Creating GRAPH programs

This chapter contains the following information:

- Basic information on GRAPH (S7-300/400)
- Settings for GRAPH (S7-300/400)
- The programming window (S7-300/400)
- Program sequencer (S7-300/400)
- Program actions and conditions (S7-300/400)
- Alarms (S7-300/400)
- Call GRAPH-FB and assign parameters (S7-300/400)
- Compile and load GRAPH-FB (S7-300/400)

Basic information on GRAPH

This chapter contains the following information:

- GRAPH programming language (S7-300/400)
- Structure of a sequential control system (S7-300/400)
- Step elements (S7-300/400)
- Permanent instructions (S7-300/400)

GRAPH programming language

Overview

GRAPH is a graphic programming language for the creation of sequential control systems.

Sequential control systems can be programmed quickly and easily. The process is broken down into individual steps each with a clear scope of functions and represented graphically. The actions to be executed are defined in the individual steps. The transitions between the steps form the transitions. These contain conditions for switching to the next step.

Sequential control modules

A sequential control controls the process in a pre-defined sequence and in dependence on certain conditions.

The complexity of the sequential control is specified by the automation task. At least three blocks always belong to a sequential control system.

- Code block to be called
  GRAPH-FB is called by a higher-level code block. This block can be an OB, an FC or another FB.

- GRAPH-FB
  GRAPH-FB describes the individual subtasks and dependencies of the sequential control system. The subtasks (steps) and dependencies (transitions) can be organized in one or more sequencers.

- Instance DB
  The instance DB contains the data and parameters of the sequential control system. The instance DB is assigned to the GRAPH-DB and can be automatically generated by the system.

The following figure shows how the blocks work together in a sequential control system:
The cycle of a GRAPH function block is defined by the cycle of the calling block. In each cycle, the permanent pre-instructions are executed first in the GRAPH function block. Then the actions of the active steps are processed. During this it is checked whether the step enabling conditions are satisfied. Last, the permanent post-instructions are executed. If the step enabling conditions of a step are not satisfied, these actions are executed again in the next cycle.

**Note**
Permanent instructions are executed in each cycle, even if no step is active.

**Sequencers**
You program sequencers in the GRAPH-FB. A sequencer consists of a series of steps that are activated in a fixed sequence depending on the transitions (step enabling).

The following figure shows examples of possible sequencers with all elements that can be contained in a sequencer structure in GRAPH:

1. GRAPH-FB with linear sequencer and jump instruction
2. GRAPH-FB with a sequencer that contains an alternative and simultaneous branch, as well as a jump instruction
3. GRAPH-FB with two sequencers, alternative branch and jump instruction
Structure of a sequential control system

Sequencer
In the GRAPH programming window you create sequencers by linking sequence elements in sequences according to the requirements of your control system. The standard instructions contain the following elements for this:

- Step and transition
- Step
- Transition
- Simultaneous branch
- Alternative branch
- Branch end
- Jump
- Sequence end

The sequence elements can be found on the "Instructions" task card or in the Requirements area of the programming window.

Processing principle of a sequencer
1. The processing of a sequencer always begins with one or more of the following initial steps: The initial steps can be at any position in a sequencer. As long as the step is active, its actions are executed in each cycle. If several steps are executed at the same time, all these steps are active.
2. An active step is exited under the following conditions:
   - The transition that follows the step is satisfied
   - There is no supervision error.
3. The next step of the transition to be satisfied becomes active.
4. The following elements can be found at the end of a sequencer:
   - Jump to any step of this or another sequencer of the FB
     This allows a cyclic operation of the sequencer.
   - Sequence end
     The task ends when the end of the sequence is reached. No step is then active any longer. However, permanent operations are executed again in the next cycle.

Step
The following figure shows a graphic illustration of a step/transition pair:

![Step/Transition Pair](image)

The tasks of a sequence control system are divided into individual steps. In the steps you formulate instructions in the form of actions that are executed by the control system under certain defined conditions. Each step of the sequence is processed individually during the execution of the program.

Active step
An active step is a step whose actions are being processed in the current cycle.
A step is activated if one of the following conditions is satisfied:
- It is defined as initial step and the sequencer was initialized.
- The conditions of the previous transition are satisfied.
- It is called by an event-dependent action.

Note
A jump from a transition to a directly preceding step is to be avoided.
To make possible this jump instruction, insert an empty step with a transition without condition.

**Empty steps**

Empty steps should be used as step/transition pair.

Empty steps are steps in which no actions or conditions are programmed. Consequently, no errors can be reported from these steps. Since interlock and supervision without condition always describe the best outcome, the following transition without condition is always satisfied.

You can use an empty step, for example, to avoid a jump after a transition to any preceding step of the transition. An empty step can also be used to lead a sequencer to a common jump or a common sequence end after the connection of branches, since a sequencer cannot be connected to a common jump or a common sequence end directly after the branches have been connected.

**Transition**

The following figure shows the graphic representation of a transition:

```
+------------------
| T1               |
| Trans1           |
```

Transitions contain the conditions for switching the sequence from one step to the next. A transition becomes valid, or is then evaluated, when all its preceding steps are active. If the step enabling conditions of a valid transition are satisfied, it switches to the next step. The step or steps belonging to the transition are hereby deactivated and the next step activated.

**Simultaneous branch**

The following figure shows the graphic representation of a simultaneous branch:

```
+------------------+
| Step1            |
| T1               |
| Trans1           |
```

```
+------------------+
| Step2            |
| T2               |
| Trans2           |
```

```
+------------------+
| Step3            |
| T3               |
| Trans3           |
```

```
+------------------+
| Step4            |
```

A simultaneous branch is an AND branch and consists of several parallel branches, each of which starts with a step. A sequence has maximum of 249 parallel branches.

The simultaneous branches of sequence are executed simultaneously. Each simultaneous branch ends with a step and is closed with a follow-on transition. The branch end terminates an open branch to the sequence running on the left next to it.

If all simultaneous branches are combined to a transition, this transition only switches to the next step if all active simultaneous branches have been completely processed.

**Alternative branch**

The following figure shows the graphic representation of an alternative branch:
If several transitions follow a step, the entry point for an alternative branch is located at these points. An alternative branch is an OR branch and consists of several parallel branches, each of which starts with a transition. A sequence can have a maximum of 125 parallel alternative branches. Only the alternative branch is executed, whose transition switches first. Each alternative branch ends with a transition and can be closed with a follow-on step, a sequence end or a jump. The branch end terminates an open branch to the sequence running on the left next to it.

If several transitions are satisfied simultaneously at the start of different branches, the transition farthest to the left has the highest priority in each case.

**Jump**

The following figure shows the graphic representation of a jump:

```
S3
Step3
```

```
T3
Trans3
```

```
S4
Step4
```

```
T4
Trans4
```

A jump is the transition from a transition to any step within the sequencer or another sequencer of the same function block. A jump is always positioned after a transition and ends the sequencer or a branch at this point.

In contrast to the sequence end, a jump causes the cycle processing of the sequencer or the repeated processing of parts of the sequencer, or the processing of a different sequencer within the GRAPH function block.

Jump and jump destination are represented as arrows.

**Sequence end**

The following figure shows the graphic representation of a sequencer:
The sequence end terminates the sequencer at the end of a linear sequencer or a branch of an alternative sequence.

At the end of branch of a simultaneous branch, a sequence end terminates only this branch. Processing continues in the other simultaneous branches.

A sequence end is always positioned after a transition.

**Note**

If all branches of a sequencer are closed with a sequence end, the sequencer can only be re-started by the INIT_SQ parameter.

**Run variants**

A sequencer can, for example, be cyclically processed to the start of the sequence by a jump at the end of the processing. If the sequence ends with a sequence end, it is only executed once.

You can also find examples of run variants under "See also".

---

### Step elements

#### Action

An action causes an instruction to execute a function when a step becomes active. The active parts of a step are the actions that are executed by the control system in the active step state. In the actions you program instructions and functions which, for example, call function blocks or steps or activate or deactivate the sequencer.

The following table shows the action examples:

<table>
<thead>
<tr>
<th>Action consists of instruction and operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction</td>
</tr>
<tr>
<td>N</td>
</tr>
<tr>
<td>-(C)- N</td>
</tr>
</tbody>
</table>

For additional information on operands, refer to "See also."

You can make the execution of an action dependent on events and interlocks.

The following table shows an action in dependence on an event.

<table>
<thead>
<tr>
<th>Action consists of event, instruction and operand</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event</td>
</tr>
<tr>
<td>S1</td>
</tr>
</tbody>
</table>

#### Event

An event is the change of the signal state of a step, a supervision or an interlock, or the acknowledgement of an alarm or an incoming registration. An event can be recorded and processed with an action.

The following events are possible:

<table>
<thead>
<tr>
<th>Event</th>
<th>Signal analysis</th>
<th>Step event</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Rising edge</td>
<td>Incoming step (signal = 1)</td>
</tr>
<tr>
<td>S0</td>
<td>Falling edge</td>
<td>Outgoing step (signal = 0)</td>
</tr>
<tr>
<td>V1</td>
<td>Rising edge</td>
<td>Supervision, incoming event (signal =1) (incoming fault)</td>
</tr>
</tbody>
</table>
The signal states of an event can be used to program additional actions. This allows you to monitor and influence not only the individual steps but also the entire sequential control.

For information on actions, refer to "See also".

Interlock
An interlock (interlocking condition) is a condition that can be programmed within a step. If the condition is satisfied, this is the best outcome: There are no faults. In this case, for example, a protective grill is closed. If you have not programmed an interlock in a step, this is treated as a best outcome. An action with which the interlock is linked is then executed. The enabling of the next step is dependent on the interlock. When a step is deactivated, a pending interlock is automatically cancelled.

For information on programming an interlock, refer to "See also".

Supervision
A supervision (step monitoring) is a condition that can be programmed within a step. If the condition is not satisfied, this the best outcome. For example, a tank is being drained. If a fault occurs, the worse outcome occurs, with which the "Tank empty" condition is satisfied. A satisfied supervision leads to an error message. You can use the "Messages" pane in the area navigation of the programming window to define the properties and contents of messages.

For information on changing and defining messages, refer to "See also".

The next step is only enabled if the monitoring error is no longer pending and the following transition is satisfied. When a step is deactivated, the pending monitoring error is automatically cancelled. Accordingly, it is also not possible to interrupt a step that is not active.

For information on programming a supervision, refer to "See also".

Transition
A transition specifies a step-enabling condition that has to be satisfied before a controller switches from one state to the follow-on state. The sequencer hereby switches from one step to the next step or steps. A transition becomes valid, or is then evaluated, when its preceding step is active. The applies analogously for several preceding steps. A valid transition switches if the step-enabling condition is satisfied and the supervision is not satisfied.

During switching, the transition terminates the preceding step and activates the follow-on step. The same applies analogously for several preceding and follow-on steps.

A transition can contain the following elements:
- Step-enabling condition
  - A step-enabling condition bit logic operation and/or comparison instruction that is programmed in LAD or FBD representation.
- Transition number
  - A transition number can be entered from the value range 1-999 or is automatically assigned by the system.
- Transition name
  - A transition name can consist of not more than 128 characters, whereby the first character must be alphanumeric. If special characters are used in transaction names, these have to be separated by inverted commas.

For information on programming a transition, refer to "See also".

Note
Interlocks, supervisions, transitions and permanent instructions can include maximum 32 program elements.

### Permanent instructions

**Introduction**
Permanent instructions are conditions or block calls that are located before or after the sequencer. They are programmed with LAD (ladder block diagram) or FBD (function block diagram) elements and processed once per cycle independently of the state of the sequencer.

You can program a maximum of 32 LAD or FBD elements per permanent instruction. A GRAPH-FB can contain a total of 250 permanent pre-instructions and 250 permanent post-instructions.

The result of the linking of conditions is stored in a LAD coil or in a FBD assignment, for which the memory functions set and reset are also available. The address used (for example, a bit memory) can be queried in transition, interlock, supervision and other permanent instruction networks.

For more information on programming LAD and FBD instructions, and on changing the programming language for conditions, refer to "See also"

**Permanent instructions**

You can program conditions and block calls as permanent instructions. You create permanent instructions in one of the panes "Permanent pre-instructions" or "Permanent post-instructions".

For information on programming a permanent instruction, refer to "See also".

In the programming window the network of a permanent instruction consists of a number, a name and a graphic network element. You can enter a separate name for each network of the permanent pre-conditions or post-conditions. You can program conditions and function or block calls in a network.

**Note**

Interlocks, supervisions, transitions and permanent instructions can include maximum 32 program elements.

---

**Settings for GRAPH**

This chapter contains the following information:

- Overview of the settings for GRAPH (S7-300/400)
- Changing the settings (S7-300/400)
- Changing programming language for conditions (S7-300/400)

**Overview of the settings for GRAPH**

**Default settings**

You can customize the following defaults for GRAPH:

<table>
<thead>
<tr>
<th>Group</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load</td>
<td>Turn off the sequence before downloading the DB</td>
<td>Sequence is turned off before downloading the DB</td>
</tr>
<tr>
<td>Default for time monitoring</td>
<td>Uninterrupted step time (U)</td>
<td>Default for time monitoring of a step with the CPM&gt;U instruction.</td>
</tr>
<tr>
<td></td>
<td>Step time (T)</td>
<td>Default for time monitoring of a step with the CPM&gt;T instruction.</td>
</tr>
<tr>
<td>Editor</td>
<td>Language in networks</td>
<td>Selecting a programming language (LAD or FBD)</td>
</tr>
<tr>
<td>Interface</td>
<td>Maximum interface set</td>
<td>New blocks are created with the maximum parameter record.</td>
</tr>
<tr>
<td>Compile</td>
<td>Create minimized DB</td>
<td>New blocks are created with a reduced record of static parameters. This means that some functions are not available. For more information, refer to “See also”.</td>
</tr>
<tr>
<td>Sequence properties</td>
<td>Skip steps</td>
<td>If the transition before a step and the</td>
</tr>
</tbody>
</table>
### Changing the settings

You can specify GRAPH defaults for function blocks and for the behavior during the loading of data blocks.

**Requirements**

A device is available.

**Procedure**

To change the settings of GRAPH, follow these steps:

1. Open the "Options > Settings" menu.
   - In the navigation view you see the panes for the various settings.
2. Open the "PLC programming" pane.
The content of the selected pane is displayed in the work area.

3. Select the "GRAPH" entry in the opened pane.
   The properties of GRAPH are adapted in the work area.

4. In this work area, navigate to the item whose settings you want to change.

5. Make your changes.

Result
The settings are changed.

---

Changing programming language for conditions

Changing the programming language in the project tree
To change the programming language for conditions in the project tree, follow these steps:

1. In the project tree, select the block whose programming language you want to change.
2. Select the "Properties" command in the shortcut menu.
   A dialog shows the general properties of the block.
3. In the "General" area under "Language in networks", select the programming language you want for the block.

Changing the programming language in the inspector window
To change the programming language in the inspector window, follow these steps:

1. Open the block whose programming language you want to change.
2. Open the inspector window.
3. In the tab "Properties", select the item "General" in the area navigation.
4. In the "General" area, select the desired programming language from the "Language in networks" drop-down list.

Result
The programming language for the block is changed. The favorites area is adapted according to the selected programming language.

---

The programming window

This chapter contains the following information:
- Overview of the programming window (S7-300/400)
- Navigation (S7-300/400)
- Sequence view (S7-300/400)
- Single step view (S7-300/400)
- Action table (S7-300/400)
- Permanent Instructions (S7-300/400)
- Navigating in the programming window (S7-300/400)
- Switching between single step and sequence view (S7-300/400)
- Inspector window (S7-300/400)
- "Sequence control" pane (S7-300/400)
- "Test settings" pane (S7-300/400)
- Keyboard shortcuts in GRAPH (S7-300/400)

---

Overview of the programming window

file:///C:/Users/Kvascev/AppData/Local/Temp/~hhC258.htm
Overview

In the GRAPH programming window, you create the sequences, steps, conditions, alarms and all other elements that you required for your sequencer.

The following figure shows the programming window of GRAPH:

All additional user interface areas that are located outside the GRAPH programming window, belong to the program editor. The size of each area can be customized to make work easier.

For additional information on the panes of the navigation view and the program editor, refer to "See also".

Navigation

In the navigation view, you can switch between the individual panes that open the corresponding programming areas. The representation in the work area changes depending on the selected pane.

Here you can change between the following panes:

- Permanent pre-instructions
- Sequences
- Permanent post-instructions
- Alarms

When you open a newly created block for the programming of your sequencer, the "Sequences" pane appears by default. The first step of the sequence is contained here.

To program a permanent pre-instruction or post-instruction, you first have to create a network in the pane.

For information on creating a network and on the panes of the navigation view, refer to "See also".

Favorites

You can find the favorites of the opened pane at the top edge of the programming window. You can use the favorites to assemble the key components of your network or your sequence in the work area. You can customize the "Favorites" pane in the "Instructions" task card. To show or hide the favorites in the programming window, select the shortcut command "Display favorites in the editor" in the favorites area in the programming window or on the task card.

For information on favorites, refer to "See also".

Work area

You program the individual components of your sequencer in the work area of the programming window. The appearance of the work area depends on which pane is open.

If you create complex sequence structures in the sequence view, it can be difficult to display all elements of the sequencer in the work area.
area. To keep a clear overview in such cases, you can use the elements at the bottom edge of the programming window to customize the size of the work area.

For information on programming the structure of a sequencer, refer to "See also".

### Navigation

#### Overview

The navigation view is used to switch quickly and easily between programming views for the individual sequence components. To open the programming views, click on the title bar of the corresponding pane in the navigation view.

Networks and sequences that are opened in the work area are displayed minimized in the navigation view also. If several sequences or networks are present, you can recognize in the navigation view in which network or in which sequence you are currently located. You can switch among the structures with one click. To keep a clear overview when there are several sequences and networks, you can use the small triangle on the left side of the respective title bar to collapse unprocessed structures.

#### Pane for permanent instructions

When you open a pane for permanent instructions, an empty operation is located by default in the work area. If a network for permanent instructions is available, this is shown in the work area as soon as you select it. The favorites for the creation of permanent instructions are located at the top edge of the work area. Below this you can enter a name and below this a comment on the permanent instruction.

For information on permanent instructions, refer to "See also".

#### "Sequences" pane

When you open a function block, the "Sequences" pane is opened by default in the navigation view. The first step of the sequence is already present in the work area. By default, this is predefined as the initial step. To program a step, double-click the required step in the navigation view or in the work area. The step is then opened in the single step view.

For information on single step view and on steps, refer to "See also".

#### "Messages" pane

In the "Alarms" pane, you can enable or disable the generation of alarms for interlock and supervision alarms. If alarm generation is enabled, you can also enable or disable the acknowledgment required for each of the alarm types. You can customize the identifier text for interlock and supervision alarms. The text can contain up to 50 characters.

For information on alarms, refer to "See also".

### Sequence view

#### Overview

If the "Sequences" pane is open in the navigation view, the selected sequencer is displayed in the work area of the programming window. If no sequence has yet been programmed, the first step of the sequence is displayed by default. In the navigation view, you can create additional sequences via the shortcut menu. If the title bar of the "Sequences" pane is selected at the same time, the new sequence is inserted before the existing sequence. If the title bar of a sequence is selected, the new sequence is inserted after the selected sequence.

#### Display of sequences

The sequence view shows the sequence with its main elements:

- Step
- Transition
- Jump
- Branch
- Sequence end

You create sequences by creating the required sequence element on the existing initial step. To do this you insert element symbols in the work area in order to create the required sequence structure here. You can find the instructions required for this in the favorites area at the top edge of the work area.

For information on favorites, refer to "See also".
Sequences can be built up in a more or less complex manner. They can consist of only one step, a linear series of steps or complex sequence structures with branches and jumps. To keep a clear overview of your sequence, you can change the display size of your sequence in the programming window.

For information on adapting the display size and on creating sequences, refer to "See also".

---

**Single step view**

**Overview**

In the single step view you program all components that a step can contain. The panes in which you carry out the programming are labeled accordingly:

- Interlock
- Supervision
- Actions
- Transition

To provide a clearer overview, the currently opened step and its transition is highlighted with a white rectangle in the navigation view. At the top edge of the work area you can enter a name and comment for the step. The comment can contain, for example, notes or information on the function of the step.

You program the conditions in the "Interlock", "Supervision" and "Transition" panes. The instructions provided in the favorites area of the programming window change according to the programming language set for the block. To open or close a pane in the work area of the single step view, click on the small triangle on the left edge of the title bar of the pane you want.

For information on switching between single step view and sequence view, refer to "See also".

**"Interlock" pane**

In the "Interlock" pane, you program the interlock conditions in the LAD and FBD programming languages. The execution of the individual actions can be influenced by an interlock.

You can assign a name to the interlock in the title bar. To do this, click behind the colon in the title bar and enter the required name.

A network with an interlock symbol "-(C)-" appears in the work area of the open "Interlock" pane. If you click on this area of the network an input field for a step-specific interlock alarm text, which can include up to 150 characters, appears in the inspector window on the "Properties" tab.

For information on interlocks, refer to "See also".

**"Supervision" pane**

In the "Supervision" pane, you program the monitoring conditions for the opened step in the LAD and FBD programming languages. The execution times for actions or steps, for example, are monitored by this.

You can assign a name to the supervision in the title bar. To do this, click behind the colon in the title bar and enter the required name.

A network with supervision symbol "-(V)-" at the end appears in the work area of the opened "Supervision" pane. If you click on this area of the network an input field for a step-specific supervision alarm text, which can include up to 150 characters, appears in the inspector window on the "Properties" tab.

For information on supervisions, refer to "See also".

**"Actions" pane**

When you open a step the "Actions" pane is opened by default in the work area. You carry out the programming in the action table by entering all instructions of the selected step here.

For information on actions, refer to "See also".

**"Transition" pane**

In the transition panes, you program the step enabling conditions in the LAD and FBD programming languages. When the transition is satisfied the sequencer switches to the next step.

You can enter in the title bar a title for the step enabling condition of the selected transition. To do this, click behind the colon in the title bar and enter the required name.

For information on transitions, refer to "See also".
### Action table

#### Overview

You program the step contents in the action table of the "Single step" view. To get to the action table, double-click a step in the sequence view.

The following table shows the information that can be recorded for each action in the action table of the single step view.

<table>
<thead>
<tr>
<th>Information</th>
<th>Representation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interlock</td>
<td>&lt;No entry&gt;</td>
<td>No condition</td>
</tr>
<tr>
<td></td>
<td>-(C)-</td>
<td>Interlock</td>
</tr>
<tr>
<td>Event</td>
<td></td>
<td>No event</td>
</tr>
<tr>
<td></td>
<td>CC 1</td>
<td>Alarm acknowledgement</td>
</tr>
<tr>
<td></td>
<td>L0</td>
<td>Incoming interlock condition</td>
</tr>
<tr>
<td></td>
<td>L1</td>
<td>Outgoing interlock condition</td>
</tr>
<tr>
<td></td>
<td>R1</td>
<td>Incoming registration</td>
</tr>
<tr>
<td></td>
<td>S0</td>
<td>Outgoing step</td>
</tr>
<tr>
<td></td>
<td>S1</td>
<td>Incoming step</td>
</tr>
<tr>
<td></td>
<td>V0</td>
<td>Outgoing monitoring error</td>
</tr>
<tr>
<td></td>
<td>V1</td>
<td>Incoming monitoring error</td>
</tr>
<tr>
<td>Identifier</td>
<td></td>
<td>No instruction</td>
</tr>
<tr>
<td></td>
<td>CD</td>
<td>Count down</td>
</tr>
<tr>
<td></td>
<td>CR</td>
<td>Reset counter</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>Set counter value</td>
</tr>
<tr>
<td></td>
<td>CU</td>
<td>Count up</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>On delay</td>
</tr>
<tr>
<td></td>
<td>L</td>
<td>Set for limited time</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>Set as long as step is active</td>
</tr>
<tr>
<td></td>
<td>ON</td>
<td>Activate step</td>
</tr>
<tr>
<td></td>
<td>OFF</td>
<td>Deactivate step</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>Set to 0</td>
</tr>
<tr>
<td></td>
<td>S</td>
<td>Set to 1</td>
</tr>
<tr>
<td></td>
<td>TD</td>
<td>Retentive on-delay</td>
</tr>
<tr>
<td></td>
<td>TF</td>
<td>Turn off timer</td>
</tr>
<tr>
<td></td>
<td>TL</td>
<td>Extended pulse</td>
</tr>
<tr>
<td></td>
<td>TR</td>
<td>Hold timer and reset</td>
</tr>
<tr>
<td>Action</td>
<td>Designation</td>
<td>Action instructions</td>
</tr>
</tbody>
</table>

You can select a specific action in the action table by clicking on the row of the required action in the first column of the table.

You can use the shortcut menu to show additional descriptions for events and identities.

The symbolic and absolute addresses are displayed in the last column of the action table. You can move this table in the "Action" column.

### Permanent Instructions

You program permanent instructions in the views of the "Permanent pre-instructions" and "Permanent post-instructions" panes. A GRAPH-FB can contain a total of 250 permanent pre-instruction and 250 permanent post-instruction networks. Each network is hereby automatically provided with a number. The permanent instructions are processed once per cycle regardless of the state of the sequencer.
Navigating in the programming window

Requirement
A GRAPH function block is open.

Procedure
To navigate in the programming window, follow these steps:

1. In the navigation view, click the required pane on the toolbar.
   The pane is opened and the structures existing in the pane are displayed in the navigation view.

2. Click on the desired element to show it in the work area.
   This applies to both the sequence view and the single step view, or the permanent instructions.
   Two switch between the sequence view and the single step view in the work area, double-click on the desired sequence
   element in the navigation view.

Result
The selected element is displayed in the work area.

Switching between single step and sequence view

Requirement
A block is open.

Procedure
To switch between single step view and sequence view, follow these steps:

1. On the toolbar, click the "Single step view" or "Sequence view" icon.
Or:

1. Double-click on an new step in the navigation.
   If the sequence view is open, the program switches to single step view.
   If the single step view is open, the program switches to sequence view.

Result
The view is changed.

Inspector window

By default, the inspector window is located below the navigation and the work area. For example, information on block properties is
displayed here.

You can make several settings for the block or for the sequencer in the inspector window.

If you have selected a block, the following block information is displayed, for example:

- Name, type, number, language
- Specified network language
- Attributes which can be set, for example:
  - ICE check of the block
Creating a minimized DB during compilation

- Specifying the sequence properties

"Sequence control" pane

The "Sequence control" pane is located on the "Testing" task card. Here you can make settings that specify the control of the sequencer during testing.

For detailed information on the testing of sequencers and the use of the "Sequence control" pane, refer to "See also".

"Test settings" pane

The "Test settings" pane is located on the "Testing" task card. Here you specify the basic behavior of the sequencer in test mode.

For detailed information on the test settings, refer to "See also".

Keyboard shortcuts in GRAPH

In the LAD and FBD networks, default keyboard shortcuts are available for editing LAD and FBD. For information on the standard keyboard shortcuts, refer to "See also".

You can also use the following keyboard shortcuts in GRAPH:

### Keyboard shortcuts in the navigation

<table>
<thead>
<tr>
<th>Function</th>
<th>Keyboard shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Up/Down</td>
<td>&lt;Page Up&gt;/</td>
</tr>
<tr>
<td>Navigating in the navigation</td>
<td>&lt;Arrow Up&gt;/</td>
</tr>
<tr>
<td>Expand object</td>
<td>&lt;+&gt; or &lt;Arrow Right&gt;</td>
</tr>
<tr>
<td>Collapse object</td>
<td>&lt;-&gt; or &lt;Arrow Left&gt;</td>
</tr>
<tr>
<td>Switching between single step view and sequence view when a step or a transition is selected</td>
<td>&lt;Enter&gt;</td>
</tr>
<tr>
<td>Switching between navigation and work area</td>
<td>&lt;ALT+F6&gt;</td>
</tr>
</tbody>
</table>

### Keyboard shortcuts in single step view

<table>
<thead>
<tr>
<th>Function</th>
<th>Keyboard shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Up/Down</td>
<td>&lt;Page Up&gt;/</td>
</tr>
<tr>
<td>Go to first element in a network</td>
<td>&lt;Home&gt;</td>
</tr>
<tr>
<td>Go to last element in a network</td>
<td>&lt;End&gt;</td>
</tr>
<tr>
<td>Switching between navigation and work area</td>
<td>&lt;ALT+F6&gt;</td>
</tr>
</tbody>
</table>

### Keyboard shortcuts in the sequence view

<table>
<thead>
<tr>
<th>Function</th>
<th>Keyboard shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Creating GRAPH programs (S7-300/400)

<table>
<thead>
<tr>
<th>Function</th>
<th>Keyboard shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Up/Down</td>
<td>&lt;Page Up&gt;/</td>
</tr>
<tr>
<td></td>
<td>&lt;Page Down&gt;</td>
</tr>
<tr>
<td>Navigating in the structure</td>
<td>Arrow keys</td>
</tr>
<tr>
<td>Go to first step</td>
<td>&lt;Home&gt; or &lt;Ctrl+Home&gt;</td>
</tr>
<tr>
<td>Go to last step</td>
<td>&lt;End&gt; or &lt;Ctrl+End&gt;</td>
</tr>
<tr>
<td>Open branch</td>
<td>&lt;Shift+F9&gt;</td>
</tr>
<tr>
<td>Close branch</td>
<td>&lt;Shift+F11&gt;</td>
</tr>
<tr>
<td>Insert sequence end</td>
<td>&lt;Shift+F7&gt;</td>
</tr>
<tr>
<td>Define jump target</td>
<td>&lt;Shift+F12&gt;</td>
</tr>
<tr>
<td>Insert step and transition</td>
<td>&lt;Shift+F5&gt;</td>
</tr>
<tr>
<td>Deleting elements</td>
<td>&lt;Del&gt;</td>
</tr>
<tr>
<td>Switching between navigation and work area</td>
<td>&lt;ALT+F6&gt;</td>
</tr>
</tbody>
</table>

**Keyboard shortcuts in permanent instructions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Keyboard shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Up/Down</td>
<td>&lt;Page Up&gt;/</td>
</tr>
<tr>
<td></td>
<td>&lt;Page Down&gt;</td>
</tr>
<tr>
<td>To the first editable element</td>
<td>&lt;Home&gt;</td>
</tr>
<tr>
<td>Go to next editable element</td>
<td>&lt;Tab&gt;</td>
</tr>
<tr>
<td>To the last editable element</td>
<td>&lt;End&gt;</td>
</tr>
<tr>
<td>Go to previous editable element</td>
<td>&lt;Shift+Tab&gt;</td>
</tr>
<tr>
<td>Switching between navigation and work area</td>
<td>&lt;ALT+F6&gt;</td>
</tr>
</tbody>
</table>

**Keyboard shortcuts in action tables**

<table>
<thead>
<tr>
<th>Function</th>
<th>Keyboard shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jump to the cell beginning (Cell &quot;Action&quot;)</td>
<td>&lt;Home&gt;</td>
</tr>
<tr>
<td>Jump to the cell end (Cell &quot;Action&quot;)</td>
<td>&lt;End&gt;</td>
</tr>
<tr>
<td>Insert new action</td>
<td>&lt;Enter&gt;</td>
</tr>
</tbody>
</table>

**Program sequencer**

This chapter contains the following information:

- Sequence (S7-300/400)
- Steps and transitions (S7-300/400)
- Branches (S7-300/400)
- Jumps (S7-300/400)
- Using favorites (S7-300/400)
Sequence

This chapter contains the following information:
- Overview of sequences (S7-300/400)
- Create new sequence (S7-300/400)
- Inserting sequence elements (S7-300/400)
- Rename sequence (S7-300/400)
- Comment on sequence (S7-300/400)
- End sequence (S7-300/400)

Overview of sequences

Overview
A GRAPH-FB can consist of one or more sequencers of various lengths and complexities. The sequencers hereby behave as follows:
- A sequencer can be processed independently of other sequencers and parallel to these.
- A sequence can be used with a jump as continuation or as branch of a sequencer.
  This allows you to display technically connected functions in one block to have a better overview of the program.
- In each sequencer an initial step can mark the first active step.

Elements of a sequencer
A sequencer consists of the following elements:
- An initial step
- A linear sequence of changing steps and transitions
- One or more alternative branches that can contain the following elements:
  - Alternative branches
  - Transitions
  - A linear sequence of steps and transitions, empty steps and empty transitions are also possible
  - Branch end
- One or more simultaneous branches that can contain the following elements:
  - Simultaneous branch
  - A linear sequence of steps and transitions, empty steps and empty transitions are also possible
  - Branch end
- Jumps to other steps
- Jumps to other sequences
- Sequence end
- Comment

The following table shows the symbols of the sequence elements in the programming window:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>🔯</td>
<td>Step and transition</td>
</tr>
<tr>
<td>🔯</td>
<td>Step</td>
</tr>
<tr>
<td>🔯</td>
<td>Transition</td>
</tr>
<tr>
<td>🔯</td>
<td>Sequence end</td>
</tr>
</tbody>
</table>
When you create a sequencer in the GRAPH programming window, you insert the used sequence elements in the work area from top to bottom and from left to right.

**Processing principle of a sequencer**

The processing of a sequencer always begins with one or more initial steps that can be located at any point of the sequence. As long as the step is active, its actions are executed in each cycle. If several steps are executed at the same time, all these steps are active.

An active step is exited if a possible pending supervision error was eliminated or confirmed and the transition following the step is satisfied. A pending interlock error does not prevent the transition to the next step. The step which follows the satisfied transition then becomes active.

A jump to any step of this or another sequencer of the same function block can be located at the end of the sequencer. Jumps make possible a cyclic operation of a sequencer.

The sequence ends when the end of the sequence is reached.

---

**Create new sequence**

**Requirement**

A GRAPH function block is open.

**Procedure**

To create a new sequence, follow these steps:

1. In the navigation view, on the title bar of the "Sequence" pane or on the list of already existing sequences and select the "Insert sequence" command.
   A new sequence is created in the navigation.

   **Note**
   If you have selected the title bar in the navigation to insert a new sequence, the new sequence is inserted before the already existing sequence. If you have selected the title line of an existing sequence in the the title bar to insert a new sequence, the new sequence is inserted after the selected sequence.

2. Repeat this procedure until the "Sequence" pane contains the required number of sequences.

**Result**

The "Sequence" pane contains additional sequencers.

---

**Inserting sequence elements**

**Requirement**

A GRAPH function block is open.

**Procedure**

To create a sequence, select one of the following options:

1. Drag the required element from task card "Instructions" to the work area.
In the work area the possible insertion points in the sequence are highlighted by rectangles. The point at which the selected element is currently located is indicated by a green rectangle.

If you want to insert a sequence element that cannot be inserted at this point because of the existing sequence structure, no insertion points will be displayed.

2. Add the element at the required point of the sequence.
3. Repeat the procedure until your sequence is completed.

Or:

1. In the work area, select the sequence element on which you want to insert a new element.
2. Select the "Insert element > <Element name>" command from the shortcut menu.
   Elements that cannot be inserted at this position are disabled.
3. Repeat the procedure until your sequence is completed.

### Rename sequence

**Requirement**
A GRAPH function block is open.

**Procedure**
To rename a sequence, follow these steps:

1. In the sequence view, open the sequence that you want to rename.
2. Position the cursor in the edit field for the sequence title.
   A default "<new sequence>" placeholder is available.
3. Enter the desired sequence name.

### Comment on sequence

**Requirement**
A GRAPH function block is open.

**Procedure**
To comment on a sequence, follow these steps:

1. In the sequence view, open the sequence for which you want to add a comment.
2. Position the cursor in the edit field for the comment.
   A default "Comment" placeholder is available.
3. Enter the desired comment.

### End sequence

**Requirement**
A GRAPH function block is open.

**Procedure**
To create a sequence, select one of the following options:
1. Drag the "Sequence end" symbol from the task card "Instructions" to the desired point in your sequencer.

Or:

1. Select the point in your sequencer at which you want to insert the end of sequence.
2. Click on the "Sequence end" symbol in the favorites area.

Alternatively, you can select the "Insert element > Sequence end" command from the shortcut menu.

**Result**
The sequencer contains an end of sequence.

---

**Steps and transitions**

This chapter contains the following information:

- Overview of steps and transitions (S7-300/400)
- Inserting steps and transitions in pairs (S7-300/400)
- Deleting steps or transitions (S7-300/400)
- Copying steps and transitions (S7-300/400)
- Inserting steps and transitions individually (S7-300/400)
- Defining an initial step (S7-300/400)
- Renumbering steps and transitions (S7-300/400)
- Renaming steps and transitions (S7-300/400)
- Commenting on steps (S7-300/400)

---

**Overview of steps and transitions**

**Overview**

A step is the active part of a sequencer. The actions of the sequencer are defined in it. The actions are processed when a step becomes active. Here the previous transition must be fulfilled.

**Step**

An active step is a step whose actions are currently being processed.

A step becomes active if one of the following conditions is satisfied:

- The conditions of the transitions that precede the step are satisfied.
- A step is defined as initial step and the sequencer was initialized.
- A step is called by an event-dependent action.

**Empty step**

An empty step is a step that contains no programmed actions. It behaves like an active step. The transition that follows it is valid.

**Step components**

A step can contain the following elements:

- Actions
- Interlock (step interlock)
- Supervision (step monitoring)
- Step comment
- Step number
- Step name

For information on the individual step components, refer to "See also".
**Initial step**

In the sequencer, an initial step can be recognized by a double border. An initial step is the sequencer step that first becomes active when a GRAPH-FB is called for the first time without previously querying conditions. Any step can be selected as initial step in each sequence of the block. All simultaneous programmed steps can be initial steps.

In the cyclic operation of a sequencer the initial step is treated like every other step, provided there is no initialization of the sequencer. The sequencer is initialized by means of the INIT_SQ parameter. If there is a positive edge at this input, all steps that are identified as initial step are activated. All other steps are deactivated.

**Transitions**

A transition contains the conditions for switching to from one step to the next in a sequencer. A transition becomes valid, or is then evaluated, when its preceding steps are active. If the step enabling conditions of a valid transition are satisfied, it switches to the next step. The step or steps belonging to the transition are hereby deactivated and the follow-on step activated.

A transition is displayed and programmed in the single step view.

The transition switches to the next step of the sequencer when the logic operation of the condition is satisfied, i.e. when the network returns the result 1. The following step of the transition becomes active.

The transition does not switch to the next step of the sequencer if the logic operation of the condition is not satisfied, i.e. if the network returns the result "0". The active step continues to remain active.

**Step/transition pair**

A new GRAPH FB is already provided with a step/transition pair onto which you can attach additional step/transition pairs. Numbers are automatically assigned to the steps and transitions when step/transition pairs are inserted. For a clearer overview you can later renumber the steps and transitions in certain areas to maintain a continuous numbering.

**Note**

A maximum of 250 steps and 250 transitions can be used in a GRAPH function block.

**Note**

Avoid jumps from a transition to a directly preceding step. To make this jump instruction possible, an empty step including a transition without conditions has to be inserted.

---

**Inserting steps and transitions in pairs**

**Requirement**

A GRAPH function block is open.

**Procedure**

To insert a step and a transition as a pair, follow these steps:

1. In the sequencer, select the point at which you want to insert a step with transition.
2. Click the "Step and transition" symbol in the task card "Instructions".
   
   Alternatively, you can insert a step with transition by dragging the symbol to the desired point in the sequence. In this case the possible connection points are indicated by rectangles. The currently active connection point is symbolized by a green rectangle.

**Result**

The sequencer contains a step with transition.

---

**Deleting steps or transitions**

**Requirement**

A GRAPH function block is open in the sequence view.
A sequence with at least two steps is available.

**Procedure**

To delete a step or a transition, follow these steps:

1. In the sequencer, select the step or transition that you want to delete.
   
   The selected element is highlighted.
2. Select the "Delete" command in the shortcut menu.
   
   When you delete a sequence element that is followed by additional elements in the sequence and for whose logic this element is necessary, GRAPH detects the corresponding inconsistencies and highlights these in red lettering.

**Result**

The selected sequence element is deleted.

**Note**

**Plausibility check of the sequence**

If a step or a transition is deleted at the end of the sequence, the sequence is logical and consistent. No error is indicated.

If a step or a transition is deleted within the sequence, the sequencer is inconsistent and an error is indicated in the form "Missing Transition" or "Missing Step".

To eliminate the inconsistency, a individual step or a transition can be inserted at the place in which the error occurs. Alternatively, you can delete the step or transition element corresponding to the occurring error to eliminate the inconsistency.

---

**Copying steps and transitions**

**Requirement**

A sequencer with a step and a transition is available.

**Procedure**

To copy a step or a transition, follow these steps:

1. Select the desired element in the sequencer.
   
   The selection is highlighted in blue.
2. In the shortcut menu, select the "Copy" command.
   
   You can also use the key combination <Ctrl+C>.
3. Select the desired insertion point in the sequencer.
   
   You can insert the copied element into the currently opened sequence or any other sequencer of the opened or other block.
4. Select the "Paste" command in the shortcut menu.
   
   You can also use the key combination <Ctrl+V>.

**Result**

A step or a transition was copied.

---

**Inserting steps and transitions individually**

You can insert a step or a transition individually at the end of a branch or at points of the sequencer at which a step or transition was deleted.

**Requirement**

A GRAPH function block is open in the sequence view.
Procedure
To insert a step or a transition individually, follow these steps:

1. In the sequencer, select the point in the sequencer where you want to insert the step or transition.
2. In the favorites area, click the symbol of the element you want to insert. Alternatively, you can drag the desired element to the corresponding point in the sequence.

Result
The sequencer contains a step or a transition.

---

Defining an initial step

Requirement
A GRAPH function block is open in the sequence view.
A sequencer is available.

Procedure
To define a step as initial step, follow these steps:

1. In the work area, select the step that you want to define as initial step.

You can declare a maximum of 250 initial steps.

2. Select the "Initial step" check box in the shortcut menu.

You can define an already existing initial step as normal step of the sequencer by clearing the "Initial step" check box.

Result
An initial step is defined.

---

Renumbering steps and transitions

You can assign new numbers to steps and transitions of branches, sequences or complete blocks.

Renumbering steps and transitions in the work area
To renumber steps or transitions, follow these steps:

1. Open a sequencer with several steps in the sequence view.

2. In the work area, double-click in the upper area of the step symbol and select the automatically assigned step number. Proceed in the same manner for transitions.

The step and transition numbers assigned within a block must be unique.

3. Enter the desired number.

A plausibility check is performed automatically during the renumbering process. If the desired number has already been assigned, you have to enter a different number. Alternatively, you can change the number of the step that contains the desired number. The same applies to transitions.

Renumbering steps and transitions using the "Renumber" dialog.
To renumber step and transitions via the "Renumber" dialog, follow these steps:

1. Open a sequence in the sequence view.

2. Select a step or a transition of the sequence in the work area.
The selected element represents the starting point for the renumbering.

3. Select the "Renumber" command in the shortcut menu. The "Renumber" dialog opens.

4. Select the elements you want to renumber.

You can also renumber only steps or only transitions. To do this, deselect the respective other element.

5. Enter the new numbers for the elements, starting from the point at which the elements are to be consecutively renumbered.

6. Select the range in which the elements are to be renumbered.

7. Click "OK" to confirm.

**Result**

Steps or transitions of the selected area contain new numbers.

**Note**

The "Renumber" dialog is not available if the "Create minimized DB" option is enabled in the settings; in this case the step and transition numbers are assigned automatically.

---

### Renaming steps and transitions

**Requirement**

A sequencer with several steps and transitions is open in the sequence view.

**Procedure**

To rename steps or transitions, follow these steps:

1. In the work area, double-click in the lower area of the step symbol and select the automatically assigned step number "Step". Follow the same steps for the transitions ("Trans").

The step and transition names assigned within a block must be unique.

2. Enter the desired name.

   A plausibility check is performed automatically during the renaming process. If the desired name has already been assigned or is too long, you have to enter a different name. Alternatively, you can change the name of the step that contains the desired name.

   The same applies to transitions.

**Result**

The step or the transition contains a new name.

---

### Commenting on steps

**Requirement**

A step is open in the single step view.

**Procedure**

To comment on a step, follow these steps:

1. Select the step you want to comment on.

2. Enter the desired text in the comment line under the step title bar.

---

### Branches
This chapter contains the following information:

- Overview of branches (S7-300/400)
- Create alternative branches (S7-300/400)
- Create simultaneous branch (S7-300/400)
- Closing branches (S7-300/400)
- Moving branches (S7-300/400)
- Deleting branches (S7-300/400)

Overview of branches

Alternative branch

An alternative branch is an OR branch in the sequencer and consists of several parallel branches (max. 125). Alternative branches begin with a transition in each case. If multiple alternative branches are available, only the alternative branch is run through, whose transition switches first. If several transitions are satisfied simultaneously at the start of different branches, the transition farthest to the left has the highest priority in each case.

Simultaneous branch

A simultaneous branch is an AND branch and consists of several parallel branches (max. 249), each of which starts with a step. If the transition switches to the next step before the simultaneous branch, all first step in all follow-up simultaneous branches become active at the same time. Each simultaneous branch ends with a step and is closed with a follow-on transition. If multiple simultaneous branches are combined to a transition, this transition only switches to the next step if all active simultaneous branches have been completely processed.
Create alternative branches

Requirement
A function block is opened for the processing in the GRAPH language.

Procedure
To create an alternative branch, follow these steps:

1. Drag the "Open alternative branch" instruction from the "Favorites" task card to the desired point behind a step in the sequencer.
2. Complete the alternative branch with the desired sequence elements.
3. Drag the "Close branch" instruction to the planned sequence end.
   You can also drag the arrow at the end of the simultaneous branch with Drag & Drop to the desired connection point.

Result
An alternative branch is created on the right next to a sequence or an already existing branch.

Create simultaneous branch

Requirement
A function block is opened for the processing in the GRAPH language.

Procedure
To create a simultaneous branch, follow these steps:

1. Drag the "Open simultaneous branch" instruction from the "Favorites" task card to the desired point behind a transition in the sequencer.
2. Complete the simultaneous branch with the desired sequence elements.
3. Drag the "Close branch" instruction to the planned sequence end.
   You can also drag the arrow at the end of the simultaneous branch to the desired connection point.
Result
A simultaneous branch is created on the right next to a sequence or an already existing branch.

Closing branches

Each alternative branch ends with a transition and can be closed with a link to the higher level sequence and follow-on step, a sequence end or a jump.

Each simultaneous branch ends with a step and is closed with a follow-on transition. When you close the simultaneous branch the simultaneous branches synchronize automatically. All branches are hereby combined to a transition. This applies if the processing of all branches have been completed.

Requirement
A sequencer with a branch is available.

Procedure
To close a branch, follow these steps:

1. In the sequencer, select the point at which you want to close the branch.
2. From the favorites, select the element "Close branch" or "Sequence end".

Or:

1. Drag one of the elements "Close branch" or "Sequence end" to the end of the branch.

Or:

1. You can also drag the arrow at the end of the branch to the desired connection point.

Result
The branch is closed.

Moving branches

Requirement
A sequencer with a branch is available.

Moving branch elements within a branch
To move branch elements within a branch, follow these steps:

1. If the branch ends with a sequence end or a jump, drag the "Open alternative branch" or "Open simultaneous branch" sequence element to the desired position of the sequence at which you want the branch to start.

Or:

1. If the branch ends with the "Close branch" sequence element on an existing branch to its left, drag the "Close branch" sequence element in the same branch to the position at which you want the branch to end.

The possible connection points are highlighted by rectangles.

2. If necessary, then drag the "Open alternative branch" or "Open simultaneous branch" sequence element in the same branch to the position at which you want the branch to start.

Moving branch elements between various branches
To move branch elements between various branches, follow these steps:
1. Select the "Close branch" sequence element and delete it.
2. Drag the "Open alternative branch" or "Open simultaneous branch" sequence element to the position of the sequencer at which you want the branch to start.
3. Insert the "Close branch" sequence element at the desired position in the sequencer.

---

**Deleting branches**

**Requirement**
A sequencer with a branch is available.

**Procedure**
To delete a branch, follow these steps:

1. In the sequencer, select the branch that you want to delete.
   The selected components are highlighted in blue.
2. Select the "Delete" command in the shortcut menu.
   You can also use the key <Del>.

---

**Jumps**

This chapter contains the following information:
- Overview of jumps (S7-300/400)
- Jump target (S7-300/400)
- Inserting jumps (S7-300/400)

---

**Overview of jumps**

Jumps make possible the transition between a transition and a step that are not graphically connected with each other. A jump can take place within a sequence or to another sequence in the same FB. A jump is always positioned after a transition and ends a branch at this point of the sequencer of the branch.

Jump and jump target are represented graphically as arrows. The actual connection is not visible in GRAPH.

Jumps make a cyclic processing possible, i.e. the repeated processing of sequence parts or the entire sequencer.

---

**Jump target**

The target of a jump is represented by an arrow that has the name of the transition through which the jump is triggered. Only the absolute name of the transition is used for this. The symbolic name is not displayed.

A jump target is always positioned directly before the step that is intended to introduce the jump. There is no transition between jump target and step.

---

**Inserting jumps**

**Requirement**
A sequencer is available.

**Procedure**
To insert a jump in a sequencer, follow these steps:

1. Select the desired starting point of the jump in the sequencer or in the branch.
   The open connection point of the sequence is represented by an arrow with two tips.
2. In favorites, click the "Jump" instruction.
   Alternatively, you can select the "Insert element > Jump" command from the shortcut menu.
   At the starting point of the jump a table opens showing the absolute and symbolic names of all steps contained in the block.
3. In the table, select the step to which the jump should lead.
   Alternatively, you can enter the step number manually in the entry field and press the Enter key.

### Result
A jump is available.

---

### Using favorites

This chapter contains the following information:
- Creating favorites (S7-300/400)
- Removing favorites (S7-300/400)

---

### Creating favorites

Three different types of favorites are available in the GRAPH editor.
- The favorites in the sequence view already contain all graphic elements that can be used in this view.
- The favorites for permanent instructions can be customized.
- The favorites in the single step view contain adjustable instructions for LAD and FBD networks and the action table.

The corresponding favorites are loaded automatically when you switch between views.

### Requirement
The sequence view, single step view or the permanent instructions are displayed in the work area.

### Procedure
To create favorites, follow these steps:

1. Open the "Instructions" task card.
2. Expand the "Favorites" pane.
3. Open the view whose favorites you want to expand.
4. Drag the desired instruction from the "Basic instructions" pane to the "Favorites" pane.
   Alternatively you can select the "Copy" command from the shortcut menu of the desired instruction and add using the "Insert" command from the shortcut menu of the "Favorites" pane.

**Note**
The favorites are only shown in the work area if the "Show favorites in editor" option is active in the shortcut menu of the "Favorites" pane.

### Result
The "Favorites" pane contains the inserted instruction.

---

### Removing favorites
Requirement
A GRAPH function block is open.

Procedure
To remove favorites, follow these steps:

1. Open the view whose favorites you want to remove.
2. In the Favorites bar or on the "Instructions" task card in the "Favorites" pane, right click on the instruction that you want to remove.
3. Select the "Remove instruction" command in the shortcut menu.
   You can also use the key <Del>.

Program actions and conditions

This chapter contains the following information:
- Programming actions (S7-300/400)
- Programming conditions (S7-300/400)

Programming actions

This chapter contains the following information:
- Overview of actions (S7-300/400)
- Entering actions (S7-300/400)
- Inserting a value assignment in actions (S7-300/400)
- Inserting block calls (S7-300/400)
- Copying, moving and deleting actions (S7-300/400)
- Standard actions (S7-300/400)
- Events (S7-300/400)
- Event-dependent actions (S7-300/400)
- Counters in actions (S7-300/400)
- Timers in actions (S7-300/400)
- Summary of all actions (S7-300/400)

Overview of actions

Overview
You program the actions of a step of the sequencer in the single step view. The actions describe the instructions for the process control. An action modifies, for example, a tag or calls a block. Programmed actions are processed when a step becomes active. Then they are executed in sequence from "top to bottom".

You have the option of entering a line comment for actions. Line comments start with the character sequence "//".

You can, for example, program the following instructions in an action:
- Value assignments
- Timer operations
- Counter operations
- Block calls
- Activating/deactivating steps of the sequencer
The following figure shows an example of the order in which actions are processed in a step:

Components of an action
An action consists of the following elements:
- Interlock (optional)
- Event (optional)
- Action identifier
- Operand, assignment or block call

Assignment of actions
Actions are differentiated as follows:
- Standard actions
- Event-dependent actions

Interlock dependence
All standard actions and event-dependent actions can be linked with an interlock condition.

Behavior in multi-line actions
You can insert additional lines in actions. You can then, for example, enter comments or adapt existing block calls without deleting the call, by adding new interface parameters.

To allow multi-line entry, the "Allow multi-line mode" command must be selected in the shortcut menu of the action table. A new line is then inserted within the selected action when you press the enter button.

For information on handling multi-line actions, refer to "See also".

Entering actions

Requirement
A step is open in the single step view.

Procedure
To enter an action in the action table, follow these steps:

1. In the "Identifier" column, select the desired action type from the drop-down list box.
   Predefined structures with placeholders are created in the "Action" column, except with "N". You can tie the action to an event in the "Event" column. Some action types must have an event.
2. In the "Interlock" column you have the option of connecting the action with an interlock condition.

3. In the "Action" column, replace the placeholder with the desired operands and values or enter the action manually.
   - You can drag the required operands from any position to the "Action" column, provided the operands are already available.
   - You can drag instructions from the "Instructions" task card or the Favorites to the "Action" column.
   - You can drag block calls from the project tree to the "Action" column.
   - Alternatively, you can select the operands from the auto-complete list. This is displayed when you select placeholders and enter the first character.

4. Press the enter key to confirm the action or click on the next line.

**Result**
A new action is created.

**Note**
A plausibility check is performed during the creation of actions. If you make an incorrect entry, a red rollout with a short error description appears in the input area. In the Inspector window, a detailed description of the error is displayed in the "Info > Syntax" tab.

**Entering actions in multiple lines**
You can insert additional lines in an action to provide a better overview of actions. Here, for example, you can add comments or adapt existing block calls without deleting these.

**Requirement**
A step is open in the single step view.

**Procedure**
To enter an action in multiple lines, follow these steps:

1. Make an entry in an action.
2. Position the cursor behind the instruction of the block call.
3. Select the "Allow multi-line mode" command from the shortcut menu.
4. Press the enter key.
   - Another line is created in the same action.
5. Enter the desired comment or adapt the block call.
6. Press the enter key to confirm or click on the next line.

**Result**
The action contains several lines.

**Note**
A plausibility check is performed during the creation of actions. If you make an incorrect entry, a red rollout with a short error description appears in the input area. In the Inspector window, a detailed description of the error is displayed in the "Info > Syntax" tab.

**Editing actions**

**Requirement**
A step is open in the single step view.
Procedure
To edit an action, follow these steps:

1. Position the cursor at the point of the action that you want to edit.
2. Edit the action. You can modify, extend or delete actions. For actions that require the specification of a time constant or a counter value, enter a comma after the operand and then the time, time constant or the counter value.
3. Press the enter key to confirm the changes.

Result
An action is changed.

Note
A plausibility check is performed during the creation of actions. If you make an incorrect entry, a red rollout with a short error description appears in the input area. In the Inspector window, a detailed description of the error is displayed in the "Info > Syntax" tab.

Note
To change the data type of an action, click on the small green square on the right below the entered action and use this to select the desired data type.

Inserting a value assignment in actions

Overview
Actions can contain value assignments. The following different types of value assignments are available:

- Direct assignment
  The value of the operand is assigned directly to another operand.
- Assignment with operator
  The value of a mathematical expression is assigned to an operand.
- Assignment with basic instructions
  The result of a basic instruction is assigned to the operand.

Inserting direct assignments
Direct assignments are entered using the syntax A:=B. The data type of both operands must be compatible. For more information on compatibility and conversion of data types, refer to "See also".

Requirement
A step is open in the single step view.

Procedure
To enter a direct assignment in the action table, follow these steps:

1. Enter the basic assignment in the form <Operand_A> := <Operand_B>. Drag the operands from any position to the desired position in the "Action" column. Alternatively, you can select the operands from the auto-complete list that appears as soon as you enter the first character or press the key combination <CTRL+I>.
2. Press the enter key to confirm the entered action or click on the next line.

Result
The second operand is assigned to the first.

Inserting assignments with operator
Assignments with operator are entered using the syntax "A:=B<Operator>C". Operand A specifies the data type of the expression.

The following table shows the operators and data types that you can use:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Assignment</th>
<th>1st operand</th>
<th>2. Operand</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition</td>
<td>A:=B+C</td>
<td>Integer, floating-point number</td>
<td>Integer, floating-point number</td>
<td>Integer, floating-point number</td>
</tr>
<tr>
<td>Subtraction</td>
<td>A:=B-C</td>
<td>Integer, floating-point number</td>
<td>Integer, floating-point number</td>
<td>Integer, floating-point number</td>
</tr>
<tr>
<td>Multiplication</td>
<td>A:=B*C</td>
<td>Integer, floating-point number</td>
<td>Integer, floating-point number</td>
<td>Integer, floating-point number</td>
</tr>
<tr>
<td>Division</td>
<td>A:=B/C</td>
<td>Integer, floating-point number</td>
<td>Integer, floating-point number</td>
<td>Integer, floating-point number</td>
</tr>
<tr>
<td>Modulo function</td>
<td>A:=B % C</td>
<td>DINT</td>
<td>DINT</td>
<td>DINT</td>
</tr>
<tr>
<td>AND operation</td>
<td>A:=B AND C</td>
<td>Binary number</td>
<td>Binary number</td>
<td>Binary number</td>
</tr>
<tr>
<td>OR operation</td>
<td>A:=B OR C</td>
<td>Binary number</td>
<td>Binary number</td>
<td>Binary number</td>
</tr>
<tr>
<td>EXCLUSIVE OR</td>
<td>A:=B XOR C</td>
<td>Binary number</td>
<td>Binary number</td>
<td>Binary number</td>
</tr>
</tbody>
</table>

**Requirement**

A step is open in the single step view.

**Procedure**

To insert an assignment with operator, follow these steps:

1. Enter a basic assignment in the form <Operand_A>:=<Operand_B> <Operator> <Operand_C>.
   
   Drag the operands from any position to the desired position in the "Action" column.
   
   Alternatively, you can select the operands from the auto-complete list that appears as soon as you enter the first character or press the key combination <CTRL+I>.

2. Press the enter key to confirm the entered action or click on the next line.

**Result**

The result of the calculation is assigned to the first operand.

**Inserting assignment with basic instructions**

Basic instructions are entered using the syntax "A:=func(B)". You can find the available instructions in the "Instructions" task card.

**Requirement**

A step is open in the single step view.

**Procedure**

To assign a basic instruction to an action, follow these steps:

1. Enter the assignment in the form <Operand_A>:=<Instruction>(<Operand_B>).
   
   Some instructions require the entry of a data type. Select the data type via the small square that appears after the entry of the instruction and a blank character.

2. You can now drag the operands from any position to the placeholder in the "Action" column. You can also select the operands from the auto-complete list that appears when you mark the placeholder and enter the first character, or enter the operand manually.

3. Press the Enter key to complete the insertion of the action.
Result
The result of the calculation is assigned to the first operand.

Inserting block calls

This chapter contains the following information:
- Principles of block calls (S7-300/400)
- Manually inserting block calls (S7-300/400)
- Inserting block calls with drag-and-drop (S7-300/400)
- Update block calls (S7-300/400)

Principles of block calls

This chapter contains the following information:
- Calling function blocks (S7-300/400)
- Calling functions (S7-300/400)
- Call example of a function block (S7-300/400)
- Call example of a function (S7-300/400)

Calling function blocks

Calling as a single instance
In GRAPH, function blocks can only be called as single instance.

Syntax for call of function block
The following syntax to call a function block:
CALL <FBName>, <DBName> (List of parameters)

Calling functions

Overview
You can call functions from within an action. Functions perform calculations and then return a result. They do not require any memory. Therefore, they can be started directly from within a step without an instance data block.

Syntax for call of a function
The following syntax to call a function:
CALL <FCName> (List of parameters)

Call example of a function block

The function block "FB_KOP" is called using the CALL command. The "CALL" instruction is always connected with the "N" identifier. Behind the symbolic name of the called block there is the symbolic name of the data block containing the data and parameters of the function block. In the brackets after the data block, the "MyTag2" tag with the absolute name "%M0.1" is assigned to the "MyInOut" parameter. The column behind the action contains the symbolic and absolute names of the function block and tags. If you want to hide this column, click on "Absolute/symbolic operands" on the toolbar.
Call example of a function

The function "FC_KOP" is called using the CALL command. In the brackets after the function name, the "MyTag1" tag with the absolute name "%M0.0" is assigned to the "MyInput" parameter. The column behind the action contains the symbolic name "FC_KOP" and the absolute name "%FC2" of the function. If you want to hide this column, click on "Absolute/symbolic operands" on the toolbar.

Manually inserting block calls

Requirement
A step is open in the single step view.

Procedure
To insert a block call manually in an action, follow these steps:

1. Position the cursor in the "Action" column of the action table.
2. Enter the "Call" command and the name of the block to be called and press the Enter key.
   The "Call options" dialog opens.
3. Select an available data block from the drop-down list of the dialog or enter a data block name if you want to create a new one.
4. Select "Manual" if you want to assign a number to the data block itself.
   By default, GRAPH automatically assigns a consecutive number.
5. Confirm your entry with "OK".

Result
The "Action" column of the action table contains the call of the function block.

Note
To change the data type of an action, click on the small green square on the right below the entered action and use this to select the desired data type.

Inserting block calls with drag-and-drop

Requirement
A step is open in the single step view.

Procedure
To insert a block call with drag-and-drop in an action, follow these steps:

1. Drag the function block from the project tree and drop it into the program.
   The "Call options" dialog box will open.
2. Select an available data block from the drop-down list of the dialog or enter a data block name if you want to create a new one.
3. Select "Manual" if you want to assign a number to the data block itself.
   By default, GRAPH automatically assigns a consecutive number.

**Result**
The "Action" column of the action table contains the call of the function block.

**Note**
To change the data type of an action, click on the small green square on the right below the entered action and use this to select the desired data type.

---

**Update block calls**

If the step parameters of a block change, the block call can under certain circumstances no longer be executed correctly. To prevent this happening, you can update block calls.

You have two options to update the block calls:

- You can update block calls explicitly in the program editor via the "Update inconsistent block calls" button.
- You can implicitly update block calls by compilation. To do this, close the block and select the "Program blocks" folder in the project tree. Then select the command "Compilation > Software (compile blocks completely)" in the shortcut menu.

**Requirement**
A block is available.

**Updating blocks in the program editor**

To update block call in the program editor, follow these steps:

1. Open the desired block in the program editor.
2. Click on the "Update inconsistent block calls" icon on the toolbar.

---

**Update block calls during compilation**

Follow these steps to update all block calls and uses of PLC data types during compilation implicitly:

1. Open the project tree.
2. Select the "Program blocks" folder.
3. Select the command "Compilation > Software (compile blocks completely)" in the shortcut menu.

**Result**
The block calls are updated.

---

**Copying, moving and deleting actions**

**Requirement**
A step is open in the single-step view.

**Copying an action**
To copy an action, follow these steps:
1. In the first column of the action table, select the action that you want to copy.
2. Select the "Copy" command from the shortcut menu.
3. Position the cursor where you want to insert the copy.
4. Select the "Paste" command from the shortcut menu.

**Result**
A copy of the original action is inserted at the destination.

**Moving an action**
You can move actions in the action table.
To move an action, follow these steps:

1. In the first column of the action table, select the action that you want to move.
2. Drag the action in the first column of the action table to the desired destination.
   The content of the source and destination cell are exchanged when the mouse key is released.

**Result**
The selected action is moved to the destination line.

**Deleting an action**
If you want to delete an action, follow these steps:

1. In the first column of the action table, select the action that you want to delete.
2. Select the "Delete" command in the shortcut menu.
   You can also use the key <Del>.

**Result**
The selected action is deleted.

---

**Standard actions**

**Overview**
Standard actions are executed as long as the step is active.
All standard actions can be linked with interlock. The actions can then only be executed if the conditions of the interlock are satisfied.

The following table shows the possible standard actions:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Data type of the operand</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>N BOOL</td>
<td>Set as long as step is active: The operand has signal 1 as long as the step is active.</td>
<td></td>
</tr>
<tr>
<td>S BOOL</td>
<td>Set retentive: As soon as the step is active, the operand is set to 1 and remains subsequently at 1.</td>
<td></td>
</tr>
<tr>
<td>R BOOL</td>
<td>Reset retentive: As soon as the step is active, the operand is set to 0 and remains subsequently at 0.</td>
<td></td>
</tr>
<tr>
<td>D BOOL,</td>
<td>On delay:</td>
<td></td>
</tr>
</tbody>
</table>
For more information on data types and possible operands, refer to "See also."

**Timer value**

Actions that contain the action identifier D or L, require a time to be specified. Times are programmed as constants or as PLC tags of data type TIME/S5-TIME.

**Examples**

The following table shows some examples of time constants:

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>MyTag</td>
<td>As long as the step is active, the signal at MyTag is set to 1.</td>
</tr>
<tr>
<td>S</td>
<td>MyTag</td>
<td>As soon as the step is active, MyTag is set to 1 and remains subsequently at 1 (retentive).</td>
</tr>
<tr>
<td>R</td>
<td>MyTag</td>
<td>The signal state of MyTag is reset to 0.</td>
</tr>
<tr>
<td>D</td>
<td>&quot;MyTag&quot;, T#2S</td>
<td>2 seconds after step activation the signal at MyTag is set to 1 for the duration of the step activation. This does not apply if the duration of the step activation is shorter than 2 seconds. The signal is reset to 1 when step is deactivated (non-retentive).</td>
</tr>
<tr>
<td>L</td>
<td>&quot;MyTag&quot;, T#20s</td>
<td>If the step is active, the signal at MyTag is 1 for 20 seconds. After this, the signal is reset (non-retentive).</td>
</tr>
<tr>
<td>N</td>
<td>Call &quot;MyTag&quot;</td>
<td>The MyTag block is called as long as the step is active.</td>
</tr>
</tbody>
</table>

---

**Events**

**Overview**

An action can be linked with a result. An event is the change of the signal state of a step, a step monitoring or an interlock, or the acknowledgement of an alarm or an incoming registration.

If an action is linked with an event, the signal state of the event is detected by edge evaluation. This means that the instructions can only be executed in the cycle in which the event occurs.

The following table shows the possible events:

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step</td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td>Step becomes active</td>
</tr>
<tr>
<td>S0</td>
<td>Step is deactivated</td>
</tr>
<tr>
<td>Supervision</td>
<td></td>
</tr>
<tr>
<td>V1</td>
<td>Monitoring error occurs (fault)</td>
</tr>
</tbody>
</table>
Registration

A registration is an event that is excited outside of the block. The event is hereby excited via positive edge on one of the input parameters "REG_S" or "REG_EF". If the event is excited via the input parameter "REG_S", the event is only transferred to the active step that is displayed at the output parameter "S_NO". If the event is excited via the input parameter "REG_EF", the event is transferred to all currently active steps.

Examples

Step:

- S1: Step becomes active
- S0: Step is deactivated

Supervision:

- V1: Monitoring error occurs (fault)
- V0: Monitoring error is eliminated (no fault)

Interlock:

- L0: Incoming interlock condition (outgoing fault)
- L1: Outgoing interlock condition (incoming fault)
- C: Interlock condition is satisfied
Event-dependent actions

Overview

All actions, except for actions with the identifiers D, L and TF, can be linked with a result.

For more information on events, refer to “See also.”

Actions with event S1, V1, A1, R1 can be linked with interlock. The actions are only executed if the conditions of the interlock are satisfied.

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-(C)-</td>
<td>S1, V1, A1, R1</td>
<td>N, R, S</td>
<td>When the event arrives (and with optional pending interlock), the action is executed once in the next cycle.</td>
</tr>
<tr>
<td>Optional</td>
<td>S0, V0, L0, L1</td>
<td>N, R, S</td>
<td>When the event arrives the action is executed once in the next cycle.</td>
</tr>
</tbody>
</table>

Event-dependent actions for activating and deactivating steps

Other steps can be activated or deactivated using the ON and OFF instructions. The instructions are always dependent on a step event. The event determines the time of activation or deactivation.

Actions that are linked with the events S1, V1, A1 and R1 can also be linked optionally with an interlock.

The following table shows the GRAPH specific actions for activating and deactivating steps:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Operand</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>-(C)-</td>
<td>S1, V1, A1, R1</td>
<td>ON, OFF</td>
<td>Step name</td>
<td>When the event arrives (and with optional pending interlock), the step is activated (ON) or deactivated (OFF).</td>
</tr>
<tr>
<td>Optional</td>
<td>S0, V0, L0, L1</td>
<td>ON, OFF</td>
<td>Step name</td>
<td>When the event arrives the step is activated (ON) or deactivated (OFF).</td>
</tr>
<tr>
<td>-(C)-</td>
<td>S1, V1</td>
<td>OFF</td>
<td>S_ALL</td>
<td>When the event arrives (and with optional pending interlock), all steps are deactivated. Excluded is the step in which the action is contained.</td>
</tr>
<tr>
<td>Optional</td>
<td>L1</td>
<td>OFF</td>
<td>S_ALL</td>
<td>When the event arrives, all steps are deactivated. Excluded is the step in which the action is contained.</td>
</tr>
</tbody>
</table>

Note

If a step is both activated and deactivated in a cycle, the deactivation has higher priority.
**Example: Action table with event-dependent actions**

The following figure shows a step that contains event-dependent actions:

![Step diagram](image)

1. Interlock
2. Optional event
3. Identifier
4. Action

The following table describes the actions in the example:

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyTag1</td>
<td>As soon as the step is active and the interlock satisfied, MyTag1 is set to 0 and remains subsequently at 0.</td>
</tr>
<tr>
<td>MyTag</td>
<td>As soon as a monitoring error occurs, MyTag is deactivated.</td>
</tr>
<tr>
<td>S_ALL</td>
<td>As soon as the interlock condition is satisfied, all steps are deactivated.</td>
</tr>
</tbody>
</table>

---

**Counters in actions**

**Overview**

All counters in actions are always dependent on an event, in other words the event determines the time at which the counter is activated.

Only the events S1, V1, A1, R1 can be linked with interlock. The actions are then only executed if the conditions of the interlock are satisfied.

Actions that contain the action identifier CS require that a timer value be specified. The count value is programmed as tag or constant of data type WORD (C#0 to C#999).

**Counters in actions**

The following table shows the possible counters in actions.

<table>
<thead>
<tr>
<th>Event, Action identifier</th>
<th>Data type of the operand</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS &lt;Count value&gt;</td>
<td>COUNTER</td>
<td>Set counter value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>As soon as the event occurs, the counter is set to the specified count value.</td>
</tr>
<tr>
<td>CU</td>
<td>COUNTER</td>
<td>Count up:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>As soon as the event has occurred, the counter value is incremented by &quot;1&quot;. The count value is incremented until the high</td>
</tr>
</tbody>
</table>
Examples

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC 1</td>
<td>CD</td>
<td>Count down:</td>
<td>%Z1</td>
</tr>
<tr>
<td>CC 1</td>
<td>CS</td>
<td>- Set counter initial value</td>
<td>%Z1, 5</td>
</tr>
</tbody>
</table>

Timers in actions

Overview
Timers in actions are generally dependent on an event. The event determines the time when the timer will be activated. The only exception is the TF timer. This timer is not dependent on an event, but instead on the step itself.

Timers can be optionally combined with interlock. The timer operations combined with an interlock are only executed if the conditions of the interlock are satisfied in addition to the event.

Timers without interlock are executed when the event occurs.

Time duration
All actions that contain the operations TL, TD or TF, require the specification of a time duration. The time is programmed with the following syntax:

\(<\text{Time}>\) = EWy, AWy, MWy, LWy, DBWy, DIWy; tag of type S5TIME, WORD; S5T#time-constant

\(Y = 0 \text{ to } 65534\)

You can also enter time constants as normal time information, such as 5s.

Timers in actions
These actions are executed once if the event occurs and the corresponding step is active. During each additional execution of the sequencer an event-dependent action is not executed again until the event occurs again.

The following table shows the possible timers in actions.

<table>
<thead>
<tr>
<th>Event</th>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TL</td>
<td>The timer starts with the event. The timer bit (status of the timer) is set to 1 for the specified time and reset to 0 when the time elapses.</td>
<td></td>
</tr>
<tr>
<td>TD</td>
<td>The timer starts with the event. The timer bit is set to 0 for the specified time and reset to 1 when the time elapses.</td>
<td></td>
</tr>
<tr>
<td>TR</td>
<td>The time is stopped with the event. Timer bit and timer value are reset to 0.</td>
<td></td>
</tr>
<tr>
<td>TF</td>
<td>With the activation of the step the status of the timer is 1. When the step is deactivated the timer runs and the timer bit is set to 0</td>
<td></td>
</tr>
</tbody>
</table>
Examples

<table>
<thead>
<tr>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>TL - Extended pulse</td>
<td>&quot;MyTimer&quot;, S5T#5S</td>
<td>With incoming step, &quot;MyTimer&quot; is set to 1 for 5 seconds.</td>
</tr>
<tr>
<td>S0</td>
<td>TD - Retentive on-delay</td>
<td>&quot;MyTimer&quot;, S5T#5S</td>
<td>With outgoing, &quot;MyTimer&quot; is set to 0 for 5 seconds.</td>
</tr>
<tr>
<td>L0</td>
<td>TR - Halt and reset timer</td>
<td>&quot;MyTimer&quot;</td>
<td>With incoming interlock condition, &quot;MyTimer&quot; is reset to 0 and the timer halted.</td>
</tr>
<tr>
<td></td>
<td>TF - Off delay</td>
<td>&quot;MyTimer&quot;, S5T#5S</td>
<td>&quot;MyTimer&quot; is set to 1 while the step is active. &quot;MyTimer&quot; is set to 0 only 5 seconds after the deactivation of the step.</td>
</tr>
</tbody>
</table>

Summary of all actions

Overview

The following table shows event-dependent actions with and without interlock:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td></td>
<td></td>
<td></td>
<td>A step is active.</td>
<td>The signal state of the operand is 1.</td>
</tr>
<tr>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The operand is set to 1.</td>
</tr>
<tr>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The operand is set to 0 and then remains at 0.</td>
</tr>
<tr>
<td>N</td>
<td>CALL</td>
<td></td>
<td></td>
<td></td>
<td>The specified block is called.</td>
</tr>
<tr>
<td>L</td>
<td>&lt;Timer value&gt;</td>
<td></td>
<td></td>
<td></td>
<td>The signal state of the operand is 1 for n seconds.</td>
</tr>
<tr>
<td>D</td>
<td>&lt;Timer value&gt;</td>
<td></td>
<td></td>
<td></td>
<td>n seconds after the step activation the signal of the operand is set to 1 for the duration of the step activation. This does not apply if the duration of the step activation is shorter than n seconds.</td>
</tr>
<tr>
<td>TF</td>
<td>&lt;Timer value&gt;</td>
<td></td>
<td></td>
<td></td>
<td>When the step is activated, the operand has the signal 1. When the step is deactivated the timer runs and the operand is set to 0 only after the time has expired.</td>
</tr>
<tr>
<td>(C)-N</td>
<td></td>
<td></td>
<td></td>
<td>A step is active and the interlock condition is</td>
<td>The signal state of the operand is 1.</td>
</tr>
<tr>
<td>(C)-S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The operand is set to 1 and then remains at 1.</td>
</tr>
</tbody>
</table>
Event-dependent actions with and without interlock

These actions are executed once if the event occurs and the corresponding step is active.

During each further execution of the sequencer an event-dependent action is only executed again if the event occurs again.

The following table shows actions that are linked to an incoming step:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>N</td>
<td>CALL</td>
<td>A step becomes active (incoming).</td>
<td>The operand has the signal 1 once.</td>
</tr>
<tr>
<td>S1</td>
<td>S</td>
<td></td>
<td></td>
<td>The operand is set to 1.</td>
</tr>
<tr>
<td>S1</td>
<td>R</td>
<td></td>
<td></td>
<td>The operand is set to 0.</td>
</tr>
<tr>
<td>S1</td>
<td>N</td>
<td>CALL</td>
<td></td>
<td>The specified block is called.</td>
</tr>
<tr>
<td>S1</td>
<td>ON</td>
<td>Step name</td>
<td></td>
<td>Step is activated</td>
</tr>
<tr>
<td>S1</td>
<td>OFF</td>
<td>Step name</td>
<td></td>
<td>Step is deactivated</td>
</tr>
<tr>
<td>S1</td>
<td>OFF</td>
<td>S_ALL</td>
<td>All steps are deactivated, except for the step in which the action is located.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>N</td>
<td>CALL</td>
<td>A step is active and the interlock condition is satisfied.</td>
<td>The operand has the signal 1 once.</td>
</tr>
<tr>
<td>S1</td>
<td>S</td>
<td></td>
<td></td>
<td>The operand is set to 1.</td>
</tr>
<tr>
<td>S1</td>
<td>R</td>
<td></td>
<td></td>
<td>The operand is set to 0.</td>
</tr>
<tr>
<td>S1</td>
<td>N</td>
<td>CALL</td>
<td></td>
<td>The specified block is called.</td>
</tr>
<tr>
<td>S1</td>
<td>ON</td>
<td>Step name</td>
<td></td>
<td>Step is activated</td>
</tr>
<tr>
<td>S1</td>
<td>OFF</td>
<td>Step name</td>
<td></td>
<td>Step is deactivated.</td>
</tr>
<tr>
<td>S1</td>
<td>OFF</td>
<td>S_ALL</td>
<td>All steps are deactivated, except for the step in which the action is located.</td>
<td></td>
</tr>
</tbody>
</table>

The following table shows actions that are linked to a step becoming active:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>N</td>
<td></td>
<td></td>
<td>A step is deactivated (outgoing).</td>
<td>The operand has the signal 1 once.</td>
</tr>
<tr>
<td>S0</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td>The operand is set to 1.</td>
</tr>
<tr>
<td>S0</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>The operand is set to 0.</td>
</tr>
</tbody>
</table>
The following table shows actions that are linked to an incoming monitoring error:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>N</td>
<td></td>
<td>CALL</td>
<td>A monitoring error occurs during an active step or a monitoring error is pending when the step becomes active.</td>
<td>The operand has the signal 1 once.</td>
</tr>
<tr>
<td>V1</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td>The operand is set to 1.</td>
</tr>
<tr>
<td>V1</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>The operand is set to 0.</td>
</tr>
<tr>
<td>V1</td>
<td>N</td>
<td>CALL</td>
<td></td>
<td></td>
<td>The specified block is called.</td>
</tr>
<tr>
<td>V1</td>
<td>ON</td>
<td>Step name</td>
<td></td>
<td></td>
<td>Step is activated</td>
</tr>
<tr>
<td>V1</td>
<td>OFF</td>
<td>Step name</td>
<td></td>
<td></td>
<td>Step is deactivated.</td>
</tr>
<tr>
<td>V1</td>
<td>OFF</td>
<td>S_ALL</td>
<td></td>
<td></td>
<td>All steps are deactivated, except for the step in which the action is located.</td>
</tr>
<tr>
<td>-(C)-</td>
<td>V1</td>
<td>N</td>
<td></td>
<td>A monitoring error occurs during an active step or a monitoring error is pending when the step becomes active and the interlock condition is satisfied.</td>
<td>The operand has the signal 1 once.</td>
</tr>
<tr>
<td>-(C)-</td>
<td>V1</td>
<td>S</td>
<td></td>
<td></td>
<td>The operand is set to 1.</td>
</tr>
<tr>
<td>-(C)-</td>
<td>V1</td>
<td>R</td>
<td></td>
<td></td>
<td>The operand is set to 0.</td>
</tr>
<tr>
<td>-(C)-</td>
<td>V1</td>
<td>N</td>
<td>CALL</td>
<td></td>
<td>The specified block is called.</td>
</tr>
<tr>
<td>-(C)-</td>
<td>V1</td>
<td>ON</td>
<td>Step name</td>
<td></td>
<td>Step is activated</td>
</tr>
<tr>
<td>-(C)-</td>
<td>V1</td>
<td>OFF</td>
<td>Step name</td>
<td></td>
<td>Step is deactivated.</td>
</tr>
<tr>
<td>-(C)-</td>
<td>V1</td>
<td>OFF</td>
<td>S_ALL</td>
<td></td>
<td>All steps are deactivated, except for the step in which the action is located.</td>
</tr>
</tbody>
</table>

The following table shows actions that are linked to an incoming interlock condition:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0</td>
<td>N</td>
<td></td>
<td>CALL</td>
<td>Outgoing monitoring error (has been eliminated or was possibly acknowledged).</td>
<td>The operand has the signal 1 once.</td>
</tr>
<tr>
<td>V0</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td>The operand is set to 1.</td>
</tr>
<tr>
<td>V0</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>The operand is set to 0.</td>
</tr>
<tr>
<td>V0</td>
<td>N</td>
<td>CALL</td>
<td></td>
<td></td>
<td>Step is activated</td>
</tr>
<tr>
<td>V0</td>
<td>ON</td>
<td>Step name</td>
<td></td>
<td></td>
<td>Step is activated</td>
</tr>
<tr>
<td>V0</td>
<td>OFF</td>
<td>Step name</td>
<td></td>
<td></td>
<td>Step is deactivated.</td>
</tr>
</tbody>
</table>

The following table shows actions that are linked to an outgoing monitoring error:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0</td>
<td>N</td>
<td></td>
<td>CALL</td>
<td></td>
<td>The operand has the signal 1 once.</td>
</tr>
<tr>
<td>V0</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td>The operand is set to 1.</td>
</tr>
<tr>
<td>V0</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>The operand is set to 0.</td>
</tr>
<tr>
<td>V0</td>
<td>N</td>
<td>CALL</td>
<td></td>
<td></td>
<td>Step is activated</td>
</tr>
<tr>
<td>V0</td>
<td>ON</td>
<td>Step name</td>
<td></td>
<td></td>
<td>Step is activated</td>
</tr>
<tr>
<td>V0</td>
<td>OFF</td>
<td>Step name</td>
<td></td>
<td></td>
<td>Step is deactivated.</td>
</tr>
</tbody>
</table>
The following table shows actions that are linked to an outgoing interlock condition:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>L0</td>
<td>N</td>
<td></td>
<td></td>
<td>An interlock condition is satisfied when the step is active.</td>
<td>The operand has the signal 1 once.</td>
</tr>
<tr>
<td>L0</td>
<td>S</td>
<td></td>
<td></td>
<td>The operand is set to 1.</td>
<td></td>
</tr>
<tr>
<td>L0</td>
<td>R</td>
<td></td>
<td></td>
<td>The operand is set to 0.</td>
<td></td>
</tr>
<tr>
<td>L0</td>
<td>N</td>
<td></td>
<td>CALL</td>
<td>The specified block is called.</td>
<td></td>
</tr>
<tr>
<td>L0</td>
<td>ON</td>
<td></td>
<td>Step name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L0</td>
<td>OFF</td>
<td></td>
<td>Step name</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The following table shows actions that are linked to an alarm acknowledgement:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC 1</td>
<td>N</td>
<td></td>
<td></td>
<td>An alarm is acknowledged.</td>
<td>The operand has the signal 1 once.</td>
</tr>
<tr>
<td>CC 1</td>
<td>S</td>
<td></td>
<td></td>
<td>The operand is set to 1.</td>
<td></td>
</tr>
<tr>
<td>CC 1</td>
<td>R</td>
<td></td>
<td></td>
<td>The operand is set to 0.</td>
<td></td>
</tr>
<tr>
<td>CC 1</td>
<td>N</td>
<td></td>
<td>CALL</td>
<td>The specified block is called.</td>
<td></td>
</tr>
<tr>
<td>CC 1</td>
<td>ON</td>
<td></td>
<td>Step name</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC 1</td>
<td>OFF</td>
<td></td>
<td>Step name</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- (C) - CC 1 N An alarm is acknowledged and the interlock condition is satisfied.
- (C) - CC 1 S The operand has the signal 1 once.
- (C) - CC 1 R The operand is set to 1.
- (C) - CC 1 N CALL The specified block is called.
- (C) - CC 1 ON Step name |

All steps are deactivated, except for the step in which the action is located.
The following table shows actions that are linked to a registration:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>N</td>
<td></td>
<td></td>
<td>Incoming registration.</td>
<td>The operand has the signal 1 once.</td>
</tr>
<tr>
<td>R1</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td>The operand is set to 1.</td>
</tr>
<tr>
<td>R1</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>The operand is set to 0.</td>
</tr>
<tr>
<td>R1</td>
<td>N</td>
<td></td>
<td>CALL</td>
<td></td>
<td>The specified block is called.</td>
</tr>
<tr>
<td>R1</td>
<td>ON</td>
<td></td>
<td>Step name</td>
<td></td>
<td>Step is activated</td>
</tr>
<tr>
<td>R1</td>
<td>OFF</td>
<td></td>
<td>Step name</td>
<td></td>
<td>Step is deactivated.</td>
</tr>
<tr>
<td>R1</td>
<td>N</td>
<td></td>
<td>A registration comes in and the interlock condition is satisfied.</td>
<td>The operand has the signal 1 once.</td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td>The operand is set to 1.</td>
</tr>
<tr>
<td>R1</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>The operand is set to 0.</td>
</tr>
<tr>
<td>R1</td>
<td>N</td>
<td></td>
<td>CALL</td>
<td></td>
<td>The specified block is called.</td>
</tr>
<tr>
<td>R1</td>
<td>ON</td>
<td></td>
<td>Step name</td>
<td></td>
<td>Step is activated</td>
</tr>
<tr>
<td>R1</td>
<td>OFF</td>
<td></td>
<td>Step name</td>
<td></td>
<td>Step is deactivated.</td>
</tr>
</tbody>
</table>

The following table shows counters in actions that are linked to an incoming step:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>CS</td>
<td></td>
<td>Operand, &lt;Count value&gt;</td>
<td>A step is active.</td>
<td>The initial counter value is loaded in the counter as soon as the step becomes active (incoming).</td>
</tr>
<tr>
<td>S1</td>
<td>CU</td>
<td></td>
<td>Operand</td>
<td></td>
<td>The counter is incremented by 1 as soon as the step becomes active.</td>
</tr>
<tr>
<td>S1</td>
<td>CD</td>
<td></td>
<td>Operand</td>
<td></td>
<td>The counter is decremented by 1 as soon as the step becomes active.</td>
</tr>
<tr>
<td>S1</td>
<td>CR</td>
<td></td>
<td>Operand</td>
<td></td>
<td>The counter is reset to 0 as soon as the step becomes active.</td>
</tr>
<tr>
<td>S1</td>
<td>CS</td>
<td></td>
<td>Operand, &lt;Count value&gt;</td>
<td>A step is active and the interlock condition is satisfied.</td>
<td>The initial counter value is loaded in the counter as soon as the step becomes active and the condition is satisfied.</td>
</tr>
<tr>
<td>S1</td>
<td>CU</td>
<td></td>
<td>Operand</td>
<td></td>
<td>The counter is incremented by 1 as soon as the step becomes active and the condition is satisfied.</td>
</tr>
<tr>
<td>S1</td>
<td>CD</td>
<td></td>
<td>Operand</td>
<td></td>
<td>The counter is decremented by 1 as soon as the step becomes active and the condition is satisfied.</td>
</tr>
<tr>
<td>S1</td>
<td>CR</td>
<td></td>
<td>Operand</td>
<td></td>
<td>The counter is reset to 0 as soon as the step becomes active and the condition is satisfied.</td>
</tr>
</tbody>
</table>
The following table shows counters in actions that are linked to an outgoing step:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>CS</td>
<td>Operand, &lt;Count value&gt;</td>
<td>A step is active.</td>
<td>The initial counter value is loaded in the counter as soon as the step is deactivated (outgoing).</td>
<td></td>
</tr>
<tr>
<td>S0</td>
<td>CU</td>
<td>Operand</td>
<td></td>
<td></td>
<td>The counter is incremented by 1 as soon as the step is deactivated.</td>
</tr>
<tr>
<td>S0</td>
<td>CD</td>
<td>Operand</td>
<td></td>
<td></td>
<td>The counter is decremented by 1 as soon as the step is deactivated.</td>
</tr>
<tr>
<td>S0</td>
<td>CR</td>
<td>Operand</td>
<td></td>
<td></td>
<td>The counter is reset to 0 as soon as the step is deactivated.</td>
</tr>
</tbody>
</table>

The following table shows counters in actions that are linked to an outgoing interlock condition:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>CS</td>
<td>Operand, &lt;Count value&gt;</td>
<td>A step is active.</td>
<td>The initial counter value is loaded in the counter as soon as the interlock condition is no longer satisfied (outgoing) during active step, or the interlock condition is not satisfied when the step becomes active.</td>
<td></td>
</tr>
<tr>
<td>L1</td>
<td>CU</td>
<td>Operand</td>
<td></td>
<td></td>
<td>The counter is incremented by 1 as soon as the interlock condition is no longer satisfied during active step, or the interlock condition is not satisfied when the step becomes active.</td>
</tr>
<tr>
<td>L1</td>
<td>CD</td>
<td>Operand</td>
<td></td>
<td></td>
<td>The counter is decremented by 1 as soon as the interlock condition is no longer satisfied during active step, or the interlock condition is not satisfied when the step becomes active.</td>
</tr>
<tr>
<td>L1</td>
<td>CR</td>
<td>Operand</td>
<td></td>
<td></td>
<td>The counter is decremented by 0 as soon as the interlock condition is no longer satisfied during active step, or the interlock condition is not satisfied when the step becomes active.</td>
</tr>
</tbody>
</table>

The following table shows counters in actions that are linked to an incoming interlock condition:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>L0</td>
<td>CS</td>
<td>Operand, &lt;Count value&gt;</td>
<td>A step is active.</td>
<td>The initial counter value is loaded in the counter as soon as the interlock condition is satisfied (incoming) during active step.</td>
<td></td>
</tr>
<tr>
<td>L0</td>
<td>CU</td>
<td>Operand</td>
<td></td>
<td></td>
<td>The counter is decremented by 1 as soon as the interlock condition is satisfied during active step.</td>
</tr>
<tr>
<td>L0</td>
<td>CD</td>
<td>Operand</td>
<td></td>
<td></td>
<td>The counter is decremented by 1 as soon as the interlock condition is satisfied during active step.</td>
</tr>
<tr>
<td>L0</td>
<td>CR</td>
<td>Operand</td>
<td></td>
<td></td>
<td>The counter is decremented by 0 as soon as the interlock condition is satisfied during active step.</td>
</tr>
</tbody>
</table>

The following table shows counters in actions that are linked to an incoming monitoring error:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>CS</td>
<td>Operand, &lt;Count value&gt;</td>
<td>A step is active.</td>
<td>The initial counter value is loaded in the counter as soon as a monitoring error is pending during</td>
<td></td>
</tr>
</tbody>
</table>
The following table shows counters in actions that are linked to an outgoing monitoring error:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0</td>
<td>CS</td>
<td>Operand, &lt;Count value&gt;</td>
<td>A step is active.</td>
<td>The initial counter value is loaded in the counter as soon as a monitoring error is cleared (i.e., eliminated or acknowledged).</td>
<td></td>
</tr>
<tr>
<td>V0</td>
<td>CU</td>
<td>Operand</td>
<td></td>
<td>The counter is incremented by 1 as soon as a monitoring error is cleared.</td>
<td></td>
</tr>
<tr>
<td>V0</td>
<td>CD</td>
<td>Operand</td>
<td></td>
<td>The counter is decremented by 1 as soon as a monitoring error is cleared.</td>
<td></td>
</tr>
<tr>
<td>V0</td>
<td>CR</td>
<td>Operand</td>
<td></td>
<td>The counter is reset to 0 as soon as a monitoring error is cleared.</td>
<td></td>
</tr>
</tbody>
</table>

The following table shows counters in actions that are linked to an incoming monitoring error:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC 1</td>
<td>CS</td>
<td>Operand, &lt;Count value&gt;</td>
<td>A step is active.</td>
<td>The initial counter value is loaded in the counter as soon as an alarm is acknowledged.</td>
<td></td>
</tr>
<tr>
<td>CC 1</td>
<td>CU</td>
<td>Operand</td>
<td></td>
<td>The counter is incremented by 1 as soon as an alarm is acknowledged.</td>
<td></td>
</tr>
</tbody>
</table>
The following table shows counters in actions that are linked to an incoming registration:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC 1</td>
<td>CD</td>
<td>Operand</td>
<td></td>
<td>The counter is decremented by 1 as soon as an alarm is acknowledged.</td>
<td></td>
</tr>
<tr>
<td>CC 1</td>
<td>CR</td>
<td>Operand</td>
<td></td>
<td>The counter is reset to 0 as soon as an alarm is acknowledged.</td>
<td></td>
</tr>
<tr>
<td>-(C)-</td>
<td>CC 1</td>
<td>CS</td>
<td>Operand, &lt;Count value&gt;</td>
<td>A step is active and the interlock condition is satisfied.</td>
<td>The initial counter value is loaded in the counter as soon as a registration comes in.</td>
</tr>
<tr>
<td>-(C)-</td>
<td>CC 1</td>
<td>CU</td>
<td>Operand</td>
<td>The counter is incremented by 1 as soon as an alarm is acknowledged and the condition is satisfied.</td>
<td></td>
</tr>
<tr>
<td>-(C)-</td>
<td>CC 1</td>
<td>CD</td>
<td>Operand</td>
<td>The counter is decremented by 1 as soon as an alarm is acknowledged and the condition is satisfied.</td>
<td></td>
</tr>
<tr>
<td>-(C)-</td>
<td>CC 1</td>
<td>CR</td>
<td>Operand</td>
<td>The counter is reset to 0 as soon as an alarm is acknowledged and the condition is satisfied.</td>
<td></td>
</tr>
</tbody>
</table>

The following table shows timers in actions that are linked to an incoming step:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC 1</td>
<td>CD</td>
<td>Operand</td>
<td></td>
<td>The counter is decremented by 1 as soon as an alarm is acknowledged.</td>
<td></td>
</tr>
<tr>
<td>CC 1</td>
<td>CR</td>
<td>Operand</td>
<td></td>
<td>The counter is reset to 0 as soon as an alarm is acknowledged.</td>
<td></td>
</tr>
<tr>
<td>-(C)-</td>
<td>CC 1</td>
<td>CS</td>
<td>Operand, &lt;Count value&gt;</td>
<td>A step is active and the interlock condition is satisfied.</td>
<td>The initial counter value is loaded in the counter as soon as a registration comes in and the interlock condition is satisfied.</td>
</tr>
<tr>
<td>-(C)-</td>
<td>CC 1</td>
<td>CU</td>
<td>Operand</td>
<td>The counter is incremented by 1 as soon as an alarm is acknowledged and the condition is satisfied.</td>
<td></td>
</tr>
<tr>
<td>-(C)-</td>
<td>CC 1</td>
<td>CD</td>
<td>Operand</td>
<td>The counter is decremented by 1 as soon as an alarm is acknowledged and the condition is satisfied.</td>
<td></td>
</tr>
<tr>
<td>-(C)-</td>
<td>CC 1</td>
<td>CR</td>
<td>Operand</td>
<td>The counter is reset to 0 as soon as an alarm is acknowledged and the condition is satisfied.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>TL</td>
<td>Operand, &lt;Timer value&gt;</td>
<td>A step is active.</td>
<td>The timer starts as soon as the step becomes active (incoming). The timer bit (status of the timer) is set to 1 for the specified time and reset to 0 when the time elapses.</td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td>TD</td>
<td>Operand, &lt;Timer value&gt;</td>
<td>A step is active.</td>
<td>The timer starts as soon as the step becomes active. The timer bit is set to 0 for the specified time and reset to 1 when the time elapses.</td>
<td></td>
</tr>
</tbody>
</table>
The following table shows timers in actions that are linked to an outgoing step:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>TL</td>
<td>Operand, &lt;Timer value&gt;</td>
<td>A step is active.</td>
<td>The timer starts as soon as the step is deactivated (outgoing). The timer bit (status of the timer) is set to 1 for the specified time and reset to 0 when the time elapses.</td>
<td></td>
</tr>
<tr>
<td>S0</td>
<td>TD</td>
<td>Operand, &lt;Timer value&gt;</td>
<td>A step is active.</td>
<td>The timer starts as soon as the step is deactivated. The timer bit is set to 0 for the specified time and reset to 1 when the time elapses.</td>
<td></td>
</tr>
<tr>
<td>S0</td>
<td>TR</td>
<td>Operand</td>
<td>A step is active and the interlock condition is satisfied.</td>
<td>The timer is stopped as soon as the step becomes active. Timer bit and timer value are reset to 0.</td>
<td></td>
</tr>
</tbody>
</table>

The following table shows timers in actions that are linked to an outgoing interlock condition:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>TL</td>
<td>Operand, &lt;Timer value&gt;</td>
<td>A step is active.</td>
<td>The timer starts as soon as the interlock condition is no longer satisfied (outgoing) during active step or the interlock condition is not satisfied when the step becomes active. The timer bit (status of the timer) is set to 1 for the specified time and reset to 0 when the time elapses.</td>
<td></td>
</tr>
<tr>
<td>L1</td>
<td>TD</td>
<td>Operand, &lt;Timer value&gt;</td>
<td>A step is active.</td>
<td>The timer starts as soon as the interlock condition is no longer satisfied during active step or the interlock condition is not satisfied when the step becomes active. The timer bit is set to 0 for the specified time and reset to 1 when the time elapses.</td>
<td></td>
</tr>
<tr>
<td>L1</td>
<td>TR</td>
<td>Operand</td>
<td>A step is active and the interlock condition is satisfied.</td>
<td>The timer is stopped as soon as the step becomes active. Timer bit and timer value are reset to 0.</td>
<td></td>
</tr>
</tbody>
</table>

The following table shows timers in actions that are linked to an incoming interlock condition:
The following table shows timers in actions that are linked to an incoming monitoring error:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>L0</td>
<td>TL</td>
<td>Operand, &lt;Timer value&gt;</td>
<td>A step is active.</td>
<td>The timer starts as soon as the interlock condition is satisfied (incoming) during active step. The timer bit (status of the timer) is set to 1 for the specified time and reset to 0 when the time elapses.</td>
<td></td>
</tr>
<tr>
<td>L0</td>
<td>TD</td>
<td>Operand, &lt;Timer value&gt;</td>
<td></td>
<td>The timer starts as soon as the interlock condition is satisfied during active step. The timer bit is set to 0 for the specified time and reset to 1 when the time elapses.</td>
<td></td>
</tr>
<tr>
<td>L0</td>
<td>TR</td>
<td>Operand</td>
<td></td>
<td>The timer is stopped as soon as the interlock condition is satisfied during active step. Timer bit and timer value are reset to 0.</td>
<td></td>
</tr>
</tbody>
</table>

The following table shows timers in actions that are linked to an outgoing monitoring error:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>TL</td>
<td>Operand, &lt;Timer value&gt;</td>
<td>A step is active.</td>
<td>The timer starts as soon as a monitoring error is pending during active step or a monitoring error is pending when the step becomes active. The timer bit (status of the timer) is set to 1 for the specified time and reset to 0 when the time elapses.</td>
<td></td>
</tr>
<tr>
<td>V1</td>
<td>TD</td>
<td>Operand, &lt;Timer value&gt;</td>
<td></td>
<td>The timer starts as soon as a monitoring error is pending during active step or a monitoring error is pending when the step becomes active. The timer bit is set to 0 for the specified time and reset to 1 when the time elapses.</td>
<td></td>
</tr>
<tr>
<td>V1</td>
<td>TR</td>
<td>Operand</td>
<td></td>
<td>The timer is stopped as soon as a monitoring error is pending during active step or a monitoring error is pending when the step becomes active. Timer bit and timer value are reset to 0.</td>
<td></td>
</tr>
<tr>
<td>-(C)-</td>
<td>V1</td>
<td>TL</td>
<td>Operand, &lt;Timer value&gt;</td>
<td>A step is active and the interlock condition is satisfied.</td>
<td>The timer starts as soon as a monitoring error is pending during active step or a monitoring error is pending when the step becomes active and the interlock condition is satisfied. The timer bit is set to 1 for the specified time and reset to 0 when the time elapses.</td>
</tr>
<tr>
<td>-(C)-</td>
<td>V1</td>
<td>TD</td>
<td>Operand, &lt;Timer value&gt;</td>
<td></td>
<td>The timer starts as soon as a monitoring error is pending during active step or a monitoring error is pending when the step becomes active and the condition is satisfied. The timer bit is set to 0 for the specified time and reset to 1 when the time elapses.</td>
</tr>
<tr>
<td>-(C)-</td>
<td>V1</td>
<td>TR</td>
<td>Operand</td>
<td></td>
<td>The timer is stopped as soon as a monitoring error is pending during active step or a monitoring error is pending when the step becomes active and the condition is satisfied. Timer bit and timer value are reset to 0.</td>
</tr>
</tbody>
</table>

The following table shows timers in actions that are linked to an incoming monitoring error:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0</td>
<td>TL</td>
<td>Operand, &lt;Timer value&gt;</td>
<td>A step is active.</td>
<td>The timer starts as soon as a monitoring error is cleared (i.e., eliminated or acknowledged). The timer bit (status of the timer) is set to 1 for the specified time and reset to 0 when the time elapses.</td>
<td></td>
</tr>
</tbody>
</table>
The following table shows timers in actions that are linked to an alarm acknowledgement:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC 1</td>
<td>TL</td>
<td>Operand, &lt;Timer value&gt;</td>
<td>A step is active.</td>
<td>The timer starts as soon as an alarm is acknowledged. The timer bit (status of the timer) is set to 1 for the specified time and reset to 0 when the time elapses.</td>
<td></td>
</tr>
<tr>
<td>CC 1</td>
<td>TD</td>
<td>Operand, &lt;Timer value&gt;</td>
<td>A step is active.</td>
<td>The timer starts as soon as an alarm is acknowledged. The timer bit is set to 0 for the specified time and reset to 1 when the time elapses.</td>
<td></td>
</tr>
<tr>
<td>CC 1</td>
<td>TR</td>
<td>Operand</td>
<td>A step is active.</td>
<td>The timer is stopped as soon as an alarm error is cleared. Timer bit and timer value are reset to 0.</td>
<td></td>
</tr>
<tr>
<td>(C)</td>
<td>CC 1</td>
<td>TL</td>
<td>Operand, &lt;Timer value&gt;</td>
<td>The timer starts as soon as an alarm is acknowledged and the interlock condition is satisfied. The timer bit is set to 1 for the specified time and reset to 0 when the time elapses.</td>
<td></td>
</tr>
<tr>
<td>(C)</td>
<td>CC 1</td>
<td>TD</td>
<td>Operand, &lt;Timer value&gt;</td>
<td>The timer starts as soon as an alarm is acknowledged and the condition is satisfied. The timer bit is set to 0 for the specified time and reset to 1 when the time elapses.</td>
<td></td>
</tr>
<tr>
<td>(C)</td>
<td>CC 1</td>
<td>TR</td>
<td>Operand</td>
<td>The timer is stopped as soon as an alarm is acknowledged and the condition is satisfied. Timer bit and timer value are reset to 0.</td>
<td></td>
</tr>
</tbody>
</table>

The following table shows timers in actions that are linked to an incoming registration:

<table>
<thead>
<tr>
<th>Interlock</th>
<th>Event</th>
<th>Identifier</th>
<th>Action</th>
<th>Requirement</th>
<th>Signal state</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>TL</td>
<td>Operand, &lt;Timer value&gt;</td>
<td>A step is active.</td>
<td>The timer starts as soon as a registration comes in. The timer bit (status of the timer) is set to 1 for the specified time and reset to 0 when the time elapses.</td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td>TD</td>
<td>Operand, &lt;Timer value&gt;</td>
<td>A step is active.</td>
<td>The timer starts as soon as a registration comes in. The timer bit is set to 0 for the specified time and reset to 1 when the time elapses.</td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td>TR</td>
<td>Operand</td>
<td>A step is active.</td>
<td>The timer is stopped as soon as a registration comes in. Timer bit and timer value are reset to 0.</td>
<td></td>
</tr>
<tr>
<td>(C)</td>
<td>R1</td>
<td>TL</td>
<td>Operand, &lt;Timer value&gt;</td>
<td>The timer starts as soon as a registration comes in and the interlock condition is satisfied. The timer bit is set to 1 for the specified time and reset to 0 when the time elapses.</td>
<td></td>
</tr>
<tr>
<td>(C)</td>
<td>R1</td>
<td>TD</td>
<td>Operand, &lt;Timer value&gt;</td>
<td>The timer starts as soon as a registration comes in and the condition is satisfied. The timer bit is set to 0 for the specified time and reset to 1 when the time elapses.</td>
<td></td>
</tr>
</tbody>
</table>
Programming conditions

This chapter contains the following information:
- Overview of conditions (S7-300/400)
- Transitions (S7-300/400)
- Interlocks (S7-300/400)
- Supervisions (S7-300/400)
- Permanent Instructions (S7-300/400)
- Creating new network components (S7-300/400)
- Copying, moving and deleting conditions (S7-300/400)
- GRAPH-specific operands in conditions (S7-300/400)
- Permanent block calls (S7-300/400)

Overview of conditions

Overview

Conditions are binary process statuses, that are linked together according to the boolean logic in the ladder block diagram or the function block diagram as LAD or FBD element, e.g. normally open contact, normally closed contact, AND box, OR box or comparator.

The result of logic operation (RLO) of conditions can influence steps, switching to the next step or the whole sequencer.

You can program the conditions in the navigation view in the following panes:
- Permanent pre-instructions
- Permanent post-instructions

You can program the conditions in the single step view in the following panes:
- Interlock
- Supervision
- Transition

Networks of transitions, supervisions, interlocks and permanent instructions can contain up to 32 instructions to which as maximum of 32 operands are interconnected.
For more information on transitions, interlocks, supervisions and permanent instructions, refer to "See also"

Transitions

Overview

The transition is the sequencer part that contains the conditions for switching from one step to the following. You can program transitions in the single step view as boolean logic operation in LAD and FBD.

Detailed information on programming in LAD and FBD, refer to "See also".

Empty transition

An "empty" transition contains no step enabling conditions and therefore no binary logic operations and no comparison operations. It is
Transition after simultaneous branch

All simultaneous branches are simultaneously active in a sequencer. The transition is valid after a simultaneous branch, if all steps are active before the simultaneous junction. The transition after the simultaneous branch only switches if all simultaneous branches are completely executed.

Skipping a step

If the transition before and after a step are valid at the same time, the step is skipped and thus not active. To do this, the "Skip steps" must be activated in "Options > Settings" under "PLC Programming > GRAPH" in the "Sequence properties" area. This option is not available, if the "Create minimized DB" option is selected.

Interlocks

Overview

An interlock or a step interlock is a condition within a step that could be programmed. The execution of individual actions can be influenced by a step interlock.

If the logic operation of the condition is satisfied, actions combined with interlock are executed. If the logic operation of the condition is not satisfied, there is a fault:

- Actions combined with interlock are not executed.
- An interlock error is reported.

You can program an interlock in the single step view. Maximum 32 LAD or FBD elements could be programmed for each interlock.

For additional information on instructions that could be used in an interlock, refer to "See also."

A programmed interlock is displayed in each view type with "-(C)" left from the step.

Note

If you program a step interlock, it will be only used in the actions if their operations are expanded with "-(C)".

Signal states

The signal states for incoming, outgoing and pending events can be collected and responded to in actions. The following signal states can be collected:

<table>
<thead>
<tr>
<th>Event</th>
<th>Signal state</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interlock is outgoing</td>
<td>L1 = 1</td>
<td>The condition is not satisfied anymore (worst outcome); the time &quot;&lt;Step_name&gt;.U&quot; stops. The time &quot;&lt;Step_name&gt;.T&quot; continues to run and the transition switches as soon as its step enabling conditions are satisfied.</td>
</tr>
<tr>
<td>Incoming interlock</td>
<td>L0 = 1</td>
<td>The condition is satisfied (best outcome).</td>
</tr>
<tr>
<td>Interlock is pending</td>
<td>C = 1</td>
<td>The condition is satisfied. The action in connection with &quot;-(C)&quot; (for example, -(C) N) is executed.</td>
</tr>
</tbody>
</table>

Empty interlock

An interlock without condition (without LAD or FBD elements) behaves as a satisfied interlock.

Acknowledgment required

You can activate or deactivate in the navigation view in the pane "Alarms" the acknowledgement required for interlock messages. If the option "Acknowledgement required for interlock alarms" is activated, the operator has to acknowledge the alarm for a pending interlock, so that the program continues.

You can define the alarm output for an error on the "Alarms" pane.

For additional information on messages, refer to "See also."
Supervisions

Overview

A supervision or step monitoring is a condition within a step that can be programmed. Here the switching from one step to the next is influenced. Only active steps are monitored.

The signal states for incoming, outgoing and pending events can be collected and responded to in actions. The following signal states can be collected:

<table>
<thead>
<tr>
<th>Event</th>
<th>Signal state</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outgoing supervision</td>
<td>V1 = 1</td>
<td>The condition is satisfied (worst outcome); the time &quot;&lt;Step_name&gt;.U&quot; stops. The time &quot;&lt;Step_name&gt;.T&quot; continues to run and the transition does not switch, even though the step enabling conditions may be satisfied.</td>
</tr>
<tr>
<td>Supervision incoming</td>
<td>V0 = 1</td>
<td>The condition is not satisfied anymore (best outcome).</td>
</tr>
</tbody>
</table>

If the supervision condition is satisfied, there is a monitoring error.

A programmed supervision is displayed in each view type with "-(V)" left from the step. It is also displayed with "-( V)" in the network.

You can program supervisions in the single step view. You can program maximum 32 LAD or FBD elements for each supervision.

Response to pending fault

If the condition logic operation is satisfied, there is a fault and the event "V1" is reported. The sequencer does not switch in the next step. The current step stays active. The step activation time "<Step_name>.U" is paused as soon as the condition is satisfied.

Response without fault

If the logic operation of the condition is not satisfied, there is a fault: If the following transition is satisfied, the sequencer switches again in the next step.

Acknowledgment required

In the navigation view in the "Alarms" pane, you can activate or deactivate the acknowledgement required for supervision alarms. If the option "Acknowledgement required for supervision alarms" is activated, the operator has to acknowledge the alarm for a pending interlock, so that the program continues.

You can define the alarm output for an error on the "Alarms" pane.

For additional information on messages, refer to "See also."

Permanent Instructions

Permanent instructions are conditions and block calls that are switched before or after the sequencer. They are processed independently from the sequencer once for a cycle. You can program in a GRAPH_FB as many permanent instructions as you want.

A comment can be created for all networks of a step. You can enter a title for each individual network. Each permanent instruction has a number. You can find the panes "Permanent pre-instructions" and "Permanent post-instructions" in the programming window in the "Navigation" area.

Programming conditions

Requirement

- A function block is open.
- A network in the pane for permanent instructions or in the single step view is available.

Procedure

To program a condition, follow these steps:
1. Open the pane in which you want to insert the condition.

2. Select one of the following options:
   - Drag an instruction from the favorites or from the task card "Instructions" in the network.
     When you drag an instruction into a network, green rectangles appear that mark the possible insert positions. The active
     green rectangle displays the insert position over which the instruction is currently located.
   - Left-click on the insert position you want.
   - Open the shortcut menu of the condition and select the command "Insert empty box".
   - Click "??" in the not yet defined edit area and select the required condition from the drop-down list box or enter it manually.

3. Select from the drop-down list box of the edit area "??" an assignment for the condition or enter it manually.
   If you move the mouse over the edit area, you can see, which formats you can use for the assignments.

4. Save the project.

**Result**
The network contains a condition.

**Note**
The time monitors "Greater than step activation time" (CMP>T) and "Greater than uninterrupted step activation time" (CMP>U) are
by default preset to 100 milliseconds. You can change these settings in the "Options > Settings" menu under "PLC programming
> GRAPH" in the "Default for time monitoring" area.

---

**Creating new network components**

**Create new rung for LAD conditions**

**Requirement**
A step is open in the single step view.

**Procedure**
To create a new rung for a LAD condition, proceed as follows:

1. Drag an element from the favorites area in the programming window to the power rail.

**Result**
The element is on a new rung.

**Create new branch for FBD conditions**

**Requirement**
A step is open in the single step view.

**Procedure**
1. Drag an element from the favorites area in the programming window to the main branch.

**Result**
The element is on a new branch.

**Creating a new network for permanent instructions**
### Requirement
A block is open.

### Procedure
1. Right-click the pane title bar permanent instructions.
2. Select the "Insert network" command in the shortcut menu.

### Result
The permanent instruction contains an additional network.

---

### Copying, moving and deleting conditions

#### Copying conditions

**Requirement**
A network contains a condition.

**Procedure**
To copy a condition, follow these steps:

1. Select the condition you want to copy.
   If you want to copy more conditions at the same time, press the <CTRL> key while you are selecting the conditions you want.
2. In the shortcut menu, select the "Copy" command.
   You can also use the key combination <Ctrl+C>.
3. Position the cursor over the position of the same or another network where you want to insert the condition.
4. Select the "Paste" command in the shortcut menu.
   You can also use the key combination <Ctrl+V>.

**Result**
The insert position contains the copied condition.

#### Moving conditions

**Requirement**
A network contains a condition.

**Procedure**
To move a condition, follow these steps:

1. Drag the condition you want to move to the required position of the same or another network.
   If you want to move more conditions at the same time, press the <CTRL> key while you are selecting the required conditions.
2. Click on the insert position you want.

**Result**
The moved condition is pasted to the desired location.

#### Deleting conditions

**Requirement**
A network contains a condition.

**Procedure**

To delete a condition, follow these steps:

1. Select the condition you want to delete.
   If you want to delete multiple conditions at the same time, press the <CTRL> key while you are selecting the desired conditions.
2. Select the "Delete" command in the shortcut menu.
   You can also use the key <Del>.

**Result**

The selected condition is deleted.

---

**GRAPH-specific operands in conditions**

**Overview**

You can use system information for steps as operands in transitions, supervisions, interlocks, actions and permanent instructions.

The following table shows the GRAPH-specific operands for conditions:

<table>
<thead>
<tr>
<th>Operand</th>
<th>Meaning</th>
<th>Use in</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Step_name&gt;.T</td>
<td>Current or last activation time of the specified step</td>
<td>Comparator, assignment</td>
</tr>
<tr>
<td>&lt;Step_name&gt;.U</td>
<td>Total activation time of the specified step without the fault time</td>
<td>Comparator, assignment</td>
</tr>
<tr>
<td>&lt;Step_name&gt;.X</td>
<td>Indicates whether the specified step is active</td>
<td>Normally opened contact/Normally closed contact</td>
</tr>
<tr>
<td>&lt;Transition_name&gt;.TT</td>
<td>Transition is satisfied Check whether all conditions of the specified transition are satisfied</td>
<td>Normally opened contact/Normally closed contact</td>
</tr>
</tbody>
</table>

**Example CMP>T**

To monitor the duration of a process, you can program a supervision condition (Step monitoring) using the "CMP>T" = "Long step activation time" instruction. The "<Step_name>.T" operand and the entire duration of a step is monitored this way. As long as the activation time of Step1 is less than 100 ms, the RLO is "0". As soon as the activation time exceeds 100 ms, the RLO changes to "1".

The following figure shows the mentioned example:

```
#STEP1.T
>
T#10MS
```

**Example CMP>U**

During many processes it is necessary to monitor the process duration less possible fault times. The necessary duration of a stirring process, for example, can take totally 50 milliseconds, regardless of the timers in which the respective step was disturbed. To perform the time comparison, the "CMP>U" = "Long uninterrupted step activation time" instruction is used here.

For this purpose, you program a supervision condition (step monitoring) in which you monitor the "Step_name,U" operand. As long as the activation time of Step1 minus possible faults is less than 100 ms, the RLO is "0". As soon as the activation time exceeds 100 ms, the RLO changes to "1".

The following figure shows the mentioned example:
Permanent block calls

This chapter contains the following information:
- Inserting block calls (S7-300/400)
- Insert block calls with Copy and Paste (S7-300/400)

Inserting block calls

Requirement
A network for permanent instructions is opened.

Inserting block calls with drag-and-drop
To insert a block call with drag-and-drop into a permanent instruction, proceed as follows:

1. In the project tree, select the block that you want to insert in the network.
2. Drag the block to the desired position in the network.
   The possible insert positions are shown with a small square in the network.
3. Release the mouse at the desired insert position.
   The "Call options" dialog opens.
   If the called block is a function block:
   ○ Select an available data block from the drop-down list of the dialog or enter a data block name if you want to create a new one.
   ○ Select "Manual" if you want to assign a number to the data block itself.
     By default, GRAPH automatically assigns a consecutive number.

Result
The permanent instruction network contains the function block call.

Inserting a block call manually
To insert manually a block call into a permanent instruction, proceed as follows:

1. Select the "Insert empty box" command from the editor shortcut menu.
   The empty box is inserted behind the last object in the network.
2. Position the cursor in the entry field.
   The not yet defined entry is characterized by "??".
3. Enter the name of the block that you want to call and press the enter key.
   The "Call options" dialog appears.
   If the called block is a function block:
   ○ Select an available data block from the drop-down list of the dialog or enter a data block name if you want to create a new one.
   ○ Select "Manual" if you want to assign a number to the data block itself.
     By default, GRAPH automatically assigns a consecutive number.
Insert block calls with Copy and Paste

**Requirement**
A network for permanent instructions is opened.

**Procedure**
To insert a block call with copy and paste into an instruction, select one of the following methods:

1. Select the function block in the project tree that you want to copy.
2. Press the key combination <CTRL+C> or select the "Copy" command in the shortcut menu.
3. Position the cursor on the network, in which you want to insert the block call. To insert a block call, press the key combination <CTRL+V> or select the "Paste" command from the shortcut menu.
   
   The function block is inserted after the last object in the network. The "Call options" dialog opens.
   
   If the called block is a function block:
   
   - Select an available data block from the drop-down list of the dialog or enter a data block name if you want to create a new one.
   
   - Select "Manual" if you want to assign a number to the data block itself.
     
     By default, GRAPH automatically assigns a consecutive number.

Or:

1. Select the network, in which you want to insert a call.
2. Insert an empty box via the shortcut menu.
3. Click twice slowly in the project tree on the name of the required function block.
   
   The edit field activates and the function block name is highlighted.
4. Press the key combination <CTRL+C> or select the "Copy" command in the shortcut menu.
5. Position the cursor in the edit field of the empty box. To insert the block call, press the key combination <CTRL+V> or select the "Paste" command in the shortcut menu and press the enter key.
   
   The block is inserted behind the last object in the network and the "Call options" dialog appears.
   
   If the called block is a function block:
   
   - Select an available data block from the drop-down list of the dialog or enter a data block name if you want to create a new one.
   
   - Select "Manual" if you want to assign a number to the data block itself.
     
     By default, GRAPH automatically assigns a consecutive number.

**Result**
The permanent instruction network contains the function block call.

---

**Alarms**

This chapter contains the following information:

- Alarms (S7-300/400)
- Editing alarms (S7-300/400)
Alarms

Overview
You can use alarms in the program window of GRAPH to monitor the operating and fault states.

To adjust interlock and supervision alarms across blocks, you can make the required settings in the "Options > Settings" menu in the "PLC programming > GRAPH" area under "Alarm properties".

You can activate or deactivate the block-specific alarms in the navigation view in the "Alarms" pane. You can enter alarm texts for interlocks and supervisions here using templates and activate or deactivate an acknowledgement required for every alarm type. The templates are highlighted in gray.

Editing alarms

Requirement
A block is open.

Procedure
Proceed as follows to enter an alarm text for interlock or a supervision:

1. Open the "Alarms" pane in the navigation view and select "Alarms" here.
2. Select the "Enable alarms" check box.
   The areas for interlock and supervision alarms are activated.
3. Select, if necessary, the "Acknowledgement required for interlock alarms" check box or "Acknowledgement required for supervision alarms".
4. Enter the required alarm text.

Result
The block contains an interlock and/or supervision alarm.

Call GRAPH-FB and assign parameters

This chapter contains the following information:
- Call GRAPH-function block (S7-300/400)
- A GRAPH block interface (S7-300/400)

Call GRAPH-function block

Call of a GRAPH-FB
To run a GRAPH-FB in the CPU, it must be called in a code block. A GRAPH-FB is always called as single instance.

Parameter assignment
At the FB call, supply the parameter from the interface with the actual parameter.

A GRAPH block interface

This chapter contains the following information:
- An overview of a GRAPH block interface (S7-300/400)
- Input parameters (S7-300/400)
An overview of a GRAPH block interface

Introduction
You can select various parameter records for creating a GRAPH block. The selection of the parameter record corresponds to the planned use of the sequencer and the memory space available in the CPU. The memory size for the GRAPH block increases simultaneously with the parameter record size and the related Instance DB.

You can make the following selections:
- Maximum interface set or default interface set
- Default DB or memory space minimized DB

Maximum interface set or default interface set
You define the number of the input and output parameter via this selection. The default parameter record facilitates the sequencer mode to operate in different operating modes and allows the acknowledgement of alarms.

The maximum parameter record makes additional diagnostics possibilities available. You can manually delete or insert individual parameters in both parameter records.

Default DB or memory space minimized DB
You define the number of the static parameter via this selection. Generally, both memory models store information on the individual steps and transitions as static parameters. For each step and transition an individual structure with detailed information is created. You receive more detailed information in a default DB than in a memory space minimized DB, however a default DB requires considerably more memory space. If you choose to use the memory space minimized DB, you cannot afterwards manually insert individual structure elements in the step and transition information.

The following restrictions apply to a memory space minimized DB:
- Some elements of the steps and transition structures are unavailable.
- The "Skip steps" option is unavailable.
- Alarms are unavailable.
- Steps and transition numbers must be consecutive. If it is not the case, the steps and transitions are automatically renumbered during compilation.
- No data is generated for the criteria analysis.

Input parameters

An overview of the GRAPH block input parameter
The following table provides an overview of the GRAPH block input parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Standard parameter record</th>
<th>Maximum parameter record</th>
</tr>
</thead>
</table>
| EN         | BOOL      | Enable input
If EN is switched, the FB is always switched. | x             | x                        |
| OFF_SQ     | BOOL      | OFF_SEQUENCE:
Turn off sequencer, i.e.deactivate all steps | x             | x                        |
| INIT_SQ    | BOOL      | INIT_SEQUENCE:
Activate initial steps Reset sequencer | x             | x                        |
<p>| ACK_EF     | BOOL      | ACKNOWLEDGE_ERROR_FAULT: | x             | x                        |</p>
<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REG_EF</td>
<td>BOOL</td>
<td>Register all errors and faults</td>
</tr>
<tr>
<td>ACK_S</td>
<td>BOOL</td>
<td>Acknowledge the indicated step in S_NO</td>
</tr>
<tr>
<td>REG_S</td>
<td>BOOL</td>
<td>Register the indicated step in S_NO</td>
</tr>
<tr>
<td>HALT_SQ</td>
<td>BOOL</td>
<td>Stop/reactivate sequencer</td>
</tr>
<tr>
<td>HALT_TM</td>
<td>BOOL</td>
<td>Stop/reactivate all step activation times and time-dependent sequence instructions (L and D)</td>
</tr>
<tr>
<td>ZERO_OP</td>
<td>BOOL</td>
<td>Reset to zero all operands with the ID N, D, L in active steps and reactivate CALL instructions in actions not executed/Operands and CALL instructions.</td>
</tr>
<tr>
<td>EN_IL</td>
<td>BOOL</td>
<td>Deactivate interlock (Sequencer performs the same as by a satisfied interlock)/reactivate</td>
</tr>
<tr>
<td>EN_SV</td>
<td>BOOL</td>
<td>Deactivate step monitoring (Supervision) (Sequencer performs the same as by an unsatisfied step monitoring)/reactivate</td>
</tr>
<tr>
<td>EN_ACKREQ</td>
<td>BOOL</td>
<td>Activate acknowledgement required</td>
</tr>
<tr>
<td>EN_SSKIP</td>
<td>BOOL</td>
<td>Activate step skipping</td>
</tr>
<tr>
<td>DISP_SACT</td>
<td>BOOL</td>
<td>Display active steps only</td>
</tr>
<tr>
<td>DISP_SEF</td>
<td>BOOL</td>
<td>Display steps with errors and disrupted steps</td>
</tr>
<tr>
<td>DISP_SALL</td>
<td>BOOL</td>
<td>Display all steps</td>
</tr>
<tr>
<td>S_PREV</td>
<td>BOOL</td>
<td>Display preceding step in S_NO (smaller numbers)</td>
</tr>
<tr>
<td>S_NEXT</td>
<td>BOOL</td>
<td>Display next step in S_NO (greater numbers)</td>
</tr>
<tr>
<td>SW_AUTO</td>
<td>BOOL</td>
<td>Operating mode-Failover: Automatic mode</td>
</tr>
<tr>
<td>SW_TAP</td>
<td>BOOL</td>
<td>Operating mode-Failover: Semiautomatic mode</td>
</tr>
<tr>
<td>SW_TOP</td>
<td>BOOL</td>
<td>Operating mode-Failover: Semiautomatic mode</td>
</tr>
</tbody>
</table>
Operating mode-Failover:
Automatic or semiautomatic mode

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Standard parameter record</th>
<th>Maximum parameter record</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW_MAN</td>
<td>BOOL</td>
<td>SWITCH_MODE_MANUAL: Operating mode-Failover: Manual mode, no independent sequence is initiated</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S_SEL</td>
<td>INT</td>
<td>STEP_SELECT: Activate step number for S_NO output parameter in manual mode, activate/deactivate with S_ON, S_OFF</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S_SELOK</td>
<td>BOOL</td>
<td>STEP_SELECT_OK: Apply value in S_SEL for S_NO</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>S_ON</td>
<td>BOOL</td>
<td>STEP_ON: Manual mode: Activate displayed step</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S_OFF</td>
<td>BOOL</td>
<td>STEP_OFF: Manual mode: Deactivate displayed step</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>T_PREV</td>
<td>BOOL</td>
<td>PREVIOUS_TRANSITION: Display previous valid transition in T_NO</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>T_NEXT</td>
<td>BOOL</td>
<td>NEXT_TRANSITION: Display next valid transition in T_NO</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>T_PUSH</td>
<td>BOOL</td>
<td>PUSH_TRANSITION: Transition switches, when condition is satisfied and T_PUSH (Edge) Requirement: Automatic mode or manual mode</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Output parameters

Overview of the GRAPH block output parameter

The following table provides an overview of the GRAPH block output parameters:

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Data type</th>
<th>Description</th>
<th>Standard parameter record</th>
<th>Maximum parameter record</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENO</td>
<td>BOOL</td>
<td>Output enable, if the FB is active, ENO has in the faultless state the value 1, otherwise 0</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S_NO</td>
<td>INT</td>
<td>STEP_NUMBER: Display step number</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S_MORE</td>
<td>BOOL</td>
<td>MORE_STEPS: Other steps are active</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S_ACTIVE</td>
<td>BOOL</td>
<td>STEP_ACTIVE: Displayed step is active</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S_TIME</td>
<td>TIME</td>
<td>STEP_TIME: Step activation time</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>S_TIMEOK</td>
<td>TIME</td>
<td>STEP_TIME_OK: Step activation time faultless</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>S_CRITLOC</td>
<td>DWORD</td>
<td>STEP_CRITERIA_INTERLOCK: Interlock criteria bits</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>S_CRITLOC</td>
<td>DWORD</td>
<td>S_CRITERIA_IL_LAST_ERROR:</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>ERR</td>
<td>Interlock criteria bits in the case of event L1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S_CRTSUP</td>
<td>DWORD</td>
<td>STEP_CRITERIA_SUPERVISION: Supervision criteria bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S_STATE</td>
<td>WORD</td>
<td>STEP_STATE: Step state bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T_NO</td>
<td>INT</td>
<td>TRANSITION: Valid transition number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T_MORE</td>
<td>BOOL</td>
<td>MORE_TRANSITIONS: Other valid transition numbers available for display</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T_CRT</td>
<td>DWORD</td>
<td>TRANSITION_CRITERIA: Transition criteria bits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T_CRTOLD</td>
<td>DWORD</td>
<td>T_CRITERIA_LAST_CYCLE: Transition criteria bits from the last cycle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T_CRTFLT</td>
<td>DWORD</td>
<td>T_CRITERIA_LAST_FAULT: Transition criteria bits in the case of event V1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERROR</td>
<td>BOOL</td>
<td>INTERLOCK_ERROR: Interlock error (any step)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FAULT</td>
<td>BOOL</td>
<td>SUPERVISION_FAULT: Supervision error (any step)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERRFLT</td>
<td>BOOL</td>
<td>IL_ERROR_OR_SV_FAULT: General fault</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQ_ISOFF</td>
<td>BOOL</td>
<td>SEQUENCE_IS_OFF: Sequence is turned off (no step active)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQ_HALT</td>
<td>BOOL</td>
<td>SEQUENCE_IS_HALT: Sequence is halted</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TM_HALTED</td>
<td>BOOL</td>
<td>TIMES_ARE_HALT: Timers are paused</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OP_ZEROED</td>
<td>BOOL</td>
<td>OPERANDS_ARE_ZEROED: Operands are reset</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IL_ENABLED</td>
<td>BOOL</td>
<td>INTERLOCK_IS_ENABLED: Interlock is taken into account</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SV_ENABLED</td>
<td>BOOL</td>
<td>SUPERVISION_IS_ENABLED: Step monitoring is taken into account</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACKREQ_ENABLED</td>
<td>BOOL</td>
<td>ACKNOWLEDGE_REQUIRED_IS_ENABLED: Acknowledgement required is activated</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSKIP ENABLED</td>
<td>BOOL</td>
<td>STEP_SKIPPING_IS_ENABLED: Step skipping is activated</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SACT_DISP</td>
<td>BOOL</td>
<td>ACTIVE_STEPS_WERE_DISPLAYED: Display only active steps in S_NO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEF_DISP</td>
<td>BOOL</td>
<td>STEPS_WITH_ERROR_FAULT_WERE_DISPLAYED: Display only steps with errors and disrupted steps in S_NO</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Static parameter

This chapter contains the following information:
- Step structure setup (S7-300/400)
- Transition structure setup (S7-300/400)
- Other static parameters (S7-300/400)

Step structure setup

Step structure in the interface

For each step that is located in the GRAPH FB sequencer, a structure with the following parameters is created in the section "Static":

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Data type</th>
<th>Memory space minimized</th>
<th>Default DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Step is activated</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>L1</td>
<td>Interlock is outgoing</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>V1</td>
<td>Supervision incoming</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>R1</td>
<td>Reserved</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>CC 1</td>
<td>Error is acknowledged</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S0</td>
<td>Step is deactivated</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>L0</td>
<td>Incoming interlock</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>V0</td>
<td>Outgoing supervision</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>X</td>
<td>Step is active</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>LA</td>
<td>Interlock is unsatisfied</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>VA</td>
<td>Pending supervision</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>RA</td>
<td>Reserved</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>AA</td>
<td>Reserved</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SS</td>
<td>System internal</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
Access to components

A writing access is not recommended because the parameter values influence the sequence.

Reading access takes place via the step name followed by the structure element name. In conditions or actions, e.g. the following accesses are thereby possible:

- TROCKNEN.X
- TROCKNEN.T

Transition structure setup

Transition structure in the interface

For each transition that is located in the GRAPH FB sequencer, a structure with the following parameters is created in the section "Static":

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Data type</th>
<th>Memory space minimized DB</th>
<th>Default DB:</th>
</tr>
</thead>
<tbody>
<tr>
<td>TV</td>
<td>Transition is valid</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>TT</td>
<td>Transition is satisfied</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>TS</td>
<td>Transition is switching</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>CF_IV</td>
<td>Entry CRIT_FLT is invalid</td>
<td>BOOL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Access to components
A writing access is not recommended because the parameter values influence the sequence.
Reading access takes place via the transition name followed by the structure element name. In conditions or actions, e.g. the following accesses are thereby possible:
- VENTIL_EIN.TT
- VENTIL_EIN.CRIT

Other static parameters

Sequence state setup:
The following information also belongs to the static parameters:
- Internal parameters S_DISPLAY, S_SEL_OLD, S_DISPIDX, T_DISPIDX
- Components "Operating mode"
- Components "Clock speeds"
- Components "Sequence bit memory"

Note
In a GRAPH-specific instance DB more parameters are contained. They are used however only for internal application. The use of these parameters in the program is not recommended.

Internal parameters S_DISPLAY, S_SEL_OLD, S_DISPIDX, T_DISPIDX
The following table shows internal parameters that are located in the section "Static". They are used to manage the step and transition numbers:

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Data type</th>
<th>Length in bytes</th>
<th>Memory space minimized DB</th>
<th>Default DB</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_DISPLAY</td>
<td>Copy of the output parameters S_NO</td>
<td>INT</td>
<td>2</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S_SEL_OLD</td>
<td>Step number from S_SEL from the last CPU cycle</td>
<td>INT</td>
<td>2</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S_DISPIDX</td>
<td>Step index for the step in S_NO</td>
<td>BYTE</td>
<td>1</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>T_DISPIDX</td>
<td>Transition index for the transition in T_NO</td>
<td>BYTE</td>
<td>1</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Operating mode
The following table shows the "MOP" structure components in the section "Static" that manage the operating modes:

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Data type</th>
<th>Memory space minimized DB</th>
<th>Default DB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUTO</td>
<td>Switch to automatic mode</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>MAN</td>
<td>Switch to manual mode</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>TAP</td>
<td>Switch to operating mode Jog</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>TOP</td>
<td>Switch to operating mode automatic or switching</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>ACK_S</td>
<td>Requirement: Acknowledge step, which is indicated on parameter in S_NO</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>REG_S</td>
<td>Requirement: Register step, which is indicated on parameter in S_NO</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>T_PREV</td>
<td>Requirement: Display the previous valid transition on the parameter T_NO</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>T_NEXT</td>
<td>Requirement: Display the next valid transition on the parameter T_NO</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>LOCK</td>
<td>Turn on /turn off interlock handling</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SUP</td>
<td>Turn on /turn off supervision handling</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>ACKREQ</td>
<td>Requirement: Acknowledge step, which is indicated on parameter in S_NO</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S_SKIP</td>
<td>Status: &quot;Skip step&quot; is activated</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>OFF</td>
<td>Deactivate sequence</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>INIT</td>
<td>Activate initial steps</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>HOLD</td>
<td>Pause/run sequence</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>TMS_HALT</td>
<td>Pause/run timers</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>OPS_ZERO</td>
<td>Switch off/switch on actions</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SACT_DISP</td>
<td>Status: Display only active steps</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SEF_DISP</td>
<td>Status: Display only steps with errors and disrupted steps</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SALL_DISP</td>
<td>Status: Display all steps</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S_PREV</td>
<td>Operating mode automatic: Display previous simultaneously active step in S_NO Operating mode manual: Display next smaller step number</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S_NEXT</td>
<td>Operating mode automatic: Display next simultaneously active step in S_NO Operating mode manual: Display next higher step number</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S_SELOK</td>
<td>Requirement: Indicate step number that is displayed on parameter S_SEL on parameter S_NO</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S_ON</td>
<td>Operating mode manual: Activate displayed step</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>S_OFF</td>
<td>Operating mode manual: Deactivate displayed step</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>T_PUSH</td>
<td>Operating mode: Jog mode Transition switches, when the satisfied condition</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
Clock speeds
The following table shows the "TICKS" structure components in the section "Static" that manages the information for activation time calculation. This information can be evaluated only internally in the system.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Data type</th>
<th>Memory space minimized DB</th>
<th>Default DB:</th>
</tr>
</thead>
<tbody>
<tr>
<td>REG</td>
<td>Requirement: Register all interlock and supervision windows</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>ACK</td>
<td>Acknowledgement of the fault</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>IL_PERM</td>
<td>Permanent interlock processing</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>T_PERM</td>
<td>Permanent transition processing</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>ILP_MAN</td>
<td>Permanent interlock processing for operating mode manual</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Sequence bit memory
The following table shows the "SQ_FLAGS" structure components in the section "Static" that monitor the internal sequence status:

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Data type</th>
<th>Memory space minimized DB</th>
<th>Default DB:</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERR_FLT</td>
<td>General fault</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>ERROR</td>
<td>Interlock error</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>FAULT</td>
<td>Monitoring error</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>RT_FAIL</td>
<td>Runtime error</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>NO_SNO</td>
<td>Selected step does not exist</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>NF_OFL</td>
<td>Too many steps that have to be activated or deactivated</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SA_OFL</td>
<td>Too many active steps</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>TV_OFL</td>
<td>Too many valid transitions</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>MSG_OFL</td>
<td>Too many alarms</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>NO_SWI</td>
<td>Do not switch in this cycle</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>CYC_OP</td>
<td>Cyclic processing of the sequence after initialization</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>AS_MSG</td>
<td>Activate or deactivate alarms during runtime on instruction</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>AS_SEND</td>
<td>Send alarms from WR_USMSG or enter only in diagnostics buffer</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>SQ_BUSY</td>
<td>Internal edge memory bit for sequence processing</td>
<td>BOOL</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
Access to components
A writing access is not recommended because the parameter values influence the sequence. Read access is however possible.

### Select parameter record

**Introduction**
You can set the parameter record used in two ways:
- Define default for generation of new blocks
- Change the parameter record for an opened block

**Note**
You can manually insert or delete individual input and output parameters in the interface anytime.

**Define default for generation of new blocks**
To define the default, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > GRAPH" group in the area navigation.
3. Select or clear the "Maximum interface set" option.
   If the option is selected, the maximum parameter record is used, if the option is cleared, the default parameter record is used.

**Change the parameter record for an opened block**
To change the parameter record of an opened block, follow these steps:

1. Select in the "Edit" menu the "Maximum interface set" command or the "Default interface set" command.
   The interface of the GRAPH block is adapted.

### Select memory space model

**Introduction**
You can set the memory space model in two ways:
- Define default for generation of new blocks
- Change the memory space model for an opened block

**Note**
You cannot manually insert GRAPH-specific parameters that are not available in the minimized data block in the interface.
To define the default, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > GRAPH" group in the area navigation.
3. Select or clear the "Create minimized DB" option.
   If the option is selected, the minimized memory space model is used, if the option is cleared, the default memory model is used.

**Change the memory space model for an opened block**
To change the memory space model of an opened block, follow these steps:

1. Select "Attributes" in the inspector window.
   The "Attributes" tab opens.
2. Select or clear the "Create minimized DB" option in the "Compile" group.
   The interface of the GRAPH block is adapted.

---

**Compile and load GRAPH-FB**

This chapter contains the following information:

- Special factors during GRAPH-FB compilation (S7-300/400)
- Special factors during loading of GRAPH-FB (S7-300/400)

---

**Special factors during GRAPH-FB compilation**

The method of GRAPH block compilation is the same as of compiling blocks that are created in other programming languages.

**Procedure**

To compile a GRAPH block in the program editor, follow these steps:

1. Open a GRAPH block in any view.
2. Select the "Compile" command in the shortcut menu.

**Result**

The code for the block is generated.

If the block to be compiled has block calls in which the interface has changed, the block calls are updated.

The alarm under "Info > Compile" in the inspector window reports whether the compilation was successful.

Depending on the memory space model, a GRAPH-specific standard FC is copied during the compiling in the "System blocks > Program resources" folder, if no FC of that type is available there yet. These standard FCs are internally required for the memory space optimization of GRAPH programs. In the case of the minimized memory space model, FC 73 is used, in the case of default memory space model, FC 72 is used.

**Reference**

For additional information on compiling, refer to "See also."

---

**Special factors during loading of GRAPH-FB**

This chapter contains the following information:
Basic information on GRAPH-FB loading

Sequencer characteristics during loading
If the instance DB is loaded, a sequencer programmed with GRAPH has different characteristics after loading:

- Loading of FB without instance DB
  The structural element, which was active before loading, is reactivated after successful loading. The program can continue to run bumplessly.

- Loading the FB with instance DB
  The sequencer processing restarts with the initialization step. Problems with the process can occur during the sequencer synchronization. For example, some required step actions could not be executed. Operands, which had to be reset before the step exit, would stay set.

Before you load a GRAPH-FB together with the instance DB, deactivate the sequencer.

Reference
For additional information on loading, refer to "See also."

Deactivate the sequencer before loading

You have the following options to deactivate the sequencer before loading:

- Set a general setting to deactivate the sequencer
- Deactivate the sequencer during the loading process

Set a general setting to deactivate the sequencer
To set the deactivation of the sequencer centrally before the loading, follow these steps:

1. Select the "Settings" command in the "Options" menu.
   The "Settings" window is displayed in the work area.
2. Select the "PLC programming > GRAPH" group in the area navigation.
3. Activate the "Turn off sequence before downloading DB" option.
   If the instance DB has to be loaded during a loading process, the sequencer is automatically deactivated beforehand.

Deactivate the sequencer during the loading process
To deactivate the sequence during the loading process, follow these steps:

1. Download the GRAPH block to the device.
   During the loading process the "Load preview" dialog opens. This dialog displays alarms and suggests the required actions for loading.
   If the instance DB is also loaded, the "Turn off sequence before downloading DB" action is displayed.
2. Activate the action.

Reference
For additional information on loading, refer to "See also."

Load GRAPH-FB
The method of GRAPH block loading is the same as of loading blocks that were created in other programming languages.

**Load GRAPH programs consistently**

During the loading process, GRAPH analyses if related blocks have to be loaded together with the GRAPH-FB, e.g. the related instance DB or the GRAPH-specific standard FCs. If necessary, you receive a message during the loading for additional blocks that have to be loaded, and you can also download all the required blocks via the "Consistent download" function.

**Procedure**

To download GRAPH block, follow these steps:

1. Select the block in the project tree.
2. Select the "Download to device > Software" command in the shortcut menu.
   The "Extended loading" dialog opens.
3. Select the type of the PG/PC interface.
4. Click the "Download" button.
   The "Load preview" dialog opens. This dialog displays messages and proposes actions necessary for loading.
   - If additional blocks have to be downloaded, the blocks are displayed. In this case select the the "Consistent download" option in the "Action" column.
   - If the instance DB is also loaded, the "Turn off sequence before downloading DB" action is displayed. Activate the action.
5. Click on "Download".

**Reference**

For additional information on loading, refer to "See also."