time in the LimitHigh 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 (check) the AlarmEnable (AlarmEn) bit. As an option, enter a numeric expression that calculates the time at runtime.
	AlarmEn	BOOL	To use the alarm bits, turn on (check) the AlarmEnable (AlarmEn) bit.
	AlarmHigh	BOOL	If the Timer (T) reaches the LimitHigh value, the AlarmHigh bit turns on. • The bit stays on until you reset it. • To use this alarm function, turn on (check) the AlarmEnable (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.
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 P0 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 to which step or stop the SFC will go 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 select the Restart Position of 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.

If you want to	Then check or set this member	Data type	Details	
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

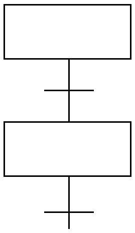
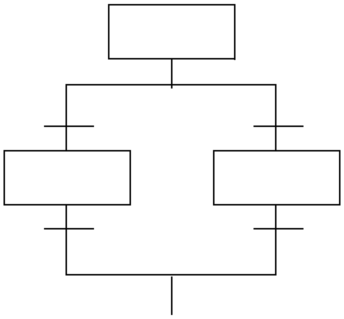
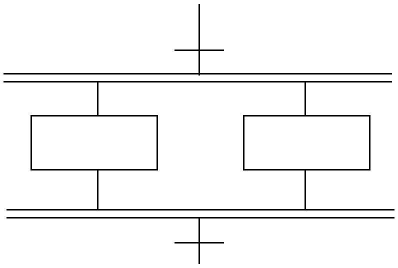
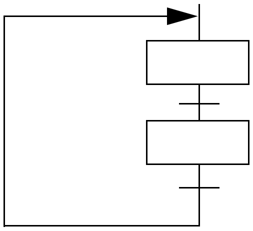
(1) The FS and LS bits are only active during a step's execution. Once a step finishes executing the code within its actions, the FS and/or LS bits are reset. If you reference either of these bits in code outside of the SFC routine in a different part of the project, the bits are always cleared (0).

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

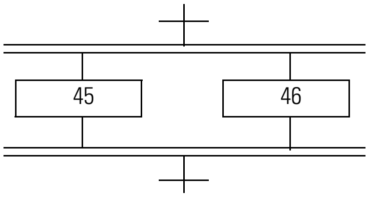
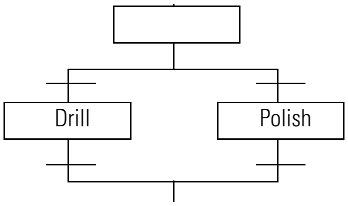
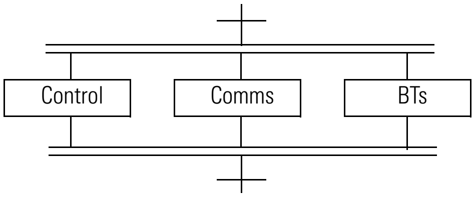
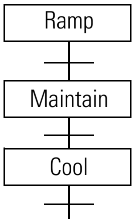
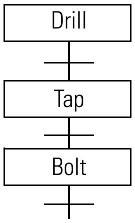
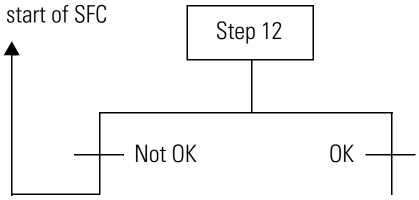


Organize the Steps

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

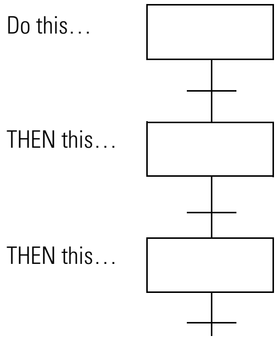
To	Use this structure	With these considerations
• •	**Sequence** 	• •
• •	**Selection Branch** 	• • • •
	Simultaneous Branch 	• •
	Wire to a Previous Step 	• •

Here are some examples of SFC structures for different situations.

Example situation	Example solution
	Simultaneous Branch 
	Selection Branch  Drill Polish
	Simultaneous Branch  Control Comms BTs
	Sequence  Ramp Maintain Cool
	Sequence  Drill Tap Bolt
• •	**Wire**  start of SFC Step 12 Not OK OK

Sequence

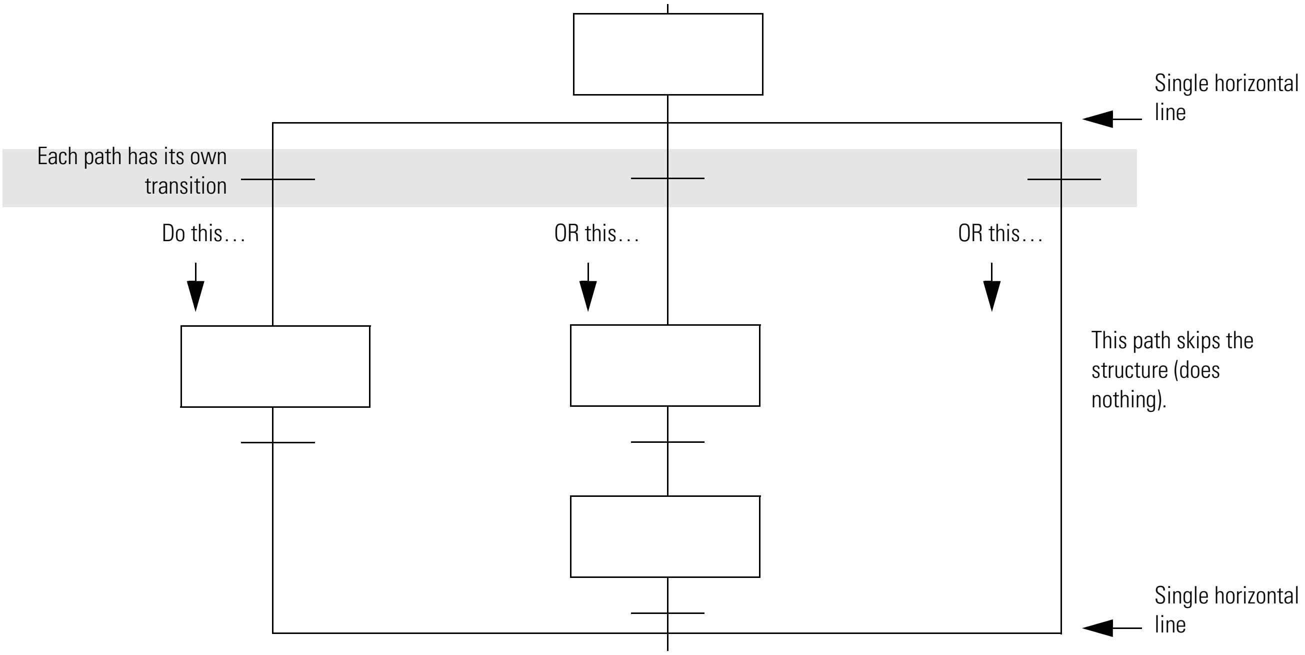
A sequence is a group of steps that execute one after the other.


Do this…
THEN this…
THEN this…

Selection Branch

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

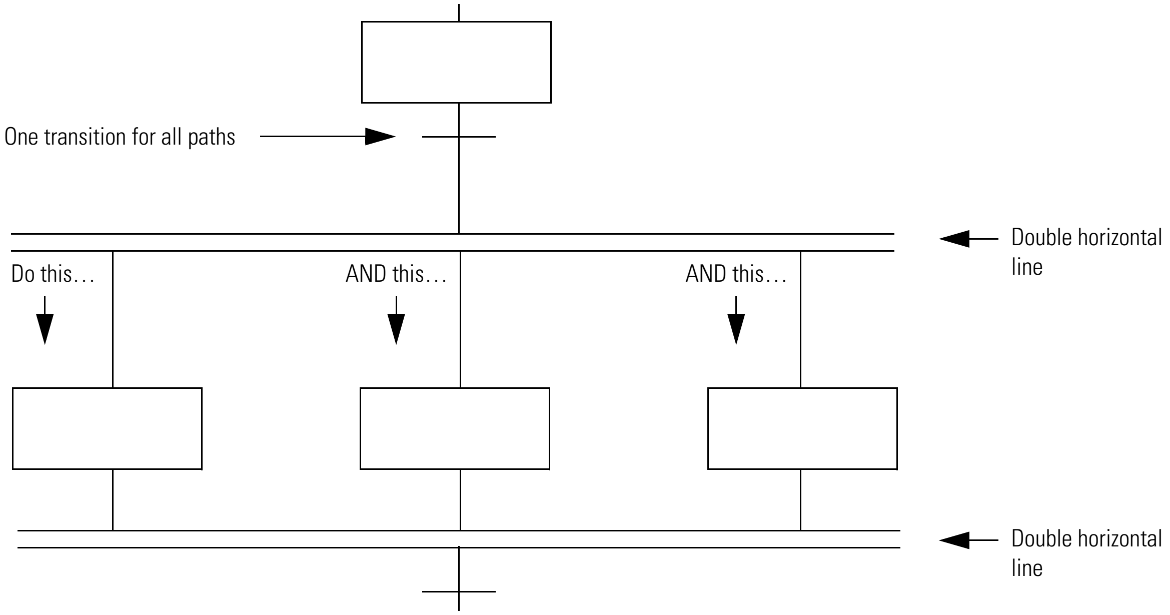
- Only one path executes.
- By default the SFC checks the transitions from left to right.
  - – The SFC takes the first true path.
  - – RSLogix 5000 software lets you change the order in which the SFC checks the transitions (see chapter 2).


Single horizontal line
Each path has its own transition
Do this…
OR this…
OR this…
This path skips the structure (does nothing).
Single horizontal line

Simultaneous Branch

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

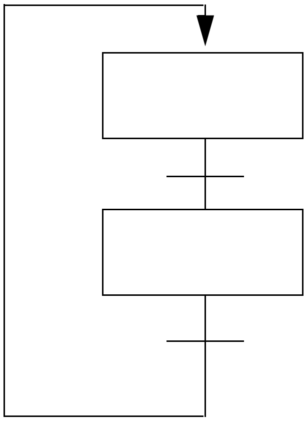
- All paths execute.
- All paths must finish before continuing with the SFC.
- The SFC checks the transition after the last step of each path has executed at least once.


One transition for all paths
Double horizontal line
Do this…
AND this…
AND this…
Double horizontal line

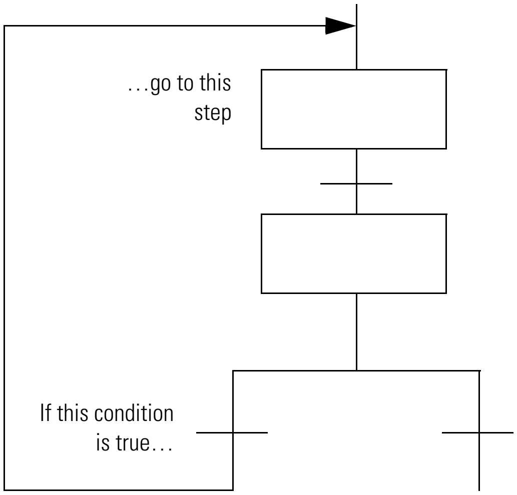
Wire to a Previous Step

In addition to connecting steps in sequences, simultaneous branches, and selection branches, you can 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



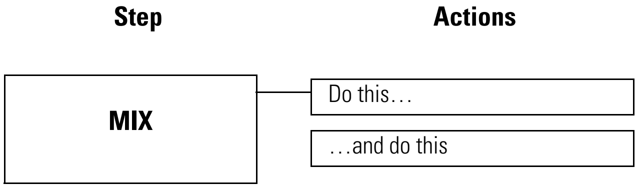
Simple loop that repeats the entire SFC


…go to this step
If this condition is true…

Path of a selection branch that returns to a previous step

Add Actions for Each Step

Use actions 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.


Step
Actions
MIX
Do this…
…and do this

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	Use a Non-Boolean Action
Call a subroutine	
Use the automatic reset option to reset data upon leaving a step	
Only set a bit and program other logic to monitor the bit to determine when to execute.	Use a Boolean Action

Use a Non-Boolean Action

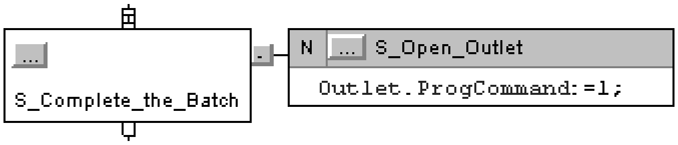
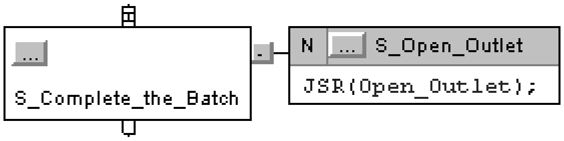
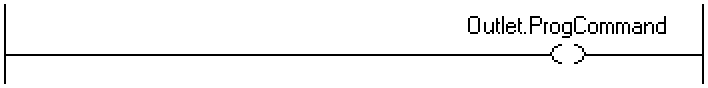
A non-boolean action contains the logic for the action. It uses Structured Text to execute assignments and instructions or call a subroutine.

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" on page 38.

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.  `Outlet.ProgCommand:=1;` 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.
• Re-use logic in multiple steps • Use another language to program the action, such as ladder logic • Nest an SFC	Call a subroutine.  `JSR(Open_Outlet);` 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.

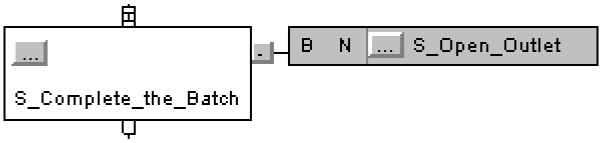
You cannot reuse a non-boolean action within the same SFC except to reset a stored action. Only one instance of a specific non-boolean action is permitted per SFC.

Use a Boolean Action

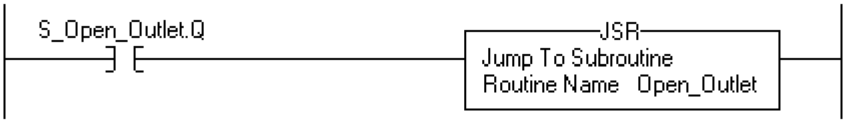
A boolean action contains no logic for the action. It simply sets a bit in its tag (SFC_ACTION structure). To do the action, other logic must monitor the bit and execute when the bit is on.

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 effect boolean actions.

EXAMPLE


...
S_Complete_the_Batch
-
B
N
...
S_Open_Outlet

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.


S_Open_Outlet.Q
JSR
Jump To Subroutine
Routine Name Open_Outlet

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 or the Monitor Tags tab of the Tags window.

<table>
<tr><th>If you want to</th><th>Then check or set this member</th><th>Data type</th><th>Details</th></tr>
<tr><td>Determine when the action is active</td><td>Q</td><td>BOOL</td><td>The status of the Q bit depends on whether the action is a boolean action or non-boolean action.
<table>
<tr><th>If the action is</th><th>Then the Q bit is</th></tr>
<tr><td>Boolean</td><td>On (1) the entire time the action is active, including the last scan of the action</td></tr>
<tr><td>Non-boolean</td><td>On (1) while the action is active but
Off (0) at the last scan of the action</td></tr>
</table>
To use a bit to determine when an action is active, use the Q bit.</td></tr>
<tr><td></td><td>A</td><td>BOOL</td><td>The A bit is on the entire time the action is active.</td></tr>
<tr><td>Determine how long an action has been active (milliseconds)</td><td>T</td><td>DINT</td><td>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.</td></tr>
<tr><td>Use one of these time-based qualifiers: L, SL, D, DS, SD</td><td>PRE</td><td>DINT</td><td>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.</td></tr>
<tr><td>Determine how many times an action has become active</td><td>Count</td><td>DINT</td><td>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.</td></tr>
<tr><td>Use one tag for the various status bits of this action</td><td>Status</td><td>DINT</td><td>
<table>
<tr><th>For this member</th><th>Use this bit</th></tr>
<tr><td>Q</td><td>30</td></tr>
<tr><td>A</td><td>31</td></tr>
</table></td></tr>
</table>

Describe Each Action in Pseudocode

To organize the logic for an action, first 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 (when 1st) and then the action to take second (what 2nd).

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 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.

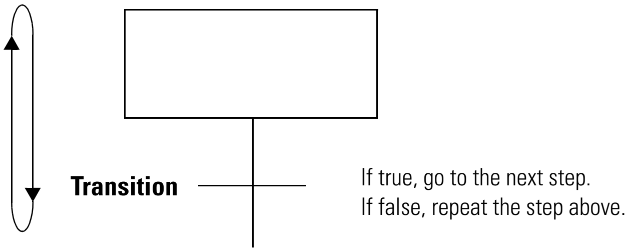
Choose a Qualifier for an Action

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

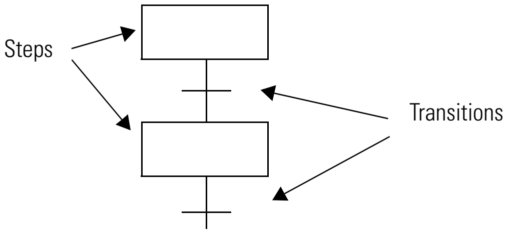
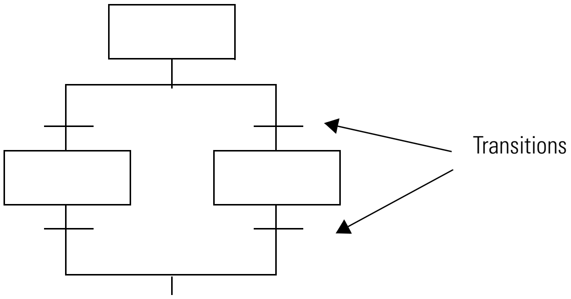
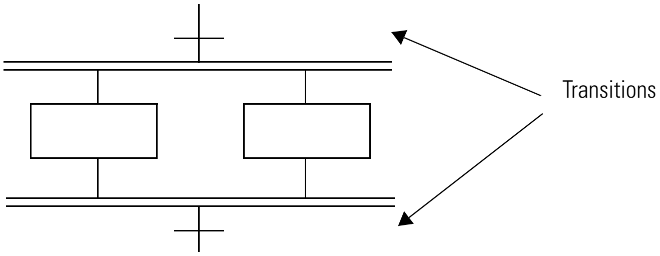
Define the Transition Conditions

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.


Transition
If true, go to the next step.
If false, repeat the step above.

Transitions occur in these structures.

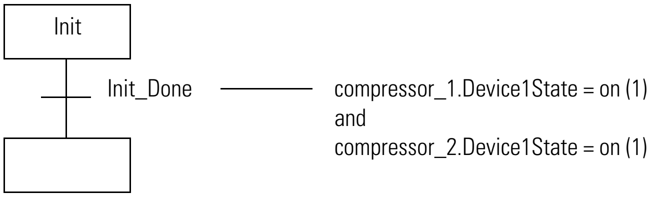
For this structure	Make sure that
Sequence	A transition is between each step.  Steps Transitions
Selection branch	Transitions are inside the horizontal lines.  Transitions
Simultaneous branch	Transitions are outside the horizontal lines.  Transitions

EXAMPLE

You want to complete these steps.

a. Turn on 2 compressors. When a compressor is on, the Device1State bit is on.

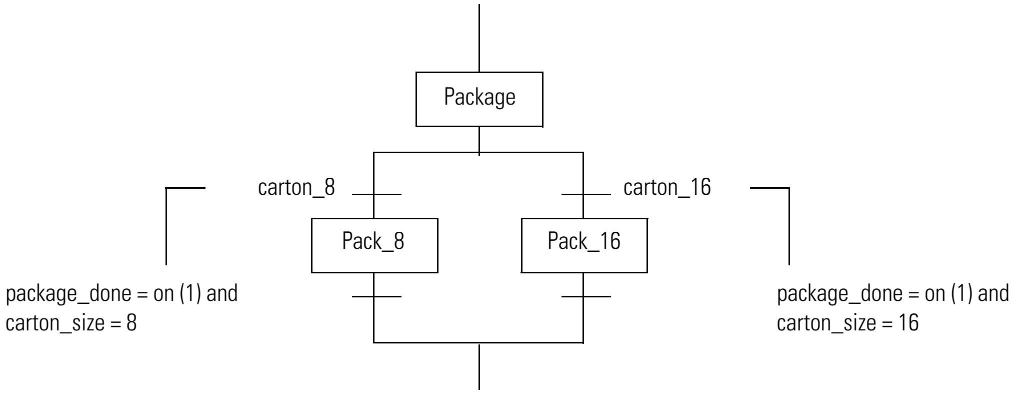
b. When both compressors are on, go to the next step.


Init
Init_Done
compressor_1.Device1State = on (1)
and
compressor_2.Device1State = on (1)

EXAMPLE

You want to complete these steps.

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


Package
carton_8
carton_16
Pack_8
Pack_16
package_done = on (1) and carton_size = 8
package_done = on (1) and carton_size = 16

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.

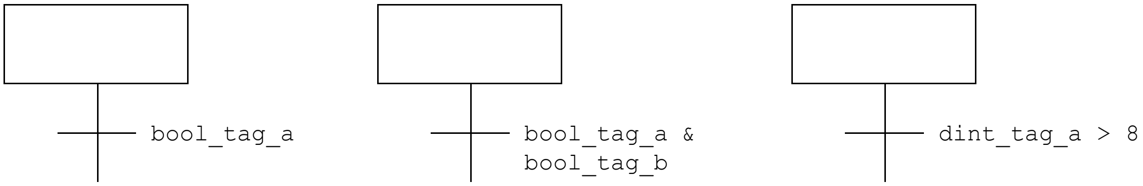
How Do You Want to Program the Transition?

To program the transition, you have these 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	

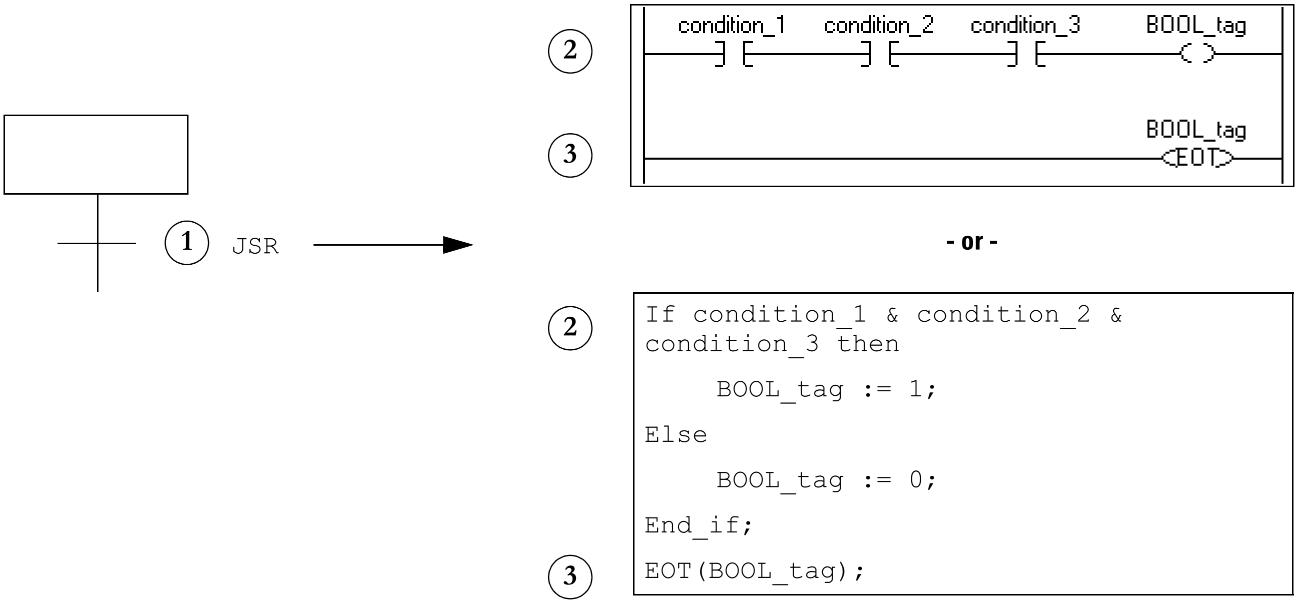
Use a BOOL Expression

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`.


bool_tag_a
bool_tag_a & bool_tag_b
dint_tag_a > 8

Call a Subroutine

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 below.


①
JSR
②
condition_1
condition_2
condition_3
BOOL_tag
③
BOOL_tag
EOT
- or -
②
③

```
If condition_1 & condition_2 &
condition_3 then
    BOOL_tag := 1;
Else
    BOOL_tag := 0;
End_if;
EOT(BOOL_tag);
```

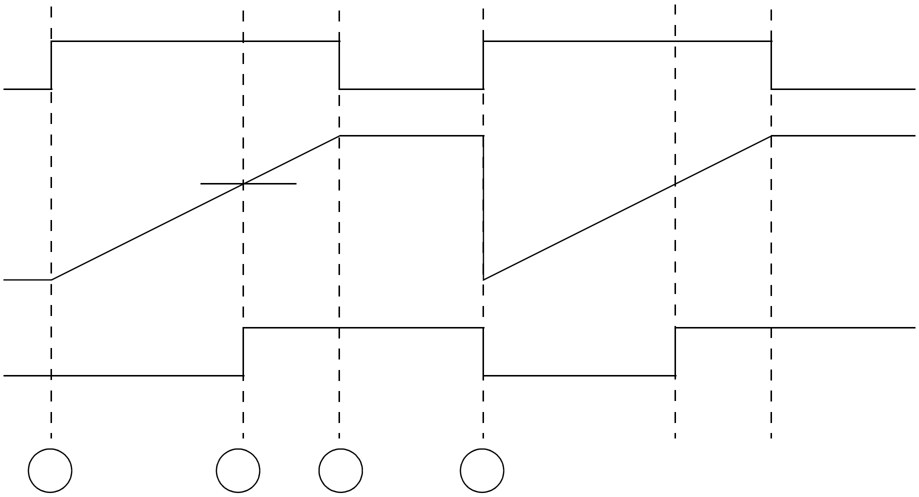
1. Call a subroutine.
2. Check for the required conditions. When those conditions are true, turn on a BOOL tag.
3. Use an EOT instruction to set the state of the transition equal to the value of the BOOL tag. When the BOOL tag is on (true), the transition is true.

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.

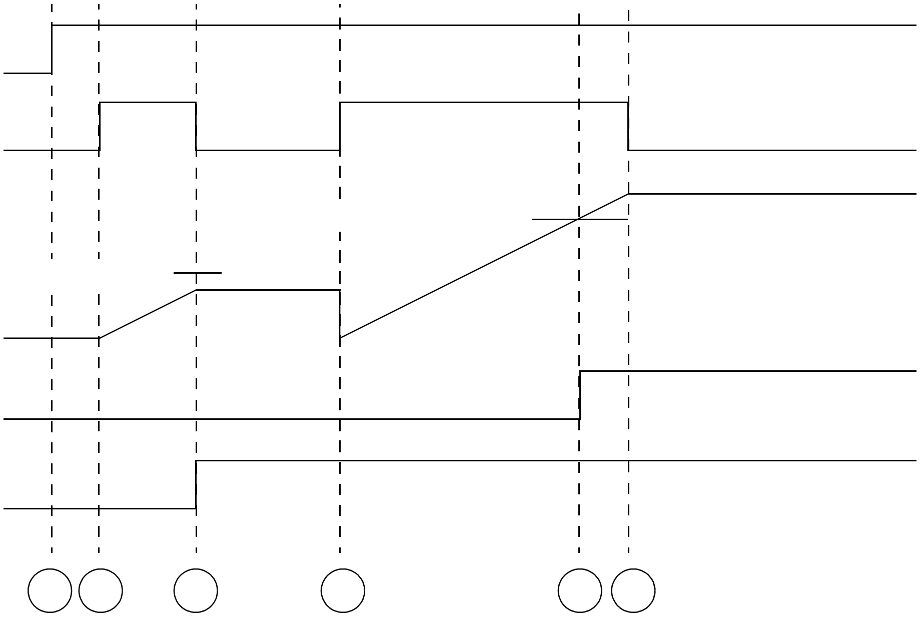
- Signal when the step has run for the required time and the SFC should go to the next step.
- Signal when the step has run too long and the SFC should go to an error step.

Figure 1.3 Action of a timer and associated bits of a step



Description

Figure 1.4 Action of the low and high alarms for a step



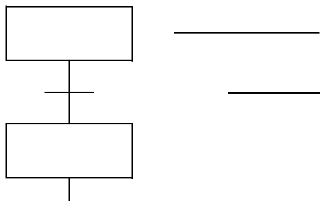
Description

Here is an example of the use of the Preset time of a step.

EXAMPLE

The functional specification has these requirements.

a. Cook the ingredients in the tank for 20 seconds.
b. Empty the tank.

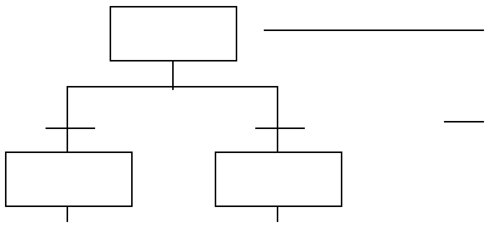


Here is an example of the use of the high alarm of a step.

EXAMPLE

The functional specification has these requirements.

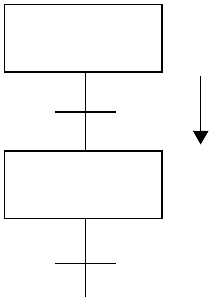
a. Home 8 devices.
b. If all 8 devices are not home within 20 seconds, then shutdown the system.



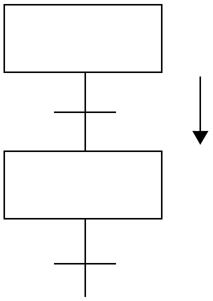
Turn Off a Device at the End of a Step

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

Programmatic Reset



Automatic Reset



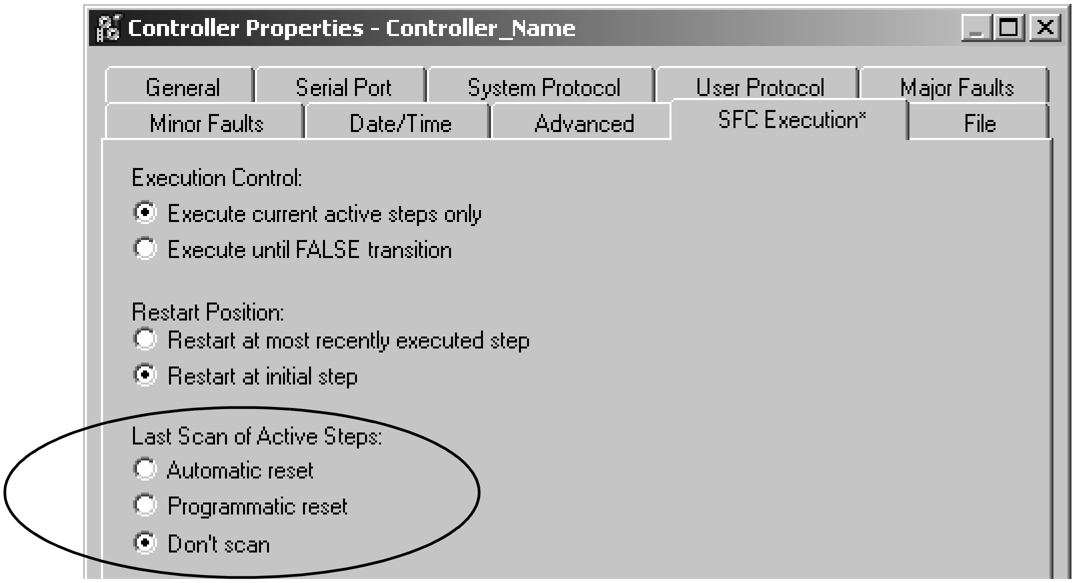
Each option requires you to make these decisions.

- Choose a last scan option.
- Based on the last scan option, develop your logic so that the last scan returns data to the desired 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 steps in all SFCs of this controller.

If you want to	And on the last scan of a step	Then	See
	• •		
	→		



Execution Control:

● Execute current active steps only

○ Execute until FALSE transition

Restart Position:

○ Restart at most recently executed step

● Restart at initial step

Last Scan of Active Steps:

○ Automatic reset

○ Programmatic reset

● Don't scan

This 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 P0 actions execute. They execute according to their logic.	All actions execute according to their logic.	• P and P0 actions execute according to their logic. • All other actions execute in Postscan mode. • On the next scan of the routine, the P and P0 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 P0 actions.	Use either of these. • Status bits of the step or action to condition logic • P and P0 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

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.

Follow these steps to turn off a device at the end of a step.

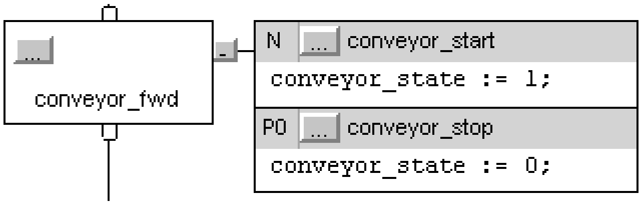
1. Make sure that the Last Scan of Active Steps property is set to the Don't scan option (default).
2. Use a P0 Pulse (Falling Edge) action to clear the required data. Make sure that the P0 action or actions are last in the order of actions for the step.

During the last scan of the step, the Don't scan option executes only P and P0 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.

EXAMPLE Use the Don't Scan Option



```
conveyor_state := 1;
```

```
conveyor_state := 0;
```

This action turns on the conveyor. When conveyor_state turns on, the conveyor turns on.

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.

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.

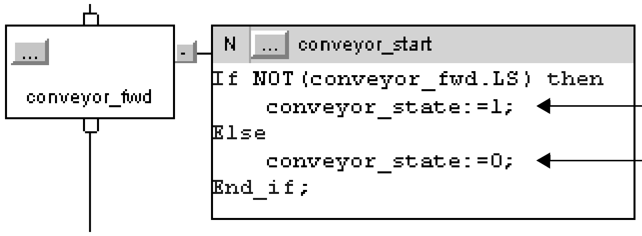
During the last scan of the step, the Programmatic reset option executes all assignments and instructions according to logic conditions.

- 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. The LS bit of the step conditions the execution of the logic. See “SFC_STEP Structure” on page 18.

EXAMPLE

Use the Programmatic Reset Option and the LS Bit



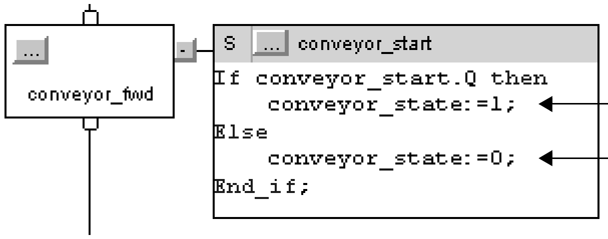
```
If NOT(conveyor_fwd.LS) then
    conveyor_state:=1;
Else
    conveyor_state:=0;
End_if;
```

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.

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 Use the Programmatic Reset Option and the Q Bit



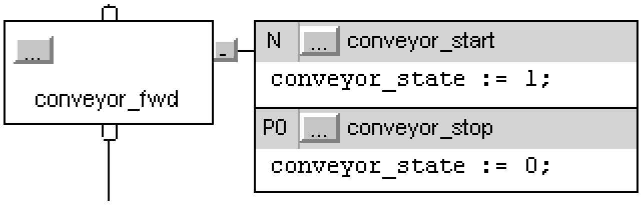
```
If conveyor_start.Q then
    conveyor_state:=1;
Else
    conveyor_state:=0;
End_if;
```

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.

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 Use the Programmatic Reset Option and a P0 Action



```
conveyor_state := 1;
```

```
conveyor_state := 0;
```

This action turns on the conveyor. When conveyor_state turns on, the conveyor turns on.

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.

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.

Follow these steps to automatically turn off (clear) devices at the end of a step.

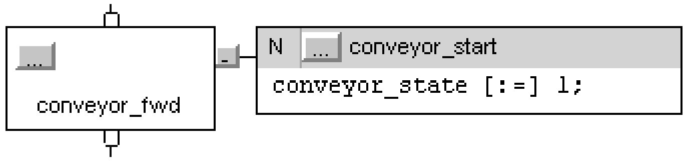
1. In the Last Scan of Active Steps property, select the Automatic reset option.
2. To turn off a device at the end of the step, control the state of the device with an assignment or instruction.
   - [:=] assignment (non-retentive assignment)
   - Output Energize (OTE) instruction in a subroutine

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

- Logix5000 Controllers General Instructions Reference Manual, publication 1756-RM003
- Logix5000 Controllers Process and Drives Instructions Reference Manual, publication 1756-RM006
- Logix5000 Controllers Motion Instruction Set Reference Manual, publication 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.

EXAMPLE Automatically Clear Data



```
conveyor_state [:=] 1;
```

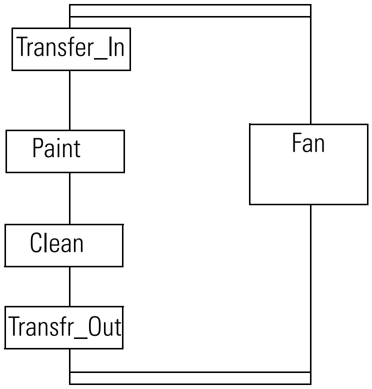
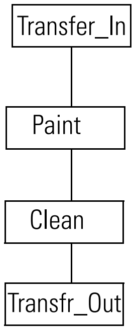
This action turns on the conveyor. When conveyor_state turns on, the conveyor turns on.

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

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 Make a separate step that controls the device.	
Store and Reset an Action 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.	 Turn on the fan Turn off the fan
Use One Large Step Use one large step that contains all the actions that occur while the device is on.	 Transfer, paint, clean, transfer, control the fan

Use a Simultaneous Branch

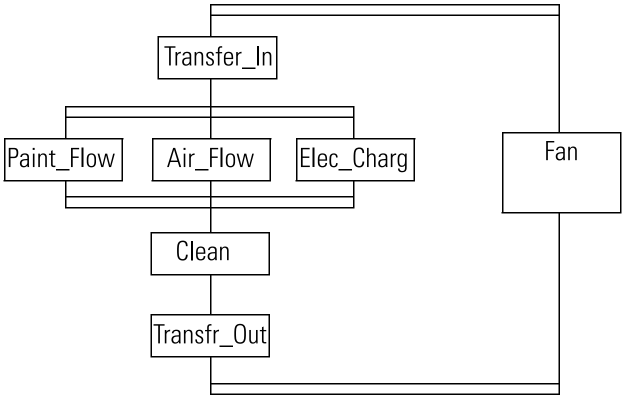
A simple way to control a device or devices during one or more steps is to create a separate step for the devices. Then use a simultaneous branch to execute the step during the rest of the process.

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.

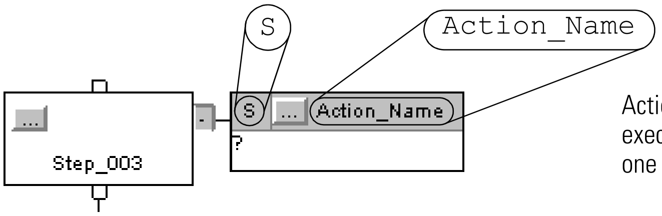


Store and Reset an Action

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.

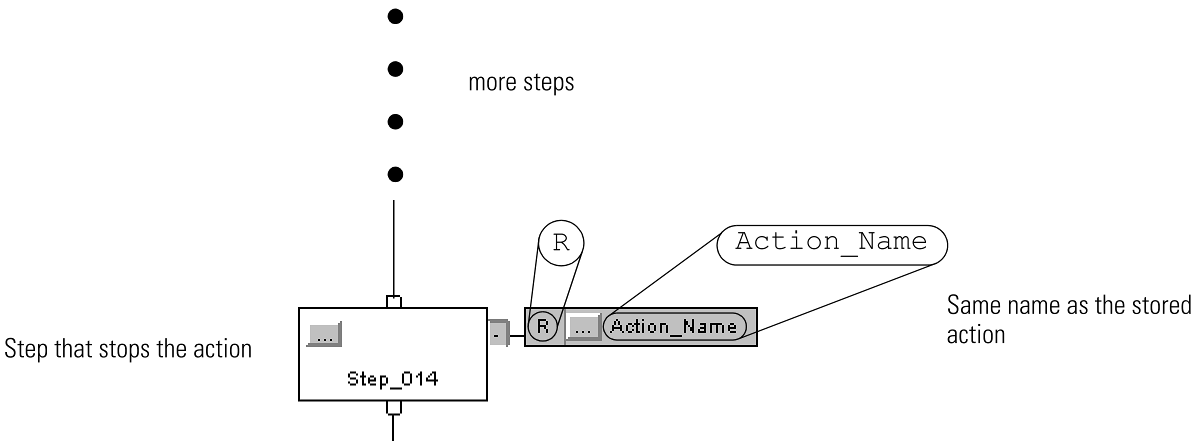
1. In the step that turns on the device, assign a stored qualifier to the action that controls the device.
2. In the step that turns off the device, use a Reset action.

This figure shows the use of a stored action.



Step that starts the action

Action that you want to execute for more than one step

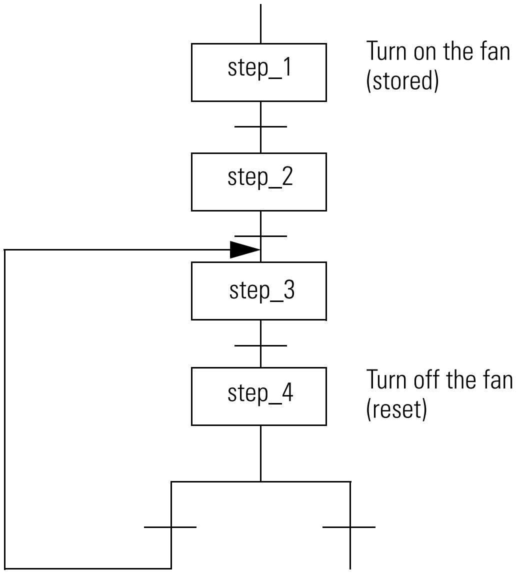


When the SFC leaves the step that stores the action, RSLogix 5000 software continues to show the stored action as active. (By default, a green border displays around the action.) This lets you know that the SFC is executing the logic of that action.

To use 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 on page 42.
- 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, steps 1 - 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.



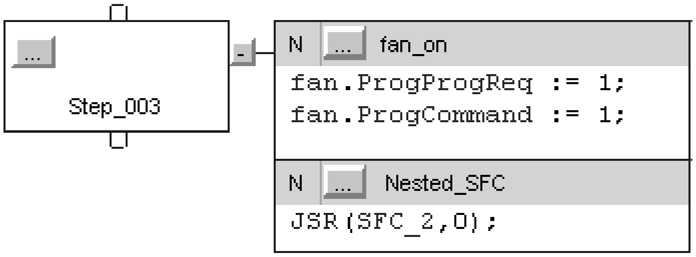
To turn the fan back on, the SFC has to jump back to step_1.

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.

EXAMPLE Use a Large Step



```
fan.ProgProgReq := 1;
fan.ProgCommand := 1;
```

```
JSR(SFC_2,0);
```

This action turns on a fan.

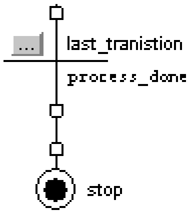
- fan.ProgProgReq lets the SFC command the state of the fan.
- fan.ProgCommand turns on the fan.

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.



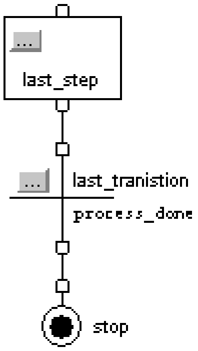
Use a Stop Element

The stop element lets you stop the execution of an entire SFC or a path of a simultaneous branch and wait to restart. When an SFC reaches a stop element, these 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	Entire SFC stops
Selection branch	
Path within a simultaneous branch	Only that path stops while the rest of the SFC continues to execute.

EXAMPLE Use a Stop Element



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 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	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.
	Don't scan	
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.

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

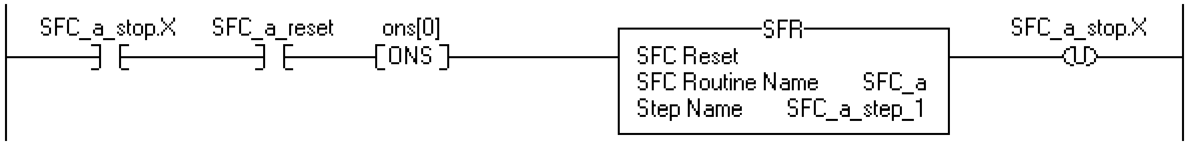
EXAMPLE

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

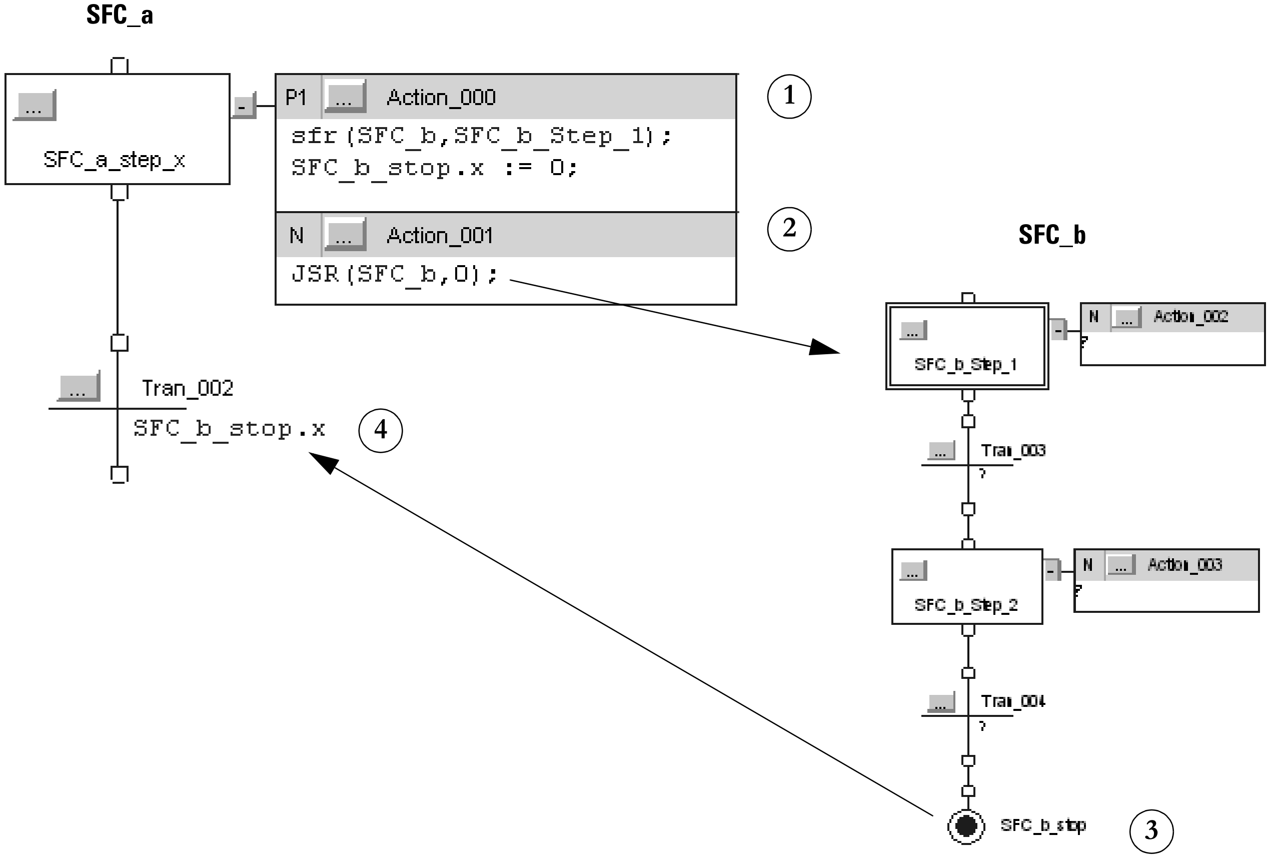
Each stop uses a tag to provide information about the stop element.

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 is unnecessary.


SFC_a
SFC_a_step_x
P1
Action_000
①
N
Action_001
②
Tran_002
④
SFC_b
SFC_b_Step_1
N
Action_002
?
Tran_003
?
SFC_b_Step_2
N
Action_003
?
Tran_004
?
SFC_b_stop
③

```
sfr(SFC_b,SFC_b_Step_1);
SFC_b_stop.x := 0;
```

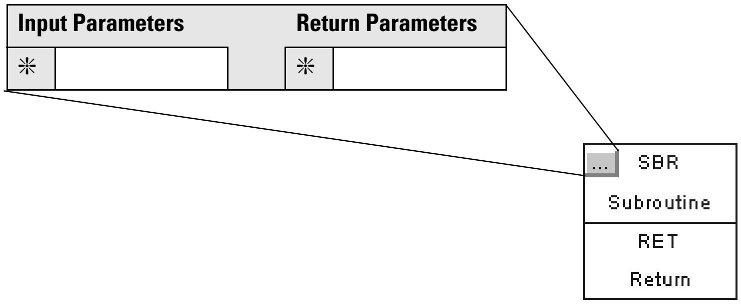
```
JSR(SFC_b,0);
```

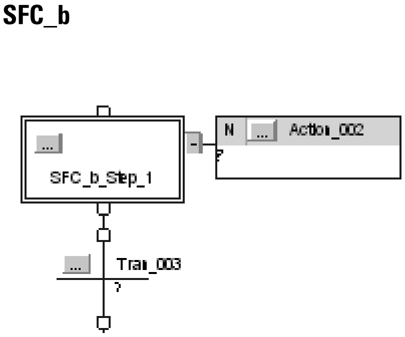
```
SFC_b_stop.x
```

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

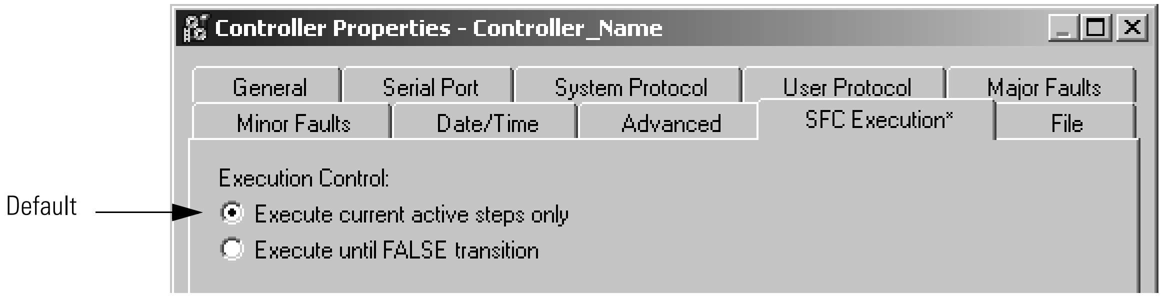
To pass parameters to or from an SFC, place a Subroutine/Return element in the SFC.


Input Parameters
Return Parameters
SBR
Subroutine
RET
Return


SFC_b
SFC_b_Step_1
N
Action_002
Tran_003

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).


Controller Properties - Controller_Name
General
Serial Port
System Protocol
User Protocol
Major Faults
Minor Faults
Date/Time
Advanced
SFC Execution*
File
Default

Execution Control:

● Execute current active steps only

○ Execute until FALSE transition

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 desired step.

Use the Execute until FALSE transition option only when either of these are true.

- You don’t 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.

For a diagram of the	Page
Execution of a Sequence	54
Execution of a Simultaneous Branch	55
Execution of a Selection Branch	56
When parameters enter and exit an SFC	56
Options for Execution Control	57

Figure 1.5 Execution of a Sequence

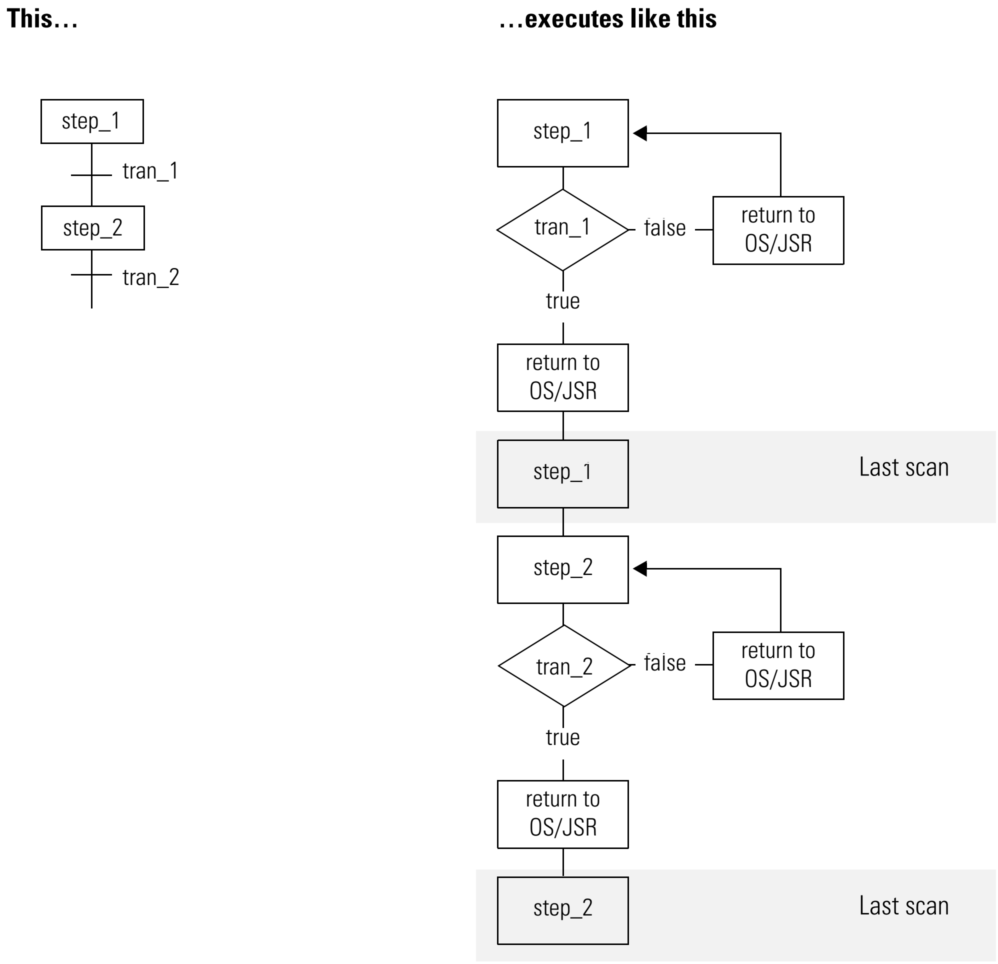
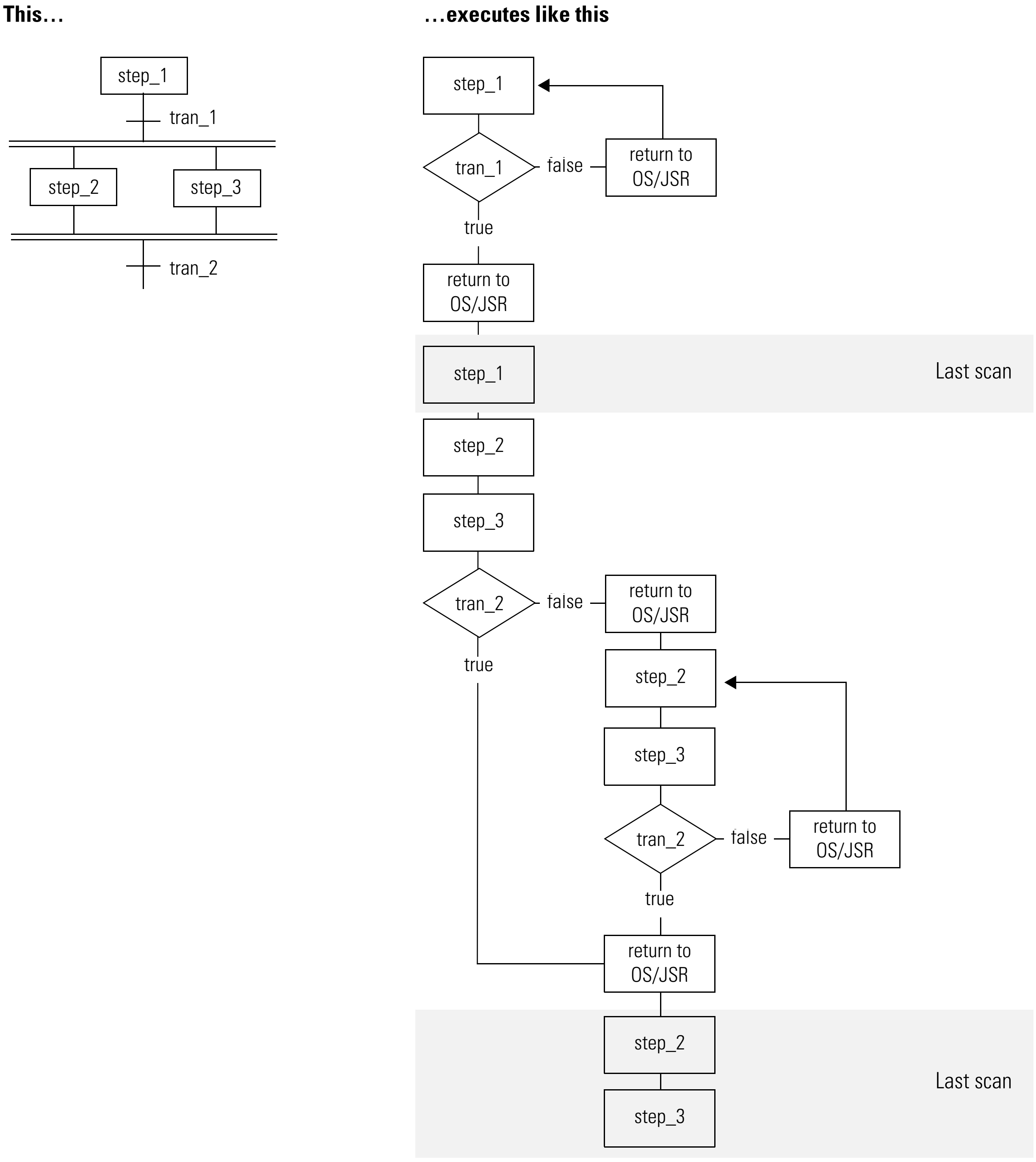

This…
…executes like this
step_1
tran_1
step_2
tran_2
step_1
tran_1
false
return to OS/JSR
true
return to OS/JSR
step_1
Last scan
step_2
tran_2
false
return to OS/JSR
true
return to OS/JSR
step_2
Last scan

Figure 1.6 Execution of a Simultaneous Branch

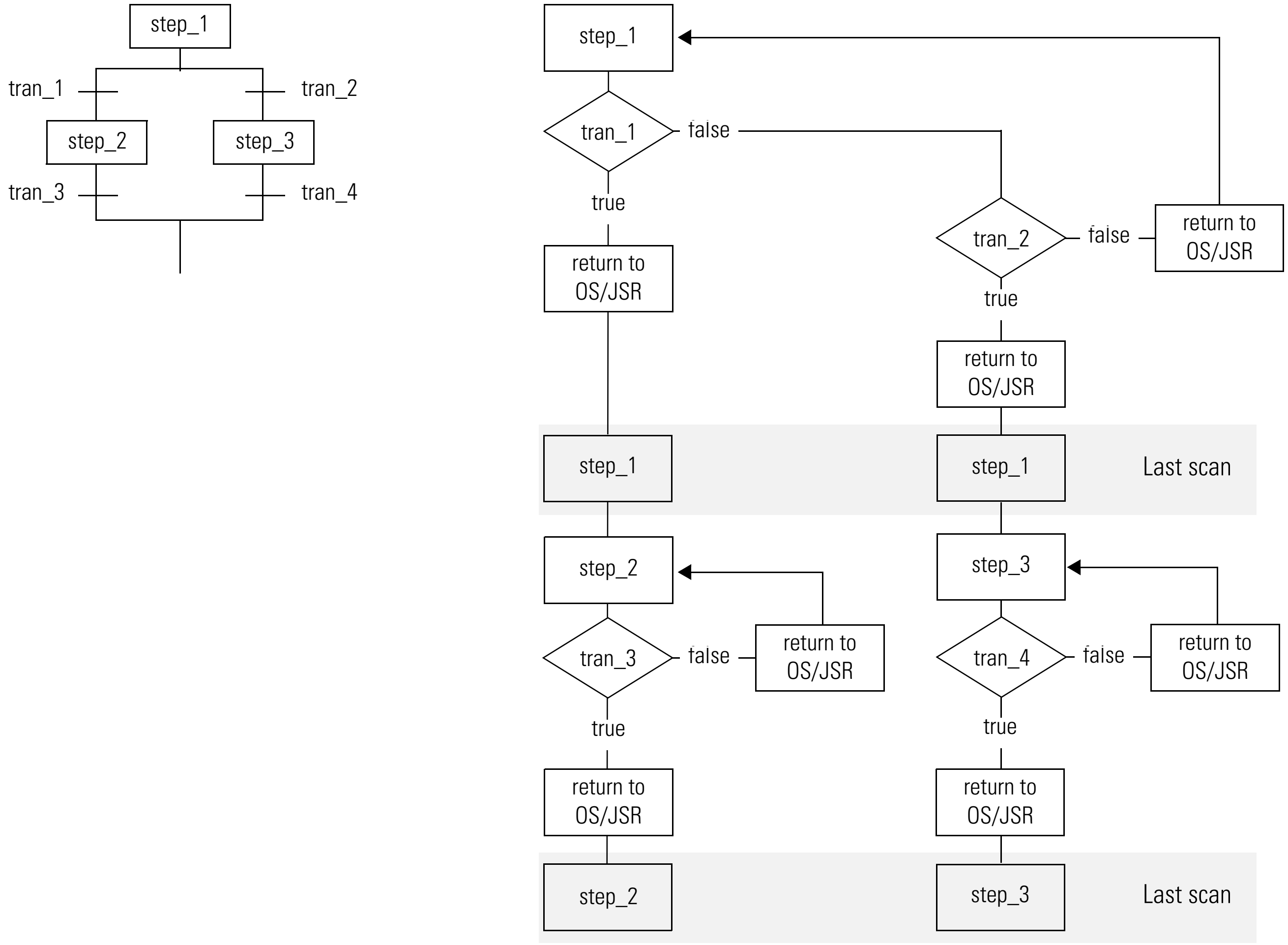


This…

…executes like this

step_1
tran_1
step_2
step_3
tran_2
step_1
tran_1
false
return to OS/JSR
true
return to OS/JSR
step_1
Last scan
step_2
step_3
tran_2
false
return to OS/JSR
true
step_2
step_3
tran_2
false
return to OS/JSR
true
return to OS/JSR
step_2
step_3
Last scan

Figure 1.7 Execution of a Selection Branch

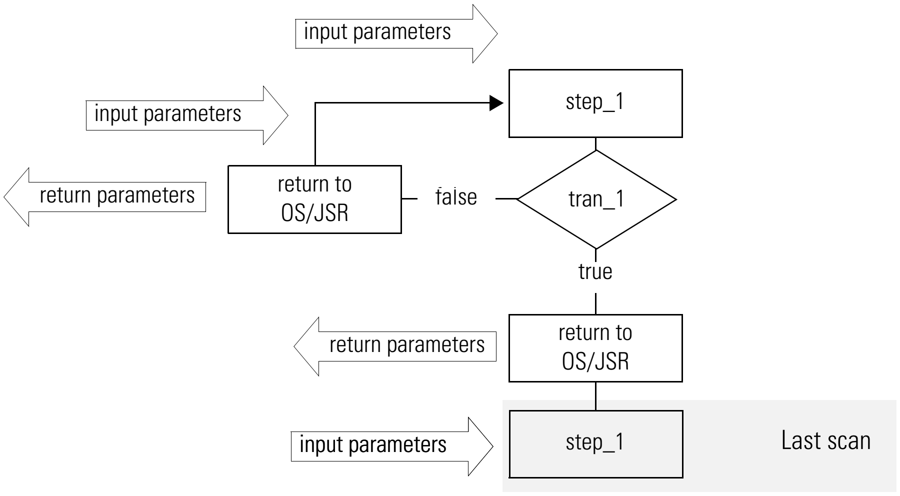


This…

…executes like this

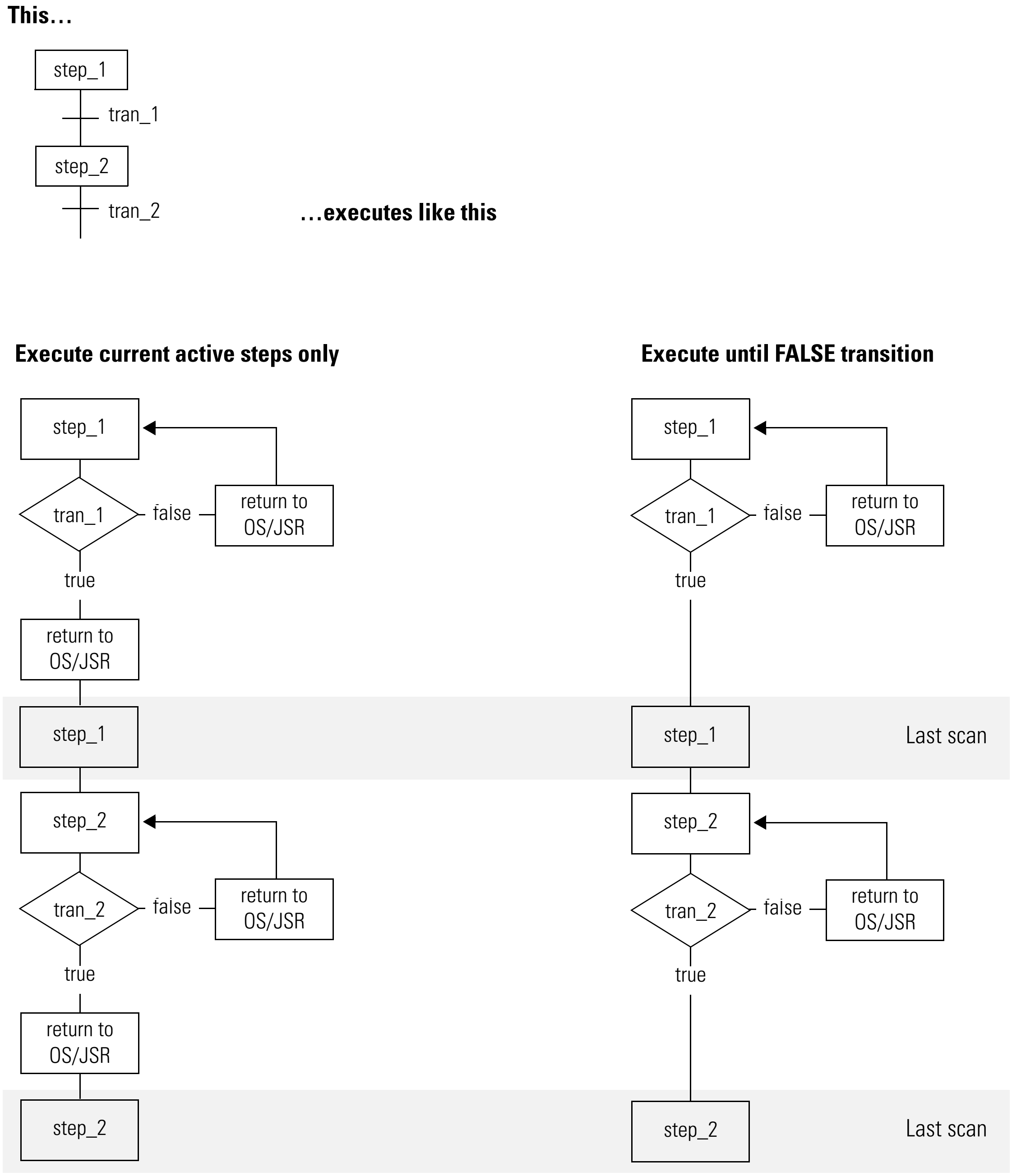
step_1
tran_1
tran_2
step_2
step_3
tran_3
tran_4
step_1
tran_1
false
true
return to OS/JSR
tran_2
false
return to OS/JSR
true
return to OS/JSR
step_1
step_1
Last scan
step_2
step_3
tran_3
false
return to OS/JSR
tran_4
false
return to OS/JSR
true
true
return to OS/JSR
return to OS/JSR
step_2
step_3
Last scan

Figure 1.8 When parameters enter and exit an SFC



input parameters
input parameters
step_1
return parameters
return to OS/JSR
false
tran_1
true
return parameters
return to OS/JSR
input parameters
step_1
Last scan

Figure 1.9 Options for Execution Control


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

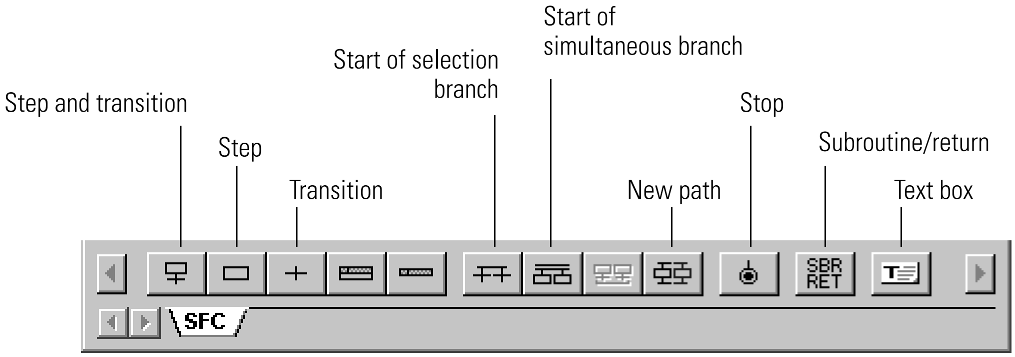
Notes:

Chapter 2

Program a Sequential Function Chart

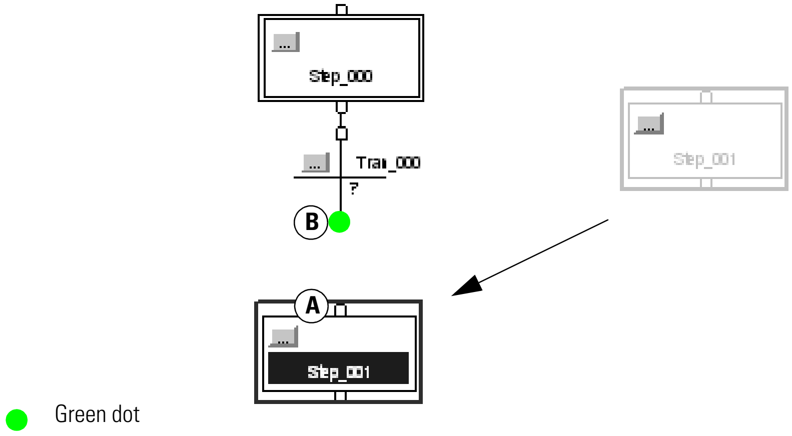
Introduction

To add SFC elements, use the SFC toolbar.


Start of simultaneous branch
Start of selection branch
Step and transition
Stop
Step
Subroutine/return
Transition
New path
Text box
SBR RET
SFC

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.


Step_000
Tran_000
?
B
Step_001
A
Step_001
Green dot

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

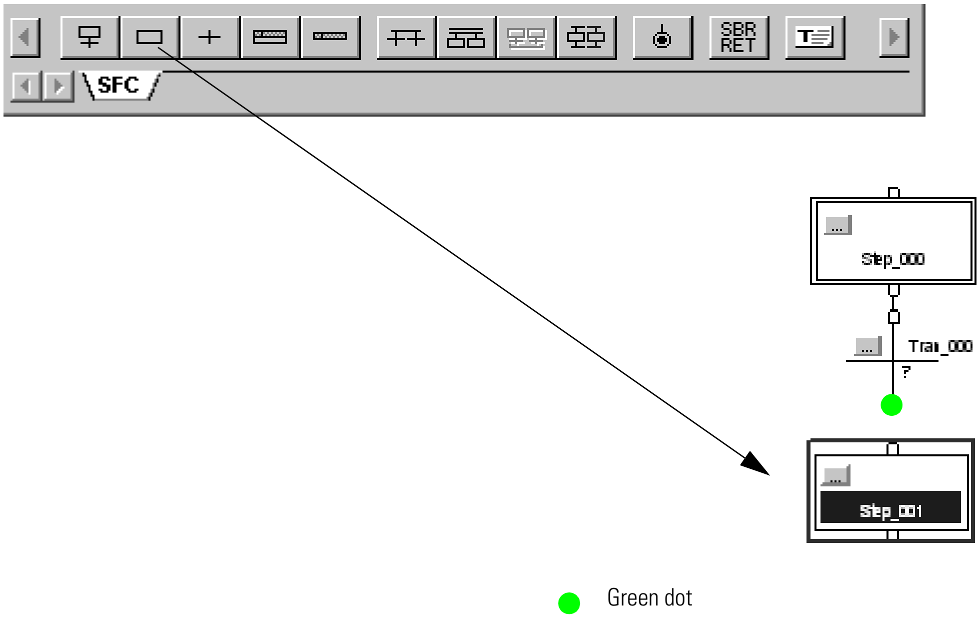
IMPORTANT Use caution when copying and pasting components between different versions of RSLogix 5000 programming software. RSLogix 5000 software only supports pasting to the same version or newer version of RSLogix 5000. Pasting to a prior version of RSLogix 5000 software is not supported. When pasting to a prior version, the paste action may succeed but the results may not be as intended.

Add and Automatically Connect Elements

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

Drag and Drop Elements

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

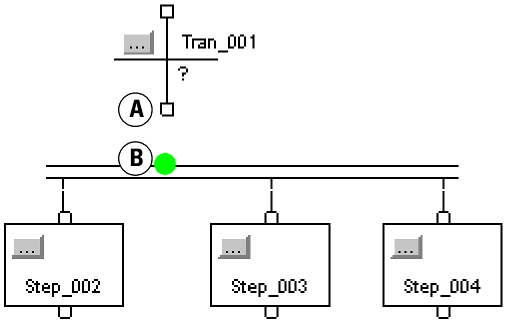


Create a Simultaneous Branch

Start a Simultaneous Branch

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

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



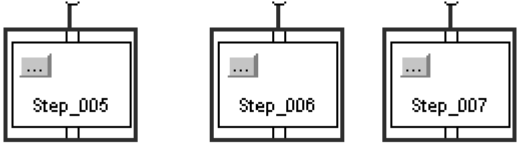
● Green dot

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

End a Simultaneous Branch

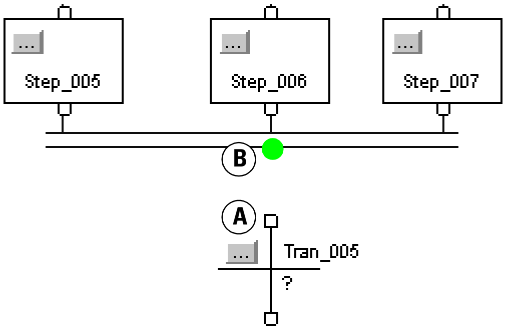
1. Select the last step of each path in the branch. To select the steps, do either of these actions.

   - Click and drag the pointer around the steps that you want to select.
   - Click the first step. Then press and hold [Shift] and click 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.



● Green dot

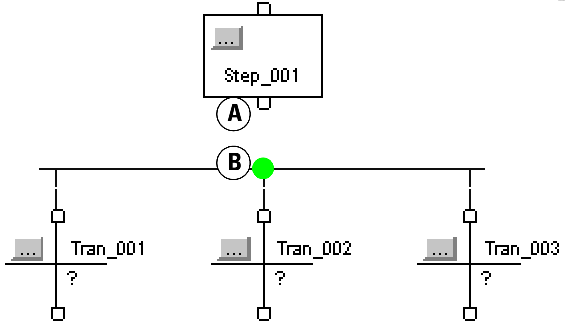
4. To wire the simultaneous branch to the transition, click the top pin of the transition Ⓐ and then click the horizontal line of the branch Ⓑ. A green dot shows a valid connection point.

Create a Selection Branch

Start a Selection Branch

1. On the SFC toolbar, click the button. Then drag the new branch to the desired location.

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



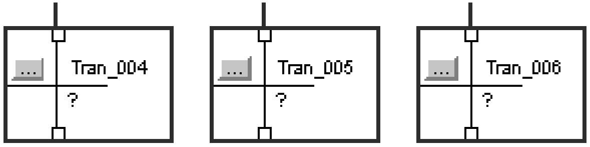
● Green dot

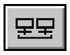
3. To wire the selection branch to the preceding step, click the bottom pin of the step Ⓐ and then click the horizontal line of the branch Ⓑ. A green dot shows a valid connection point.

End a Selection Branch

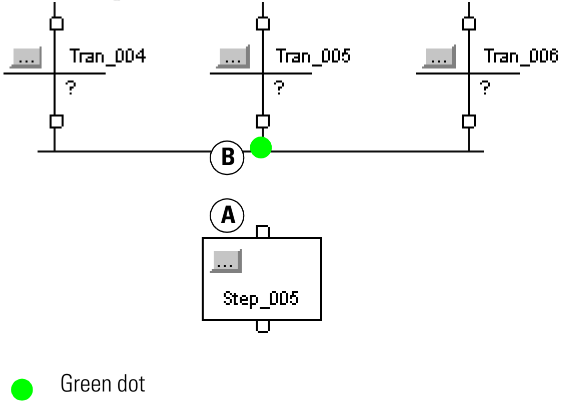
1. Select the last transition of each path in the branch. To select the transitions, do either of these actions.

   - Click and drag the pointer around the transitions that you want to select.
   - Click the first transition. Then press and hold [Shift] and click 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.



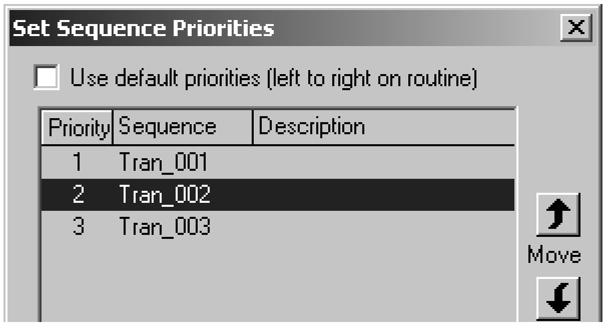
4. To wire the selection branch to the step, click the top pin of the step Ⓐ and then click the horizontal line of the branch Ⓑ. A green dot shows a valid connection point.

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.

1. Right-click the horizontal line that starts the branch and select Set Sequence Priorities.

2. Clear (uncheck) the Use default priorities check box and select a transition.



Set Sequence Priorities

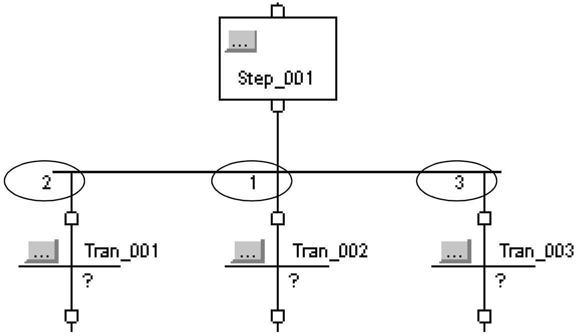
☐ Use default priorities (left to right on routine)

Priority	Sequence	Description
1	Tran_001	
2	Tran_002	
3	Tran_003	

Use the Move buttons to raise or lower the priority of the transition.

3. When all the transitions have the desired priority, click OK.

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



Connect a Wire to the Step

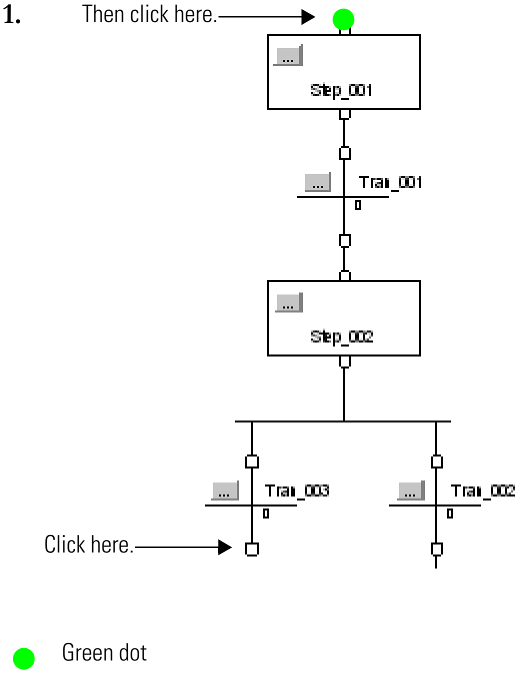
1. Click the lower pin of the transition that signals the jump. Then click the top pin of the step to which you want to go. A green dot shows a valid connection point.

   Typically, the resulting connection orients itself along the center of the flowchart and is hard to see.

2. 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.

This example shows how to go to Step_001 from Tran_003.

1.

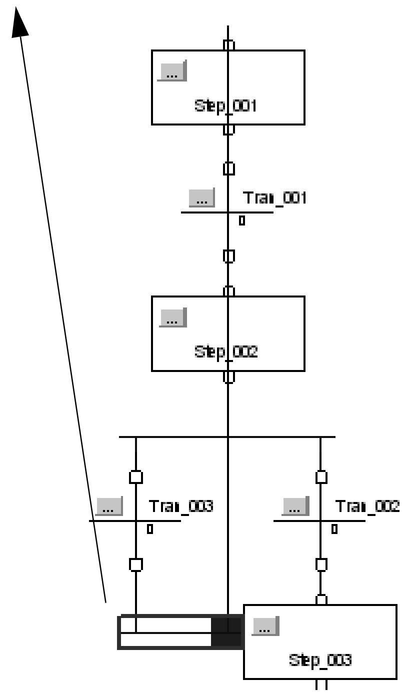

Step_001
Tran_001
Step_002
Tran_003
Tran_002

Then click here.

Click here.

● Green dot

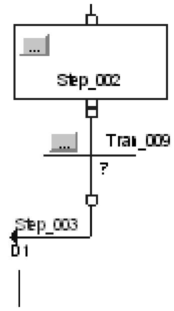
2.


Step_001
Tran_001
Step_002
Tran_003
Tran_002
Step_003

Drag the horizontal bar here.

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.

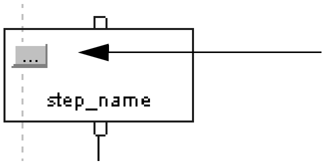

Step_002
Tran_009
?
Step_003
D1

Location to which the wire goes

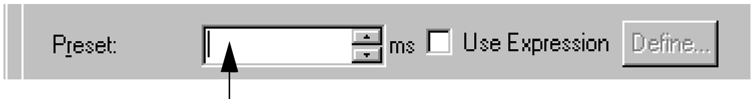
To see the SFC element to which the wire goes, click the grid location on the wire.

Configure a Step

Assign the Preset Time for a Step



1. Click the ... button of the step.


Preset:
ms
☐ Use Expression
Define...

2. On the General tab, type the time for the step, in milliseconds.

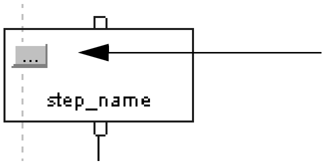
3. Click OK.

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” on page 67.

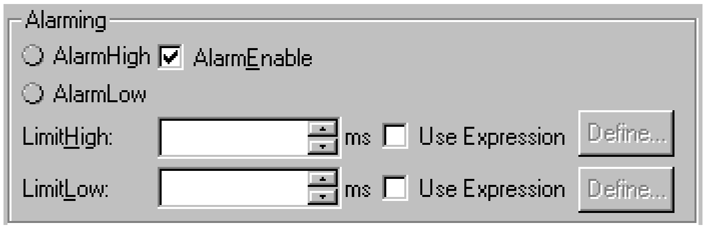
Configure Alarms for a Step

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



1. Click the ... button of the step.

2. Check the AlarmEnable check box.


Alarming
○ AlarmHigh
☑ AlarmEnable
○ AlarmLow
LimitHigh:
ms
☐ Use Expression
Define...
LimitLow:
ms
☐ Use Expression
Define...

Type the time for the high alarm and low alarm, in milliseconds.

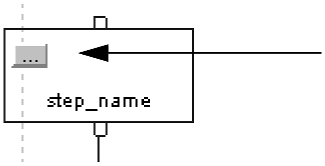
3. Click OK.

Use an Expression to Calculate a Time

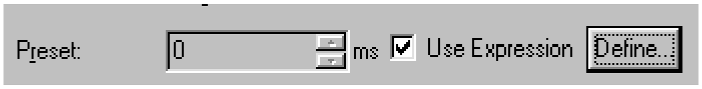
To calculate a time based on tags in your project, enter the time as a numeric expression. You can use an expression to calculate these values.

- Preset
- LimitHigh
- LimitLow

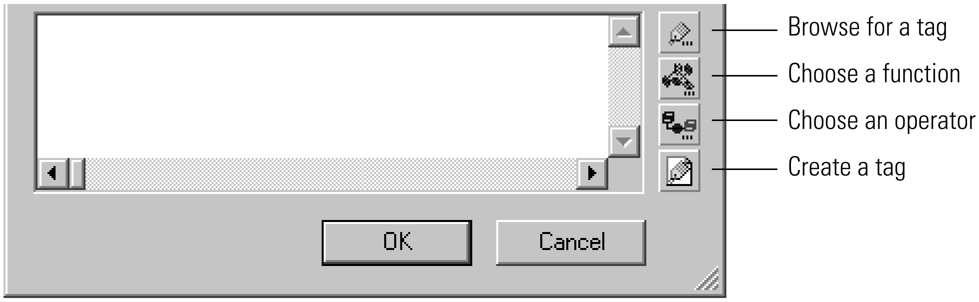
Follows these steps to enter a time as an expression.



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


Preset: 0 ms ☑ Use Expression

3. Click the Define button and enter an expression.

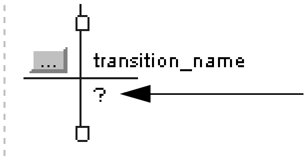


Browse for a tag
Choose a function
Choose an operator
Create a tag

Type a numeric expression that defines the time. Use the buttons alongside the dialog box to help you complete the expression.

4. Click OK.
5. To close the Step Properties dialog box, click OK.

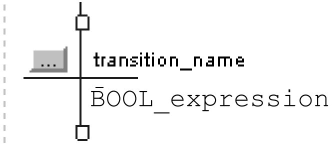
Program a Transition


...
transition_name
?

Enter a BOOL Expression

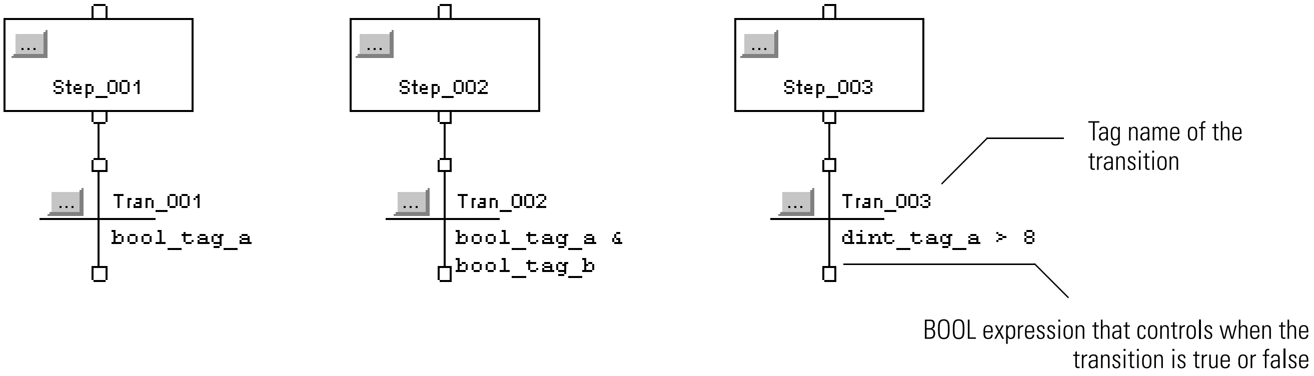
The simplest way to program the transition is to enter the conditions as a BOOL expression in structured text.

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].

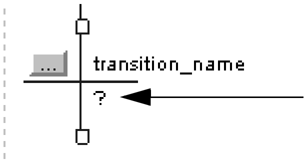

...
transition_name
BOOL_expression

This example shows three transitions that use a BOOL expression.

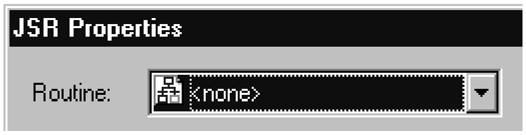
EXAMPLE Enter a BOOL Expression


...
Step_001
...
Tran_001
bool_tag_a
...
Step_002
...
Tran_002
bool_tag_a &
bool_tag_b
...
Step_003
...
Tran_003
dint_tag_a > 8
Tag name of the transition
BOOL expression that controls when the transition is true or false

Call a Subroutine


...
transition_name
?

1. Right-click the transition and select Set JSR.
2. Select the routine that contains the logic for the transition.

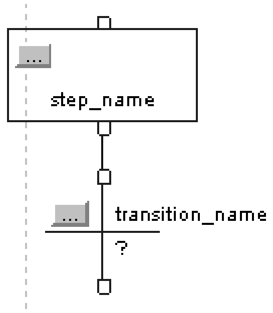


JSR Properties

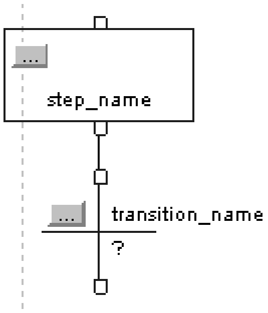
Routine: <none>

3. Click OK.

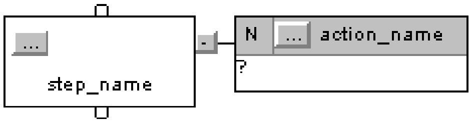
`JSR(routine_name)`


...
step_name
...
transition_name
?

Add an Action


...
step_name
...
transition_name
?

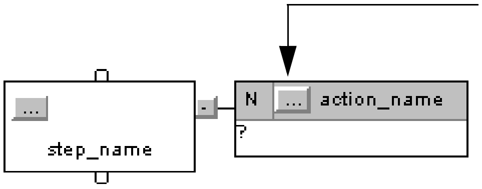
To add an action to a step, right-click the step in which the action executes and select Add Action.


...
step_name
-
N
...
action_name
?

Configure an Action

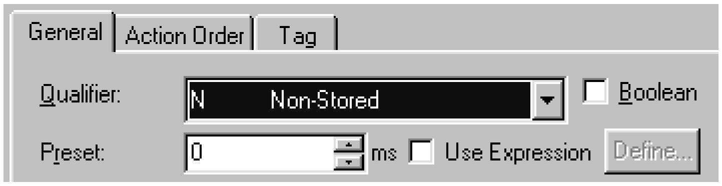
Change the Qualifier of an Action

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.


...
step_name
-
N
...
action_name
?

1. Click the ... button of the action.

2. On the General tab, select the qualifier for the action.


General
Action Order
Tag
Define...

Qualifier: N Non-Stored

Preset: 0 ms

☐ Boolean

☐ Use Expression

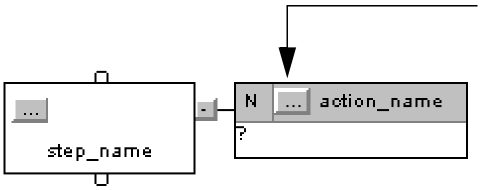
If you chose a timed qualifier, type the time limit or delay for the action, in milliseconds. These are the timed qualifiers.

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

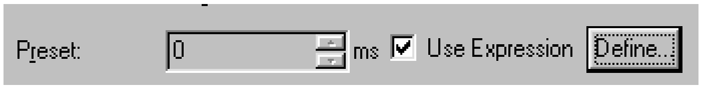
3. Click OK.

Calculate a Preset Time at Runtime

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

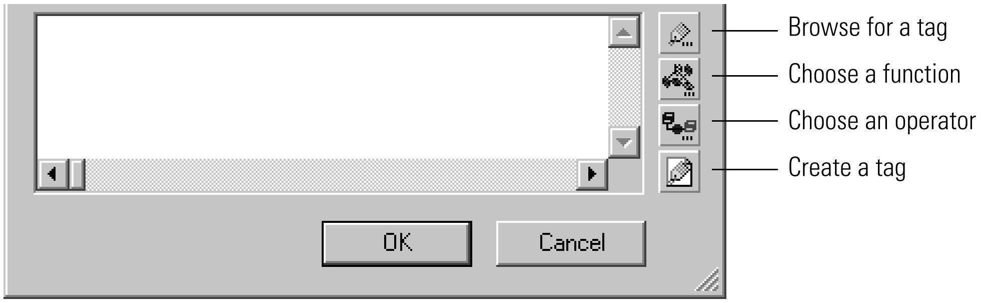

...
step_name
N
...
action_name
?

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



Preset: 0 ms ☑ Use Expression

3. Click the Define button.


Browse for a tag
Choose a function
Choose an operator
Create a tag
OK
Cancel

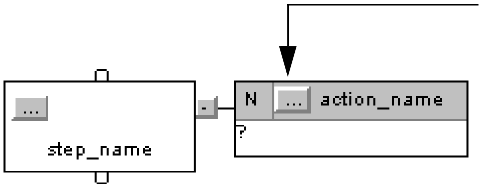
Type a numeric expression that defines the preset time. Use the buttons alongside the dialog box to help you complete the expression.

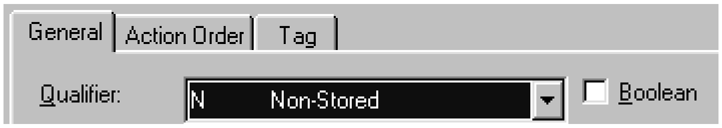
4. Click OK.
5. To close the Action Properties dialog box, click OK.

Mark an Action as a Boolean Action

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

1. Click the ... button of the action.


...
step_name
N
...
action_name
?


General
Action Order
Tag

Qualifier: N Non-Stored

☐ Boolean

2. Check the Boolean check box.

3. Click OK.

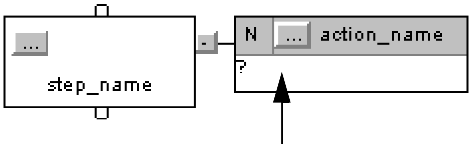
Program an Action

Completes these steps to program an action.

- Enter Structured Text
- Call a Subroutine

Enter Structured Text

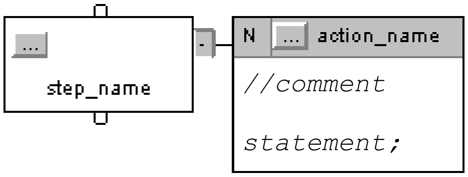
The easiest way to program an action is to write the logic as structured text within the body of the action. When the action turns on, the controller executes the structured text.


...
step_name
N
...
action_name
?

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

2. Type the required structured text.

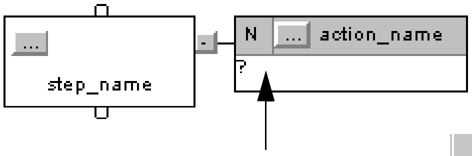
3. To close the text entry window, press [Ctrl] + [Enter].


...
step_name
N
...
action_name

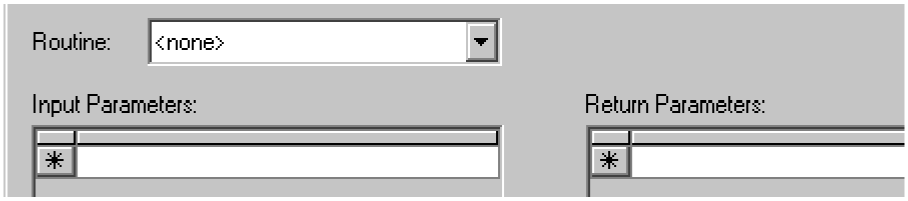
```
//comment

statement;
```

Call a Subroutine

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



1. In the SFC, right-click the text entry area of the action and select Set JSR.



Routine: <none>

Input Parameters:

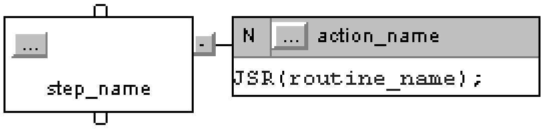
Return Parameters:

2. Choose the routine that you want to call.

3. To pass a parameter to the routine, click an empty Input Parameters text box. Then use the down arrow to select the tag that contains the parameter.

4. To receive a parameter from the routine, click an empty Return Parameters text box. Then use the down arrow to select the tag in which to store the parameter from the routine.

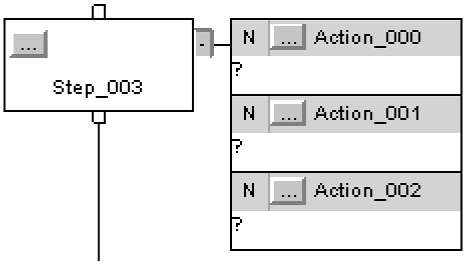
5. Click OK.



```
JSR(routine_name);
```

Assign the Execution Order of Actions

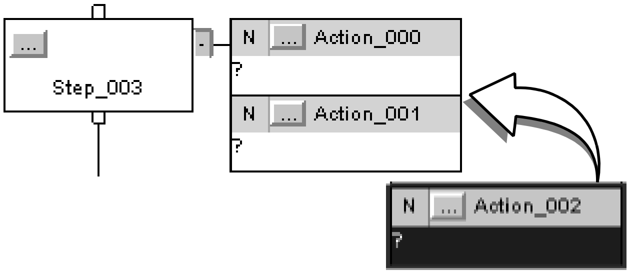
Actions execute in the order in which they appear.



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 desired location in the sequence. A green bar shows a valid placement location.



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
Step	→	Add a Text Box -or- Add a Tag Description
Transition	Download the documentation to the controller	Add Structured Text Comments
	Have the option of showing or hiding the documentation	Add a Text Box -or- Add a Tag Description
	Position the documentation anywhere in the SFC	
Action	Download the documentation to the controller	Add Structured Text Comments
Stop	→	Add a Text Box -or- Add a Tag Description
Other element (such as a selection branch)	→	

Language Switching

With RSLogix 5000 software, version 17, you have the option to display project documentation, such as tag descriptions and rung comments for any supported localized language. You can store project documentation for multiple languages in a single project file rather than in language-specific project files. You define all the localized languages that the project will support and set the current, default, and optional custom localized language. The software uses the default language if the current language's content is blank for a particular component of the project. However, you can use a custom language to tailor documentation to a specific type of project file user.Enter the localized descriptions in your RSLogix 5000 project, either when programming in that language or by using the import/export utility to translate the documentation off-line and then import it back into the project. Once you enable language switching in RSLogix 5000 software, you can dynamically switch between languages as you use the software.

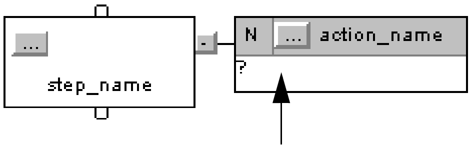
Project documentation that supports multiple translations within a project includes these logic elements.

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

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 ST section of an action are downloaded into controller memory and are available for upload.



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

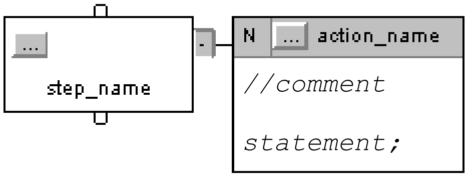
2. Type the comments.

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	
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].

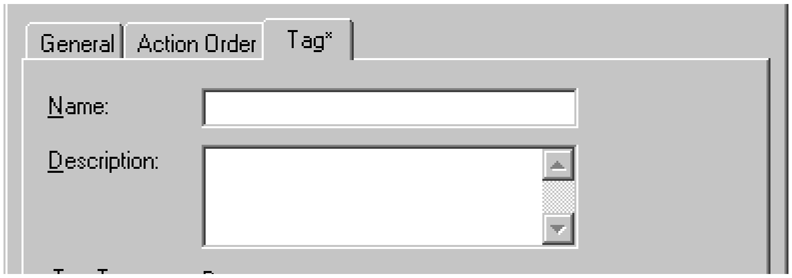


```
//comment
statement;
```

Add a Tag Description

1. Click the … button of the element.

2. Click the Tag tab and type the description for the element.



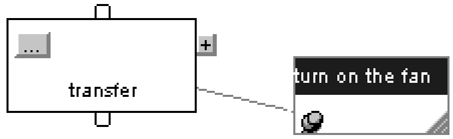
3. Click OK.

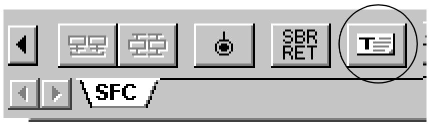
4. Drag the description box to the desired location on the SFC.

Add a Text Box

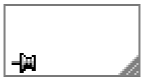
A text box lets you add notes that clarify the function of an SFC element (step, transition, or stop. Text boxes are only stored in the offline, ACD project file. Text boxes are not downloaded into controller memory.

Or use a text box to capture information that you will use later on.



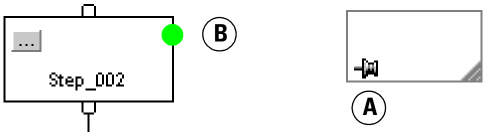


1. Click

   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.



● Green dot

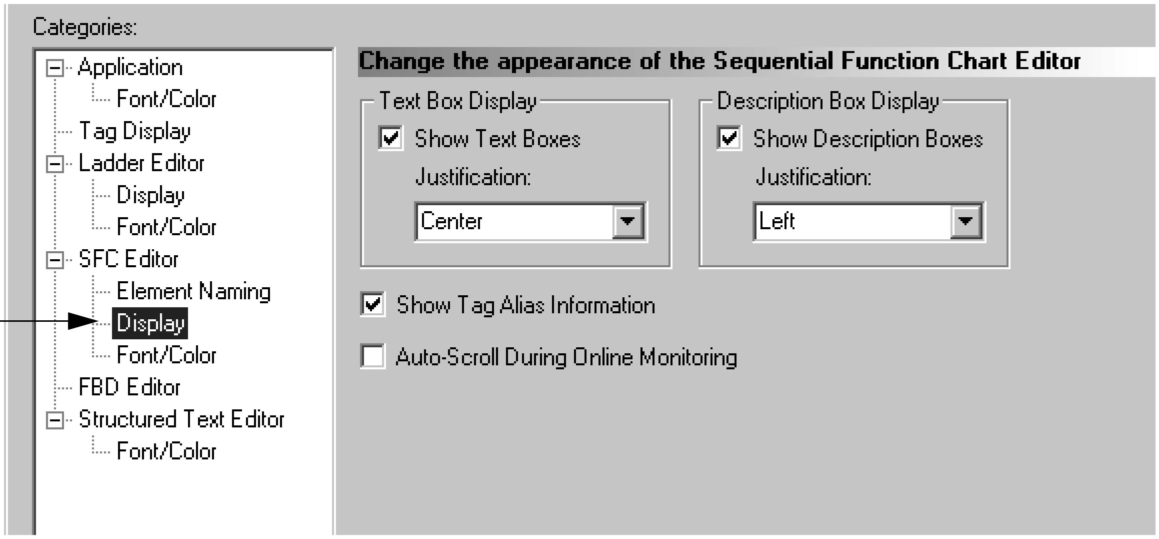
5. Click the pin symbol in the text box and then click the SFC element to which you want to attach the text box. A green dot shows a valid connection point.

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.



Categories:

- Application
  - Font/Color
- Tag Display
- Ladder Editor
  - Display
  - Font/Color
- SFC Editor
  - Element Naming
  - Display
  - Font/Color
- FBD Editor
- Structured Text Editor
  - Font/Color

Change the appearance of the Sequential Function Chart Editor

Text Box Display

☑ Show Text Boxes

Justification: Center

Description Box Display

☑ Show Description Boxes

Justification: Left

☑ Show Tag Alias Information

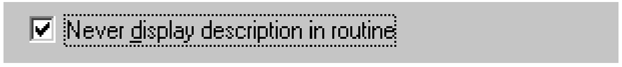
☐ Auto-Scroll During Online Monitoring

Hide an Individual Tag Description

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

1. Click the ... button of the element whose description you want to hide.

2. Check the Never display description in routine check box.



☑ Never display description in routine

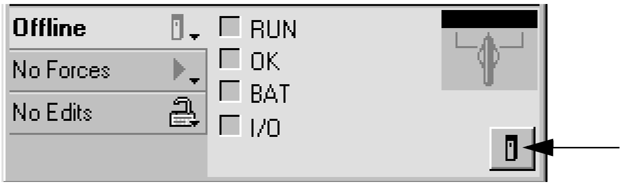
3. Click OK.

Configure the Execution of the SFC

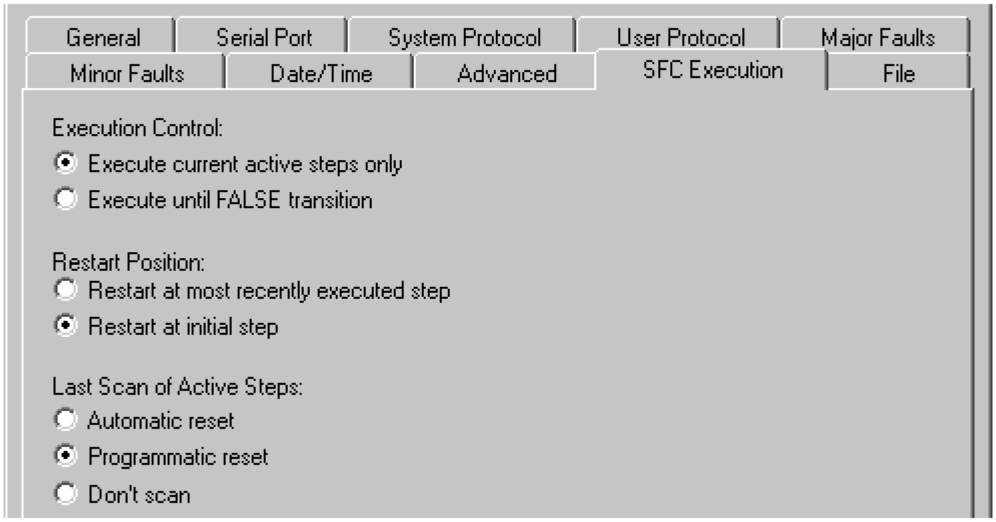
The SFC Execution tab of the controller properties lets you configure these decisions.

- 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.


Offline
No Forces
No Edits
☐ RUN
☐ OK
☐ BAT
☐ I/O

2. Select the SFC Execution tab.


General
Serial Port
System Protocol
User Protocol
Major Faults
Minor Faults
Date/Time
Advanced
SFC Execution
File

Execution Control:
- ● Execute current active steps only
- ○ Execute until FALSE transition

Restart Position:
- ○ Restart at most recently executed step
- ● Restart at initial step

Last Scan of Active Steps:
- ○ Automatic reset
- ● Programmatic reset
- ○ Don't scan

3. Choose one of the following.
   - whether or not to return to the OS/JSR if a transition is true.
   - where to restart the SFC.

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

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

     The restart position isn't 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.

4. Click OK.

Verify the Routine

As you program your routine, periodically verify your work.

1. In the top-most toolbar of the RSLogix 5000 window, click
2. Follows these steps if any errors are listed at the bottom of the window.
   a. To go to the first error or warning, press [F4].
   b. Correct the error according to the description in the Results window.
   c. Go to step 1.
3. To close the Results window, press [Alt] + [1].

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

- force transitions
- step through the SFC

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 untest edits to coincide with the SFC executing 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 desired step.

Maintain Active SFC Step

As of firmware revision 18, these 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

Notes:

Chapter 3

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

ATTENTION ⚠

Forcing can cause unexpected machine motion that could injure personnel. Before you use a force, determine how the force will effect 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.

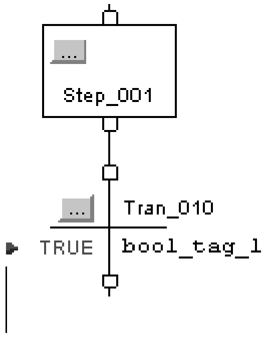
Enable Forces

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

- 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 ▶ appears next to the forced element.



Forces are in effect (enabled)

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

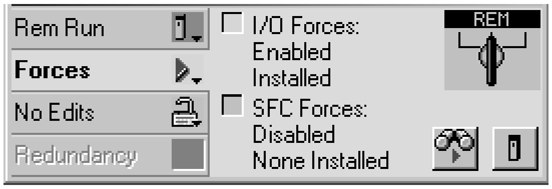
Changes to forces can cause unexpected machine motion that could injure personnel. Before you disable or remove forces, determine how the change will effect your machine or process and keep personnel away from the machine area.

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

☐ I/O Forces:
Enabled
Installed

☐ SFC Forces:
Disabled
None Installed

This	Means
Enabled	• If the project contains any forces of this type, they 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.

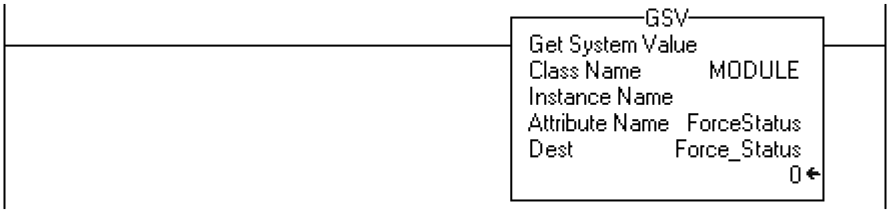
IMPORTANT The FORCE LED shows only the status of I/O forces. It does not show that status of SFC forces.

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

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

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



GSV
Get System Value
Class Name MODULE
Instance Name
Attribute Name ForceStatus
Dest Force_Status
0

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

When to Use Step Through

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.

Step Through a Transition or a Force of a Path

Follows these steps 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 choose 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

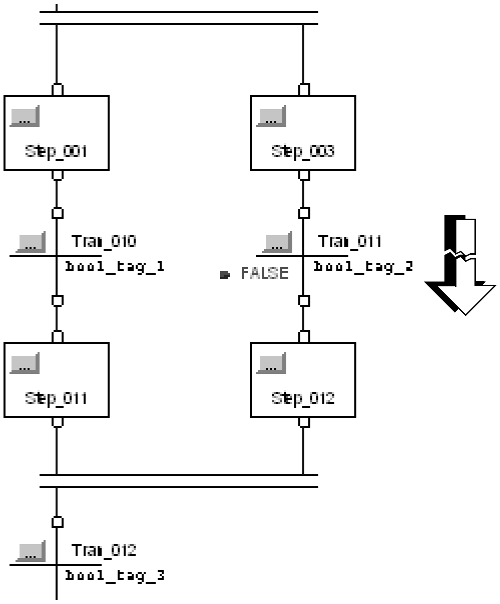
Force a Transition

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 true

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.


Step_001
Step_003
Tran_010
bool_tag_1
FALSE
Tran_011
bool_tag_2
Step_011
Step_012
Tran_012
bool_tag_3

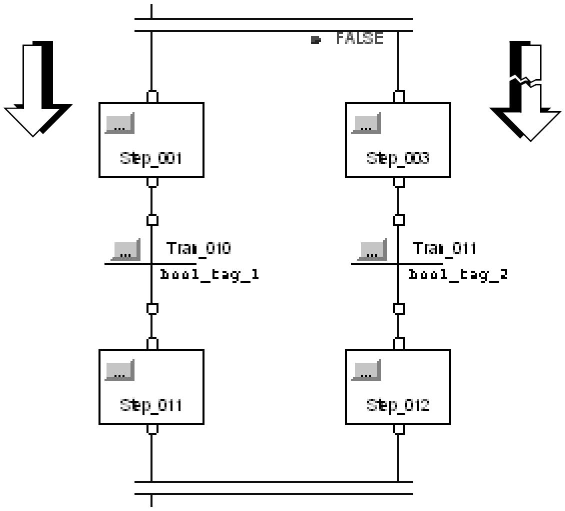
For example, to exit this branch, the SFC must be able to complete these actions.

- Execute Step_011 at least once
- Get past Tran_011 and execute Step_012 at least once
- Determine that Tran_012 is true

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.

This path executes.


FALSE
Step_001
Step_003
Tran_010
bool_tag_1
Tran_011
bool_tag_2
Step_011
Step_012

This path does not execute.

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.

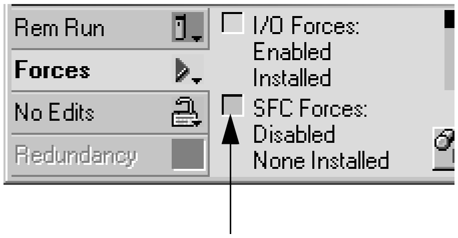
Add an SFC Force

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 effect 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.



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 choose 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

ATTENTION

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

Disable All SFC Forces

From the Logic menu, choose SFC Forcing > Disable All SFC Forces. Then choose Yes to confirm.

Remove All SFC Forces

From the Logic menu, choose SFC Forcing > Remove All SFC Forces. Then choose Yes to confirm.

Rockwell Automation Support

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

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

Installation Assistance

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

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

New Product Satisfaction Return

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

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

Documentation Feedback

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

www.rockwellautomation.com

Power, Control and Information Solutions Headquarters

Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
Europe/Middle East/Africa: Rockwell Automation, Vorstlaan/Boulevard du Souverain 36, 1170 Brussels, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640
Asia Pacific: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Publication 1756-PM006D-EN-P - August 2010
Supersedes Publication 1756-PM006C-EN-P - October 2009

Copyright © 2010 Rockwell Automation, Inc. All rights reserved. Printed in the U.S.A.