mapp Services

1 mapp Sequence: Dynamic machine processes

mapp Sequencer provides management for a wide range of machine processes.

1.1 Concept

- PictureLink_Configuration
- PictureLink_IEC
- PictureLink_Diagnosis
- PictureLink_Visualization

1.1.1 Terminology and definitions

This section lists and explains all relevant terminology and definitions.

Sequence

A sequence consists of several steps that each execute a certain command. The user defines the order as well as the command for each step.
**Step**

A step is an executable unit in the sequence. A step is defined via a command and the associated parameters.

<table>
<thead>
<tr>
<th>Command</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command.Timeout</td>
<td></td>
<td>0.00</td>
</tr>
<tr>
<td>Time</td>
<td></td>
<td>2000.00</td>
</tr>
<tr>
<td>BlinkTime</td>
<td></td>
<td>100.00</td>
</tr>
</tbody>
</table>
Command

A command defines executable commands for a machine. Examples include an absolute movement of an axis or setting a PV to a certain value.

Commands are predefined, can be structured in a hierarchy (column "Group" in the image) and define the options for the machine. The hierarchy is displayed on widget Sequencer.

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Appearance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Commands</td>
</tr>
<tr>
<td></td>
<td>LedYellow</td>
</tr>
<tr>
<td></td>
<td>LedRed</td>
</tr>
<tr>
<td></td>
<td>LedBlue</td>
</tr>
</tbody>
</table>

Parameter

Parameters are used to provide a command with additional information (e.g. target position of an axis). Command "Movement" can be combined with any number of parameters. All other Commands have predefined parameters.

- Configuration

- Linking
Interlocks
Interlocks define conditions in which command "Movement" is not permitted to be executed. An interlock can be defined directly on command "Movement" (static interlock) and then queried each time the command is executed. Axes can also be interlocked with one another (dynamic interlocks). In this case, the system monitors whether the axes behave as expected within the sequence.

<table>
<thead>
<tr>
<th>Dynamic interlock</th>
<th>Static interlock</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part</td>
<td>Start</td>
</tr>
<tr>
<td>Part</td>
<td>LedYellow</td>
</tr>
<tr>
<td>Part</td>
<td>LedBlue</td>
</tr>
</tbody>
</table>

Macros
A macro can be used to group together several steps in a sequence, e.g. to simplify the display of a complex sequence. For more information about macros, see here.

Axes
An axis defines a moving part on a machine (e.g. cylinder). An axis can be used in combination with actuators (for priority control) and interlocks. Each axis has its own state (e.g. in motion, at end position, stopped, etc.). For more information about axes, see Axes and actuators.

<table>
<thead>
<tr>
<th>Axis: Start</th>
<th>Name</th>
<th>Start</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis: LedYellow</td>
<td>Name</td>
<td>LedYellow</td>
</tr>
<tr>
<td></td>
<td>Axis State</td>
<td>Axis information</td>
</tr>
<tr>
<td>Axis: LedBlue</td>
<td>Name</td>
<td>LedBlue</td>
</tr>
<tr>
<td></td>
<td>Axis State</td>
<td>Axis information</td>
</tr>
</tbody>
</table>

Actuators
An actuator is used to limit simultaneous access to machine units (e.g. controlling a pump). The actuator being used can be queried by the code. For more information about actuators, see Axes and actuators.
Priorities

Priorities are used in combination with actuators. If a higher priority is defined for one command over another that should be executed at the same time, then the command with the higher priority is the one that accesses the actuator. For more information, see Using priorities.

Example

Assume the following system:

3 hydraulic cylinders are controlled by 3 servo pumps. Pumps "P1" and "P2" are needed to extend cylinder 1 ("C1"). Since a pump cannot move several cylinders at a time, it must be ensured that each pump is only used by one cylinder at any time.

1.1.2 Axes and actuators

An axis defines a movable unit on a machine (e.g. cylinder). An actuator is used to assign an axis movement to an element that generates force (e.g. pump).

Axes are used to record the state of a movable unit. Actuators are used to limit the access to an actuator so that an actuator can only be requested by only one command at a time.
Configuration of the cylinders:
For this machine, the cylinders represent axes for MpSequence. They are used to represent a certain machine function.

<table>
<thead>
<tr>
<th>Axes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis: C1</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>C1</td>
</tr>
<tr>
<td>Axis State</td>
<td></td>
</tr>
<tr>
<td>Axis: C2</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>C2</td>
</tr>
<tr>
<td>Axis State</td>
<td></td>
</tr>
<tr>
<td>Axis: C3</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>C3</td>
</tr>
<tr>
<td>Axis State</td>
<td></td>
</tr>
</tbody>
</table>

Configuration of the pumps:
The pumps represent the actuators in this case. The use of the pumps is limited. Each pump is only permitted to be used by only one cylinder at a time.
Actuators "P1", "P2" and "P3" directly represent the actuators for the pumps. Actuators "Comb1", "Comb2" and "Comb3" represent a combination of these individual actuators. This makes it possible to ensure that each pump is used simultaneously for max. one "movement".

In this case, it would not be possible to extend cylinder "C3" if another cylinder is being extended or retracted since the extension of "C3" already requires all actuators ("Comb3").

It is still possible to retract cylinders "C2" and "C3" simultaneously, however. Only actuator "P1" is needed to retract "C2"; actuator "Comb2" ("P2" and "P3") is needed to retract "C3".

It would not be possible to extend cylinders "C1" and "C2" simultaneously. Cylinder "C2" uses actuator "P1" for extension, while "C1" needs actuator "Comb1" ("P1" and "P2")..

1.1.3 Definition of dynamic interlocks

A dynamic interlock can be used to protect the tool or product. Dynamic interlocks check whether the sequence is executed correctly. This means they check whether all steps before a step to be executed were executed correctly and no more axes were moved. This ensures that no axis has been moved outside the sequence by another program part or mechanically since this would prevent the sequence from running correctly.

The state of an axis is always compared with "FinalState" of the command that was executed before the current command and used the interlocking axis.
If the same axis is used in several parallel branches, `MpsSequenceCore` can no longer ensure that the correct status of the axis is queried within the parallel execution and afterwards since it cannot be determined which movement ends first. A clear decision as to which state of the axis must be queried is only possible again if an axis has only moved in one branch.

If a dynamic interlock is used already in the first step of a sequence, then the state of the axis must be compared with "FinalState" of the last command using this axis.

**Example**

4 axes are defined in the configuration. The dynamic interlocks are defined such that all axes interlock each other.

<table>
<thead>
<tr>
<th>Configuration of the axes</th>
<th>Configuration of the interlocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axes 1</td>
<td>Axis 1</td>
</tr>
<tr>
<td>Name</td>
<td>Interlocks</td>
</tr>
<tr>
<td>Axis State</td>
<td>Axis 2</td>
</tr>
<tr>
<td>Axes 2</td>
<td>Interlocks</td>
</tr>
<tr>
<td>Name</td>
<td>Axis 3</td>
</tr>
<tr>
<td>Axis State</td>
<td>Interlocks</td>
</tr>
<tr>
<td>Axes 3</td>
<td>Axis 4</td>
</tr>
<tr>
<td>Name</td>
<td>Interlocks</td>
</tr>
<tr>
<td>Axis State</td>
<td></td>
</tr>
</tbody>
</table>

Representation of the steps:

The number of the current step is also used as "FinalStep" for that step. Step "1" is defined as follows in this case:
The "interlocking axes" define the axes for which the current state is compared with "FinalState".

Sequence used:
Due to the dynamic interlocks, each step would have to check the state of each other axis. There are exceptions here, however, due to the parallel branches.

**Step 1**
- "Axis2" is moved only once in this sequence (step 2). For this reason, "FinalState" of step 2 must be compared with the state of "Axis2" in this step.
- "Axis3" is moved the last time in step 7. For this reason, "FinalState" of step 7 must be compared with the state of "Axis3" in this step.
- "Axis4" is moved only once in this sequence (step 6). For this reason, "FinalState" of step 6 must be compared with the state of "Axis4" in this step.

**Step 2**
Step 2 contains a parallel branch. Since the state of the interlocking axes is not defined here, fewer interlocks need to be checked here.
- "Axis1": Moved in step 1. For this reason, "FinalState" of step 1 is compared with the state of "Axis1" in this step.
• "Axis3": A movement of "Axis3" is performed in a parallel branch (step 3 and step 5). For this reason, the state of the axis cannot be identified.
• "Axis4": A movement of "Axis4" is performed in a parallel branch (step 6). For this reason, the state of the axis cannot be identified.

Step 3
• "Axis1": A movement of "Axis1" is performed in a parallel branch (step 4). For this reason, the state of the axis cannot be identified.
• "Axis2": A movement of "Axis2" is performed in a parallel branch (step 2). For this reason, the state of the axis cannot be identified.
• "Axis4": This axis is not used in a parallel branch. For this reason, the status of the axis can be compared with "FinalState" from step 6.

Step 4
• "Axis2": Not used in a parallel branch. For this reason, the status of the axis is compared with "FinalState" from step 2.
• "Axis3": A movement of "Axis3" is performed in a parallel branch (step 3 and step 5). For this reason, the state of the axis cannot be identified.
• "Axis4": A movement of "Axis4" is performed in a parallel branch (step 6). For this reason, the state of the axis cannot be identified.

Step 5
• "Axis1": A movement of "Axis1" is performed in a parallel branch (step 4). For this reason, the state of the axis cannot be identified.
• "Axis2": A movement of "Axis2" is performed in a parallel branch (step 2). For this reason, the state of the axis cannot be identified.
• "Axis4": A movement of "Axis4" is performed in a parallel branch (step 6). For this reason, the state of the axis cannot be identified.

Step 6
• "Axis1": A movement of "Axis1" is performed in a parallel branch (step 4). For this reason, the state of the axis cannot be identified.
• "Axis2": A movement of "Axis2" is performed in a parallel branch (step 2). For this reason, the state of the axis cannot be identified.
• "Axis3": A movement of "Axis3" is performed in a parallel branch (step 5). For this reason, the state of the axis cannot be identified.

Step 7
• "Axis1": The state of this axis is compared with "FinalState" from step 4.
• "Axis2": The state of this axis is compared with "FinalState" from step 2.
• "Axis4": The state of this axis is compared with "FinalState" from step 6.

1.1.4 Start mode of the sequence

Normal (mpSEQUENCE_START_NORMAL)

The sequent is started at its starting point. The starting point is defined by a step with command "SequenceStart". This command must also be the first step in the sequence.
Manual selection of the step (mpSEQUENCE_START_SELECT_STEP)

Mode `mpSEQUENCE_START_SELECT_STEP` is defined in the parameter structure for the start of the sequence (`MpSequenceCoreStartType`). Parameter "Step" defines the step at which the sequence is started.

In this example, step "4" is specified to start the sequence.
Automatic selection of the step (mpSEQUENCE_START_AUTO_STEP)
(reserved for future use)

Mode mpSEQUENCE_START_AUTO_STEP is defined in the parameter structure for the start of the sequence (MpSequenceCoreStartType). The sequencer will then select the first executable step as the starting point for the sequence based on the interlocks and conditions.

In this example, step "4" is the first executable step. All preceding steps are blocked (by interlocks).
1.1.5 Execution mode of the sequence

**Automatic (mpSEQUENCE_MODE_AUTOMATIC)**

In automatic mode, the sequence is executed based on the definition in the sequence file. Timeouts and conditions are checked. After the end of the sequence is reached, it starts again from the first step.
Semi-automatic or single-cycle (mpSEQUENCE_MODE_SINGLE_CYCLE)

Semi-automatic mode is the same as automatic mode. The only difference is that the sequence does not restart when it has ended.
Single-step (mpSEQUENCE_MODE_SINGLE_STEP)
In this mode, only one step is executed at a time. Each step must be started with a positive edge on input "Start" of MpSequenceCore.
Manual (mpSEQUENCE_MODE_MANUAL)

In manual mode, only one step is executed at a time. The command to be executed is specified in MpSequence-ManualType. Here, the first executable step (all interlocks are okay) is executed with the specified command. Step "2" is executed in this example.
Single command (mpSEQUENCE_MODE_SINGLE_COMMAND)

In this mode, only one command is executed. The command to be executed is defined in structure MpSequenceSingleCommandType. For this command, all interlocks and conditions defined in this configuration are checked. The command is only executed after this check has taken place. In addition, parameters can be specified for the command to be executed.

Initialization/Exit sequence

In automatic and semi-automatic mode, it is possible to execute an initialization or exit sequence. These sequences are executed only once before or after the sequence, respectively.

1.1.6 Stop mode of the sequence

Immediate (mpSEQUENCE_STOP_IMMEDIATE)

If this mode is selected, the sequence is stopped immediately on a stop command. This mode behaves the same as a falling edge on input "Start".
End of step (mpSEQUENCE_STOP_END_OF_STEP)

If mode `mpSEQUENCE_STOP_MODE_END_OF_STEP` is defined in structure `MpSequenceCoreStopType`, the sequence is only ended on a stop command after all active steps have completed execution.
End of cycle (mpSEQUENCE_STOP_END_OF_CYCLE)

If mode `mpSEQUENCE_STOP_MODE_END_OF_CYCLE` is defined in structure `MpSequenceCoreStopType`, the sequence is only ended on a stop command after the current cycle.
1.1.7 Using priorities

Priorities can be defined for command "Movement".

If this priority is used in combination with an actuator, then the command with the higher priority has access to the actuator if two commands that want to use the same actuator are started simultaneously.

The actuator must be configured as "single access"!

The actuator available for the command is passed on to the application and can be queried there. The application can then react according to the actuator being used (e.g. whether pump 1 or pump 2 is being used).

In this example, command "LedBlue" is given priority 1 while command "LedRed" is given priority 0. Both commands have been configured to use actuator P1. Because command "LedBlue" has a higher priority, this command has access to actuator P1. Command "LedRed" will then try to access its fallback actuator. In this example, this is actuator P2.
If 2 commands have the same priority, then the command started first will have access to the actuator. In the following case, command "LedRed" is started before command "LedBlue" since step "LedYellow" requires more time to execute than step "LedRed". For this reason, command "LedRed" accesses actuator P1 and command "LedBlue" tries to access its fallback actuator.
1.1.8 Structure of the sequence file

The sequence file consists of XML nodes that define the structure of the sequence.

- Each sequence must begin with a step from command "SequenceStart"!
- Each sequence must end with a step from command "SequenceEnd"!

Example of a complete sequence file with an inline macro:

For more information about macros, see Macros.

```
<?xml version="1.0"?>
<Data>
  <Sequence>
```
DATA
Node "DATA" is the root node of the XML file.

Sequence
Node "Sequence" defines a sequence. It must be located within node "DATA".

Macro
Node "Macro" can be used to better group elements together. It must be located within node "Sequence". A name and value must be specified for node "Macro". The name can be used as a reference to external files. "Value" is the instance name for referencing purposes.

Step
Node "Step" defines a step in the sequence. It must be defined either within node "Sequence" or node "Macro". Each "Step" node must have a unique value.

  Each "Step" node requires node "Command".
<Sequence>
  <Step Value="0">
    ...
  </Step>
</Sequence>

<Macro Name="MyMacro" Value="MyMac">
  <Step Value="Step8">
    ...
  </Step>
</Macro>

Command
Node "Command" must be located within node "Step". It requires the name of one of the commands defined in the configuration as a value.

  Each "Step" node is only permitted to have one "Command" node.

<Step Value="StepX">
  <Command Value="MyCommand"/>
</Step>

Parameter
Node "Parameter" must be located within node "Step". This node can be used to pass parameters to the command. The parameters must have already been linked in the configuration.

The name specified in the configuration must be defined for each parameter. In addition, each parameter must also have a value.

<Step Value="StepX">
  <Command Value="MyCommand"/>
  <Parameter Name="MyFirstParameter" Value="1"/>
  <Parameter Name="MySecondParameter" Value="10"/>
</Step>

Timeout
Node "Timeout" must be located within node "Step". The time that should be waited until a timeout occurs is specified under "Value". The value for the timeout is specified in seconds.

<Step Value="StepX">
  <Command Value="MySetPvCommand"/>
  <Timeout Value="1"/>
</Step>

PvValue
Node "PvValue" must be located within a "Step" node that has "SetPv", "WaitForPV" or "ConditionalExecution" as a command. The PV to be modified or checked must be specified in the configuration.

<Step Value="StepX">
  <Command Value="MySetPvCommand"/>
  <PvValue Value="1"/>
</Step>

StepPrevious
Node "StepPrevious" defines the structure of the sequence. This node must be located within node "Step".

  The sequence file must contain a "Step" node without "StepPrevious".

The value of this node must refer to a value from another step.

  Creating a loop of steps is not permitted!

<Step Value="StepX">
  <Command Value="MyCommand"/>
  <Parameter Name="MyFirstParameter" Value="1"/>
  <Parameter Name="MySecondParameter" Value="10"/>
</Step>
1.1.9 Binding to mapp View

If MpSequence is used in combination with widget “Sequencer”, it is possible to edit or display the sequence visually. The widget provides several actions and events for this. The entire sequence can be edited as needed using this widget.

Requirements

The following commands must exist in the configuration:

**SequenceStart**: Defines the start of the sequence.
**SequenceEnd**: Defines the end of the sequence.

1.2 Guides

1.2.1 Getting started

1.2.1.1 Creating a project

You must first create a new project in Automation Studio.

1.2.1.2 Adding axes

A new axis is added to the hardware configuration. This axis should be called "gAxis01".
1.2.1.3 Adding MpAxisBasic

First, select the desired mapp component from the Toolbox and add the configuration to folder "Motion" using drag-and-drop.

Technology Package "ACP10/ARNC0 5.x.x" is required for this example!

After the configuration is added, a task is needed to control the axis.

This task is called "AxisCtrl".

In this task, all necessary variables are added first. To start, only one variable of type "MpAxisBasic" and one variable of type "MpAxisBasicParType" are needed.

If the variables have been added, MpAxisBasic is configured in the program code.

```plaintext
PROGRAM _CYCLIC
    MpAxisBasic_0.Enable := TRUE;
    MpAxisBasic_0.MpLink := ADR(gAxisBasic);
    MpAxisBasic_0.Parameters := ADR(Parameters);
    MpAxisBasic_0.Axis := ADR(gAxis01);
    MpAxisBasic_0();
END_PROGRAM
```

The parameters needed for a movement are also initialized. This can take place either in the initialization subroutine of the task or cyclically.

```plaintext
Parameters.Acceleration := 100;
Parameters.Deceleration := 100;
Parameters.Velocity := 10;
```

1.2.1.4 Creating the data storage device

The next step is to create the data storage device ("HD"). To do so, we need to select or create a folder in Windows. Then a data storage device (file device) must be created for this folder in Automation Runtime.
1.2.1.5 Adding MpSequenceCore

An MpSequenceCore configuration is added first.

The data storage device created earlier is then entered in the configuration. Parameter "TargetPosition" and axis "BasicAxis" are then configured.

Before command "Movement" is entered, "AxisCtrl" must be prepared. To do so, a new type is entered in the type file. This type contains all of the parameters specified in the configuration.

This type is then used to create a variable. This variable is the parameter structure used by the commands. A command structure is created in addition to the parameter structure. This is used to return information about the current state to MpSequence.

If all previous steps are completed, command "Movement" can be added. 3 commands are added to the configuration. Command "SequenceStart" should be starting point for the sequence. This command only checks whether the axis is ready to be moved. The second command is "Movement". This is used to move the axis to a certain position. The last command is "SequenceEnd". This command is only used as the endpoint of the sequence.
1.2.1.6 Configuring MpSequenceCore

First, a new task called "SequenceCtrl" is added.

After this task has been added, 2 variables are created.

MpSequenceCore is then configured and called in the cyclic code.

```plaintext
PROGRAM _CYCLIC

Parameters.Sequence := 'Sequence.xml';
MpSequenceCore_0.Enable := TRUE;
MpSequenceCore_0.MpLink := ADR(gSequencerCore);
MpSequenceCore_0.Parameters := ADR(Parameters);
MpSequenceCore_0();

END_PROGRAM
```

1.2.1.7 Writing control logic for axis movement

After MpSequence and MpAxisBasic are working, a "switch case" is added to task "AxisCtrl". First, a step is needed that does not execute any actions (axis standstill). The sequencer is informed in this step that the axis is not executing any actions.

```plaintext
CASE State OF
  0:
    CommandStructure.MovementState := mpSEQUENCE_MOVEMENT_IDLE;
    IF CommandStructure.Start THEN
      Parameters.Position := ParameterStructure.TargetPosition;
      MpAxisBasic_0.Update := TRUE;
      MpAxisBasic_0.MoveAbsolute := TRUE;
      State := 1;
    END_IF;
END_CASE
```

A step is then added in which the axis should be moved. In this step, MpSequence is informed by mpSEQUENCE_MOVEMENT_ACTIVE that a movement is currently active. When the movement is completed, the sequence is informed of this using mpSEQUENCE_MOVEMENT_DONE. This step is then considered completed by MpSequence and the next step is carried out.

```plaintext
CASE State OF
  0:
    CommandStructure.MovementState := mpSEQUENCE_MOVEMENT_IDLE;
    IF CommandStructure.Start THEN
      Parameters.Position := ParameterStructure.TargetPosition;
      MpAxisBasic_0.Update := TRUE;
      MpAxisBasic_0.MoveAbsolute := TRUE;
      State := 1;
    END_IF;
  1:
    CommandStructure.MovementState := mpSEQUENCE_MOVEMENT_ACTIVE;
    MpAxisBasic_0.Update := FALSE;
```
mapp Services

```plaintext
IF MpAxisBasic_0.InPosition THEN
    MpAxisBasic_0.MoveAbsolute := FALSE;
    CommandStructure.MovementState := mpSEQUENCE_MOVEMENT_DONE;
    State := 0;
END_IF;
END_CASE;
```

1.2.1.8 Creating the sequence file

The sequence file can be created in any text editor. The first step in the sequence file is "SequenceStart" form the configuration. This step only receives as a parameter the comparison value to which the PV specified in the configuration should be compared.

The second step is specified with command "AbsoluteMovement". This step receives as a parameter the target position of the axis. The previous step must also be defined. This is "Start" in this case. The last step and end of the sequence is "SequenceEnd". This step can be configured to wait a certain amount of time. In this case, this step is used to identify the end of the sequence.

This file is then saved to the folder configured on the data storage device.

1.2.1.9 Generating the file structure

Next, a file structure is generated for ARsim.

1.2.1.10 Testing the sequence in the program

First, monitor mode is enabled in task "AxisCtrl". The inputs and outputs of MpAxisBasic_0 look like this:
The axis is then switched on ("Power" = TRUE). Here, we must wait until output "PowerOn" = TRUE.

After the axis is switched on, it needs to be homed.
We then switch to task "SequenceCtrl" and enable monitor mode there. The inputs and outputs of MSequenceCore_0 must look like this:

Before the sequence can now be started, it must be imported. Here, we must wait until output "ImportDone" = TRUE.

The sequence is then started. Here, we must wait until output "Running" = TRUE.
As soon as execution of the sequence begins, the currently active step or currently executing command in the Info structure can be read out.

Whether the axis is following the sequence is then checked in task "AxisCtrl". Outputs "Position", "Velocity" and "MoveActive" are observed for this.

Widget "Sequencer" can be used to display the sequence visually. For information about how this widget must be configured, see section Concept.
1.3 Configuration

1.3.1 MpSequenceCore configuration

The configuration defines possible commands that can be used for a sequence. For certain commands, parameters or interlocks can additionally be defined to further influence the execution of the sequence. The execution sequence of commands and parameter values cannot be defined in the configuration. A sequence file is required for this. This sequence file can use the commands as needed and thus defines the execution of the sequence.

For more information about the structure of the sequence file, see Structure of the sequence file.

Parameter
Parameters are used to provide commands with additional information. For more information, see Parameter.

Axes
Axes can be used in Commands "Movement" to delay or stop the command. Dynamic interlocks can also be configured with axes. For more information, see Axes.

Actuators
Actuators can be used to limit the simultaneous access of multiple Commands to a machine unit. For more information, see Actuators.

Commands
Commands are executed in a sequence. A command can control a particular machine function (moving an axis, controlling LEDs, etc.). For more information, see Commands.

Dynamic interlocks
A dynamic interlock can limit access to an axis if other axes are already in use. For more information, see Dynamic interlocks.

Display
Here, all parameters can be configured with units and limits. For more information, see Display.

1.3.1.1 Parameter
Parameters can be used to provide additional information for command "Movement" in the sequence. If command "MoveAbsolute" is defined, it can be extended with parameters "MovementSpeed" and "TargetPosition".

Each parameter must have a unique name. The individual parameters are then assigned to a command. A parameter can also be assigned to several Commands. For example, parameter "MovementSpeed" can be used for command "MoveAbsolute" and command "MoveRelative".

A separate description text can also be defined for each parameter. The text system is used for this. "TextID" defines where the description text is located. If a sequence is then edited using widget "Sequencer", then a description text will be available there for each parameter.

Parameters can also be graphically modified and adjusted using widget "Sequencer".

1.3.1.2 Axes

The axes that can be used in the sequence are defined here. Each axis must have a unique name. A PV of data type DINT that defines the current state of the axis can be connected to "Axis state".

An axis defines a moving part on a machine (e.g. cylinder). Several commands can move the same axis.

For more information about axes, see Axes and actuators.

1.3.1.3 Actuators

An actuator can be used to limit access to a machine unit (e.g. a pump). The actuator being used must be checked in the code for this.

An actuator must have a unique name. There are several configuration options for actuators:
For more information about actuators, see Axes and actuators.

**Single access**

An actuator defined with "Single access" can only be used by one command. If a second command wants to use the actuator at the same time, MpSequence will not permit this access.

**Multiple access**

An actuator defined with "Multiple access" can be used by several commands simultaneously.

**Single actuator**

An actuator defined as "Single actuator" does not consist of several actuators.

**Multiple actuator**

An actuator defined as "Multiple actuator" consists of several different actuators. This makes it possible to assign multiple actuators to a command as a group. MpSequence then checks not only the group actuator, but the individual actuators in the group as well. Simple as well as complex mechanism can thus be defined in order to enable priority control for accessing the actuators (e.g. pumps).

### 1.3.1.4 Commands

A command describes an action that can be executed by the sequencer. Together with parameters, the command becomes a step. Multiple steps make up a sequence.

Possible commands:

- Start injection (injection molding)
- Extend cylinder
- Wait for a certain value (PV value)
- Set alarm

Each command must have a unique name. A separate description text can also be defined for each command. The text system is used for this. "TextID" defines where the description text is located.

If a sequence is then edited using widget "Sequencer", then a description text will be available there for each command.

The hierarchy of commands is used to visually display and group commands in the HMI application. Widget "Sequencer" is used for this visual display. Since there is a very wide range of different commands, they can be grouped together. In other words, individual commands are assigned to a group. The group is then displayed first in the HMI application when editing a sequence. Once selected, it is possible to select one of the commands assigned to that group. This results in several different hierarchy levels. For more information about how commands are displayed visually, see here.

There are several types of commands:

**Movement**

Command "Movement" can be used to perform any machine function. An axis as well as the command and parameter structures are assigned to the command in this case.

To execute command "Movement", a PV of type MpSequencerCommandType is specified under "Command structure". This command structure is used as an interface between the sequencer and a task.

In addition, parameters can be specified for command "Movement". For this purpose, a PV containing all the values specified under "Parameters" is required. For example, if "BlinkTime" is specified under "Parameter 1", then a value called "BlinkTime" must be included in the parameter structure. This PV is specified under "Parameter structure".

"Final state" informs the sequence of the value for the axis state when the movement is ended.

- **The axis state must always be set by the application!**

Execution of command "Movement" can also be limited by conditions (static interlocks). In this case, either the state of an axis or a PV is compared to an expected value. If this condition is met, the reaction specified under "Reaction" is carried out.

Possible reactions:
- **Wait**: The command waits until the condition is met before execution.
- **Stop**: The sequence is ended with an error.

These interlocks can be used to prevent undesirable movements of the machine if these can endanger the machine or are not possible from the process sequence. It makes it possible to create any number of protection functions that are checked before the command is started and during its execution. For example, a static interlock can prevent an axis from moving if it would collide with another axis due to its position.

In addition, an actuator can be assigned to command "Movement".

The priority of the command can be set under the advanced parameters. If an actuator has been assigned, then the command with higher priority is always executed. If a parallel command has the same actuator, then the “fallback actuator” is used according to the settings.

**Interface to the task**

The interface to the task is defined by the "command structure" and "parameter structure". The command structure must be structured as follows:

<table>
<thead>
<tr>
<th>Element</th>
<th>Data type</th>
<th>Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>BOOL</td>
<td>Sequencer</td>
<td>The sequencer sets this PV to TRUE when the step should be started. It is re-set when the step is ended or stopped by the sequencer.</td>
</tr>
<tr>
<td>Stop</td>
<td>BOOL</td>
<td>Sequencer</td>
<td>The sequencer sets this PV whenever a stop is requested.</td>
</tr>
<tr>
<td>Actuator</td>
<td>STRING</td>
<td>Sequencer</td>
<td>This element contains information about which actuator is being used for the current movement. The sequencer takes into account the priority and fallback actuator.</td>
</tr>
<tr>
<td>CurrentMode</td>
<td>MpSequenceModeEnum</td>
<td>Sequencer</td>
<td>Optional: This element (if present) contains information about the currently selected mode.</td>
</tr>
<tr>
<td>MovementState</td>
<td>MpSequenceMovementStateEnum</td>
<td>Task</td>
<td>This element is used to provide the sequencer information about the state of the axis. The sequencer monitors this element and reacts to certain values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• IDLE: The sequence is waiting.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• ACTIVE: Command &quot;Start&quot; was evaluated and a movement is active. The timeout check is active.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• DONE: The movement has completed. The sequencer can continue executing the sequence.</td>
</tr>
</tbody>
</table>

The parameter structure can be structured as needed. It must contain all the values specified under "Parameters", however.

The values for the parameters in the structure are specified in the sequence file using `<Parameter Name="My-ParameterName" Value="xxx"/>`. These values must then be evaluated in the task. A parameter can be used to define the target position of an axis. Each use of the command can therefore specify a new target position in the sequence file.

**Set PV**

"Set PV" allows a process variable to be assigned a certain value. The name of the PV must be specified.

The new value of the PV must be specified in the sequence file with `<PvValue Value="xxx"/>`. For information about specifying comparison values, see Structure of the sequence file.

**Wait for time**

Command "Wait for time" can be used to delay the execution of the next step by a certain amount of time.

If the command is configured using widget "Sequencer", the parameter is displayed automatically in the parameter list.

A PV of data type REAL can also be defined as the source. The time is specified in seconds.

If the command is entered externally in the sequence, then a node of type "Duration" must exist in the command.

```xml
<Duration Value="100"/>
```
Wait for process variable

Command "Wait for PV" can be used to wait until a PV meets a certain condition before the next step is executed. A PV must be specified for this command. This PV is then compared to a value based on the selected condition. The comparison value must be specified in the sequence file with `<PvValue Value="xxx"/>`. The sequence continues with the next step only when the condition is met.

For information about specifying comparison values, see Structure of the sequence file.

Set alarm

This command triggers an alarm. The name of the alarm to be triggered is specified under "Alarm".

The alarm must already be defined in MpAlarmX!

Conditional execution

"Conditional execution" can be used to execute or skip certain steps in a sequence based on a condition. A PV and condition are specified for this. The PV is compared to a value based on the condition. If the condition is met, the steps are executed in accordance with the conditional execution. Otherwise, the sequence at the end of the condition is executed. The end is defined by a command of type "End conditional execution".

Each "Conditional execution" requires an "End conditional execution".

<table>
<thead>
<tr>
<th>Execution with condition met</th>
<th>Execution with condition unmet</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SequenceStart</strong></td>
<td><strong>SequenceStart</strong></td>
</tr>
<tr>
<td>Start of Execution</td>
<td>Start of execution</td>
</tr>
<tr>
<td>Checking of the condition</td>
<td>Checking of the condition</td>
</tr>
<tr>
<td><strong>StartCondition</strong></td>
<td><strong>StartCondition</strong></td>
</tr>
<tr>
<td><strong>LedYellow</strong></td>
<td><strong>LedYellow</strong></td>
</tr>
<tr>
<td><strong>LedRed</strong></td>
<td><strong>LedRed</strong></td>
</tr>
<tr>
<td><strong>EndCondition</strong></td>
<td><strong>EndCondition</strong></td>
</tr>
<tr>
<td>End of the condition</td>
<td></td>
</tr>
<tr>
<td><strong>LedBlue</strong></td>
<td><strong>LedBlue</strong></td>
</tr>
<tr>
<td></td>
<td>Continuation after the condition</td>
</tr>
</tbody>
</table>
If the condition is met, then both steps ("LedYellow" and "LedRed") are executed. If the condition of "StartCondition" is not met, none of the steps up to the next "End conditional execution" are executed ("LedYellow" and "LedRed" in this case). Execution of the sequence resumes after "End conditional execution".

End conditional execution
"End conditional execution" defines the end of a "Conditional execution". This command has no parameters since the next associated "End conditional execution" is used as the end of a "Conditional execution".

None
This command can be used to define a step in the sequence where no action should be executed. This can be used for steps "SequenceStart" and "SequenceEnd", for example.

1.3.1.5 Dynamic interlocks
A dynamic interlock defines that a certain axis is only permitted to move if all axes specified for the interlock have a certain state.
For more information about usage, see Definition of dynamic interlocks.

1.3.1.6 Macros
Macros can be used to group together several steps visually (only in combination with widget "Sequencer"). Only the hierarchy of the macro is defined for this. All macros found on the specified data storage device are displayed there.
The following types of macros are available:

- **Inline macros**: These macros exist only in the sequence where they were created. These macros can be used to visually group together several steps for widget "Sequencer". These macros have no effect on the execution of the sequence.
- **External macros**: These macros are created in a separate file with file extension .mac and can be referenced as needed by sequence files. Referencing looks like this:

```xml
<Macro Value="5" Name="Macro1">
  <StepPrevious Value="1"/>
</Macro>
```
A file with the name of the macro being referenced must exist (e.g. Macro1.mac).

1.3.1.7 Display
This section defines units and limits for defined Parameter.
The name of the defined parameter is entered under "Identifier".
"Engineering unit" describes the unit that is used for calculations on the controller.
"Display unit" is only seen by the user. Conversion between units takes place based on the selected unit system.
If the advanced properties are displayed, parameters "Number of decimal digits", "Minimum integer digits" and "Maximum integer digits" can be edited. This allows you to specify the maximum/minimum number of digits and the number of decimal places. These properties affect widget "SequencerStepItemParameterForm" in the HMI application.

1.4 Libraries
1.4.1 Data types and enumerators
1.4.1.1 Data types
1.4.1.1.1 MpSequenceActuatorConfigType
Configuration structure for an actuator.
### 1.4.1.1.2 MpSequenceAxisConfigType
Configuration structure for function block MpSequenceAxisConfig.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AxisState</td>
<td>STRING[255]</td>
<td>Defines the state of the axis.</td>
</tr>
</tbody>
</table>

### 1.4.1.1.3 MpSequenceCommandConfigType
Configuration structure for the function block.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>STRING[255]</td>
<td>Text ID for the name of the command on the widget.</td>
</tr>
<tr>
<td>Hierarchy</td>
<td>STRING[255]</td>
<td>Hierarchy of the command for grouping.</td>
</tr>
<tr>
<td>Type</td>
<td>MpSequenceCommandTypeEnum</td>
<td>Type of command.</td>
</tr>
<tr>
<td>Movement</td>
<td>MpSequenceMovementConfigType</td>
<td>Structure for configuring a movement.</td>
</tr>
<tr>
<td>SetPv</td>
<td>MpSequenceSetPvConfigType</td>
<td>Structure for configuring a command to change a PV value.</td>
</tr>
<tr>
<td>WaitForTime</td>
<td>MpSequenceWaitForTimeConfigType</td>
<td>Structure for configuring a command to delay the time.</td>
</tr>
<tr>
<td>WaitForPv</td>
<td>MpSequenceWaitForPvConfigType</td>
<td>Structure for configuring a command to wait for a condition.</td>
</tr>
<tr>
<td>SetAlarm</td>
<td>MpSequenceSetAlarmConfigType</td>
<td>Structure for configuring a command to set an alarm.</td>
</tr>
<tr>
<td>ConditionalExecution</td>
<td>MpSequenceCondExecConfigType</td>
<td>Structure for configuring a command that checks for a condition.</td>
</tr>
</tbody>
</table>

### 1.4.1.1.4 MpSequenceCondExecConfigType
Configuration structure for command "ConditionalExecution".

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConditionPv</td>
<td>STRING[255]</td>
<td>Name of the PV used in the condition.</td>
</tr>
<tr>
<td>ConditionType</td>
<td>MpSequenceCondCheckEnum</td>
<td>Defines the type of condition.</td>
</tr>
<tr>
<td>ExpectedState</td>
<td>DINT</td>
<td>Expected state of the axis.</td>
</tr>
<tr>
<td>Text</td>
<td>STRING[255]</td>
<td>Text ID for the name of the PV on the widget. This must exist in the widget namespace.</td>
</tr>
</tbody>
</table>

### 1.4.1.1.5 MpSequenceConditionType
Configuration structure for a condition for command "Movement".

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConditionAxis</td>
<td>STRING[255]</td>
<td>Axis name for the condition. Only if &quot;ConditionType&quot; = mpSEQUENCE_CONDITION_AXIS_STATE.</td>
</tr>
<tr>
<td>ConditionProcessVariable</td>
<td>STRING[255]</td>
<td>Name of the process variable for the condition. Only if &quot;ConditionType&quot; = mpSEQUENCE_CONDITION_GENERIC.</td>
</tr>
<tr>
<td>ConditionType</td>
<td>MpSequenceCondCheckEnum</td>
<td>Defines the type of condition.</td>
</tr>
<tr>
<td>ExpectedState</td>
<td>DINT</td>
<td>Expected state of the axis.</td>
</tr>
<tr>
<td>Reaction</td>
<td>MpSequenceCondCheckReactionEnum</td>
<td>Type of reaction when the condition is met.</td>
</tr>
</tbody>
</table>

### 1.4.1.1.6 MpSequenceCoreInfoType
Information structure for MpSequenceCore.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadyToStart</td>
<td>BOOL</td>
<td>A sequence was loaded using command &quot;Import&quot; or transferred from the HMI application and can be executed.</td>
</tr>
<tr>
<td>ReadyToResume</td>
<td>BOOL</td>
<td>A sequence was interrupted by command &quot;Suspend&quot; and can be resumed.</td>
</tr>
<tr>
<td>ActiveSteps</td>
<td>ARRAY[0..9] of STRING[255]</td>
<td>Currently active steps (up to 10 active steps at one time)</td>
</tr>
</tbody>
</table>
1.4.1.1.7 MpSequenceCoreParType

Parameter structure that defines how the sequence is executed.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence</td>
<td>STRING[255]</td>
<td>Name of the sequence to be executed.</td>
</tr>
<tr>
<td>InitSequence</td>
<td>STRING[255]</td>
<td>Name of the initialization sequence to be executed.</td>
</tr>
<tr>
<td>ExitSequence</td>
<td>STRING[255]</td>
<td>Name of the exit sequence to be executed.</td>
</tr>
<tr>
<td>Start</td>
<td>MpSequenceCoreStartType</td>
<td>Parameter for starting the sequence.</td>
</tr>
<tr>
<td>Stop</td>
<td>MpSequenceCoreStopType</td>
<td>Parameter for stopping the sequence.</td>
</tr>
<tr>
<td>Suspend</td>
<td>MpSequenceSuspendType</td>
<td>Parameter for interrupting the sequence.</td>
</tr>
<tr>
<td>Manual</td>
<td>MpSequenceManualType</td>
<td>Parameter for manual mode</td>
</tr>
<tr>
<td>SingleCommand</td>
<td>MpSequenceSingleCommandType</td>
<td>Parameter for mode mpSEQUENCE_MODE_SINGLE_COMMAND.</td>
</tr>
</tbody>
</table>

1.4.1.1.8 MpSequenceCoreStartType

This data type describes how the sequence is started.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode</td>
<td>MpSequenceStartModeEnum</td>
<td>mpSEQUENCE_START_NORMAL</td>
<td>Defines the starting mode for the sequence.</td>
</tr>
<tr>
<td>Step</td>
<td>STRING[255]</td>
<td></td>
<td>Defines the step at which the sequence should start (only if mpSEQUENCE_START_SELECT_STEP)</td>
</tr>
</tbody>
</table>

1.4.1.1.9 MpSequenceCoreStopType

This data type describes how the sequence is stopped.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode</td>
<td>MpSequenceStopModeEnum</td>
<td>mpSEQUENCE_STOP_IMMEDIATE</td>
<td>Stop mode of the sequence</td>
</tr>
</tbody>
</table>

1.4.1.1.10 MpSequenceCycleType

This data type provides additional information about the execution of the sequence.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>UDINT</td>
<td>Number of cycles that have been executed since the last time this component was enabled.</td>
</tr>
<tr>
<td>CurrentTime</td>
<td>REAL</td>
<td>Elapsed time of the current cycle.</td>
</tr>
<tr>
<td>LastTime</td>
<td>REAL</td>
<td>Elapsed time of the previous cycle.</td>
</tr>
</tbody>
</table>

1.4.1.1.11 MpSequenceDiagType

This data type is used as a substructure within the structure to hold additional information for diagnostic purposes as well as to supply additional data about the status ID.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StatusID</td>
<td>MpSequenceStatusIDType</td>
<td>StatusID diagnostic structure</td>
</tr>
</tbody>
</table>

1.4.1.1.12 MpSequenceInfoType

This data type provides additional information for the function block.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diag</td>
<td>MpSequenceDiagType</td>
<td>Diagnostic structure for the status ID</td>
</tr>
</tbody>
</table>

1.4.1.1.13 MpSequenceManualParType

This data type defines a parameter for manual mode.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>STRING[255]</td>
<td>Name of the parameter.</td>
</tr>
<tr>
<td>Value</td>
<td>LREAL</td>
<td>Value of the parameter.</td>
</tr>
</tbody>
</table>
1.4.1.14 MpSequenceManualType

This data type contains all necessary parameters for manual mode.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>STRING[255]</td>
<td>Name of the command to be executed.</td>
</tr>
</tbody>
</table>

1.4.1.15 MpSequenceMovementConfigType

Configuration structure for command "Movement".

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis</td>
<td>STRING[255]</td>
<td>Name of the axis</td>
</tr>
<tr>
<td>CommandStructure</td>
<td>STRING[255]</td>
<td>Name of the command structure to be used</td>
</tr>
<tr>
<td>ParameterStructure</td>
<td>STRING[255]</td>
<td>Name of the parameter structure to be used</td>
</tr>
<tr>
<td>Actuator</td>
<td>STRING[255]</td>
<td>Name of the actuator</td>
</tr>
<tr>
<td>Priority</td>
<td>UINT</td>
<td>Priority of the movement</td>
</tr>
<tr>
<td>FallbackActuator</td>
<td>STRING[255]</td>
<td>Name of the actuator to be used if &quot;Actuator&quot; is already in use</td>
</tr>
<tr>
<td>ParameterArraySize</td>
<td>UDINT</td>
<td>Number of parameters to be specified.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Pointer to ARRAY[ ] of STRING[255]</td>
<td>Parameter list of the command. The size of the array is not permitted to be less than defined by &quot;ParameterArraySize&quot;.</td>
</tr>
<tr>
<td>FinalState</td>
<td>DINT</td>
<td>Defines the state of the axis when the command is ended.</td>
</tr>
<tr>
<td>ConditionsArraySize</td>
<td>UDINT</td>
<td>Defines the number of conditions.</td>
</tr>
<tr>
<td>Conditions</td>
<td>Pointer to ARRAY[ ] of MpSequenceConditionType</td>
<td>The conditions for this command are defined here.</td>
</tr>
</tbody>
</table>

1.4.1.16 MpSequencerCommandType

Command structure for axis movements.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>BOOL</td>
<td>Start command</td>
</tr>
<tr>
<td>Stop</td>
<td>BOOL</td>
<td>Stop command</td>
</tr>
<tr>
<td>Actuator</td>
<td>STRING[255]</td>
<td>Actuator used for the movement</td>
</tr>
<tr>
<td>CurrentMode</td>
<td>MpSequenceModeEnum</td>
<td>Currently active mode.</td>
</tr>
<tr>
<td>MovementState</td>
<td>MpSequenceMovementStateEnum</td>
<td>Feedback: Current state of the movement.</td>
</tr>
</tbody>
</table>

If command "Movement" is executed and execution of the sequence is suspended by "Suspend" on function block MpSequenceCore, then parameter "Start" = "FALSE". To indicate to the application that execution is paused, parameter "Stop" = TRUE.

If "Stop" = TRUE, the application must ensure that the current movement (e.g. movement of an axis, controlling a pump, etc.) is ended and "MovementState" is set to mpSEQUENCE_MOVEMENT_IDLE or mpSEQUENCE_MOVEMENT_DONE.

**Warning!**

The application must ensure a proper reaction to "Stop"! MpSequenceCore does not check whether the current function (e.g. moving a robot) has ended!

As soon as "MovementState" has changed, "Stop" = FALSE.

If "MovementState" = mpSEQUENCE_MOVEMENT_IDLE, "Resume" on MpSequenceCore can be used to resume execution of the sequence. Parameter "Start" = TRUE for this.

1.4.1.17 MpSequenceSetAlarmConfigType

Configuration structure for command "SetAlarm".

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alarm</td>
<td>STRING[255]</td>
<td>Name of the alarm that should be set.</td>
</tr>
</tbody>
</table>

1.4.1.18 MpSequenceSetPvConfigType

Configuration structure for command "SetPv".

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pv</td>
<td>STRING[255]</td>
<td>Name of the PV to be modified.</td>
</tr>
<tr>
<td>Text</td>
<td>STRING[255]</td>
<td>Text ID for the name of the PV on the widget.</td>
</tr>
</tbody>
</table>
1.4.1.19 MpSequenceSingleCommandType

All parameters needed for mode `mpSEQUENCE_MODE_SINGLE_COMMAND`.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command</td>
<td>STRING[255]</td>
<td>Name of the command to be executed.</td>
</tr>
<tr>
<td>ParametersArraySize</td>
<td>UDINT</td>
<td>Number of elements in &quot;Parameters&quot;.</td>
</tr>
<tr>
<td>Parameters</td>
<td>Pointer to ARRAY[ ]</td>
<td>Parameters to be used for the command.</td>
</tr>
</tbody>
</table>

1.4.1.20 MpSequenceStatusTypeIDType

This data type is used as a substructure within the structure to hold additional information for diagnostic purposes as well as to supply additional data about the status ID.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>MpSequenceErrorEnum</td>
<td>Error code for mapp component.</td>
</tr>
<tr>
<td>Severity</td>
<td>MpComSeveritiesEnum</td>
<td>Describes the type of information supplied by the status ID (success, information, warning, error)</td>
</tr>
<tr>
<td>Code</td>
<td>UINT</td>
<td>Code for the status ID. This error number can be used to search for additional information in the help system.</td>
</tr>
</tbody>
</table>

1.4.1.21 MpSequenceSuspendType

This data type contains all of the parameters necessary for interrupting the sequence.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode</td>
<td>MpSequenceSuspendModeEnum</td>
<td>mpSEQUENCE_SUSPEND_IMMEDIATE</td>
<td>Defines how the sequence is interrupted.</td>
</tr>
</tbody>
</table>

1.4.1.22 MpSequenceWaitForPvConfigType

Configuration structure for command "WaitForPv".

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WaitPv</td>
<td>STRING[255]</td>
<td>Name of the PV to be checked.</td>
</tr>
<tr>
<td>WaitCondition</td>
<td>MpSequenceConditionEnum</td>
<td>Condition used for the check.</td>
</tr>
<tr>
<td>Text</td>
<td>STRING[255]</td>
<td>Text ID for the name of the PV displayed on the widget.</td>
</tr>
</tbody>
</table>

1.4.1.23 MpSequenceWaitForTimeConfigType

Configuration structure for command "WaitForTime".

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>STRING[255]</td>
<td>Text ID for the name of the delay time displayed on the widget.</td>
</tr>
</tbody>
</table>

1.4.1.2 Enumerators

1.4.1.2.1 MpSequenceActuatorAccessTypeEnum

This enumerated data type defines the possible access types for the actuator.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpSEQUENCE_ACCESS_SINGLE</td>
<td>Only one axis can access this actuator at any time.</td>
</tr>
<tr>
<td>mpSEQUENCE_ACCESS_MULTIPLE</td>
<td>Multiple axes can access this actuator at the same time.</td>
</tr>
</tbody>
</table>

1.4.1.2.2 MpSequenceActuatorTypeEnum

This enumerated data type describes the type of actuator.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpSEQUENCE_ACTUATOR_SINGLE</td>
<td>The actuator is by itself.</td>
</tr>
<tr>
<td>mpSEQUENCE_ACTUATOR_MULTIPLE</td>
<td>The actuator consists of several different actuators.</td>
</tr>
</tbody>
</table>

1.4.1.2.3 MpSequenceCommandTypeEnum

This enumerated data type defines the possible commands that can be used by the sequencer.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpSEQUENCE_COMMAND_MOVEMENT</td>
<td>The command is a movement.</td>
</tr>
<tr>
<td>mpSEQUENCE_COMMAND_SET_PV</td>
<td>The command should assign a value to a PV.</td>
</tr>
<tr>
<td>mpSEQUENCE_COMMAND_WAIT_FOR_TIME</td>
<td>The command should wait a certain amount of time.</td>
</tr>
</tbody>
</table>
### MpSequenceCommandWaitForPv
The command waits until the value of a PV meets a condition.

### MpSequenceCommandSetAlarm
The command sets an alarm.

### MpSequenceCommandCondExec
The command checks whether a condition is met.

### MpSequenceCommandEndCondExec
Defines the end of command "mpSequenceCommandCondExec".

### MpSequenceCondCheckReactionEnum
This enumerated data type defines the reaction to an unmet condition.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpSEQUENCE_CONDITION_REACT_WAIT</td>
<td>The sequence will wait until the condition is met.</td>
</tr>
<tr>
<td>mpSEQUENCE_CONDITION_REACT_STOP</td>
<td>Execution of the sequence is stopped with an error if the condition is not met.</td>
</tr>
</tbody>
</table>

### MpSequenceConditionEnum
This enumerated data type defines possible operators for conditions.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpSEQUENCE_CONDITION_EQUAL</td>
<td>Checks whether the operands are the same.</td>
</tr>
<tr>
<td>mpSEQUENCE_CONDITION_NOT_EQUAL</td>
<td>Checks whether the operands are not the same.</td>
</tr>
<tr>
<td>mpSEQUENCE_CONDITION_LESS</td>
<td>Check whether the first operand is less than.</td>
</tr>
<tr>
<td>mpSEQUENCE_CONDITION_LESS_EQ</td>
<td>Check whether the first operand is less than or equal to.</td>
</tr>
<tr>
<td>mpSEQUENCE_CONDITION_GREATER</td>
<td>Check whether the first operand is greater than.</td>
</tr>
<tr>
<td>mpSEQUENCE_CONDITION_GREATER_EQ</td>
<td>Check whether the first operand is greater than or equal to.</td>
</tr>
</tbody>
</table>

### MpSequenceCondCheckEnum
This enumerated data type defines possible conditions for static interlocks.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpSEQUENCE_CONDITION_AXIS_STATE</td>
<td>The condition checks the state of an axis.</td>
</tr>
<tr>
<td>mpSEQUENCE_CONDITION_GENERIC</td>
<td>The condition checks the state of a PV.</td>
</tr>
</tbody>
</table>

### MpSequenceModeEnum
Execution modes of the sequence. For more information, see Execution mode of the sequence.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpSEQUENCE_MODE_AUTOMATIC</td>
<td>Automatic mode</td>
</tr>
<tr>
<td>mpSEQUENCE_MODE_SINGLE_CYCLE</td>
<td>Semi-automatic mode (waits after the end of a cycle)</td>
</tr>
<tr>
<td>mpSEQUENCE_MODE_SINGLE_STEP</td>
<td>Single-step mode (only one step at a time)</td>
</tr>
<tr>
<td>mpSEQUENCE_MODE_SINGLE_COMMAND</td>
<td>Single-function mode (only one function, no sequence file necessary)</td>
</tr>
<tr>
<td>mpSEQUENCE_MODE_MANUAL</td>
<td>Manual mode. The command is specified via the parameter structure, but it is looked for in the currently loaded sequence.</td>
</tr>
</tbody>
</table>

### MpSequenceMovementStateEnum
Movement mode of the current movement.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpSEQUENCE_MOVEMENT_IDLE</td>
<td>Idle: No movement is currently active.</td>
</tr>
<tr>
<td>mpSEQUENCE_MOVEMENT_ACTIVE</td>
<td>Active: A movement is currently active.</td>
</tr>
<tr>
<td>mpSEQUENCE_MOVEMENT_DONE</td>
<td>Finished: A movement command was completed.</td>
</tr>
</tbody>
</table>

### MpSequenceStartModeEnum
Start modes for sequence execution. For more information, see Start mode of the sequence.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpSEQUENCE_START_NORMAL</td>
<td>Begin at the sequence starting point</td>
</tr>
<tr>
<td>mpSEQUENCE_START_SELECT_STEP</td>
<td>Begin at a selected step (input)</td>
</tr>
<tr>
<td>mpSEQUENCE_START_AUTO_STEP</td>
<td>Begin at an automatically determined step (based on the machine step). Reserved for future use.</td>
</tr>
</tbody>
</table>

### MpSequenceStopModeEnum
Stop modes for stopping the sequence. For more information, see Stop mode of the sequence.
### 1.4.1.2.11 MpSequenceSuspendModeEnum

Interruption modes for pausing the sequence.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpSEQUENCE_SUSPEND_IMMEDIATE</td>
<td>Interrupts execution immediately</td>
</tr>
<tr>
<td>mpSEQUENCE_SUSPEND_END_OF_STEP</td>
<td>Interrupts execution after all active steps have ended</td>
</tr>
</tbody>
</table>

### 1.4.2 Function blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MpSequenceCore</td>
<td>This function block checks the execution of the sequence.</td>
</tr>
<tr>
<td>MpSequenceCommandConfig</td>
<td>This function block configures a command.</td>
</tr>
<tr>
<td>MpSequenceActuatorConfig</td>
<td>This function block configures an actuator.</td>
</tr>
<tr>
<td>MpSequenceAxisConfig</td>
<td>This function block is used to configure an axis.</td>
</tr>
</tbody>
</table>

#### 1.4.2.1 MpSequenceCore

Function block MpSequenceCore checks the execution of the sequence.

**When using widget "Sequencer", MpSequenceCore must be called in the program's initialization subroutine!**

Function block

```plaintext
## Optional parameters

<table>
<thead>
<tr>
<th>MpSequenceCore</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;MpComIdentType</td>
</tr>
<tr>
<td>BOOL</td>
</tr>
<tr>
<td>BOOL</td>
</tr>
<tr>
<td>&amp;MpSequenceCoreParType</td>
</tr>
<tr>
<td>BOOL</td>
</tr>
<tr>
<td>MpSequenceModeEnum</td>
</tr>
<tr>
<td>BOOL</td>
</tr>
<tr>
<td>BOOL</td>
</tr>
<tr>
<td>BOOL</td>
</tr>
<tr>
<td>BOOL</td>
</tr>
</tbody>
</table>
```
**mapp concept**

Section Component design explains how mapp components are structured. In addition, it provides important notes for correctly using mapp components (e.g. for downloads).

**1.4.2.1.1 Description**

MpLink from MpSequenceCore configuration is used for this function block. This function block controls the entire execution process for a sequence.

**Import/Export**

"Import" reads the sequence file defined in the parameter structure. An import is always necessary if the sequence is changed or a sequence has not yet been imported.

"Export" exports the active sequence to a file.
Start/Stop

"Start" starts execution of the sequence. For more information about how sequences can be started, see Start mode of the sequence.

"Stop" stops execution of the sequence. For more information about how sequences can be stopped, see Stop mode of the sequence.

Suspend/Resume

"Suspend" pauses execution of the sequence after the currently active step. "Resume" resumes execution of this sequence at the next step.

Displaying the sequence in an HMI application

The sequence can be displayed visually using widget "Sequencer". For information about how the widgets must be configured, see Concept.

1.4.2.1.2 Timing diagrams

Timing diagrams indicate the behavior of the sequence when started, stopped, paused as well as with different starting, stopping and execution modes.

Starting/Stopping a sequence

This timing diagram shows the behavior of the sequence with different stop modes. As long as command "Stop" is active, no sequence can be started.

Pausing/Resuming a sequence

This timing diagram shows the behavior of the sequence when execution is paused with different modes. If mode "mpSEQUENCE_SUSPEND_MODE_IMMEDIATE" is used, execution of the sequence is paused immediately on a rising edge of input "Suspend". With mode "mpSEQUENCE_SUSPEND_MODE_END_OF_STEP", execution of the current step is completed. Only then is the sequence paused.
Behavior with mode "mpSEQUENCE_MODE_SINGLE_CYCLE"
This timing diagram shows the behavior of mode "mpSEQUENCE_MODE_SINGLE_CYCLE". After a cycle has ended, output "Running" = FALSE. A new cycle can be started with a rising edge on input "Start".

Behavior with mode "mpSEQUENCE_MODE_SINGLE_STEP"
This timing diagram shows the behavior of mode "mpSEQUENCE_MODE_SINGLE_STEP". After a step has ended, output "Suspended" = TRUE. The next step in the sequence can be started with a rising edge on input "Resume".
Step Execution Finished
Whenever one step has finished its execution, output "Suspended" is set to TRUE. Execution can be continued using input "Resume".

1.4.2.2 MpSequenceAxisConfig
This function block is used to configure an axis.

Function block
## Optional parameters

```
MpSequenceAxisConfig
&MpComIdentType MpLink
BOOL Enable
BOOL ErrorReset
&STRING Name
&MpSequenceAxisConfigType Configuration
BOOL Load
BOOL Save
BOOL Active
BOOL Error
DINT StatusID
BOOL CommandDone
MpSequenceInfoTypeInfo
```
Interface

<table>
<thead>
<tr>
<th>I/O</th>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MpLink</td>
<td>Pointer to MpComIdentType</td>
<td>Connection to mapp (MpLink of an MpSequenceCore configuration)</td>
</tr>
<tr>
<td>IN</td>
<td>Enable</td>
<td>BOOL</td>
<td>The function block is active as long as this input is set.</td>
</tr>
<tr>
<td>IN</td>
<td>ErrorReset</td>
<td>BOOL</td>
<td>Resets function block errors.</td>
</tr>
<tr>
<td>IN</td>
<td>Name</td>
<td>Pointer to STRING[255]</td>
<td>Unique name of the axis. The name is specified in the MpSequenceCore configuration in section Axes.</td>
</tr>
<tr>
<td>IN</td>
<td>Configuration</td>
<td>Pointer to MpSequenceAxisConfigType</td>
<td>Structure used to specify the configuration.</td>
</tr>
<tr>
<td>IN</td>
<td>Load</td>
<td>BOOL</td>
<td>Loads the configuration of the component.</td>
</tr>
<tr>
<td>IN</td>
<td>Save</td>
<td>BOOL</td>
<td>Saves the configuration of the component.</td>
</tr>
<tr>
<td>OUT</td>
<td>Active</td>
<td>BOOL</td>
<td>Function block active.</td>
</tr>
<tr>
<td>OUT</td>
<td>Error</td>
<td>BOOL</td>
<td>Error occurred during execution.</td>
</tr>
<tr>
<td>OUT</td>
<td>StatusID</td>
<td>DINT</td>
<td>Status information.</td>
</tr>
<tr>
<td>OUT</td>
<td>CommandBusy</td>
<td>BOOL</td>
<td>Function block currently executing command.</td>
</tr>
<tr>
<td>OUT</td>
<td>CommandDone</td>
<td>BOOL</td>
<td>Execution successful. Function block is finished.</td>
</tr>
<tr>
<td>OUT</td>
<td>Info</td>
<td>MpSequenceInfoType</td>
<td>Additional information about the component.</td>
</tr>
</tbody>
</table>

mapp concept

Section mapp components explains how mapp components are structured. In addition, it provides important notes for correctly using mapp components (e.g. for downloads).

For mapp function blocks, asynchronous handling does not have to carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

1.4.2.2.1 Description

The same MpLink used by MpSequenceCore is used for this component as well.

This function block reads and writes the configuration. The configuration of an individual command takes place using the configuration structure on input "Configuration".

Command "Load" is used to read out the currently configured parameters. The currently active configuration on the controller is written to the PV on input "Configuration". This overwrites the data in the structure. Values can then be edited as needed.

Command "Save" applies the values from structure "Configuration" to the active configuration; they are then applied immediately, if possible.

The axis to be edited or created is defined via input parameter "Name". Creation and editing can take place at runtime.

Configuring structure MpSequenceAxisConfigType

This structure allows you to change the variable from which the state of an axis configured in section Axes of the MpSequenceCore configuration is taken.

If a new axis should be added, a name must be specified on input parameter "Name" that was not defined in section Axes. This allows the new axis to be added and used.

1.4.2.3 MpSequenceCommandConfig

This function block configures a command.

Function block

## Optional parameters
### Interface

<table>
<thead>
<tr>
<th>I/O</th>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MpLink</td>
<td>Pointer to MpComIdentType</td>
<td>Connection to mapp (MpLink of an MpSequenceCore configuration)</td>
</tr>
<tr>
<td>IN</td>
<td>Enable</td>
<td>BOOL</td>
<td>The function block is active as long as this input is set.</td>
</tr>
<tr>
<td>IN</td>
<td>ErrorReset</td>
<td>BOOL</td>
<td>Resets function block errors.</td>
</tr>
<tr>
<td>IN</td>
<td>Name</td>
<td>Pointer to STRING[255]</td>
<td>Unique name of the command.</td>
</tr>
<tr>
<td>IN</td>
<td>Configuration</td>
<td>Pointer to MpSequenceCommandConfigType</td>
<td>Structure used to specify the configuration.</td>
</tr>
<tr>
<td>IN</td>
<td>Load</td>
<td>BOOL</td>
<td>Loads the configuration of the component.</td>
</tr>
<tr>
<td>IN</td>
<td>Save</td>
<td>BOOL</td>
<td>Saves the configuration of the component.</td>
</tr>
<tr>
<td>OUT</td>
<td>Active</td>
<td>BOOL</td>
<td>Function block active.</td>
</tr>
<tr>
<td>OUT</td>
<td>Error</td>
<td>BOOL</td>
<td>Error occurred during execution.</td>
</tr>
<tr>
<td>OUT</td>
<td>StatusID</td>
<td>DINT</td>
<td>Status information.</td>
</tr>
<tr>
<td>OUT</td>
<td>CommandBusy</td>
<td>BOOL</td>
<td>Function block currently executing command.</td>
</tr>
<tr>
<td>OUT</td>
<td>CommandDone</td>
<td>BOOL</td>
<td>Execution successful. Function block is finished.</td>
</tr>
<tr>
<td>OUT</td>
<td>Info</td>
<td>MpSequenceInfoType</td>
<td>Additional information about the component.</td>
</tr>
</tbody>
</table>

### mapp concept

Section mapp components explains how mapp components are structured. In addition, it provides important notes for correctly using mapp components (e.g. for downloads).

For mapp function blocks, asynchronous handling does not have to carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

#### 1.4.2.3.1 Description

The same MpLink used by MpSequenceCore is used for this component as well.

This function block reads and writes a configuration. The configuration of an individual command takes place using the configuration structure on input "Configuration".

The "Load" command is used to read the currently configured parameters. The configuration currently active on the controller is written to the PV on input "Configuration". This overwrites the data in the structure. Values can then be edited as needed.

The "Save" command applies the values from the "Configuration" structure to the active configuration, where they are used immediately if possible.

The name of the command to be created or edited is defined on input "Name". Creation and editing can take place at runtime.

#### Configuring structure MpSequenceCommandConfigType

This structure can be used to reconfigure existing commands or create new ones.

The type of command must be specified. Depending on the type selected, a different substructure must be used for configuration.

#### 1.4.2.4 MpSequenceActuatorConfig

This function block modifies the configuration of an actuator.
Function block

## Optional parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;MpComIdentType</td>
<td>MpLink</td>
<td>Connection to mapp (MpLink of an MpSequenceCore configuration)</td>
</tr>
<tr>
<td>BOOL Enable</td>
<td>BOOL</td>
<td>The function block is active as long as this input is set.</td>
</tr>
<tr>
<td>BOOL ErrorReset</td>
<td>BOOL</td>
<td>Resets function block errors.</td>
</tr>
<tr>
<td>&amp;STRING Name</td>
<td>STRING[255]</td>
<td>Unique name of the actuator.</td>
</tr>
<tr>
<td>&amp;MpSequenceActuatorConfigType Configuration</td>
<td>MpSequenceActuatorConfig</td>
<td>Structure used to specify the configuration.</td>
</tr>
<tr>
<td>BOOL Load</td>
<td>BOOL</td>
<td>Loads the configuration of the component.</td>
</tr>
<tr>
<td>BOOL Save</td>
<td>BOOL</td>
<td>Saves the configuration of the component.</td>
</tr>
<tr>
<td>BOOL CommandBusy</td>
<td>BOOL</td>
<td>Function block currently executing command.</td>
</tr>
<tr>
<td>BOOL CommandDone</td>
<td>BOOL</td>
<td>Execution successful. Function block is finished.</td>
</tr>
<tr>
<td>DINT StatusID</td>
<td>DINT</td>
<td>Status information.</td>
</tr>
<tr>
<td>MpSequenceInfoType Info</td>
<td>MpSequenceInfoType</td>
<td>Additional information about the component.</td>
</tr>
</tbody>
</table>

**Interface**

<table>
<thead>
<tr>
<th>I/O</th>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MpLink</td>
<td>Pointer to MpComIdentType</td>
<td>Connection to mapp (MpLink of an MpSequenceCore configuration)</td>
</tr>
<tr>
<td>IN</td>
<td>Enable</td>
<td>BOOL</td>
<td>The function block is active as long as this input is set.</td>
</tr>
<tr>
<td>IN</td>
<td>ErrorReset</td>
<td>BOOL</td>
<td>Resets function block errors.</td>
</tr>
<tr>
<td>IN</td>
<td>Name</td>
<td>Pointer to STRING[255]</td>
<td>Unique name of the actuator.</td>
</tr>
<tr>
<td>IN</td>
<td>Configuration</td>
<td>Pointer to MpSequenceActuatorConfigType</td>
<td>Structure used to specify the configuration.</td>
</tr>
<tr>
<td>IN</td>
<td>Load</td>
<td>BOOL</td>
<td>Loads the configuration of the component.</td>
</tr>
<tr>
<td>IN</td>
<td>Save</td>
<td>BOOL</td>
<td>Saves the configuration of the component.</td>
</tr>
<tr>
<td>OUT</td>
<td>Active</td>
<td>BOOL</td>
<td>Function block active.</td>
</tr>
<tr>
<td>OUT</td>
<td>Error</td>
<td>BOOL</td>
<td>Error occurred during execution.</td>
</tr>
<tr>
<td>OUT</td>
<td>StatusID</td>
<td>DINT</td>
<td>Status information.</td>
</tr>
<tr>
<td>OUT</td>
<td>CommandBusy</td>
<td>BOOL</td>
<td>Function block currently executing command.</td>
</tr>
<tr>
<td>OUT</td>
<td>CommandDone</td>
<td>BOOL</td>
<td>Execution successful. Function block is finished.</td>
</tr>
<tr>
<td>OUT</td>
<td>Info</td>
<td>MpSequenceInfoType</td>
<td>Additional information about the component.</td>
</tr>
</tbody>
</table>

**mapp concept**

Section mapp components explains how mapp components are structured. In addition, it provides important notes for correctly using mapp components (e.g. for downloads).

For mapp function blocks, asynchronous handling does not have to carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

### 1.4.2.4.1 Description

The same MpLink used by MpSequenceCore is used for this component as well.

This function block reads and writes a configuration. The configuration of an individual actuator takes place using the configuration structure on input "Configuration".

The "Load" command is used to read the currently configured parameters. The configuration currently active on the controller is written to the PV on input "Configuration". This overwrites the data in the structure. Values can then be edited as needed.
The "Save" command applies the values from the "Configuration" structure to the active configuration, where they are used immediately if possible.

The name of the actuator to be created or edited is defined on input "Name". Creation and editing can take place at runtime.

**Configuring structure** **MpSequenceActuatorConfigType**

This configuration can be used to reconfigure existing actuators or create new ones.

The type of actuator must be specified. An actuator can either be by itself or composed of several actuators.

In addition, an access type must be specified for the actuator. This defines whether only one command is permitted to access an actuator or multiple commands are permitted to use the actuator at the same time.

Parameters "UsedActuatorArraySize" and "UsedActuator" define the individual actuators that make up the actuator.

This is only used if "Access" = mpSEQUENCE_ACCESS_MULTIPLE.

### 1.4.3 Status numbers

#### 1.4.3.1 -1064004587: Execution of command failed

**Description:**
Command {2:Command} was started but aborted in the application due to an error.

**Reaction:**
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

**Cause/Solution:**
- Check the command in the application.

**Additional information:**
- {2:Command}: Name of the affected command.

**Constant**

mpSEQUENCE_ERR_COMMAND_STATE_ERR

**These function blocks / functions can report this error:**
- MpSequenceCore

#### 1.4.3.2 -1064004588: No steps selected

**Description:**
The sequence could not be exported since no steps are selected.

**Reaction:**
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

**Cause/Solution:**
- Generate a sequence using widget "Sequencer". Then try exporting a sequence again.

**Constant**

mpSEQUENCE_ERR_NO_STEPSDEFINED

**These function blocks / functions can report this error:**
- MpSequenceCore
1.4.3.3 -1064004589: No step defined beforehand

Description:
No previous step was defined in a macro.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
• Check whether the macro was edited manually.
• Check whether the sequence was edited manually.

Constant
mpSEQUENCE_ERR_NO_STEPS_PREVIOUS

These function blocks / functions can report this error:
• MpSequenceCore

1.4.3.4 -1064004590: Command not found

Description:
Command {2:Command} could not be found in the list of available commands. See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
• Checks whether the command is entered in the list.

Additional information:
• {2:Command}: Name of the affected command.

Constant
mpSEQUENCE_ERR_COMMAND_NOT_FOUND

These function blocks / functions can report this error:
• MpSequenceCore

1.4.3.5 -1064004588: Undefined axis

Description:
The specified axis {2:AxisName} cannot be found in the configuration. See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
• Check whether the axis is defined in the configuration.
Additional information:

- \{2:AxisName\}: Name of the specified axis.

Constant

\texttt{mp\_SEQUENCE\_ERR\_AXIS\_MISSING}

These function blocks / functions can report this error:

- \texttt{MpSequenceCommandConfig}

1.4.3.6 -1064004592: No sequence loaded

Description:
A sequence has not yet been loaded. Command "Start" cannot be executed. See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- Command "Start" was issued before a sequence was loaded.

Constant

\texttt{mp\_SEQUENCE\_ERR\_NO\_SEQ\_IMPORTED}

These function blocks / functions can report this error:

- \texttt{MpSequenceCore}

1.4.3.7 -1064004593: Step not found

Description:
Specified step \{2:Step\} could not be found in the currently loaded configuration. See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- Check whether the name of the step was specified correctly.

Additional information:

- \{2:Step\}: Name of the affected step.

Constant

\texttt{mp\_SEQUENCE\_ERR\_STEP\_NOT\_FOUND}

These function blocks / functions can report this error:

- \texttt{MpSequenceCore}
1.4.3.8 -1064004594: Parameter not found

Description:
Specified parameter \{2:ParameterName\} cannot be found in the configuration. See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- Check whether the specified parameter exists in the configuration.

Additional information:
- \{2:ParameterName\}: Name of the affected parameter.

Constant
mpSEQUENCE_ERR_PAR_MISSING

These function blocks / functions can report this error:
- MpSequenceCommandConfig

1.4.3.9 -1064004595: Too few linked actuators

Description:
At least 2 actuators must be linked to an actuator of type "Multiple actuators". See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- Check whether at least 2 actuators are linked.

Constant
mpSEQUENCE_ERR_ACTUATOR_COUNT

These function blocks / functions can report this error:
- MpSequenceActuatorConfig

1.4.3.10 -1064004596: Actuator not found

Description:
An actuator that should be used is not defined in the configuration. See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- Check whether the actuator was defined in the configuration.
Constant

mpSEQUENCE_ERR_ACTUATOR_MISSING

These function blocks / functions can report this error:
  • MpSequenceActuatorConfig

1.4.3.11 -1064004598: Name is null pointer

Description:
A null pointer was provided to function block input "Name". See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
  • Provide a valid value on the input.

Constant

mpSEQUENCE_ERR_NAME_NULL

These function blocks / functions can report this error:
  • MpSequenceCommandConfig
  • MpSequenceActuatorConfig

1.4.3.12 -1064004599: Parameter array NULL

Description:
The pointer to the parameter/actuator array is a null pointer. See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
  • Check whether the array was assigned correctly.

Constant

mpSEQUENCE_ERR_ARRAY_NULL

These function blocks / functions can report this error:
  • MpSequenceCommandConfig
  • MpSequenceActuatorConfig

1.4.3.13 -1064004600: Command not found

Description:
Specified command {2:CommandName} could not be found in the configuration. See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.
Cause/Solution:
- Check whether the command exists in the configuration.

Additional information:
- \{2:CommandName\}: Name of the affected command

Constant
\texttt{mpSEQUENCE\_ERR\_NAME\_NOT\_FOUND}

These function blocks / functions can report this error:
- \texttt{MpSequenceCommandConfig}
- \texttt{MpSequenceActuatorConfig}

1.4.3.14 -1064004601: Invalid definition of steps

Description:
The steps in the loaded sequence are not defined correctly. See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- Check whether the order of execution in the sequence file is valid.

Additional information
- \{2:ErrorType\}: Describes the error in the sequence definition.

Constant
\texttt{mpSEQUENCE\_ERR\_CONDITION\_INVALID}

These function blocks / functions can report this error:
- \texttt{MpSequenceCore}

1.4.3.15 -1064004602: Timeout during execution

Description:
A timeout for a step occurred during execution of command \{2:Command\}. See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- Check whether the delay time for the timeout is high enough to execute the command.

Additional information
- \{2:Command\}: Name of the command for which the timeout occurred.

Constant
\texttt{mpSEQUENCE\_ERR\_STEP\_TIMEOUT}
These function blocks / functions can report this error:

- MpSequenceCore

1.4.3.16 -1064004603: Interlock active

Description:
The condition of an interlock was not met during execution of sequence {2:SequenceName}, {3:StepID}. See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- Check whether the interlock was defined correctly.

Additional information:
- {2:SequenceName}: Name of the affected sequence.
- {3:StepID}: ID of the affected step.

Constant
mpSEQUENCE_ERR_INTERLOCK

These function blocks / functions can report this error:

- MpSequenceCore

1.4.3.17 -1064004604: PV access failed

Description:
An error occurred while accessing PV {2:PVName}. Reason for the error: {1:ErrorNumber}. See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- Check whether the PV exists.

Additional information:
- {1:ErrorNumber}: Reason for the error.
- {2:PVName}: Name of the process variable.

Constant
mpSEQUENCE_ERR_PV_NOT_FOUND

These function blocks / functions can report this error:

- MpSequenceCore

1.4.3.18 -1064004605: No sequence defined

Description:
No sequence was found in specified sequence file {2:Filename} on data storage device {3:DeviceName}. See the Logger for additional information.
Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
• Check whether a sequence was defined in the file.

Additional information
• {2:Filename}: Name of the specified sequence file.
• {3:DeviceName}: Name of the specified data storage device.

Constant
mpSEQUENCE_ERR_NO_SEQUENCE_DEF

These function blocks / functions can report this error:
• MpSequenceCore

1.4.3.19 -1064004606: Sequence file export failed

Description:
The export of sequence file {2:Filename} to specified data storage device {3:DeviceName} failed. See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
• Check whether access to the data storage device is permitted.

Additional information
• {2:Filename}: Specified filename.
• {3:DeviceName}: Specified data storage device.

Constant
mpSEQUENCE_ERR_EXPORT

These function blocks / functions can report this error:
• MpSequenceCore

1.4.3.20 -1064004607: Sequence file import failed

Description:
The import of sequence file {2:Filename} from specified data storage device {3:FileDevice} failed. See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
• Check whether the file exists on the specified data storage device.
Additional information:

- {2:Filename}: Specified filename.
- {3:FileDevice}: Specified data storage device.

Constant

`mpSEQUENCE_ERR_IMPORT`

These function blocks / functions can report this error:

- MpSequenceCore

1.4.3.21 -1064004608: Blank sequence name defined

Description:
No name was specified for the sequence. See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- Check whether a sequence name was specified.

Constant

`mpSEQUENCE_ERR_NAME_EMPTY`

These function blocks / functions can report this error:

- MpSequenceCore

1.4.3.22 -1064239091: Invalid configuration

Description:
Could not read configuration while creating components. See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- The configuration is damaged.

These function blocks / functions can report this error:

- MpSequenceCore
- MpSequenceCommandConfig
- MpSequenceActuatorConfig

1.4.3.23 -1064239092: Error saving configuration

Description:
An error occurred while saving to the configuration file (from the PV). See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.
Cause/Solution:

- Configuration file damaged
- Difference between the contents of the configuration file and the PV (e.g. different data types, data outside valid limits, etc.)

These function blocks / functions can report this error:

- MpSequenceCommandConfig
- MpSequenceActuatorConfig

1.4.3.24 -1064239094: Error loading configuration

Description:
An error occurred while loading the configuration file (in the PV). See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:

- Configuration file damaged
- Difference between the contents of the configuration file and the PV (e.g. different data types, data outside valid limits, etc.)

These function blocks / functions can report this error:

- MpSequenceCommandConfig
- MpSequenceActuatorConfig

1.4.3.25 -1064239095: Configuration pointer not a PV

Description:
The specified pointer is not a pointer to a valid PV. See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:

- Check the value on input "Configuration".

These function blocks / functions can report this error:

- MpSequenceCommandConfig
- MpSequenceActuatorConfig

1.4.3.26 -1064239096: Configuration structure is a null pointer

Description:
Input "Configuration" is not connected, null pointer.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.
Cause/Solution:
- Check input "Configuration" on the function block.

These function blocks / functions can report this error:
- MpSequenceCommandConfig
- MpSequenceActuatorConfig

1.4.3.27 -1064239097: Parameter structure is a null pointer

Description:
Input "Parameter" is not connected, null pointer.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- Check input "Parameter" on the function block.

These function blocks / functions can report this error:
- MpSequenceCore

1.4.3.28 -1064239098: MpLink already in use

Description:
This MpLink is already in use.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- If "Overload" is the selected download mode, then the mapp component should be disabled in the exit subroutine with "Enable = FALSE". It is also possible to use "Copy" or "One cycle" mode. In these cases, it is not necessary to disable the mapp component.
- Check which components are already using this MpLink.
- Create a new MpLink.

These function blocks / functions can report this error:
- MpSequenceCore
- MpSequenceCommandConfig
- MpSequenceActuatorConfig

1.4.3.29 -1064239099: Invalid MpLink contents

Description:
The value in the "MpLink" variable on the function block input is invalid.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.
Cause/Solution:
- Writing to the values in structure MpComIdentType is not permitted.
- A corresponding configuration for this mapp component must be available (see first paragraph of the description for the functions / function blocks listed below).

These function blocks / functions can report this error:
- MpSequenceCore
- MpSequenceCommandConfig
- MpSequenceActuatorConfig

1.4.3.30 -1064239100: MpLink modified

Description:
The value on input "MpLink" was modified while the components were running ("Enable" = TRUE).

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- The value of input "MpLink" can only be changed while the component is inactive ("Enable" = FALSE).

These function blocks / functions can report this error:
- MpSequenceCore
- MpSequenceCommandConfig
- MpSequenceActuatorConfig

1.4.3.31 -1064239101: MpLink connection not permitted

Description:
The value on input "MpLink" is not allowed.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- The component is connected directly to mpCOM_MAIN or mpCOM_STANDALONE. This is not supported.

These function blocks / functions can report this error:
- MpSequenceCore
- MpSequenceCommandConfig
- MpSequenceActuatorConfig

1.4.3.32 -1064239102: MpLink is null pointer

Description:
Input "MpLink" is not connected, null pointer.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.
Cause/Solution:
- Check input "MpLink" on the function block.

These function blocks / functions can report this error:
- MpSequenceCore
- MpSequenceCommandConfig
- MpSequenceActuatorConfig

1.4.3.33 -1064239103: Could not create component

Description:
The mapp component could not be created and is not enabled. See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- Service for configuring the function block not available -> Problem with MpCom
- Unable to read registry -> Problem with MpCom
- Details about the cause of error in the logger

These function blocks / functions can report this error:
- MpSequenceCore
- MpSequenceCommandConfig
- MpSequenceActuatorConfig

1.4.3.34 -1064293635: Text ID not found

Description:
Specified text ID {2:Text ID} could not be found in namespace {3:Namespace}. See the Logger for additional information.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- The specified text ID does not exist.

Additional information:
- {2:Text ID}: Specified text ID.
- {3:Namespace}: Namespace that should contain the text ID.

Constant
mpSEQUENCE_ERR_TEXT_ID_NOT_FOUND

1.4.3.35 -1064293637: Received sequence corrupt

Description:
Sequence {2:SequenceName} received by the widget contains errors. See the Logger for additional information.
Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- Check the loaded sequence.

Additional information:
- {2:SequenceName}: Name of the affected sequence.

Constant
mpSEQUENCE_ERR_RECEIVE_SEQUENCE

These function blocks / functions can report this error:
- MpSequenceCore

1.4.3.36 1083190008: Empty sequence

Description:
No steps were defined in the specified sequence. See the Logger for additional information.

Reaction:
The function block indicates this information on output "StatusID".

Cause/Solution:
- Define steps in the sequence.

Constant
mpSEQUENCE_INF_SEQUENCE_EMPTY

These function blocks / functions can report this error:
- MpSequenceCore

1.4.3.37 1083190009: "Movement" blocked by condition

Description:
At least one condition (static or dynamic interlock) is active. Command "Movement" cannot be executed yet. See the Logger for additional information.

Reaction:
The function block indicates this information on output "StatusID".

Cause/Solution:
- Check the interlocks.

Constant
mpSEQUENCE_INF_CONDITION_ACTIVE

These function blocks / functions can report this error:
- MpSequenceCore
1.4.3.38 1083190010: Sequence executed

**Description:**
Execution of the sequence was completed. See the Logger for additional information.

**Reaction:**
The function block indicates this information on output "StatusID".

**Constant**
\texttt{mpSEQUENCE-INF SEQUENCE DONE}

**These function blocks / functions can report this error:**
- \texttt{MpSequenceCore}

1.4.3.39 -2137746421: Array too small

**Description:**
The parameter/actuator array is too small to load all of the parameters/actuators specified in the configuration. See the Logger for additional information.

**Reaction:**
The function block indicates an active warning on output "StatusID".

**Cause/Solution:**
- Check whether all actuators/parameters can be entered in the array.

**Constant**
\texttt{mpSEQUENCE_WRN ARRAY SHORT}

**These function blocks / functions can report this error:**
- \texttt{MpSequenceCommandConfig}
- \texttt{MpSequenceActuatorConfig}

1.4.3.40 -2137980917: Warning loading configuration

**Description:**
A warning occurred while loading the configuration file (in the PV). See the Logger for additional information.

**Reaction:**
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

**Cause/Solution:**
- Configuration file damaged
- Difference between the contents of the configuration file and the PV (e.g. more data in the configuration file than in the PV)

**These function blocks / functions can report this error:**
- \texttt{MpSequenceCommandConfig}
- \texttt{MpSequenceActuatorConfig}
1.4.3.41 -2138035460: Cannot modified sequence

Description:
The sequence cannot be modified since it is currently being executed. See the Logger for additional information.

Reaction:
The function block indicates an active warning on output "StatusID".

Cause/Solution:
• Stop execution.

Constant
mpSEQUENCE_WRN_SEQUENCE_ACTIVE

These function blocks / functions can report this error:
• MpSequenceCore

1.4.4 Alarms

1.4.4.1 mpSEQUENCE_ALM_IMPORT_SEQUENCE: Import failed

Description:
The import of the sequence failed. See the Logger for additional information.

Reaction:
A mapp alarm is triggered if a MpAlarmX component is active.

Alarm details:
The following details about this alarm are available and can be embedded in the alarm text using the key:

<table>
<thead>
<tr>
<th>Key</th>
<th>Format</th>
<th>Additional information</th>
</tr>
</thead>
<tbody>
<tr>
<td>SequenceName</td>
<td>{&amp;SequenceName}</td>
<td>Name of the specified sequence</td>
</tr>
<tr>
<td>DeviceName</td>
<td>{&amp;DeviceName}</td>
<td>Data storage device that should contain the sequence</td>
</tr>
</tbody>
</table>

Cause/Solution:
• Check whether the data storage device for the sequence file exists.

Behavior
Edge alarm

1.4.4.2 mpSEQUENCE_ALM_EXPORT_SEQUENCE: Export failed

Description:
The export of the sequence failed. See the Logger for additional information.

Reaction:
A mapp alarm is triggered if a MpAlarmX component is active.

Alarm details:
The following details about this alarm are available and can be embedded in the alarm text using the key:

<table>
<thead>
<tr>
<th>Key</th>
<th>Format</th>
<th>Additional information</th>
</tr>
</thead>
<tbody>
<tr>
<td>SequenceName</td>
<td>{&amp;SequenceName}</td>
<td>Name of the specified sequence</td>
</tr>
<tr>
<td>DeviceName</td>
<td>{&amp;DeviceName}</td>
<td>Data storage device that should contain the sequence</td>
</tr>
</tbody>
</table>

Cause/Solution:
• Check whether the data storage device for the export file exists.
**1.4.4.3 mpSEQUENCE_ALM_TIMEOUT: Timeout during execution**

**Description:**
A timeout occurred while executing the sequence. See the Logger for additional information.

**Reaction:**
A mapp alarm is triggered if a MpAlarmX component is active.

**Alarm details:**
The following details about this alarm are available and can be embedded in the alarm text using the key:

<table>
<thead>
<tr>
<th>Key</th>
<th>Format</th>
<th>Additional information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step</td>
<td>(&amp;Step)</td>
<td>Step at which the timeout occurred.</td>
</tr>
</tbody>
</table>

**Cause/Solution:**
- Executing the command took longer than defined by "Timeout".

**1.4.4.4 mpSEQUENCE_ALM_INTERLOCK: Active interlock**

**Description:**
A condition for an interlock was not met. See the Logger for additional information.

**Reaction:**
A mapp alarm is triggered if a MpAlarmX component is active.

**Alarm details:**
The following details about this alarm are available and can be embedded in the alarm text using the key:

<table>
<thead>
<tr>
<th>Key</th>
<th>Format</th>
<th>Additional information</th>
</tr>
</thead>
<tbody>
<tr>
<td>InterlockType</td>
<td>(&amp;InterlockType)</td>
<td>Indicates whether the interlock is static or dynamic.</td>
</tr>
<tr>
<td>Type</td>
<td>(&amp;Type)</td>
<td>Indicates the type of interlock (axis or condition)</td>
</tr>
<tr>
<td>ExpectedState</td>
<td>(&amp;ExpectedState)</td>
<td>Expected state of the axis</td>
</tr>
<tr>
<td>CurrentState</td>
<td>(&amp;CurrentState)</td>
<td>Current state of the axis</td>
</tr>
<tr>
<td>Axis</td>
<td>(&amp;Axis)</td>
<td>Axis whose condition is not met</td>
</tr>
<tr>
<td>StepAxis</td>
<td>(&amp;StepAxis)</td>
<td>Axis that should currently be moved</td>
</tr>
</tbody>
</table>

**Cause/Solution:**
- An interlock was enabled.

**1.4.4.5 mpSEQUENCE_ALM_ACTUATOR_BLOCKED: Actuator blocked**

**Description:**
An actuator is not available for this command since it is already being used in a parallel command. See the Logger for additional information.

**Reaction:**
A mapp alarm is triggered if a MpAlarmX component is active.
Alarm details:

The following details about this alarm are available and can be embedded in the alarm text using the key:

<table>
<thead>
<tr>
<th>Key</th>
<th>Format</th>
<th>Additional information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step</td>
<td>(&amp;Step)</td>
<td>Step whose actuator is blocked</td>
</tr>
<tr>
<td>Actuator</td>
<td>(&amp;Actuator)</td>
<td>Affected actuator</td>
</tr>
<tr>
<td>BlockingStep</td>
<td>(&amp;BlockingStep)</td>
<td>Step blocking the actuator</td>
</tr>
</tbody>
</table>

Cause/Solution:

- An actuator was blocked by another command.

Behavior

Edge alarm