Logix5000 Controllers Sequential Function Charts

Programming Manual

Catalog Numbers 1756 ControlLogix, 1769 CompactLogix, 1789 SoftLogix, 1794 FlexLogix, PowerFlex 700S with DriveLogix
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://literature.rockwellautomation.com) describes some important differences between solid state equipment and hard-wired electromechanical devices. Because of this difference, and also because of the wide variety of uses for solid state equipment, all persons responsible for applying this equipment must satisfy themselves that each intended application of this equipment is acceptable.

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

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

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

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

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

<table>
<thead>
<tr>
<th>WARNING</th>
<th>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.</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMPORTANT</td>
<td>Identifies information that is critical for successful application and understanding of the product.</td>
</tr>
<tr>
<td>ATTENTION</td>
<td>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</td>
</tr>
<tr>
<td>SHOCK HAZARD</td>
<td>Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present.</td>
</tr>
<tr>
<td>BURN HAZARD</td>
<td>Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.</td>
</tr>
</tbody>
</table>

Allen-Bradley, Rockwell Automation, and TechConnect are trademarks of Rockwell Automation, Inc.

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

Purpose of this Manual ........................................... 7
How to Use this Manual ....................................... 7

Chapter 1

Introduction ....................................................... 9
What is a Sequential Function Chart? ....................... 10
Define the Tasks ................................................. 13
Choose How to Execute the SFC ............................ 14
Define the Steps of the Process .............................. 14
  Step Guidelines .............................................. 15
  SFC_STEP Structure ........................................ 16
Organize the Steps .............................................. 19
  Sequence .................................................. 21
  Selection Branch ......................................... 21
  Simultaneous Branch ..................................... 22
  Wire to a Previous Step .................................. 22
Add Actions for Each Step .................................. 23
  How Do You Want to Use the Action? ................. 23
  Use a Non-Boolean Action ................................ 24
  Use a Boolean Action .................................... 25
  SFC_ACTION Structure .................................... 26
Describe Each Action in Pseudocode ....................... 27
Choose a Qualifier for an Action ........................... 27
Define the Transition Conditions ......................... 28
  Transition Tag ............................................ 31
  How Do You Want to Program the Transition? ...... 31
  Use a BOOL Expression .................................. 31
  Call a Subroutine ....................................... 32
Transition After a Specified Time ......................... 33
Turn Off a Device at the End of a Step .................... 36
  Choose a Last Scan Option .............................. 36
  Use the Don’t Scan Option ............................... 38
  Use the Programmatic Reset Option .................. 39
  Use the Automatic Reset Option ....................... 41
Keep Something On From Step-to-Step ..................... 42
  How Do You Want to Control the Device? .......... 42
  Use a Simultaneous Branch ............................. 43
  Store and Reset an Action ............................. 43
  Use One Large Step ...................................... 45
End the SFC ................................................... 46
  Use a Stop Element ..................................... 46
  Restart (Reset) the SFC ................................ 47
  SFC_STOP Structure ...................................... 48
Nest an SFC .................................................... 49
  Pass Parameters .......................................... 50
Configure When to Return to the OS/JSR .................. 50
Pause or Reset an SFC ...................................... 51
# Programming a Sequential Function Chart

## Chapter 2

**Introduction** ................................................................. 57

- Add and Manually Connect Elements .................................. 57
- Add and Automatically Connect Elements ............................. 58
- Drag and Drop Elements .................................................. 58

**Create a Simultaneous Branch** ......................................... 58

- Start a Simultaneous Branch .......................................... 58
- End a Simultaneous Branch ........................................... 59

**Create a Selection Branch** .................................................. 60

- Start a Selection Branch ............................................... 60
- End a Selection Branch ................................................. 61

**Set the Priorities of a Selection Branch** ......................... 61

**Connect a Wire to the Step** ............................................... 62

**Hide a Wire** ................................................................. 63

**Configure a Step** .............................................................. 64

- Assign the Preset Time for a Step .................................... 64
- Configure Alarms for a Step ......................................... 64
- Use an Expression to Calculate a Time ............................ 65

**Program a Transition** ......................................................... 66

- Enter a BOOL Expression ............................................... 66
- Call a Subroutine ......................................................... 66

**Add an Action** ................................................................. 67

**Configure an Action** .......................................................... 67

- Change the Qualifier of an Action ................................. 67
- Calculate a Preset Time at Runtime ............................... 68
- Mark an Action as a Boolean Action .............................. 69

**Program an Action** ............................................................. 69

- Enter Structured Text .................................................... 69
- Call a Subroutine .......................................................... 70

**Assign the Execution Order of Actions** ......................... 71

**Document an SFC** ............................................................. 71

- Add Structured Text Comments ...................................... 72
- Add a Tag Description .................................................... 73
- Add a Text Box ............................................................... 73
- Show or Hide Text Boxes or Tag Descriptions ...................... 74
- Hide an Individual Tag Description .................................. 75
- Language Switching ......................................................... 75

**Configure the Execution of the SFC** .......................... 76

**Verify the Routine** .......................................................... 77

**Edit an SFC Online** ............................................................ 77

## Chapter 3

**Introduction** ................................................................. 79
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precautions</td>
<td>79</td>
</tr>
<tr>
<td>Enable Forces</td>
<td>79</td>
</tr>
<tr>
<td>Disable or Remove a Force</td>
<td>80</td>
</tr>
<tr>
<td>Check Force Status</td>
<td>80</td>
</tr>
<tr>
<td>FORCE LED</td>
<td>81</td>
</tr>
<tr>
<td>GSV Instruction</td>
<td>82</td>
</tr>
<tr>
<td>When to Use Step Through</td>
<td>82</td>
</tr>
<tr>
<td>Step Through a Transition or a Force of a Path</td>
<td>82</td>
</tr>
<tr>
<td>When to Use an SFC Force</td>
<td>83</td>
</tr>
<tr>
<td>Force a Transition</td>
<td>83</td>
</tr>
<tr>
<td>Force a Simultaneous Path</td>
<td>84</td>
</tr>
<tr>
<td>Add an SFC Force</td>
<td>85</td>
</tr>
<tr>
<td>Remove or Disable Forces</td>
<td>86</td>
</tr>
<tr>
<td>Disable All SFC Forces</td>
<td>86</td>
</tr>
<tr>
<td>Remove All SFC Forces</td>
<td>86</td>
</tr>
</tbody>
</table>
Preface

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 Logix 5000 Controllers Common Procedures Programming Manual, publication 1756-PM001.

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

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

How to Use this Manual

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

<table>
<thead>
<tr>
<th>Text that is</th>
<th>Identifies</th>
<th>For example</th>
<th>Means</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Italic</em></td>
<td>the actual name of an item that you see on your screen or in an example</td>
<td>Right-click User-Defined …</td>
<td>Right-click the item that is named User-Defined.</td>
</tr>
<tr>
<td>courier</td>
<td>information that you must supply based on your application (a variable)</td>
<td>Right-click name_of_program …</td>
<td>You must identify the specific program in your application. Typically, it is a name or variable that you have defined.</td>
</tr>
<tr>
<td>enclosed in brackets</td>
<td>a keyboard key</td>
<td>Press [Enter].</td>
<td>Press the Enter key.</td>
</tr>
</tbody>
</table>
Notes:
Chapter 1

Designing 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. Use an SFC to:

- 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. In addition, RSLogix 5000 software lets you:
  - 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:

**SFC Example**

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

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.

A JSR instruction calls a subroutine.

Show or hide an action.

(continued on next page)
A selection branch chooses between different execution paths.

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 on the previous page.
To design a sequential function chart:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define the Tasks</td>
<td>13</td>
</tr>
<tr>
<td>Choose How to Execute the SFC</td>
<td>14</td>
</tr>
<tr>
<td>Define the Steps of the Process</td>
<td>14</td>
</tr>
<tr>
<td>Organize the Steps</td>
<td>19</td>
</tr>
<tr>
<td>Add Actions for Each Step</td>
<td>23</td>
</tr>
<tr>
<td>Describe Each Action in Pseudocode</td>
<td>27</td>
</tr>
<tr>
<td>Choose a Qualifier for an Action</td>
<td>27</td>
</tr>
<tr>
<td>Define the Transition Conditions</td>
<td>28</td>
</tr>
<tr>
<td>Transition After a Specified Time</td>
<td>33</td>
</tr>
<tr>
<td>Turn Off a Device at the End of a Step</td>
<td>36</td>
</tr>
<tr>
<td>Keep Something On From Step-to-Step</td>
<td>42</td>
</tr>
<tr>
<td>End the SFC</td>
<td>46</td>
</tr>
<tr>
<td>Nest an SFC</td>
<td>49</td>
</tr>
<tr>
<td>Configure When to Return to the OS/JSR</td>
<td>50</td>
</tr>
<tr>
<td>Pause or Reset an SFC</td>
<td>51</td>
</tr>
<tr>
<td>Execution Diagrams</td>
<td>51</td>
</tr>
</tbody>
</table>
Chapter 1  Designing a Sequential Function Chart

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:

<table>
<thead>
<tr>
<th>These functions</th>
<th>Go here</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure and regulate devices</td>
<td>Periodic task</td>
</tr>
<tr>
<td>Command a device to a specific state</td>
<td>SFC in the continuous task</td>
</tr>
<tr>
<td>Sequence the execution of your process</td>
<td></td>
</tr>
</tbody>
</table>

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

- 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.

<table>
<thead>
<tr>
<th>If</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>The SFC is the only routine in the program.</td>
<td>Configure the SFC as the main routine for the program.</td>
</tr>
<tr>
<td>The SFC calls all the other routines of the program.</td>
<td></td>
</tr>
<tr>
<td>The program requires other routines to execute independent of the SFC.</td>
<td>1. Configure another routine as the main routine for the program.</td>
</tr>
<tr>
<td>The SFC uses boolean actions.</td>
<td>2. Use the main routine to call the SFC as a subroutine.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Step</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIZ</td>
<td>Do this…</td>
</tr>
<tr>
<td></td>
<td>…and do this</td>
</tr>
</tbody>
</table>

A step executes continuously until a transition tells the SFC to go to the next step.

<table>
<thead>
<tr>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>• If true, go to the next step.</td>
</tr>
</tbody>
</table>

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

As you define the steps of your process, follow these guidelines:

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

<table>
<thead>
<tr>
<th>First pass</th>
<th>Second pass</th>
<th>Third pass</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paint</td>
<td>Transfer_In</td>
<td>Transfer_In</td>
</tr>
<tr>
<td></td>
<td>Paint</td>
<td>Paint_Flow</td>
</tr>
<tr>
<td></td>
<td>Clean</td>
<td>Air_Flow</td>
</tr>
<tr>
<td></td>
<td>Transfr_Out</td>
<td>Elec_Charg</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Clean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transfr_Out</td>
</tr>
</tbody>
</table>

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

The controller executes the initial step:

- 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 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. For example:

<table>
<thead>
<tr>
<th>This organization of steps</th>
<th>Is</th>
</tr>
</thead>
<tbody>
<tr>
<td>produce_solution</td>
<td>Probably too large</td>
</tr>
<tr>
<td>set_mode, close_outlet, set_temperature, open_inlet_a, close_inlet_a, set_timer, reset_temperature, open_outlet, reset_mode</td>
<td>Probably too small</td>
</tr>
<tr>
<td>preset_tank, add_ingredient_a, cook, drain</td>
<td>Probably about right</td>
</tr>
</tbody>
</table>

### 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:

<table>
<thead>
<tr>
<th>If you want to</th>
<th>Then check or set this member</th>
<th>Data type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine how long a step has been active (milliseconds)</td>
<td>T</td>
<td>DINT</td>
<td>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.</td>
</tr>
<tr>
<td>Flag when the step has been active for a specific length of time (milliseconds)</td>
<td>PRE</td>
<td>DINT</td>
<td>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.</td>
</tr>
<tr>
<td>Flag if a step did not execute long enough</td>
<td>LimitLow</td>
<td>DINT</td>
<td>Enter the time in the LimitLow member (milliseconds).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• If the step goes inactive before the Timer (T) reaches the LimitLow value, the AlarmLow bit turns on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The AlarmLow bit stays on until you reset it.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 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.</td>
</tr>
<tr>
<td></td>
<td>AlarmEn</td>
<td>BOOL</td>
<td>To use the alarm bits, turn on (check) the AlarmEnable (AlarmEn) bit.</td>
</tr>
<tr>
<td></td>
<td>AlarmLow</td>
<td>BOOL</td>
<td>If the step goes inactive before the Timer (T) reaches the LimitLow value, the AlarmLow bit turns on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The bit stays on until you reset it.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• To use this alarm function, turn on (check) the AlarmEnable (AlarmEn) bit.</td>
</tr>
</tbody>
</table>
### Chapter 1  
Designing a Sequential Function Chart

<table>
<thead>
<tr>
<th>If you want to</th>
<th>Then check or set this member</th>
<th>Data type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flag if a step is executing too long</td>
<td>LimitHigh</td>
<td>DINT</td>
<td>Enter the time in the LimitHigh member (milliseconds).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• If the Timer (T) reaches the LimitHigh value, the AlarmHigh bit turns on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The AlarmHigh bit stays on until you reset it.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• To use this alarm function, turn on (check) the AlarmEnable (AlarmEn) bit.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>As an option, enter a numeric expression that calculates the time at runtime.</td>
</tr>
<tr>
<td></td>
<td>AlarmEn</td>
<td>BOOL</td>
<td>To use the alarm bits, turn on (check) the AlarmEnable (AlarmEn) bit.</td>
</tr>
<tr>
<td></td>
<td>AlarmHigh</td>
<td>BOOL</td>
<td>If the Timer (T) reaches the LimitHigh value, the AlarmHigh bit turns on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The bit stays on until you reset it.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• To use this alarm function, turn on (check) the AlarmEnable (AlarmEn) bit.</td>
</tr>
<tr>
<td>Do something while the step is active (including first and last scan)</td>
<td>X</td>
<td>BOOL</td>
<td>The X bit is on the entire time the step is active (executing).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Typically, we recommend that you use an action with a Non-Stored qualifier to accomplish this.</td>
</tr>
<tr>
<td>Do something one time when the step becomes active</td>
<td>FS(1)</td>
<td>BOOL</td>
<td>The FS bit is on during the first scan of the step.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Typically, we recommend that you use an action with a Pulse (Rising Edge) qualifier to accomplish this.</td>
</tr>
<tr>
<td>Do something while the step is active, except on the first and last scan</td>
<td>SA</td>
<td>BOOL</td>
<td>The SA bit is on when the step is active except during the first and last scan of the step.</td>
</tr>
<tr>
<td>Do something one time on the last scan of the step</td>
<td>LS(1)</td>
<td>BOOL</td>
<td>The LS bit is on during the last scan of the step.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Use this bit only if you do the following: On the Controller Properties dialog box, SFC Execution tab, set the Last Scan of Active Step to Don't Scan or Programmatic reset.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Typically, we recommend that you use an action with a Pulse (Falling Edge) qualifier to accomplish this.</td>
</tr>
<tr>
<td>Determine the target of an SFC Reset (SFR) instruction</td>
<td>Reset</td>
<td>BOOL</td>
<td>An SFC Reset (SFR) instruction resets the SFC to a step or stop that the instruction specifies.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The Reset bit indicates to which step or stop the SFC will go to begin executing again.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Once the SFC executes, the Reset bit clears.</td>
</tr>
<tr>
<td>Determine the maximum time that a step has been active during any of its executions</td>
<td>Tmax</td>
<td>DINT</td>
<td>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.</td>
</tr>
<tr>
<td>Determine if the Timer (T) value rolls over to a negative value</td>
<td>OV</td>
<td>BOOL</td>
<td>Use this for diagnostic purposes.</td>
</tr>
</tbody>
</table>
This diagram shows the relationship of the X, FS, SA, and LS bits.

<table>
<thead>
<tr>
<th>If you want to</th>
<th>Then check or set this member</th>
<th>Data type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine how many times a step has become active</td>
<td>Count</td>
<td>DINT</td>
<td>This is not a count of scans of the step.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The count increments each time the step becomes active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• It increments again only after the step goes inactive and then</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>active again.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The count resets only if you configure the SFC to restart at the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>initial step. With that configuration, it resets when the controller</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>changes from program mode to run mode.</td>
</tr>
<tr>
<td>Use one tag for the various status bits of this step</td>
<td>Status</td>
<td>DINT</td>
<td>For this member</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AlarmHigh</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AlarmLow</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AlarmEn</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SA</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Footnote: 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).
Organize the Steps

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

<table>
<thead>
<tr>
<th>To</th>
<th>Use this structure</th>
<th>With these considerations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute 1 or more steps in sequence:</td>
<td>Sequence</td>
<td>The SFC checks the transition at the end of the step:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If true, the SFC goes to the next step.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If false, the SFC repeats the step.</td>
</tr>
<tr>
<td>• Choose between alternative steps or groups of steps depending on logic conditions</td>
<td>Selection Branch</td>
<td>• It is OK for a path to have no steps and only a transition. This lets the SFC skip the selection branch.</td>
</tr>
<tr>
<td>• Execute a step or steps or skip the step or steps depending on logic conditions</td>
<td></td>
<td>• By default, the SFC checks from left to right the transitions that start each path. It takes the first true path.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• If no transitions are true, the SFC repeats the previous step.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• RSLogix 5000 software lets you change the order in which the SFC checks the transitions.</td>
</tr>
<tr>
<td>Execute 2 or more steps at the same time. All paths must finish before continuing the SFC</td>
<td>Simultaneous Branch</td>
<td>• A single transition ends the branch.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The SFC checks the ending transition after the last step in each path has executed at least once. If the transition is false, the SFC repeats the previous step.</td>
</tr>
<tr>
<td>Loop back to a previous step</td>
<td>Wire to a Previous Step</td>
<td>• Connect the wire to the step or simultaneous branch to which you want to go.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Do not wire into, out of, or between a simultaneous branch.</td>
</tr>
</tbody>
</table>
Here are some examples of SFC structures for different situations:

<table>
<thead>
<tr>
<th>Example situation</th>
<th>Example solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Station 45 and 46 of an assembly line work on parts simultaneously. When both stations are done, the parts move down 1 station.</td>
<td>Simultaneous Branch</td>
</tr>
<tr>
<td></td>
<td><img src="image1" alt="Simultaneous Branch Diagram" /></td>
</tr>
<tr>
<td>Depending on the build code, a station either drills or polishes.</td>
<td>Selection Branch</td>
</tr>
<tr>
<td></td>
<td><img src="image2" alt="Selection Branch Diagram" /></td>
</tr>
<tr>
<td>To simplify my programming, I want to separate communications and block transfers from other control logic. All occur at the same time.</td>
<td>Simultaneous Branch</td>
</tr>
<tr>
<td></td>
<td><img src="image3" alt="Simultaneous Branch Diagram" /></td>
</tr>
<tr>
<td>In a heat treating area, the temperature ramps up at a specific rate, maintains that temperature for a specific duration, and then cools at a specific rate.</td>
<td>Sequence</td>
</tr>
<tr>
<td></td>
<td><img src="image4" alt="Sequence Diagram" /></td>
</tr>
<tr>
<td>At station 12, the machine drills, taps, and bolts a part. The steps occur one after the other.</td>
<td>Sequence</td>
</tr>
<tr>
<td></td>
<td><img src="image5" alt="Sequence Diagram" /></td>
</tr>
<tr>
<td>Step 12 inspects a process for the correct mix of chemicals.</td>
<td>Wire</td>
</tr>
<tr>
<td>• If OK, then continue with the remaining steps.</td>
<td><img src="image6" alt="Wire Diagram" /></td>
</tr>
<tr>
<td>• If not OK, go to the top of the SFC and purge the system.</td>
<td><img src="image7" alt="Wire Diagram" /></td>
</tr>
</tbody>
</table>
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 (i.e., 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).
**Simultaneous Branch**

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

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

**Wire to a Previous Step**

In addition to connecting steps in sequences, simultaneous branches, and selection branches, you can connect a step to a previous point in your SFC. This lets you:

- loop back and repeat steps.
- return to the beginning of the SFC and start over.
For example:

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.

How Do You Want to Use the Action?
There are two types of actions:

<table>
<thead>
<tr>
<th>If you want to</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute structured text directly in the SFC</td>
<td>Use a Non-Boolean Action</td>
</tr>
<tr>
<td>Call a subroutine</td>
<td></td>
</tr>
<tr>
<td>Use the automatic reset option to reset data upon</td>
<td></td>
</tr>
<tr>
<td>leaving a step</td>
<td></td>
</tr>
<tr>
<td>Only set a bit and program other logic to monitor</td>
<td>Use a Boolean Action</td>
</tr>
<tr>
<td>the bit to determine when to execute.</td>
<td></td>
</tr>
</tbody>
</table>
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 36.

To program a non-boolean action, you have these options:

<table>
<thead>
<tr>
<th>If you want to</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Execute your logic without additional routines</td>
<td>Embed structured text.</td>
</tr>
<tr>
<td>• Use structured text assignments, constructs, and instructions</td>
<td>For example:</td>
</tr>
<tr>
<td></td>
<td><img src="example1.png" alt="Structured Text Example" /></td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>• Re-use logic in multiple steps</td>
<td>Call a subroutine.</td>
</tr>
<tr>
<td>• Use another language to program the action, such as ladder logic</td>
<td>For example:</td>
</tr>
<tr>
<td></td>
<td><img src="example2.png" alt="Structured Text Example" /></td>
</tr>
<tr>
<td></td>
<td>When the S_Complete_the_Batch step is active, the S_Open_Outlet action executes. The action calls the Open_Outlet routine.</td>
</tr>
<tr>
<td>• Nest an SFC</td>
<td></td>
</tr>
</tbody>
</table>

Open_Outlet Routine

![Diagram](diagram.png)

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.

For example:

**EXAMPLE**

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

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

You can reuse a boolean action multiple times within the same SFC.
## SFC_ACTION Structure

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

<table>
<thead>
<tr>
<th>If you want to</th>
<th>Then check or set this member</th>
<th>Data type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<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:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>If the action is:</strong>  <strong>Then the Q bit is:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Non-boolean</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.</td>
</tr>
<tr>
<td>Use one tag for the various status bits of this action</td>
<td>Status</td>
<td>DINT</td>
<td>For this member</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td></td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td></td>
<td>31</td>
</tr>
</tbody>
</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:

<table>
<thead>
<tr>
<th>For a comment</th>
<th>Use one of these formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>On a single line</td>
<td>// comment</td>
</tr>
<tr>
<td>That spans more than one line</td>
<td>(<em>start of comment . . . end of comment</em>)</td>
</tr>
<tr>
<td></td>
<td>/<em>start of comment . . . end of comment</em>/</td>
</tr>
</tbody>
</table>

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

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

- S  Stored
- SL  Stored and Time Limited
- DS  Delayed and Stored
- SD  Stored and Time Delayed

### 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.*

![Diagram](attachment:image.png)
Transitions occur in the:

<table>
<thead>
<tr>
<th>For this structure</th>
<th>Make sure that</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence</td>
<td>A transition is between each step.</td>
</tr>
<tr>
<td>Selection branch</td>
<td>Transitions are inside the horizontal lines.</td>
</tr>
<tr>
<td>Simultaneous branch</td>
<td>Transitions are outside the horizontal lines.</td>
</tr>
</tbody>
</table>
Here are two examples of transitions:

**EXAMPLE** You want to:

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.

Solution:

```
Init
  └─ Init_Done ─ compressor_1.Device1State = on (1)
      └─ and
```

**EXAMPLE** You want to:

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.

Solution:

```
Package
  └─ carton_8 ─ package_done = on (1) and carton_size = 8
  └─ carton_16 ─ package_done = on (1) and carton_size = 16
    └─ Pack_8
    └─ Pack_16
```
Transition Tag

Each transition uses a BOOL tag to represent the true or false state of the transition.

<table>
<thead>
<tr>
<th>If the transition is</th>
<th>The value is</th>
<th>And</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>1</td>
<td>The SFC goes to the next step.</td>
</tr>
<tr>
<td>False</td>
<td>0</td>
<td>The SFC continues to execute the current step.</td>
</tr>
</tbody>
</table>

How Do You Want to Program the Transition?

To program the transition, you have these options.

<table>
<thead>
<tr>
<th>If you want to</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter the conditions as an expression in structured text</td>
<td>Use a BOOL Expression</td>
</tr>
<tr>
<td>Enter the conditions as instructions in another routine</td>
<td>Call a Subroutine</td>
</tr>
<tr>
<td>Use the same logic for multiple transitions</td>
<td></td>
</tr>
</tbody>
</table>

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`.

Here are some examples of BOOL expressions.

- `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. 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:

- 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.

**Description**

1. Step becomes active.
   
   X bit turns on.
   
   Timer (T) begins to increment.

2. Timer reaches the Preset (PRE) value of the step.
   
   DN bit turns on.
   
   Timer continues to increment.

3. Step becomes inactive.
   
   X bit turns off.
   
   Timer retains its value.
   
   DN remains on.

4. Step becomes active.
   
   X bit turns on.
   
   Timer clears and then begins to increment.
   
   DN bit turns off.
Action of the low and high alarms for a step:

1. AlarmEn is on. To use the low and high alarms turn this bit on. Turn the bit on via the properties dialog box or the tag for the step.

2. Step becomes active.  
   X bit turns on.  
   Timer (T) begins to increment.

3. Step becomes inactive.  
   X bit turns off.  
   Timer retains its value.  
   Since Timer is less than LimitLow, AlarmLow bit turns on.

4. Step becomes active.  
   X bit turns on.  
   Timer clears and then begins to increment.  
   AlarmLow stays on. (You have to manually turn it off.)

5. Timer reaches the LimitHigh value of the step.  
   AlarmHigh bit turns on.  
   Timer continues to increment.

   X bit turns off.  
   Timer retains its value.  
   AlarmHigh stays on. (You have to manually turn it off.)
Here is an example of the use of the Preset time of a step.

**EXAMPLE**

Functional specification says:

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

Solution:

```
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cook</td>
<td>Cook.PRE = 20000 ms</td>
</tr>
<tr>
<td>Cook_Done</td>
<td>Cook.ON = on (1)</td>
</tr>
<tr>
<td>Empty_Tank</td>
<td></td>
</tr>
</tbody>
</table>
```

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

**EXAMPLE**

Functional specification says:

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

Solution:

```
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Init</td>
<td>Init.LimitHigh = 20000 ms</td>
</tr>
<tr>
<td>Init_OK</td>
<td>Init.AlarmHigh</td>
</tr>
<tr>
<td>Init_Not_OK</td>
<td></td>
</tr>
<tr>
<td>Step_1</td>
<td></td>
</tr>
<tr>
<td>Shutdown</td>
<td></td>
</tr>
</tbody>
</table>
```
**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**

Use logic to clear data

**Automatic Reset**

Let the controller automatically clear data

Each option requires you to make these choices:

- 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.

<table>
<thead>
<tr>
<th>If you want to</th>
<th>And on the last scan of a step</th>
<th>Then</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control which data to clear</td>
<td>Execute only P and P0 actions and use them to clear the required data.</td>
<td>Use the Don’t Scan Option</td>
<td>page 38</td>
</tr>
<tr>
<td></td>
<td>Execute all actions and use either of these options to clear the required data:</td>
<td>Use the Programmatic Reset Option</td>
<td>page 39</td>
</tr>
<tr>
<td></td>
<td>• status bits of the step or action to condition logic</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• P and P0 actions</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Let the controller clear data</td>
<td></td>
<td>Use the Automatic Reset Option</td>
<td>page 41</td>
</tr>
</tbody>
</table>
This table compares the different options for handling the last scan of a step:

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>During the last scan of a step, this option does</th>
<th>Automatic reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution actions</td>
<td>Don't scan: Only P and P0 actions execute. They execute according to their logic.</td>
<td>• P and P0 actions execute according to their logic.</td>
</tr>
<tr>
<td></td>
<td>Programmatic reset: All actions execute according to their logic.</td>
<td>• All other actions execute in Postscan mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• On the next scan of the routine, the P and P0 actions execute in Postscan mode.</td>
</tr>
<tr>
<td>Retention of data values</td>
<td>All data keeps its current values.</td>
<td>• Data reverts to its values for postscan.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Tags to the left of [:=] assignments clear to zero.</td>
</tr>
<tr>
<td>Method for clearing data</td>
<td>Use P and P0 actions.</td>
<td>Use either:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Status bits of the step or action to condition logic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• P and P0 actions</td>
</tr>
<tr>
<td>Reset of a nested SFC</td>
<td>A nested SFCs remains at its current step.</td>
<td>For the Restart Position property, if you choose the Restart at initial step option, then:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• a nested SFC resets to its initial step.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• the X bit of a stop element in a nested SFC clears to zero.</td>
</tr>
</tbody>
</table>

The table shows the differences in behavior for different last scan options, including execution actions, retention of data values, method for clearing data, and reset of a nested SFC.
Use the Don’t Scan Option

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

To turn off a device at the end of a step:

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

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 16.

EXAMPLE

Use the Programmatic Reset Option and the LS Bit

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**

```
S  convoy_start
If convoy_start.Q then
  convoy_state:=1;
Else
  convoy_state:=0;
End_if;
```

When the action is not on its last scan (conveyor_start.Q =1), this statement turns on convoy_state. When convoy_state turns on, the conveyor turns on.

On the last scan of the action (conveyor_start.Q =0), this statement turns off convoy_state. When convoy_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**

```
N  convoy_start
  convoy_state := 1;
```

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

```
P0  convoy_stop
  convoy_state := 0;
```

Before the SFC leaves the step, the P0 action turns off the conveyor. On the last scan of the step, convoy_state turns off. This turns off the conveyor.
### Use the Automatic Reset Option

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 such as:
   - \([:=]\) assignment (non-retentive assignment)
   - Output Energize (OTE) instruction in a subroutine

During the last scan of each step, the Automatic reset option:
   - executes P and P0 actions according to their logic conditions
   - clears tags to the left of \([:=]\) assignments
   - executes a postscan of embedded Structured Text
   - executes a postscan of any subroutine that an action calls via a Jump to Subroutine (JSR) instruction
   - resets any nested SFC (SFC that an action calls as a subroutine)

---

**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.

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

- *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 1756-RM007
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

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:

<table>
<thead>
<tr>
<th>Option</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use a Simultaneous Branch</strong></td>
<td><img src="image.png" alt="Diagram" /></td>
</tr>
<tr>
<td>Make a separate step that controls the device.</td>
<td>Turn on the fan</td>
</tr>
<tr>
<td></td>
<td>Paint</td>
</tr>
<tr>
<td></td>
<td>Clean</td>
</tr>
<tr>
<td></td>
<td>Transf_Out</td>
</tr>
<tr>
<td><strong>Store and Reset an Action</strong></td>
<td><img src="image.png" alt="Diagram" /></td>
</tr>
<tr>
<td>Note the step that turns on the device and the step that turns off the device.</td>
<td>Turn off the fan</td>
</tr>
<tr>
<td>Later, define a Stored and Reset Action pair to control the device.</td>
<td>Transfer_In</td>
</tr>
<tr>
<td></td>
<td>Paint</td>
</tr>
<tr>
<td></td>
<td>Clean</td>
</tr>
<tr>
<td></td>
<td>Transf_Out</td>
</tr>
<tr>
<td><strong>Use One Large Step</strong></td>
<td><img src="image.png" alt="Diagram" /></td>
</tr>
<tr>
<td>Use one large step that contains all the actions that occur while the device is on.</td>
<td>Transfer, paint, clean, transfer, control the fan</td>
</tr>
</tbody>
</table>
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:

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.

Solution:

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.

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 41.
- 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**

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.

<table>
<thead>
<tr>
<th>To</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatically loop back to an earlier step</td>
<td>Wire the last transition to the top of the step to which you want to go.</td>
</tr>
<tr>
<td>Stop and wait for a command to restart</td>
<td>Use a Stop Element.</td>
</tr>
</tbody>
</table>

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:

- 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:

<table>
<thead>
<tr>
<th>If the stop element is at the end of a</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence</td>
<td>Entire SFC stops</td>
</tr>
<tr>
<td>Selection branch</td>
<td></td>
</tr>
<tr>
<td>Path within a simultaneous branch</td>
<td>Only that path stops while the rest of the SFC continues to execute.</td>
</tr>
</tbody>
</table>

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:

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

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:

<table>
<thead>
<tr>
<th>If you want to</th>
<th>Then check or set this member</th>
<th>Data type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Determine when the SFC is at the stop</td>
<td>X</td>
<td>BOOL</td>
<td>• When the SFC reaches the stop, the X bit turns on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The X bit clears if you configure the SFCs to restart at the initial step and the controller changes from program to run mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 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.</td>
</tr>
<tr>
<td>Determine the target of an SFC Reset (SFR) instruction</td>
<td>Reset</td>
<td>BOOL</td>
<td>An SFC Reset (SFR) instruction resets the SFC to a step or stop that the instruction specifies.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The Reset bit indicates to which step or stop the SFC will go to begin executing again.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Once the SFC executes, the Reset bit clears.</td>
</tr>
<tr>
<td>Determine how many times a stop has become active</td>
<td>Count</td>
<td>DINT</td>
<td>This is not a count of scans of the stop.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The count increments each time the stop becomes active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• It increments again only after the stop goes inactive and then active again.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 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 stop</td>
<td>Status</td>
<td>DINT</td>
<td>For this member</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
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.

1. Reset the nested SFC.
   - The SFR instruction restarts SFC_b at SFC_b_Step_1. Each time SFC_a leaves this step and then returns, you have to reset SFC_b.
   - The action also clears the X bit of the stop element.

2. Call SFC_b.

3. Stop SFC_b. This sets the X bit of the stop element.

4. Use the X bit of the stop element to signal that SFC_b is done and it is time to go to the next step.
Pass Parameters

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

Configure When to Return to the OS/JSR

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

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

Use the Execute until FALSE transition option only when:

- 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:

<table>
<thead>
<tr>
<th>If you want to</th>
<th>Then use this instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pause an SFC</td>
<td>Pause SFC (SFP)</td>
</tr>
<tr>
<td>Reset an SFC to a specific step or stop</td>
<td>Reset SFC (SFR)</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>For a diagram of the</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution of a Sequence</td>
<td>52</td>
</tr>
<tr>
<td>Execution of a Simultaneous Branch</td>
<td>53</td>
</tr>
<tr>
<td>Execution of a Selection Branch</td>
<td>54</td>
</tr>
<tr>
<td>When parameters enter and exit an SFC</td>
<td>54</td>
</tr>
<tr>
<td>Options for Execution Control</td>
<td>55</td>
</tr>
</tbody>
</table>
Execution of a Sequence

This...

...executes like this

```
step_1
   tran_1
step_2
   tran_2
```

```
step_1
   tran_1 true
       return to OS/JSR
   tran_1 false
       return to OS/JSR

step_1
   return to OS/JSR
   step_2
   tran_2 false
       return to OS/JSR
   tran_2 true
       return to OS/JSR

step_2
   return to OS/JSR
   Last scan
```

Last scan
Chapter 1  Designing a Sequential Function Chart

Execution of a Simultaneous Branch

This... ...executes like this

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

This... ...executes like this

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

Execution of a Selection Branch

…executes like this

When parameters enter and exit an SFC
Chapter 1  Designing a Sequential Function Chart

Options for Execution Control

This…

 Execute current active steps only

 Execute until FALSE transition

...executes like this
Notes:
### Programming a Sequential Function Chart

#### Introduction

To add SFC elements, use the SFC toolbar.

#### Add and Manually Connect Elements

1. On the SFC toolbar, click the button for the item that you want to add.

2. Drag the element to the required location on the SFC.

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

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

End a Simultaneous Branch

1. Select the last step of each path in the branch. To select the steps, you can either:
   - 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  

   ![Diagram of sequential function chart with steps and transitions marked for end a simultaneous branch process]
3. Add the transition that follows the simultaneous branch.

4. To wire the simultaneous branch to the transition, click the top pin of the transition A and then click the horizontal line of the branch B. 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

3. To wire the selection branch to the preceding step, click the bottom pin of the step A and then click the horizontal line of the branch B. 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, you can either:
   - 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](image_url)

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.
Chapter 2  Programming a Sequential Function Chart

For example, to go to Step_001 from Tran_003:

1. Then click here.

2. 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.

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.

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 65.

Configure Alarms for a Step

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.

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:

- Preset
- LimitHigh
- LimitLow

To enter a time as an expression:

1. Click the button of the step.
2. Check the Use Expression check box.
3. Click the Define button and enter an expression.

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

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].

This example shows three transitions that use a BOOL expression.

EXAMPLE Enter a BOOL Expression

Call a Subroutine

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.

Add an Action

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

Configure an Action

Change the Qualifier of an Action

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

1. Click the button of the action.

2. On the General tab, select the qualifier for the action.
If you chose a timed qualifier, type the time limit or delay for the action, in milliseconds. Timed qualifiers include:

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

3. Click OK.

**Calculate a Preset Time at Runtime**

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

1. Click the \( \text{\textasciitilde} \) button of the action.
2. Check the Use Expression check box.
3. Click the Define button.

![Diagram showing action properties dialog box]

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.
2. Check the Boolean check box.
3. Click OK.

Program an Action

To program an action, you can:

- 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.

1. Double-click the text area of the action.
2. Type the required structured text.
3. To close the text entry window, press [Ctrl] + [Enter].
Call a Subroutine

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.

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.
## 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

To document an SFC, you can:

<table>
<thead>
<tr>
<th>To document this</th>
<th>And you want to</th>
<th>Do this</th>
</tr>
</thead>
<tbody>
<tr>
<td>General information about the SFC</td>
<td></td>
<td>Add a Text Box</td>
</tr>
<tr>
<td>Step</td>
<td></td>
<td>Add a Text Box</td>
</tr>
<tr>
<td>Transition</td>
<td>Download the documentation to the controller</td>
<td>Add Structured Text Comments</td>
</tr>
<tr>
<td></td>
<td>Have the option of showing or hiding the documentation</td>
<td>Add a Text Box</td>
</tr>
<tr>
<td></td>
<td>Position the documentation anywhere in the SFC</td>
<td>Add a Text Box</td>
</tr>
<tr>
<td>Action</td>
<td>Download the documentation to the controller</td>
<td>Add Structured Text Comments</td>
</tr>
<tr>
<td>Stop</td>
<td></td>
<td>Add a Text Box</td>
</tr>
<tr>
<td>Other element (such as a selection branch)</td>
<td></td>
<td>Add a Text Box</td>
</tr>
</tbody>
</table>
Add Structured Text Comments

Comments embedded in the ST section of an action are downloaded into controller memory and are available for upload. To enter comments:

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

2. Type the comments.

<table>
<thead>
<tr>
<th>To add a comment</th>
<th>Use one of these formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>On a single line</td>
<td>//comment</td>
</tr>
<tr>
<td>At the end of a line of structured text</td>
<td>(<em>comment</em>)</td>
</tr>
<tr>
<td></td>
<td>/<em>comment</em>/</td>
</tr>
<tr>
<td>Within a line of structured text</td>
<td>(<em>comment</em>)</td>
</tr>
<tr>
<td></td>
<td>/<em>comment</em>/</td>
</tr>
<tr>
<td>That spans more than one line</td>
<td>(<em>start of comment ... end of comment</em>)</td>
</tr>
<tr>
<td></td>
<td>/<em>start of comment ... end of comment</em>/</td>
</tr>
</tbody>
</table>

3. To close the text entry window, press [Ctrl] + [Enter].
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, stop, etc.). 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. For example:

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?

<table>
<thead>
<tr>
<th>If you the text box to</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stay in the same spot</td>
<td>Stop. You are done.</td>
</tr>
<tr>
<td>Move with the element</td>
<td>Go to step 5.</td>
</tr>
<tr>
<td>to which it applies</td>
<td></td>
</tr>
</tbody>
</table>

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.
Hide an Individual Tag Description

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.

3. Click OK.

Language Switching

With RSLogix 5000 software, version 17, you have the option to display project documentation, such as tag descriptions and rung comments for any supported localized language. You can store project documentation for multiple languages in a single project file rather than in language-specific project files. You define all the localized languages that the project will support and set the current, default, and optional custom localized language. The software uses the default language if the current language's content is blank for a particular component of the project. However, you can use a custom language to tailor documentation to a specific type of project file user.

Enter the localized descriptions in your RSLogix 5000 project, either when programming in that language or by using the import/export utility to translate the documentation 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:

- 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.
Configure the Execution of the SFC

The SFC Execution tab of the controller properties lets you configure:

- what to do when a transition is true.
- where to start after a transition to the Run mode or recovery from a power loss.
- what to do on the last scan of a step.

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

2. Select the SFC Execution tab.

3. Choose:
   - 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:

<table>
<thead>
<tr>
<th>If you want to restart at the</th>
<th>Select</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last step that was running</td>
<td>Restart at most recently executed step</td>
</tr>
<tr>
<td>Initial step</td>
<td>Select Restart at initial step</td>
</tr>
</tbody>
</table>

   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. 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:
- 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. 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.
Notes:
Force Steps

Introduction

Use a force to override data that your logic either uses or produces. For example, use forces to:

- 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

When you use forces, take these 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.

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.

Check Force Status

Before you use a force, determine the status of forces for the controller. You can check force status:

<table>
<thead>
<tr>
<th>To determine the status of</th>
<th>Use any of the following</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O forces</td>
<td>• Online toolbar</td>
</tr>
<tr>
<td></td>
<td>• FORCE LED</td>
</tr>
<tr>
<td></td>
<td>• GSV instruction</td>
</tr>
<tr>
<td>SFC forces</td>
<td>Online Toolbar</td>
</tr>
</tbody>
</table>
The Online toolbar shows the status of forces. It shows the status of I/O forces and SFC forces separately.

<table>
<thead>
<tr>
<th>This</th>
<th>Means</th>
</tr>
</thead>
</table>
| Enabled         | • If the project contains any forces of this type, they *are* overriding your logic.  
                  | • If you add a force of this type, the new force immediately takes effect |
| Disabled        | Forces of this type are inactive. If the project contains any forces of this type, they *are not* overriding your logic. |
| Installed       | At least one force of this type exists in the project.              |
| None Installed  | No forces of this type exist in the project.                        |

**FORCE LED**

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

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

<table>
<thead>
<tr>
<th>If the FORCE LED is</th>
<th>Then</th>
</tr>
</thead>
</table>
| 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.

```
<table>
<thead>
<tr>
<th>Get System Value</th>
<th>MODULE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance Name</td>
<td>ForceStatus</td>
</tr>
<tr>
<td>Attribute Name</td>
<td>Force_Status</td>
</tr>
</tbody>
</table>
```

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. With 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**

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:

<table>
<thead>
<tr>
<th>If you want to</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>Override the conditions of a transition each</td>
<td>Force a Transition</td>
</tr>
<tr>
<td>time the SFC reaches the transition</td>
<td></td>
</tr>
<tr>
<td>Prevent the execution of one or more paths</td>
<td>Force a Simultaneous Path</td>
</tr>
<tr>
<td>of a simultaneous branch</td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>If you want to</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prevent the SFC from going to the next step</td>
<td>Force the transition false</td>
</tr>
<tr>
<td>Cause the SFC go to the next step regardless of</td>
<td>Force the transition true</td>
</tr>
<tr>
<td>transition conditions</td>
<td></td>
</tr>
</tbody>
</table>

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.
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.

For example, to exit this branch, the SFC must be able to:

- 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**

This path executes.

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:

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

<table>
<thead>
<tr>
<th>If</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>No SFC forces currently exist.</td>
</tr>
<tr>
<td>Flashing</td>
<td>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.</td>
</tr>
<tr>
<td>Solid</td>
<td>SFC forces are enabled (active). When you install (add) a force, it immediately takes effect.</td>
</tr>
</tbody>
</table>

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.)

<table>
<thead>
<tr>
<th>If</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>From the Logic menu, choose SFC Forcing &gt; Enable All SFC Forces. Then choose Yes to confirm.</td>
</tr>
<tr>
<td>Yes</td>
<td>Stop.</td>
</tr>
</tbody>
</table>
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 provides technical information on the Web to assist you in using its products. At http://support.rockwellautomation.com, 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://support.rockwellautomation.com.

Installation Assistance

If you experience a problem within the first 24 hours of installation, please review the information that's contained in this manual. You can also contact a special Customer Support number for initial help in getting your product up and running.

<table>
<thead>
<tr>
<th></th>
<th>United States</th>
<th>Outside United States</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.440.646.3434</td>
<td>Please contact your local Rockwell Automation representative for any technical support issues.</td>
</tr>
<tr>
<td></td>
<td>Monday – Friday, 8am – 5pm EST</td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th></th>
<th>United States</th>
<th>Outside United States</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Contact your distributor. You must provide a Customer Support case number (call the phone number above to obtain one) to your distributor in order to complete the return process.</td>
<td>Please contact your local Rockwell Automation representative for the return procedure.</td>
</tr>
</tbody>
</table>

www.rockwellautomation.com