mapp Services

1 mapp AlarmX: Alarm management

mapp AlarmX collects and manages both mapp alarms and user alarms. The alarms are configured using Automation Studio, managed in the application and then displayed in an HMI application or exported as a file.

Tutorials about this topic are also available on our B&R online tutorial platform.

Several tutorials about this topic can be found on our YouTube channel.

1.1 Concept

In a given project, it is possible to use either MpAlarm or MpAlarmX, but not both at the same time. MpAlarmX is a standalone alarm system, independent of the HMI application. MpAlarmX also offers a broad range of functions that allow the alarm system to be configured and provide advanced functionality at runtime.

MpAlarm is designed as an extension of the alarm system for VC4.
### User stories

#### Machine operator

- As a machine operator, I would like to see informative and simple alarm messages in my own language.
- As a machine operator, I would like to always know unequivocally which alarm is most important or critical if multiple alarms are being displayed.
- As a machine operator or service technician, I would like easy access to information or instructions about how to correct the source of an alarm. This could be from an informative alarm text, video or more extensive reference information (PDF, HTML, etc.).
- As a machine operator, I would like to be able to export the alarm history for further analysis.

#### Service technician

- As a service technician, I would like access to the alarm history so I can understand how the machine has been used.
- As a service technician, I would like access to more detailed alarm information than is available to the machine operator. This would include the source of the alarm (e.g. conveyor belt, heating station, etc.), possible error numbers, links to additional information, etc.

#### Application engineer

- As an application engineer, I would like my alarms to behave intelligently. For example, I would like my alarms to be able to trigger different responses, to escalate when needed and to be grouped together.
- As an application engineer, I would like to be able to use the same alarm system that is used internally by mapp.
- As an application engineer, I would like to be able to display alarms on different levels. I would like to make it possible for the machine operator to clear an alarm before it is passed on.
- As an application engineer, I would like to have access to all of the functions available in the HMI application in my application as well. For example, I want to be in a position to be able to acknowledge all alarms from my application.
- As an application engineer, I would to be able to add any information whatsoever to my alarms.

### 1.1.1 Basic alarm concept

This section will shed some light on the basic concept behind alarm management. It will explain which states an alarm can assume and how these states function in relation to one another. It will also explain the difference between errors and alarms as well as the difference between a mapp alarm and a user alarm.

### Basic state of an alarm

An alarm can take on the following basic states:
The alarm states occur in combination with one another:

- **Active and Acknowledged**: The alarm is active. mapp alarms are enabled by the respective component, while user-defined alarms are set using `MpAlarmXSet`. It is also possible that the active alarm has already been acknowledged. Acknowledgment can take place in the application using `MpAlarmXAcknowledge` or in the HMI application using `MpAlarmXListUI`.
- **Active and not acknowledged**: The alarm is active. mapp alarms are enabled by the respective component, while user-defined alarms are set using `MpAlarmXSet`. If the alarm has not yet been acknowledged, then it will also have status "Not acknowledged".
- **Inactive and not acknowledged**: If the alarm is not active and therefore not acknowledged, the alarm has status "Inactive" and "Not acknowledged".
- **None**: The alarm is not enabled.

The status of an alarm can be determined using the `MpAlarmXCheckState` function. One of the 5 states above is specified, and Boolean feedback is returned about whether the specified state is active. It is also possible to use the alarm configuration to specify whether an alarm must be acknowledged and/or reset or not.

**Errors, user-defined alarms and mapp alarms**

With mapp components, it is necessary to differentiate between errors and alarms. Errors are indicated via "StatusID" on the output of a function or function block or entered in Logger. "StatusID" is used to troubleshoot what did not work when developing the application or later when performing detailed diagnostics.

Alarms provide the machine operator with information about the state of a machine. Alarms should be written up in such a way that the machine operator can respond to them effectively. Alarm handling is not intended to cover detailed diagnostics (e.g. evaluate "StatusID", display ACOPOS error texts, etc.).

Alarms and errors should be considered separate from one another as a rule. Alarms do not necessarily have to be the consequence of an error; they can also occur independently (e.g. user blocked due to incorrectly entering a password triggers an alarm). Errors provide information about the incorrect use of hardware or software and can occur at the same time as an alarm, though this is not required (e.g. loading a recipe indicated via "StatusID" with an alarm generated additionally). The following table shows some of these differences:
Error

Alarms

mapp Alarm

User alarm

mapp alarms

Alarms have already been conceived for each mapp component. These are default alarms that are meant to help the machine operator determine the state of the machine. A few examples:

- Load recipe failed
- Encoder error
- User blocked due to too many failed login attempts
- Data storage device not found

mapp alarms can be found in the configuration of the respective component. In the configuration, the user must decide whether the alarms should be used in combination with MpAlarmX or MpAlarm:

It is also possible to modify mapp alarms further here, e.g. adjusting the text source, defining a reaction or completely disabling them.

In addition to the default text, mapp components can also provide additional information as well. This can include the name of the recipe that could not be loaded or the ARNC0 error number when an error occurs in the CNC system. Users are free to include this information in the alarm text. The process is the same as with integrating application data into the alarm text. The only difference is that the text snippet does not need to be created because it already exists. The key that needs to be used can be found in the help system for the respective alarms. For the alarm mpROBOX_ALM_SYSTEM_ERROR of MpRoboX, for example, the ARNC0 error number can be embedded in the alarm text using the key Arnc0Code.

User-defined alarms

The developer can specify user alarms using MpAlarmXCore configuration. Here, the behavior of each alarm can be adjusted and various settings made, e.g. defining the reaction, links to further information or specifying the level (severity) of the alarm:

For more information about the settings that can be made in the alarm configuration, see section Alarm properties.
1.1.2 Alarm properties

If you use MpAlarmX, each alarm can be treated as a separate object. Each object has certain properties. These alarm properties can be modified in the configuration. The following section provides more detailed information about the properties possible for an alarm.

Some properties are only visible by enabling the "Advanced parameters" option.

General properties

An alarm has the following general properties:

- **Name**: The name of the alarm. This is the name used to identify the alarm. The alarm can be set (MpAlarmXSet), reset (MpAlarmXReset) or acknowledged (MpAlarmXAcknowledge, MpAlarmXListUI). In addition, the current basic state (MpAlarmXCheckState) can be determined for each alarm. The name of an alarm must not contain any special characters (e.g. ü, ö, ä). The character "," is also not allowed.
- **Message**: This is where the alarm text is entered. This can be done either directly or via a link to the text system. For more information, see section Alarm texts.
- **Code**: A number can be assigned here for each alarm. This parameter can be used in the event that you are working with alarm numbers.
- **Severity**: The severity specifies the "level" of an alarm. This is a number (UDINT) that can be chosen and defined as needed. For example, non-critical alarms can be given a severity of 10 whereas critical alarms can be given a severity of 100.
- **Behavior**: The behavior of the alarm is determined more exactly by this parameter. See further below for details.
- **Disable**: This property can be used to disable the alarm, see Basic state of an alarm. As long as this property is active, this alarm cannot be set or appear in another form.
- **AdditionalInformation1 and AdditionalInformation2**: These STRINGs can be used to link additional information to an alarm, e.g. a PDF file, video or HTML file. In the following example, 2 HTML files are linked that provide information about the cause of an alarm and its correction via the HMI application.

Behavior of the alarm

You can also specify the behavior of the alarm in the configuration:

There are two predefined types of behavior. One of these is "edge alarms", which represent short-term alarms, i.e. an event. "Persistent alarms" are also possible, which remain active until a certain condition is met, i.e. a state.
In addition, there are alarms for which set and reset conditions can be configured directly in the configuration. These alarms are described in more detail in section Alarm monitoring.

**Edge alarms**

Edge alarms are short-term alarms. You can think of them like an event. As soon as they are set, they are automatically reset. As can be seen in the timing diagram, an edge alarm is set using `MpAlarmXSet`. The reset takes place automatically and does not need to be explicitly triggered by the user. The user just has to acknowledge the edge alarm using `MpAlarmXAcknowledge` or from the HMI application.

Edge alarms can be instanced multiple times, i.e. edge alarms can be set more than once. If an edge alarm is set twice using `MpAlarmXSet`, then 2 entries are shown in the alarm list. Edge alarms must be acknowledged. Examples of typical edge alarms:

- Unable to load recipe: "Recipe1"
- Could not send SMS to "Shift supervisor"
- Data storage device not found
- Operation not permitted

**Persistent alarms**

Persistent alarms remain active until a certain condition is met, i.e. until the state of an alarm changes. These alarms are set and reset manually. As shown in the timing diagram, a persistent alarm is set using `MpAlarmXSet`. After a certain condition is met (e.g. temperature back in normal range), they are reset using `MpAlarmXReset`. Persistent alarms can be acknowledged using `MpAlarmXAcknowledge` or from the HMI application any time after they are enabled. Acknowledgment is possible regardless of whether the alarm is still active or has already been reset. Note that `MpAlarmXAcknowledge` cannot be used to acknowledge persistent alarms with a period in their name, e.g. "Temp.High".

Persistent alarms can only be instanced once, i.e. a persistent alarm only appears in the alarm list once. A persistent alarm will remain in the alarm list until it is acknowledged. If it becomes active again, a new entry is not entered in the alarm list; the already existing alarm becomes active again. Persistent alarms must be acknowledged. Examples of typical persistent alarms:

- Temperature (123°C) not in normal range (100 - 120°C)
- Water level in tank too high
- Emergency switch-off pressed
- X20DI8371 not connected

**User-defined alarms**

If neither of these two default behaviors is suitable, the user has the option of defining the behavior. In this case, user-defined behavior is used. Keep in mind that the "Advanced parameters" must be displayed in order to view the various configuration options.

The following section provides more detailed information about the individual settings:
• **Auto reset**: This setting determines whether the alarm should be reset automatically immediately after it is set (TRUE) or whether the reset must take place manually using MpAlarmXReset (FALSE).

• **Acknowledge**: Defines whether acknowledgment is necessary. There are 3 variants here:
  - Disabled: No acknowledgment necessary
  - Required: Acknowledgment necessary
  - Required after active: Acknowledgment necessary but only possible after the alarm is inactive

• **Multiple instances**: Defines whether an alarm can be instanced once or multiple times, i.e. the same alarm can be enabled and appear in the alarm list more than once.

• **Reaction until acknowledge**: Defines how long the reaction remains in effect. This is either until the alarm or alarms have been acknowledged or until they are reset.

• **History report**: This section allows you to define which state changes should appear in the alarm history.
  - Inactive to active: Defines whether setting an alarm should be recorded in the alarm history.
  - Active to inactive: Defines whether resetting an alarm should be recorded in the alarm history.
  - Unacknowledged to acknowledged: Defines whether acknowledging an alarm should be recorded in the alarm history.

1.1.3 Alarm instances

The behavior of an alarm also includes whether the alarm can only be active singularly or whether multiple instances of the same alarm can be active. If an alarm is displayed when a recipe cannot be opened, for example, then this alarm could potentially occur more than once. If you try to open a damaged recipe called “Recipe1”, then an alarm entry should be created with the text “The following recipe could not be opened: Recipe1”. If you now attempt to open another recipe, Recipe2, that also does not exist, then a new alarm should appear in the list: “The following recipe could not be opened: Recipe2”. This should not replace the previous alarm, however, since it is a different alarm. In this case, we refer to multiple instanced alarms for MpAlarmX.

On the other side, an alarm that indicates when the temperature is too high cannot be enabled more than once. If the temperature is too high, then this one alarm will remain active until the temperature returns to the normal range. Only then can the alarm be enabled again.

Whether an alarm can be instanced only once or multiple times is basically determined by its behavior, though it can also be adjusted as necessary.
Singular instanced alarms

Singular instanced alarms are intended for alarms that, once set, cannot be active again at the same time. This type of alarm only never appears more than once at a time in the alarm list. Examples of singular instanced alarms:

- Water level in tank too high
- Temperature in extruder (154°C) above the maximum limit value (150°C)
- Emergency switch-off pressed

Each persistent alarm is preconfigured as a singular instanced alarm.

Multiple instanced alarms

Multiple instanced alarms can be set more than once at a time. These alarms can appear in the alarm list more than once. Examples of multiple instanced alarms:

- Unable to load following recipe: "Recipe1"
- Unable to send SMS to "John Doe"
- Module not OK: "X20DI2653"

Each edge alarm is preconfigured as a multiple instanced alarm. Multiple alarm instances can no longer be identified by their name since multiple alarms of the same type can be active. The instance ID is available for this. It is a value returned by MpAlarmXSet. An alarm instance can be clearly identified using its instance ID.

Managing instanced alarms

ID functions are available to ensure access to a certain alarm instance from the application. This includes MpAlarmXResetID, MpAlarmXAcknowledgeID and MpAlarmXCheckStateID, for example. These functions work exactly the same as the same functions without the "ID" suffix. The only difference with these functions is that alarms are specified using their instance ID instead of their name.

1.1.4 Alarm monitoring

Alarm monitoring makes it possible to define the Behavior of the alarm, including setting and resetting, via the configuration. Various conditions are monitored, and the alarm state is adjusted accordingly. Cyclic monitoring of alarm conditions takes place in the context of MpAlarmXCore. The following alarm monitoring methods are possible:

- **Level monitoring:** Monitors a PV. An alarm is triggered if its value exceeds or falls below a certain level.
- **Deviation monitoring:** Monitors a PV. An alarm is triggered if the value deviates by a certain amount from the setpoint.
- **Rate of change monitoring:** Monitors a PV. An alarm is triggered if its value exceeds or falls below a certain rate of change.
- **Discrete value monitoring:** Monitors a PV. An alarm is triggered if it has a certain value.

The methods for alarm monitoring will be described in more detail in the following sections.

Level monitoring

(level monitoring)

Monitors the level of a PV. The PV is defined in the configuration. It is possible to define two lower limit values and two upper limit values. The alarm is triggered if a limit value is exceeded. Limit values are disabled by default (Disabled). If both an upper and lower limit value are defined, you can specify whether the limit values are static or dynamic.

- **Static:** The limit value is entered directly in the configuration. The image below shows static limit values.
- **Dynamic:** The limit value is defined using another PV. This limit value PV is connected to the configuration.

It is also possible to use parameter Limit text to enter a separate text as a text snippet for each limit value. This text snippet can then be used as the alarm text (see image below). The text in snippet Limit text can change depending on the limit value exceeded:
• Temperature <= 50°C: Temperature: Low limit reached
• Temperature <= 45°C: Temperature: Critical low limit reached

The alarm is reset if the PV falls below the limit value again.

Level monitoring alarms are exclusive alarms by default. This means that only one limit transgression at a time is displayed in the alarm list. In the example above, this means that either an alarm exists that fell below 50°C or an alarm exists that fell below 45°C. Both of these will not be displayed at the same time, however. More details about exclusive alarms can be found further down.

**Deviation monitoring**

(deviation monitoring)

Monitors the deviation from a defined level. One PV specified the current value while a second PV defines the setpoint. An alarm is generated if the current value deviates from the setpoint by a defined tolerance. It is possible to define two lower limit values and two upper limit values. The alarm is triggered if a limit value is reached/exceeded. Limit values are disabled by default (Disabled). If both an upper and lower limit value are defined, you can specify whether the limit values are static or dynamic.

• **Static**: The tolerance is entered directly in the configuration. The image below shows static tolerance values.
• **Dynamic**: The tolerance value is defined using another PV. This tolerance PV is connected to the configuration.

It is also possible to use parameter *Limit text* to enter a separate text as a text snippet for each instance that the tolerance is exceeded or fallen below. This text snippet can then be used as the alarm text (see image below). The text in snippet *Limit text* can change depending on the tolerance exceeded or fallen below:

• Temperature: Below tolerance level
• Temperature: Below critical tolerance level

The alarm is reset when the current value is back in the tolerance range. In the example, variable "Temperature" must be greater than 50 for the alarm to be reset.

Deviation monitoring alarms are exclusive alarms by default. This means that only one tolerance transgression at a time is displayed in the alarm list. In the example above, this means that either an alarm exists that fell below the lower tolerance level or an alarm exists that fell below the critical tolerance level. Both of these will not be displayed at the same time, however. More details about exclusive alarms can be found further down.

**Rate of change monitoring**

(rate of change monitoring)
Monitors the rate of change of a PV. The PV is defined in the configuration. It is possible to define two lower limit values and two upper limit values. The alarm is triggered if a limit value is exceeded. Limit values are disabled by default (*Disabled*). If both an upper and lower limit value are defined, you can specify whether the limit values are static or dynamic.

- **Static:** The limit value is entered directly in the configuration. The image below shows static limit values.
- **Dynamic:** The limit value is defined using another PV. This limit value PV is connected to the configuration.

It is also possible to use parameter *Limit text* to enter a separate text as a text snippet for each limit value. This text snippet can then be used as the alarm text (see image below). The text in snippet *Limit text* can change depending on the limit value exceeded:

- Temperature: Rate of change limit reached
- Temperature: Critical rate of change limit reached

The alarm is reset if the PV exceeds the limit value again.

Rate of change monitoring alarms are exclusive alarms by default. This means that only one limit transgression at a time is displayed in the alarm list. In the example above, this means that either an alarm exists that has reached the rate of change limit value or an alarm exists that has reached the critical rate of change limit value. Both of these will not be displayed at the same time, however. More details about exclusive alarms can be found further down.

Parameter "Time constant" is the time constant of a PT1 filter.

**Discrete value monitoring**

(discrete value monitoring)

Monitors certain PV values. The PV is defined in the configuration. An alarm is triggered the PV takes on a specific value. One or more values can be defined. The definition is made using *alarm trigger values*. Characters "*" and "," can be used to defined multiple values. For example:

- **1..5:** An alarm is set in the event that the PV takes on value 1, 2, 3, 4 or 5.
- **3:** An alarm is set in the event that the PV takes on value 3.
- **TRUE:** An alarm is set in the event that the PV takes on value TRUE.

Discrete value monitoring alarms are especially interesting when processing function block errors. It is possible to monitor whether a certain error is indicated on the function block and whether an alarm should be used to respond to it. In the example below, "StatusID" of function block MTBasicsPID is monitored. If an error is returned on the function block output, then an alarm is generated that contains the error number.

The alarm is reset when the current value is back outside the defined range.

**PVs defined for discrete value monitoring in the configuration must be BOOL, INT, DINT, SINT, UDINT, UINT or USINT.**
Exclusive and non-exclusive alarms

Monitoring alarms are exclusive alarms:

- **Exclusive**: If multiple limit values are overshot or undershot at a time, then only one active alarm with the highest (or lowest) limit value will be displayed in the alarm list along with other alarms that are inactive but not acknowledged. This means that one active alarm is indicated on output "ActiveAlarms" for component MpAlarmXCore. Multiple alarms are indicated on output "PendingAlarms" since several limit values were overshot or undershot.

- **Non-exclusive**: If multiple limit values are exceeded or fallen below at a time, then an alarm is displayed in the alarm list for each limit value exceeded or fallen below.

Blocking

Parameter Inhibit-PV can be used to block monitoring alarms. This makes it possible to prevent alarms from becoming enabled under defined conditions. For example, no temperature alarms should be enabled whenever the heating element of a machine is being serviced.

This is done by selecting a process variable of data type BOOL in the configuration. As long as this variable has value TRUE, the alarm will not be triggered. Already active alarms of this type are reset. If Inhibit-PV is set from TRUE to FALSE, then the system checks whether the alarm condition is active; the alarm is then set taking delay and hysteresis into account.

Deceleration

Parameter Delay can be used to define a delay time for triggering the alarm. If a delay time of 3 seconds is configured, for example, then the alarm is set 3 seconds after the trigger condition is met. The trigger condition must last through the entire time specified with Delay.

Hysteresis

Parameter "Hysteresis" can be used to define hysteresis that prevents possible rapid switching. The hysteresis value applies in both the positive and negative direction from the setpoint. If a hysteresis of 1°C is configured for a setpoint of 80°C, for example, then the alarm is triggered at 81°C and reset at 79°C.

Additional information can be passed to the alarm using text snippets.

Adding additional information using text snippets

Text snippets can be used to pass additional information to the alarm.

The following text snippets can be integrated into the alarm text:

- **LimitText**: Text snippet "LimitText" can be used to pass the text defined by parameter "Limit text" to the alarm message. The text snippet is specified on parameter "Message" as follows: {&LimitText}.

![Example of LimitText](image)

The text snippet can be used with level monitoring, deviation monitoring and rate-of-change monitoring alarm behavior.

- **LimitText**: Text snippet "Limit" can be used to pass the limit defined in the configuration to the alarm text. If a limit of 100 is defined, for example, then this value will additionally be shown in the alarm message. The text snippet is specified on parameter "Message" as follows: {&Limit}.
12

The text snippet can be used with level monitoring, deviation monitoring and rate-of-change monitoring alarm behavior.

- **MonitoredValue**: Text snippet "MonitoredValue" can be used to pass the value of the process variable specified on parameter "Monitored PV". If the variable has value 120 when the alarm is enabled, then value 120 will be displayed in the alarm message.

The text snippet is specified on parameter "Message" as follows: {&MonitoredValue}.

### 1.1.5 Alarm texts

This section explains how a user can put together alarm texts. The following questions are answered here:

- Where do I enter my alarm texts?
- How can I localize/translate my alarm texts?
- How can I incorporate application data into my alarm texts?

#### Alarm text directly in the configuration

An alarm text can be entered in the **configuration** for each alarm. In this case, the alarm text is entered for the **Message** parameter:

The text can only be entered in one language here. The Automation Studio text system can be used to localize text.

#### Localizing alarm texts with the text system

The Automation Studio text system is used to localize alarm texts. Here, the alarm text is not directly specified for the **Message** parameter, but is linked to the text system itself. This could look like this:
A file with the actual alarm texts is added to the Logical View beforehand. This file contains the alarm text in each project language. The connection is established via the namespace of the text and the text ID. The namespace can be defined in the properties for the file containing the texts. The text ID is then defined for each text in the file to be localized. The namespace and text ID are specified inside curly brackets. The connection to the text system is established with the $ character.

\[
\text{\{Namespace/TextID\}}
\]

Figure 1: Use of text system

Figure 2: Project languages

Figure 3: Namespace localizable text file

Figure 4: Text ID and localized texts

The TMX file must also be specified in the text configuration:

Integrating application data into the alarm text

You can use text snippets if you would like to include application data in your alarm texts. These text snippets can be defined in the configuration.

This is where all text snippets are defined that are to be used for the different alarm texts. A single text snippet can be used for one or more alarms. A key is defined for each text snippet that is used to identify the text snippet in the alarm text. The PV that supplies the application data is also defined. In our example, the current temperature should be integrated in the alarm text for the alarms TemperatureHigh and TemperatureLow. This could look like this:

The text snippet still needs to be incorporated in the alarm text.
Alarm text snippets can be incorporated in the alarm text regardless of whether the text comes from the alarm configuration or is read from the text system. The syntax looks the same in both cases:

```
{&Key}
```

The & character inside the curly brackets indicates a text snippet. The corresponding Key then follows the & character.

PV data can be formatted as needed. For example, it is possible to define how many digits after the decimal point should be displayed. This is done by appending the corresponding format string next to the text snippet:

```
{&Key} \{\#d\}
```

More details about these options is available in section Format strings.

The following data types can be integrated into the alarm text using a snippet:

- BOOL
- SINT
- INT
- DINT
- USINT
- UINT
- UDINT
- REAL
- LREAL
- STRING
- WSTRING

1.1.6 Auditing alarm management

Events for these components can be recorded using MpAudit. For more information about how to create an entry, see section Creating entries.

The following events can occur for these mapp components:

**MpAlarmX event**

This event type allows events involving the alarm system, such as when an alarm occurs, to be recorded.

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>MpAlarmX components</td>
</tr>
</tbody>
</table>

**Events**

- `ev`: Event ID
- `name`: Alarm name
- `code`: Alarm code
- `stold`: Last state of the alarm
- `stnew`: Current state of the alarm
- `sev`: Alarm level (severity)
- `Time`: Time at which the alarm state changed. This can deviate from the time at which the event was logged.

**Examples**


<table>
<thead>
<tr>
<th>Event</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>MpAlarmX components</td>
</tr>
</tbody>
</table>

Table 1: MpAlarmX event
1.1.7 Mapping alarms

Use case Advanced alarm management is available for this topic.

Use case 3: Advanced alarm management for a production line

In addition to Alarm properties, the user can also define how the alarm behaves in the entire system. It is up to the user to decide what should happen with the alarm or which actions should occur with a certain trigger. This is done using alarm mapping.

Each trigger is connected to one or more actions. Possible triggers include:

1.1.7.1 Alarm

One or more actions are executed if a specific alarm occurs. In the example image, reaction Stop is triggered when alarm TemperatureHigh occurs.

1.1.7.2 Severity

One or more actions are executed if the severity (of one or more alarms) takes on a certain value or falls within a defined range. In the example, an action is defined as the response to a severity within a defined range; in the second case, an action is defined as a response to a certain severity.

Severity is specified as follows:

- For a range of values: 1..10
- For a certain value: 11

With mapping, it is important to understand which actions can be used in which context. There are basically actions that can always be used and actions that are only beneficial in machines with multiple groups, including multiple HMI devices.

Multiple actions can follow an alarm. If these are contradictory, then the first action in the mapping assigned to the alarm is always assigned. For example, if an alarm is assigned actions Escalate alarm and Remain, then the first action listed in the mapping is carried out.

1.1.7.3 Actions for each machine

These actions can be used for any machine:

Reaction

A reaction refers to the behavior of a system when a new state occurs (e.g. triggering a warning light to blink when the water level is low). Reactions can be defined for each alarm. The following must be defined:

- Possible reactions: Immediate stop, accept no new parameters, turn off motor, turn off heater, etc.
- Reactions of each alarm: Which alarm triggers which reaction(s) above?

The MpAlarmXCheckReaction function can be used to check whether a certain reaction is active. The application software can then deal with this reaction appropriately. The configuration of reactions and assignment to alarms is done in the alarm mapping.
1. **Light curtain interrupted** alarm becomes active.
2. Alarm triggers defined reaction - **Switch off motors**.
3. The user sees the reaction and stops the motor with his software.

1. Alarm **Temperature too high** becomes active.
2. Alarm triggers defined reaction - **Switch off motors** and **Switch off heating**.
3. The user sees the reaction and stops the motor and switches off the heating with his software.

1. Alarm **No user logged in** becomes active.
2. Alarm triggers defined reaction - **No new parameters**.
3. The user sees the reaction and locks the input fields in the HMI application.

**Alarm mapping could look like this for this example:**

The reaction depends on the specific application at hand.

**SendMessage**

Action "SendMessage" allows an SMS text message notification to be sent to a specified user in the event of an alarm. Component MpTweet must be added in order to use this option.

This configuration could look something like this:
As soon as alarm "TempHigh" is active, a message is sent to a user. The user is defined in the MpTweetCore configuration. For more information, see Connection to alarm management.

1.1.7.4 Additional actions for machines with multiple HMI devices - Intelligent alarm handling

These actions are particularly beneficial when there are several HMI devices in a system and alarms should be concentrated at the top. Let's take the example of the following production line:

Each group has its own alarms and own HMI device. It is now necessary to decide for each group and each alarm whether they should be managed locally or concentrated at the top. The use case Alarm management for a production line is dedicated to this example in detail. The following actions support intelligent alarm handling:

- **Escalate alarm:** Escalates an alarm or reaction to the next higher level
- **Replace alarm:** Replaces one or more alarms with another alarm and escalates the new alarm to the next higher level
- **Remain:** Manages the alarm locally

**Escalate alarm**

Action *Escalate alarm* escalates the alarm to the next higher group without changing it. If our example, the *Saw battered* alarm should be escalated since it causes the entire machine process to be stopped.

The configuration for escalating the *Saw battered* alarm could look like this:

In addition to escalating the alarm, it is also possible to escalate a reaction with action *Escalate reaction.*

If an alarm should be acknowledged that has been escalated by action "Escalate alarm" to a higher-level group, the mapp Link of the higher-level alarm system must be specified in order to acknowledge the alarm.
mapp Services

Remain

If action *Remain* is selected for an alarm, then it remains at the same level as configured when it becomes active. In our example, the *Remain* option is selected for the stacker's *No product* option since the alarm only indicates that no product has currently arrived to process; the machine process is not affected in any way by this alarm.

The configuration for keeping the *No product* alarm local could look like this:

---

Replace alarm

Action *Replace alarm* replaces one alarm with another, which is escalated immediately. In our example, the *TemperatureHigh* and *TemperatureLow* alarms should be replaced by a *HeatingFailure* alarm since it is important that the operator of the line HMI is informed of this state.

The configuration in which the *TemperatureHigh* and *TemperatureLow* alarms are replaced by *HeatingFailure*, which is escalated, could look like this:

---

Default action

*Default action* specifies what happens by default for all alarms where nothing different has been mapped. Each of the described actions can be selected here. For example, the user can define that the same "YellowLamp" reaction should become active regardless of which alarm becomes active. At the same time, this reaction causes a yellow warning lamp to blink on the machine. This configuration could look something like this:

---

The default *Default action* is *Escalate alarm*. For systems with just one HMI device / group, this is irrelevant since alarms cannot be escalated further anyway. In the production line example above, this would mean that all configured alarms would be escalated to *Line HMI* by default.

Wildcard support

Alarm mapping supports the wildcard principle. If you would like to trigger the same reaction for a group of alarms, then multiple alarms can be selected using a wildcard character. If you want to trigger the *ExtruderSlow* reaction for all alarms involving temperature (*TemperatureHigh* and *TemperatureLow*), this can be done via mapping using the following wildcard:

---

Another possibility is to replace all of the alarms of a mapp component. If you would like to replace all of the alarms for an axis (mapp Link: *gConveyor*) that controls a conveyor belt, this can also be done using a wildcard:
1.1.8 Displaying units for variables

To determine which variable should have which unit, see Unit management.

Parameter "Language" for MpAlarmXHistory, MpAlarmXHistoryUIConnectType and MpAlarmXListUIConnectType can be used to define the language or unit to be used when displaying alarms.

In mapp Services 5.1.x, only REAL data types can be used to display units.
In mapp Services 5.2.x, all data types for displaying the unit are supported.

Defining the language and unit for display and export

How a unit is displayed is defined under "Displaying the unit within a text". For export or display, however, it is necessary to specify both the language as well as the unit system. This is done for the components listed above using input/parameter "Language". In order to define the language and unit system for export or display, input "Language" can be used in the following way:

'Language|UnitSystem'

A few examples:

• 'de': Language is German, values are shown with the engineering unit
• 'en|imperial-us': Language is English, unit system is imperial-US
• 'de|metric': Language is German, unit system is metric
• 'imperial': Default language, unit system is imperial

For details about defining display units, see section MpComUnit configuration.

Displaying the unit within a text

In addition to displaying and converting process values that are part of a text (e.g. alarms, audit events), it is also possible to define how a unit should be displayed:

• Symbol %s: This defines that the unit symbol for the current display unit will be displayed (e.g. "mm").
• Short text %s: This defines that the full name of the current display unit will be displayed (e.g. "millimeters").
• Description %d: This defines that the description for the current display unit will be displayed (e.g. "Distance measured in millimeters"). A description does not necessarily exist for each unit.

Conversion takes place automatically; how the unit is displayed is defined with the following syntax:

{&key}{&key[UNIT= xxx]}

• {&key}: Defines the process variable whose value should be displayed. This value will be converted automatically to the current display unit. If no display unit is defined, then the engineering unit for the value is output.
• {&key[UNIT=xx]}: Displays the unit text for the unit assigned to the value. The parameters mentioned above (%s, %n, %d) can be specified in place of "xx".

A few examples:

• "Current temperature {&actualTemperature}{&actualTemperature[UNIT=%s]} has reached a critical level!"  
  Current temperature 120°C has reached a critical level!
• "Axis stopped at limit position: {&actualPosition}{&actualPosition[UNIT= %n]}!"  
  Axis stopped at limit position: 500.0 millimeters!
• "Current pressure {&currentPressure}{&currentPressure[UNIT= %s]} exceeds the recommended maximum value!"  
  Current pressure 50 bar exceeds the recommended maximum value!

1.2 Guides

1.2.1 Getting started

"Getting started" sections are located here.
1.2.1.1 User alarms with VC4 mapp template

This section provides a step-by-step explanation of how to create user alarms using MpAlarmXCore, MpAlarmXHistory, MpAlarmXSet and MpAlarmXReset, as well as how they can be used with the VC4 mapp template.

1.2.1.1.1 Creating a project

You must first create a new project in Automation Studio.
1.2.1.1.2 Add mapp component

Adding MpAlarmXCore and MpAlarmXHistory configurations

The next step is to select the desired mapp component in the Toolbox and then drag-and-drop the configuration into the "mapp Services" folder.

Name the MpLink

When a configuration is added, the MpLink is automatically added along with it. When the configuration is opened, the MpLink is shown at the very top. "Name" is used to specify the name of the user alarm used later in the application to identify the alarm when it comes time to set and reset it. The alarm text that will also be shown later in the HMI application is added under "Message". "Code" and "Severity" specify the alarm code and severity of the alarm. The behavior of the alarm is determined by "Behavior". In our case, we are dealing with a persistent alarm since the alarm should remain active until the temperature returns to its normal range.
1.2.1.1.3 Adding a program

The next step is to add a program. In this case, we will add a Ladder Diagram program. We could use any of the other programming languages, however.
1.2.1.4 Adding MpAlarmXCore to the program

Searching for MpAlarmXCore in the Ladder Diagram Catalog

We can select the "mapp" option in the Ladder Diagram Catalog. This will display all of the available mapp components that are in the project. In our case, "Alarm system" counts as well.

Selecting a function block

Function block MpAlarmXCore is added to the program using drag-and-drop.
Adding function block

A function block instance is generated after we have pulled the function block into the program.
1.2.1.5 Configuring MpAlarmXCore

Connecting MpLink
First, connect input "MpLink" to the MpLink previously created in the Configuration View. Its address is passed on to the function block.

Enabling the component
The component is enabled by setting input "Enable". This is indicated by output "Active".
1.2.1.6 Adding MpAlarmXHistory to the program

Searching for MpAlarmXHistory in the Ladder Diagram Catalog

We can select the "mapp" option in the Ladder Diagram Catalog. This will display all of the available mapp components that are in the project. In our case, "Alarm system" counts as well.

Selecting a function block

Function block MpAlarmXHistory is added to the program using drag-and-drop.
**Adding function block**

A function block instance is generated after we have pulled the function block into the program.

**1.2.1.1.7 Configuring MpAlarmXHistory**

**Connecting MpLink**

First, connect input "MpLink" to the MpLink previously created in the Configuration View. Its address is passed on to the function block.
Enabling the component

The component is enabled by setting input "Enable". This is indicated by output "Active".

Specifying the data storage device

MpAlarmXHistory can be used to export the entire alarm history to a specified data storage device. For this reason, we will specify data storage device "CF" for parameter "DeviceName".
The data storage device is created in the hardware configuration under "FileDevice".
1.2.1.1.8 Adding MpAlarmXSet to the program

Searching for MpAlarmXSet in the Ladder Diagram Catalog

Navigate back to "Alarm system" in the catalog.
Adding the function

User alarms are set using the \texttt{MpAlarmXSet} function. The function is added to the program using drag-and-drop. To prevent the function from being called cyclically, we will right-click on the function to add "EN/ENO" to it.

1.2.1.9 Configuring \texttt{MpAlarmXSet}

\textbf{Enabling the component}

In order to ensure that the function is only called once and not cyclically, a switch is used that only reacts to a rising edge of "UserAlarm". It is positioned before input "EN".

The function or alarm is therefore only true if the "UserAlarm" variable becomes \texttt{TRUE}. 
Connecting MpLink

Now connect input "MpLink" to the MpLink previously created in the Configuration View.

Specify the alarm name.

Input "Name" specifies the name of the user alarm that was created in the MpAlarmXCore configuration. When the function is enabled, the alarm with the specified name is set.

Status of the MpAlarmXSet function

The "AlarmXSetStatus" variable is created to see the current status of the function.
1.2.1.10 Adding MpAlarmXReset to the program

Searching for MpAlarmXReset in the Ladder Diagram Catalog

Navigate back to "Alarm system" in the catalog.

Adding the function

MpAlarmXReset is used to reset user alarms. The function is added to the program using drag-and-drop. To prevent the function from being called cyclically, we will right-click on the function to add "EN/ENO" to it.
1.2.1.11 Configuring MpAlarmXReset

Enabling the component

In order to ensure that the function is only called once and not cyclically, a switch is used that only reacts to a falling edge of "UserAlarm". It is positioned before input "EN".

The function is therefore only true if the "UserAlarm" variable becomes FALSE. The alarm is reset if the function is active.

Connecting MpLink

Connect input "MpLink" to the MpLink previously created in the Configuration View.

Specify the alarm name.

Input "Name" specifies the name of the user alarm that was created in the MpAlarmXCore configuration. When the function is enabled, the alarm with the specified name is reset.
The "AlarmXResetStatus" variable is created to see the current status of the function.

1.2.1.12 Generating the file structure

Now generate a file structure for the memory card for ARsim.

1.2.1.13 Testing the user alarm in the program

Enabling the user alarm

Monitor mode must be active in order to test the user alarm. To start, the "UserAlarm" variable is set to \textit{FALSE}. Outputs "ActiveAlarms" and "PendingAlarms" are both still zero.
In order to display only one alarm, the "UserAlarms" variable must be set to TRUE. This enables the MpAlarmXSet function and sets the user alarm. The two outputs of MpAlarmXCore indicate that an alarm is active.

Resetting the user alarm

To disable the alarm once again, set the "UserAlarm" variable back to FALSE. This enables the MpAlarmXReset function and resets the alarm.

Since the alarm has not yet been acknowledged in the HMI application, output "PendingAlarms" from MpAlarmXCore is still set to 1.
1.2.1.14 Adding MpAlarmXListUI to the program

Searching for MpAlarmXListUI in the Ladder Diagram Catalog

Navigate back to "Alarm system" in the catalog.

Selecting a function block

MpAlarmXListUI is used to manage the display of user alarms via VC4. The function block is added to the program using drag-and-drop.
Adding function block
A function block instance is generated after we have pulled the function block into the program.

1.2.1.15 Configuring MpAlarmXListUI

Connecting MpLink
First, connect input "MpLink" to the MpLink previously created in the Configuration View. Its address is passed on to the function block.
Enabling the component

The component is enabled by setting input "Enable". This is indicated by output "Active".

Defining UIConnect

To connect the program and the HMI application later, you must create a variable of type `MpAlarmXListUIConnectType`.
1.2.1.16 Adding MpAlarmXHistoryUI to the program

Searching for MpAlarmXHistoryUI in the Ladder Diagram Catalog

Navigate back to "Alarm system" in the catalog.

Selecting a function block

MpAlarmXHistoryUI is used to manage the display of the alarm history via VC4. The function block is added to the program using drag-and-drop.
Adding function block
A function block instance is generated after we have pulled the function block into the program.

1.2.1.17 Configuring MpAlarmXHistoryUI

Connecting MpLink
First, connect input "MpLink" to the MpLink previously created in the Configuration View. Its address is passed on to the function block.
Enabling the component

The component is enabled by setting input "Enable". This is indicated by output "Active".

Defining UIConnect

To connect the program and the HMI application later, you must create a variable of type `MpAlarmXHistoryUIConnectType`. 
1.2.1.18 Creating an HMI application

Adding a visualization object

To test the axis alarms, we will create a VC4-based HMI application.

Selecting a template

In order to show the mapp HMI application, the "mapp Technology VGA 640x480 landscape" template must be selected.
Configuring VNC server for the HMI application

The HMI application is then connected to a VNC server.

1.2.1.1.19 Integrating the template into the project

Creating the connection between the task and visualization object

After opening the visualization object, select¹ and update² the data source. After the update, you will see which variables are connected in the HMI application but do not yet exist in the project³.

¹ select
² update
³ exists in the project
Select the variable from the list that should be replaced by your own. Right-click on `MpAlarmXListUIConnect` and select the `Refactor` option.
Now select your own "UIConnect" structure in the dialog box that opens.
You must then confirm the replacement. All connections are now created automatically.
After refactoring, the data source is updated again. The previous `MpAlarmXListUIConnect` connection should now no longer be available. This procedure is repeated for the `MpAlarmXHistoryUIConnect` structure.
Alarm page

All existing data points on the alarm page have been replaced by the own structure.
1.2.1.20 Generating the file structure

Now generate a file structure for the memory card for ARsim.

![Create ARsim Runtime Structure](image)

1.2.1.21 Testing the HMI application

Opening the VNC viewer

The HMI application can be opened with VNC Viewer.

![VNC Viewer: Connection Details](image)

Home screen of the HMI application

The start page is displayed after successfully connecting to the HMI application using the VNC viewer. Clicking on the "MpAlarmX" button opens up to the alarm page.
**Alarm page**

Active alarms can now be displayed on the alarm page. An alarm must be set first, however.
Enabling alarms

An alarm is now displayed in VNC Viewer if variable "UserAlarm" is set to TRUE as described in section Testing the user alarm in the program.
Acknowledging alarms

If the variable is set back to FALSE in the program, then the alarm will be displayed on the "Alarm history" page. Selecting the "Alarm history" opens up the alarm page where the alarm history is displayed.
1.3 Configuration

1.3.1 MpAlarmXCore configuration

For a tutorial on the topic of alarm configuration, see our YouTube channel.

Mapping alarms

If an alarm is active, an action can be triggered. This is defined in section "Alarm mapping". For information about the type of actions that can be selected, see section Mapping alarms.

User alarms

User alarms can be created in the configuration in section "Alarm list". For more information, see section Alarm properties.

Text snippets

Text snippets can be used to provide additional information for an alarm. They are specified in section "Alarm text snippets". For more information, see Alarm texts in section "Integrating application data into the alarm text".

Query

Queries can be created under "Data queries". These queries can be used to filter or search for specific alarms in the alarm list. A unique name for the query must be specified under "Name". If the request is to be started via MpAlarmXQuery, the name defined here must be specified on input parameter "Name".
The value of the variable specified on "Update count" is automatically increased by 1 as soon as new information is available. This means that as soon as new data corresponding to the specified filter criteria (WHERE) is available, the counter value of the variable on "Update count" changes. This can be used as a trigger to start command "Execute" on MpAlarmXQuery.

As the query source, you can determine whether to search for pending alarms or for all configured alarms. The information to be included in the query is defined via "Select" and "Where":

**SELECT**

"Select" determines which information is to be requested and where it is to be copied. A variable must be specified on "Process variable". If the query is started via MpAlarmXQuery, the information is displayed on this process variable. An array variable can also be specified. The array variable must be specified as follows: "::MyVar[]".

"Column" determines which information is to be queried. For more information about individual types, see Alarm properties.

**WHERE**

The filter properties can be further configured under "Where". Under "Column", it is possible to determine which information is to be filtered. For more information about individual types, see Alarm properties.

Using "Operator" and "Compare to", the query can be provided with additional filters, for example alarms with a certain name or alarms from a specific point in time. This means that certain values can be queried, for example. "Compare to" can be a fixed value ("Value") or process variable ("PV").

If "LIKE" is used as an operator, similar entries can be searched for. Wildcard symbol "*" must be used. If "Test" is used as "Value", for example, entries "Test01", "MyTest" and "MyTest01" can be searched for.

**Example**

All currently active alarms are to be searched in the alarm list. This means that "Pending alarms" is selected as the "Query source". It should be filtered by the alarm name. The information is to be displayed on a variable specified under "Process variable". Variable "AlarmName" of data type STRING[200][100] was specified. This means that the variable is an array variable with 100 elements. Since it is an array variable, the variable must be specified with ::AlarmName[].

Next, additional filter criteria were defined under "Where". The severity of the alarm should be ≥10 and the alarm should be active. The filter options were specified under "Connect", i.e. filter criteria 1 and 2 are used.

```
1.3.2 MpAlarmXHistory configuration
```

![Diagram of MpAlarmXQuery configuration](image-url)
Export
Export settings can be defined in section "Export". The name of the file to be saved can be defined under "Filename pattern". "Timestamp pattern" defines the timestamp for the alarm history.
For more information, see "Adding the time to an entry" in Creating entries.
Option "Overwrite file" can be used if the advanced parameters are enabled. If "Overwrite file" = TRUE, a file that has the same name will be overwritten. If FALSE, a new file is created with appended ".n", e.g. "xxx_2".

Memory location
Parameter "Memory" defines the way in which recorded data is saved. One entry in the alarm history needs approximately 350 bytes.
The following modes can be selected:
Temporary: Data is saved to DRAM. It is important to note that this data is not protected in the event of power failure or restart!
UserROM: The data is saved to UserROM and persists after a power failure.
SRAM: When using battery-backed memory, there must be sufficient memory reserved in the controller configuration. Can only be used if the hardware being used is equipped with SRAM memory.

Auditing
"Auditing" can be used to define whether events triggered in the framework of alarm management should be recorded by MpAudit. For more information, see Auditing alarm management.

1.4 Use cases
This section outlines several different use cases for MpAlarmX components.

1.4.1 Use case 1: Alarm management with an HMI application

Requirement
A conveyor belt on a machine delivers products to a packaging station. The machine is equipped with an alarm management system that collects all alarms from the conveyor belt and displays them in an HMI application.
It should be possible to save all alarms displayed in the HMI application on a USB flash drive.

Solution
Component list
- **MpAlarmXCore** (own MpLink): Collects all alarms from mapp components
- **MpAlarmXHistory** (own MpLink): Saves all alarms and their state change
- **MpAlarmXHistoryUI** (MpLink from MpAlarmXHistory): Contains all necessary parameters and structures for establishing a connection to an HMI application and displaying all alarms saved by MpAlarmXHistory
- **MpAxisBasic** (own MpLink): Axis control for the conveyor belt
**Connection diagram**

![Connection diagram](image)

**Alarm configuration**

The MpAlarmXCore and MpAlarmXHistory configurations are added for alarm management.

![Configuration View](image)

**Using the mapp components**

The **MpAlarmXCore**, **MpAlarmXHistory** and **MpAlarmXHistoryUI** components are added for alarm management, and the **MpAxisBasic** component is added to control the conveyor belt.

The components are connected to one another as depicted in "Connection diagram". **MpAxisBasic** must be located in the same group as the alarm function blocks.
All components are then configured. For information about making a connection to an HMI application using `MpAlarmXHistoryUI`, see section VC4 templates.

Each axis also has default alarms that the user does not have to configure such as for lag errors, encoder errors and much more. If an error is now triggered, it will be shown in the HMI application. The "MpAlarmXHistory.Export = TRUE" command exports a CSV file containing the entire alarm history to the specified data storage device. This data storage device can be a USB flash drive, for example.

### 1.4.2 Use case 2: User alarm management with an HMI application

#### Requirement

A conveyor belt on a machine is transporting PET bottles. During the process, the conveyor belt travels through a heating zone. Monitoring the temperature of the heating zone is implemented in the application. An alarm is triggered as soon as the temperature in the heating zone exceeds the upper limit value.

All alarms should be visible in the HMI application.

#### Solution

**Component list**

- **MpAlarmXCore** (own MpLink): Collects all alarms from mapp components
- **MpAlarmXSet** (MpLink from MpAlarmXCore): Sets a user alarm
- **MpAlarmXReset** (MpLink from MpAlarmXCore): Resets a user alarm
- **MpAlarmXListUI** (MpLink from MpAlarmXCore): Contains all necessary parameters and structures for establishing a connection to an HMI application and displaying all alarms from MpAlarmXListUI.
- **MpAxisBasic** (own MpLink): Axis control for the conveyor belt

**Connection diagram**

**Alarm configuration**

Alarm properties can be edited in the MpAlarmXCore configuration. The name of the user alarm is specified under "Name". This name is used later to identify the alarm in the application when it comes time to set, reset and acknowledge it. The alarm text that will also be shown later in the HMI application is added under "Message". "Code" and "Severity" specify the alarm code and severity of the alarm. The behavior of the alarm is determined by "Behavior". In our case, we are dealing with a persistent alarm since the alarm should remain active until the temperature returns to its normal range.

**Using the mapp components**

The **MpAlarmXCore**, **MpAlarmXSet**, **MpAlarmXReset** and **MpAlarmXListUI** components are added for alarm management, and the **MpAxisBasic** component is added to control the conveyor belt.
The components are connected to one another as depicted in "Connection diagram".

All components are then configured. For information about making a connection to an HMI application using MpAlarmXListUI, see section VC4 templates.

The name of the user alarm (TemperatureHigh) is specified on input "Name" of components MpAlarmXSet and MpAlarmXReset. If the alarm is now triggered in the heating zone, the MpAlarmXSet component must be called. This causes the user alarm to be active and displayed in the HMI application.

If the temperature of the heating zone returns to the normal range, calling the MpAlarmXReset component resets the alarm so that it is no longer active.

1.4.3 Use case 3: Advanced alarm management for a production line

Requirement

A production line that produces plastic profiles consists of several stations, such as an extruder, stacker and saw. A controller oversees the line, but there is a separate HMI system at each station for monitoring purposes. Another HMI device is used to monitor the entire line.

A mapp hierarchy must be established in order to implement this production line and its individual stations. For more information about setting up a mapp hierarchy, see section mapp in machine groups (mapp hierarchy). This makes it possible for the user to manage the individual stations independently on one another.

The goal now is to implement advanced alarm management. Less severe alarms at a station are only processed locally. If a new product does not make it to the stacker, for example, then the alarm is displayed locally at the stacker station and can be acknowledged by the machine operator responsible for this area. If a critical alarm occurs, it can be escalated directly to the line monitoring system. If the saw gets stuck in the material, for example, then this would be escalated immediately to the line monitoring system.

Our example contains the following alarms:

<table>
<thead>
<tr>
<th>Station</th>
<th>Alarm</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stacker</td>
<td>StackFull</td>
<td>Stacker full. The alarm should be acknowledged locally.</td>
</tr>
<tr>
<td></td>
<td>NoProduct</td>
<td>No product available. The alarm should be acknowledged locally.</td>
</tr>
<tr>
<td>Saw</td>
<td>InvalidProduct</td>
<td>Product faulty. The alarm should be acknowledged locally.</td>
</tr>
<tr>
<td></td>
<td>SawBattered</td>
<td>Saw stuck in product. This alarm should be acknowledged locally.</td>
</tr>
<tr>
<td>Extruder</td>
<td>TemperatureLow</td>
<td>The temperature in the extruder is too low. The conveyor belt in the extruder should be slowed down in response. The temperature alarm should be visible as &quot;Heating failure&quot; on the HMI system for line monitoring.</td>
</tr>
<tr>
<td></td>
<td>TemperatureHigh</td>
<td>The temperature in the extruder is too high. The conveyor belt in the extruder should be slowed down in response. The temperature alarm should be visible as &quot;Heating failure&quot; on the HMI system for line monitoring.</td>
</tr>
</tbody>
</table>
Solution

Component list

- **MpAlarmXCore** (own MpLink): Collects all alarms from mapp components globally and per machine group.
- **MpAlarmXSet** (MpLink from MpAlarmXCore): Sets a user alarm
- **MpAlarmXReset** (MpLink from MpAlarmXCore): Resets a user alarm
- **MpAlarmXCheckReaction** (MpLink from MpAlarmXCore): Used to check in the application whether a certain reaction is active.
- **MpAlarmXListUI** (MpLink from MpAlarmXCore): Contains all necessary parameters and structures for establishing a connection to an HMI application and displaying all alarms from MpAlarmXListUI locally and globally.

Connection diagram

![Connection diagram](image)

Configuration

**Setting up the production line using a mapp hierarchy**

The "MpComGroupDefault" configuration is added for each station. The mapp Link of the component that should be located in the respective machine group is now specified under "Child components".
The configuration for MpAlarmXCore is added for complete line monitoring as well as for each station. The alarm properties can be edited in this configuration. This includes the behavior of the alarm (edge or persistent). The configuration also defines which alarms are managed locally, which are escalated to line monitoring and which reaction is triggered by which alarm. The latter is configured with "Action" in the alarm mapping.

The effect of the individual options is explained in section Mapping alarms.

Stacker configuration

The two alarms "StackerFull" and "NoProduct" are mapped for the stacker in the MpAlarmXCore configuration. These alarms should be acknowledged at the station, i.e. locally. For this reason, "Remain" is selected underneath "Action". This means that the alarm is not forwarded to line monitoring and is handled locally instead.
Saw configuration

The alarms "InvalidProduct" and "SawBattered" are mapped in the saw's alarm configuration. The first alarm should be handled locally. The "SawBattered" alarm is so severe that it must be forwarded on to line monitoring. In order for the alarm to be passed on, "Escalate alarm" is selected as its action.

Extruder configuration

The two temperature alarms "TemperatureLow" and "TemperatureHigh" are mapped in the MpAlarmXCore configuration for the extruder. If one of the two alarms is triggered, then it ("TemperatureLow" or "TemperatureHigh") will be replaced by "HeatingFailure" in line monitoring. To set this up, "Replace alarm" is specified as the action and the replacement name is provided under "Name". In reaction to a temperature alarm, the extruder's conveyor belt is slowed down. For this reason, the name of the reaction "SlowDown" is specified under "Action2". The "Heating-Failure" alarm is passed on for line monitoring using the "Escalate alarm" action.
Line monitoring configuration

The escalated "SawBattered" alarm from the saw station is mapped in the alarm configuration for line monitoring. To do this, the mapp Link of the saw station and the name of the alarm is specified. If the alarm is triggered, then the entire production line should be stopped. To do this, "StopMachine" is specified as the reaction.

![Diagram of alarm configuration]

Using the mapp components

The MpAlarmXCore component is added for alarm management for line monitoring as well as for each station. The MpAlarmXSet and MpAlarmXReset components are used for setting and resetting the alarms. The MpAlarmXCheckReaction component is used to check whether a certain reaction, such as stopping the production line, is enabled. If the reaction is enabled, then the production line can be stopped.

The components are connected to one another as depicted in "Connection diagram" and then configured. For information about making a connection to an HMI application using MpAlarmXListUI, see section VC4 templates. Alarms are acknowledged directly via the HMI application.

1.4.4 Use case 4: Displaying units in the alarm text

Requirement

As soon as the temperature value rises too high in a heating coil, an alarm with text "The temperature is too high: xxx°C" is triggered. It should be possible to display the alarm message in German or English. If the alarm is displayed in German, the metric unit system should be used (°C). If the alarm message is displayed in English, the imperial unit system (°F) should be used.

The temperature should be converted automatically depending on the unit.

The unit system as well as the language are toggled in the HMI application. It is possible to switch between the metric, imperial and imperial-US systems. When using the metric system, the temperature value should be displayed in degrees Celsius; in imperial and imperial-US, degrees Fahrenheit should be displayed.

Degrees Celsius are used when working in the software.

Solution

Component list

- **MpAlarmXCore** (own MpLink): Collects all alarms from mapp components
- **MpAlarmXSet** (MpLink from MpAlarmXCore): Sets a user alarm
- **MpAlarmXReset** (MpLink from MpAlarmXCore): Resets a user alarm
- **MpAlarmXListUI** (MpLink from MpAlarmXCore): Contains all necessary parameters and structures for establishing a connection to an HMI application and displaying all alarms from MpAlarmXCore.
Preparation for using the unit system

In order to use the unit system, the OPC UA server must be enabled in the target system configuration:

The OPC UA default view configuration must then be added. The variables used in the project can be enabled as OPC UA variables in the file. For more information, see View.

Variable TemperatureValue is enabled and degrees Celsius is used as the unit since degrees Celsius is used when working in the application.

The user role system must be added in order to use the unit system. User "Anonymous" must be assigned to role "Everyone".
MpAlarmXCore configuration

To use the alarm system, the MpAlarmXCore configuration must be added. User alarm "TempHigh" is created. "Persistent alarm" is selected as the alarm behavior.

The alarm message should show the temperature value and temperature unit. For the language-independent configuration of the alarm message, the message is specified via the text system. For more information, see section "Localizing alarm texts with the text system" in Alarm texts.

To obtain the temperature value, text snippet "TempValue" is created. So that the value can be displayed in the alarm message, it is then specified in the text system. For more information, see section "Integrating application data into the alarm text" in Alarm texts.

Text system configuration

The text used to output the alarm is defined by the text system. A TMX file, the project languages to be used and the configuration of the text system are added. The TMX file must be specified in the configuration under "TMX files for target". For more information about the text system, see Text system.

The namespace and text ID must be defined in the text file. Namespace "AlarmText" and text ID "TempText" are used, just like in the MpAlarmXCore configuration. For more information, see section "Localizing alarm texts with the text system" in Alarm texts.

As soon as alarm "TempHigh" is active, the text specified under text ID "TempText" is used as the alarm message. The value of the temperature variable is displayed by {&TempValue}. The unit (e.g. °C) of the variable is displayed with [UNIT=%s].

For more information about specifying the unit, see Displaying units for variables.
MpComUnit configuration

The definition of which unit should be displayed in which unit system takes place in the MpComUnit configuration. Variable *TemperatureValue* to be recorded is specified under "Data points". Which display unit should be used per unit system is defined in section "DisplayUnits". For more information, see Unit management.

Degrees Celsius should be used in the metric system. Degrees Fahrenheit should be used for the imperial and imperial-US systems. Unit abbreviations CEL and FAH are therefore used.

For which token is used for which unit, see [here](#).

Using the mapp components

Component **MpAlarmXCore** for alarm management is added.

Components **MpAlarmXSet** and **MpAlarmXReset** are used for setting and resetting the user alarm.

The components are connected to one another as depicted in "Connection diagram" and then configured.

For information about making a connection to an HMI application using **MpAlarmXListUI**, see section VC4 templates.

To display the alarm in German or English in the HMI application, parameter "Language" contained in structure **MpAlarmXListUIConnectType** is used.

For example, if the alarm message should use English and thus the imperial unit system, "en|imperial" must be specified for "Language". For more information, see Displaying units for variables.
1.5 Libraries

1.5.1 Function blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MpAlarmXCore</td>
<td>This function block records and manages alarms.</td>
</tr>
<tr>
<td>MpAlarmXListUI</td>
<td>This function block is used to establish a connection between MpAlarmX and Visual Components 4. MpAlarmXListUI creates a list of active or unacknowledged alarms.</td>
</tr>
<tr>
<td>MpAlarmXSet</td>
<td>This function enables user-defined alarms.</td>
</tr>
<tr>
<td>MpAlarmXReset</td>
<td>This function resets user-defined alarms.</td>
</tr>
<tr>
<td>MpAlarmXResetID</td>
<td>This function resets user-defined alarms using their instance ID.</td>
</tr>
<tr>
<td>MpAlarmXAcknowledge</td>
<td>This function acknowledges specified alarms.</td>
</tr>
<tr>
<td>MpAlarmXAcknowledgeID</td>
<td>This function can acknowledge specified alarms using their instance ID.</td>
</tr>
<tr>
<td>MpAlarmXCheckState</td>
<td>This function checks the state of the specified alarm.</td>
</tr>
<tr>
<td>MpAlarmXCheckStateID</td>
<td>This function checks the current status of a specified alarm using its instance ID.</td>
</tr>
<tr>
<td>MpAlarmXCheckReaction</td>
<td>This function checks whether a certain reaction is active.</td>
</tr>
<tr>
<td>MpAlarmXConfigAlarm</td>
<td>This function block makes it possible to create and configure an alarm at runtime.</td>
</tr>
<tr>
<td>MpAlarmXConfigMapping</td>
<td>This function block can be used to export the alarm history to a specified data storage device.</td>
</tr>
<tr>
<td>MpAlarmXHistory</td>
<td>This function block is used to establish a connection between MpAlarmX and Visual Components 4. MpAlarmXHistoryUI creates a list of all alarms with status changes.</td>
</tr>
<tr>
<td>MpAlarmXHistoryUI</td>
<td>A query can be started using MpAlarmXQuery. This allows you to search for certain alarms in the alarm list.</td>
</tr>
</tbody>
</table>

1.5.1.1 MpAlarmXAcknowledge

This function acknowledges specified alarms.

Function

MpComIdentType MpLink
STRING[255] Name
MpAlarmXAcknowledge
DINT Return value
mapp Services

### 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>MpComIdentType</td>
<td>Connection to mapp (MpLink of an MpAlarmXCore configuration)</td>
</tr>
<tr>
<td>IN</td>
<td>Name</td>
<td>STRING[255]</td>
<td>Name of the alarm</td>
</tr>
<tr>
<td>OUT</td>
<td>Return value</td>
<td>DINT</td>
<td>Return value of function</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 be carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

#### 1.5.1.1.1 Description

This function uses the MpLink of an MpAlarmXCore configuration.

If an alarm should be acknowledged that has been escalated by action "Escalate alarm" to a higher-level group, the mapp Link of the higher-level alarm system must be specified in order to acknowledge the alarm. For more information, see Mapping alarms.

Executing this function acknowledges the alarm defined using parameter "Name".

Note that MpAlarmXAcknowledge cannot be used to acknowledge persistent alarms with a period in their name, e.g. "Temp.High".

"Return value" returns the function's return value. The following return values are possible:

- mpALARMX_ERR_ALARM_NOT_ACTIVE
- mpALARMX_ERR_NAME_EMPTY
- mpALARMX_ERR_NAME_NULL
- mpALARMX_ERR_MPLINK_NULL
- mpALARMX_ERR_MPLINK_INVALID
- mpALARMX_ERR_MPLINK_CORRUPT

#### 1.5.1.2 MpAlarmXAcknowledgeID

This function can acknowledge specified alarms using their instance ID.

### Function

```
MpAlarmXAcknowledgeID
MpComIdentType MpLink
UDINT InstanceID
DINT Return value
```

### 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>MpComIdentType</td>
<td>Connection to mapp (MpLink of an MpAlarmXCore configuration)</td>
</tr>
<tr>
<td>IN</td>
<td>InstanceID</td>
<td>UDINT</td>
<td>Instance ID of the alarm</td>
</tr>
<tr>
<td>OUT</td>
<td>Return value</td>
<td>DINT</td>
<td>Return value of function</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 be carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

#### 1.5.1.2.1 Description

This function uses the MpLink of an MpAlarmXCore configuration.
If an alarm should be acknowledged that has been escalated by action "Escalate alarm" to a higher-level group, the mapp Link of the higher-level alarm system must be specified in order to acknowledge the alarm. For more information, see Mapping alarms.

Executing this function acknowledges the alarm defined using parameter "InstanceID". The instance ID is the return value from function MpAlarmXSet.

**Working with ID functions is only necessary using alarms with multiple instances.**

Executing this function acknowledges the alarm defined using parameter "Name".

"Return value" returns the function's return value. The following return values are possible:

- mpALARMX_ERR_ALARM_NOT_ACTIVE
- mpALARMX_ERR_NAME_EMPTY
- mpALARMX_ERR_NAME_NULL
- mpALARM_ERR_MPLINK_NULL
- mpALARMX_ERR_MPLINK_INVALID
- mpALARMX_ERR_MPLINK_CORRUPT

### 1.5.1.3 MpAlarmXCheckReaction

This function checks whether a certain reaction is active.

**Function**

```plaintext
MpComIdentType MpLink
STRING[255] Reaction

Return value

BOOL
```

**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>MpComIdentType</td>
<td>Connection to mapp (MpLink of an MpAlarmXCore configuration)</td>
</tr>
<tr>
<td>IN</td>
<td>Reaction</td>
<td>STRING[255]</td>
<td>Name of the reaction</td>
</tr>
<tr>
<td>OUT</td>
<td>Return value</td>
<td>BOOL</td>
<td>Status of the reaction. The output is TRUE if an alarm is triggered; otherwise, it is FALSE.</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).

### 1.5.1.3.1 Description

The MpLink of an MpAlarmXCore configuration is used.

This function checks whether a certain reaction is active. This is indicated by the return value. If the value is TRUE, then the reaction is active.

The reaction to be checked is specified via parameter "Reaction".

### 1.5.1.4 MpAlarmXCheckState

This function checks the state of the specified alarm.
Function

MpAlarmXCheckState

MpComIdentType MpLink STRING[255] Name MpAlarmXStateEnum State

BOOL Return value

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>MpComIdentType</td>
<td>Connection to mapp (MpLink of an MpAlarmXCore configuration)</td>
</tr>
<tr>
<td>IN</td>
<td>Name</td>
<td>STRING[255]</td>
<td>Name used to identify the alarm</td>
</tr>
<tr>
<td>IN</td>
<td>State</td>
<td>MpAlarmXStateEnum</td>
<td>Specifies the status to be checked</td>
</tr>
<tr>
<td>OUT</td>
<td>ReturnValue</td>
<td>BOOL</td>
<td>Checks whether the instance ID of the alarm is currently in the specified status</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 be carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

1.5.1.4.1 Description

This function checks whether the specified alarm is in a certain state. The same MpLink is used as with MpAlarmXCore.

Input "State" defines the state of the alarm that should be checked. If the alarm is in the specified status, then the function returns TRUE.

The following model indicates the various states of an alarm:

Alarm States

Disabled

None

Enabled

Acknowledged

Not Acknowledged

Active

Inactive

For more information about the state of an alarm, see section Basic alarm concept.

1.5.1.5 MpAlarmXCheckStateID

This function checks the state of the specified alarm.
Function

<table>
<thead>
<tr>
<th>MpComIdentType</th>
<th>MpComIdentType</th>
<th>MpLink</th>
<th>DINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>MpComIdentType</td>
<td>MpLink</td>
<td>InstanceID</td>
<td>UDINT</td>
</tr>
<tr>
<td>MpComIdentType</td>
<td>MpLink</td>
<td>State</td>
<td>MpAlarmXStateEnum</td>
</tr>
</tbody>
</table>

**Interface**

<table>
<thead>
<tr>
<th>ID</th>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MpLink</td>
<td>MpComIdentType</td>
<td>Connection to mapp (MpLink of an MpAlarmXCore configuration)</td>
</tr>
<tr>
<td>IN</td>
<td>InstanceID</td>
<td>UDINT</td>
<td>Instance ID used to identify the alarm</td>
</tr>
<tr>
<td>IN</td>
<td>State</td>
<td>MpAlarmXStateEnum</td>
<td>Specifies the status to be checked</td>
</tr>
<tr>
<td>OUT</td>
<td>ReturnValue</td>
<td>BOOL</td>
<td>Checks whether the instance ID of the alarm is currently in the specified status</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 be carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

**1.5.1.5.1 Description**

This function checks whether the specified alarm is in a certain state. The same MpLink is used as with MpAlarmXCore.

Input "State" defines the state of the alarm that should be checked. If the alarm is in the specified status, then the function returns *TRUE*.

The following model indicates the various states of an alarm:

For more information about the state of an alarm, see section *Basic alarm concept*.

The alarm is specified on input "InstanceID" using its instance ID. The instance ID is the return value returned when setting the alarm using MpAlarmXSet.

Working with ID functions is only necessary using alarms with multiple instances.
1.5.1.6 MpAlarmXConfigAlarm

This function block makes it possible to create and configure an alarm at runtime.

Function block

## Optional parameters

<table>
<thead>
<tr>
<th>Interface</th>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MpLink</td>
<td>Pointer to Mp-</td>
<td>Connection to mapp (MpLink of an MpAlarmXCore 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>Name of the alarm that should be created or configured.</td>
</tr>
<tr>
<td>IN</td>
<td>Configuration</td>
<td>Pointer to MpAlarmXAlarmConfigType</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>MpAlarmXInfoType</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 be carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

1.5.1.6.1 Description

The same MpLink used by MpAlarmXCore is used for this component as well.
This function block reads and writes a configuration. The complete configuration 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 alarm to be created or edited is defined on input "Name". The alarm can be created or edited at runtime. Changes are applied immediately.

**Configuring structure MpAlarmXAlarmConfigType**

This configuration can be used to reconfigure existing alarms or create new ones. It is possible to change all the different properties of an alarm. Details about each of these properties can be found in the section Alarm properties.

The data type of the configuration structure describes which value is required for each parameter in order to represent an edge or a persistent alarm.

**1.5.1.7 MpAlarmXConfigMapping**

This function block makes it possible to create and configure the alarm mapping at runtime.

**Function block**

```
## Optional parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MpLink</td>
<td>Pointer to Mp-ComIdentType</td>
<td>Connection to mapp.</td>
</tr>
<tr>
<td>Enable</td>
<td>BOOL</td>
<td>The function block is active as long as this input is set.</td>
</tr>
<tr>
<td>ErrorReset</td>
<td>BOOL</td>
<td>Resets function block errors.</td>
</tr>
<tr>
<td>Type</td>
<td>MpAlarmXMappingTypeEnum</td>
<td>Defines the trigger.</td>
</tr>
<tr>
<td>Name</td>
<td>&amp;STRING</td>
<td></td>
</tr>
<tr>
<td>Configuration</td>
<td>&amp;MpAlarmXMappingConfigType</td>
<td></td>
</tr>
<tr>
<td>Load</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>Save</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>CommandBusy</td>
<td>BOOL</td>
<td></td>
</tr>
<tr>
<td>CommandDone</td>
<td>BOOL</td>
<td></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 Mp-ComIdentType</td>
<td>Connection to mapp.</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>Type</td>
<td>MpAlarmXMappingTypeEnum</td>
<td>Defines the trigger.</td>
</tr>
</tbody>
</table>
**mapp Services**

<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>Name</td>
<td>Pointer to STRING[255]</td>
<td>Defines the name of the alarm or range of values for severity.</td>
</tr>
<tr>
<td>IN</td>
<td>Configuration</td>
<td>Pointer to MpAlarmXMapping-ConfigType</td>
<td>Contains the actions that should be carried out as a reaction to the trigger.</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>MpAlarmXInfoType</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).

1.5.1.7.1 Description

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

This function block reads and writes a configuration. The configuration for a mapping takes place using the configuration structure on input "Configuration".

The "Load" command is used to read the currently configured parameters. The configuration currently active for a mapping 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, specifically to the specified mapping, where they are used immediately if possible. The name of the mapping to be created or edited is defined via the combination from inputs "Type" and "Name".

**Configuring structure MpAlarmXMappingConfigType**

This structure can be used to modify an existing alarm mapping or create a completely new one. If you would like to modify an existing mapping for a certain trigger, then first specify "Type" and "Name" and then load the current mapping. A certain action can then be modified or added using MpAlarmXMappingConfigType. If the "Type" / "Name" combination does not yet exist in the alarm mapping, then it will be created.

If an alarm is defined as trigger "Type", then the alarm name is defined using parameter "Name".

If a severity is defined as trigger "Type", then the range of values is specified using "Name". It looks something like this:

- For a range of values: 1..10
- For a certain value: 11

1.5.1.8 MpAlarmXCore

This function block records and manages alarms.

When using widget "AlarmList", MpAlarmXCore must be called in the program's initialization subroutine!

For a tutorial on the topic of alarm management, see our YouTube channel.
**Function block**

<table>
<thead>
<tr>
<th>&amp;MpComIdentType</th>
<th>MpLink</th>
<th>Active</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOO</td>
<td>Enable</td>
<td>Error</td>
<td>BOOL</td>
</tr>
<tr>
<td>BOO</td>
<td>ErrorReset</td>
<td>StatusID</td>
<td>DINT</td>
</tr>
<tr>
<td></td>
<td>ActiveAlarms</td>
<td>UDINT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PendingAlarms</td>
<td>UDINT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Info</td>
<td>MpAlarmXInfoType</td>
<td></td>
</tr>
</tbody>
</table>

**Interface**

<table>
<thead>
<tr>
<th>IO</th>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MpLink</td>
<td>Pointer to Mp-ComIdentType</td>
<td>Connection to mapp (MpLink of an MpAlarmXCore 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>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>ActiveAlarms</td>
<td>UDINT</td>
<td>Number of currently active alarms.</td>
</tr>
<tr>
<td>OUT</td>
<td>PendingAlarms</td>
<td>UDINT</td>
<td>Indicates the number of alarms that are either active or not yet acknowledged. In this way, it is an exact reproduction of the alarms being shown on the visualization device.</td>
</tr>
<tr>
<td>OUT</td>
<td>Info</td>
<td>MpAlarmXInfoType</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).

**1.5.1.8.1 Description**

The MpLink of an MpAlarmXCore configuration is used for this function block. The entire alarm system is managed using this component. This function block collects all alarms along with their current status. The alarms themselves were created and configured beforehand using MpAlarmXCore configuration. The status is modified using the different functions of MpAlarmX (MpAlarmXSet for setting, MpAlarmXAcknowledge for acknowledging, MpAlarmXListUI for visual display, etc.). In addition, all states of all reactions are managed using this function block.

To ensure that MpAlarmXCore alarms are detected immediately when booting, it is recommended to already have this function block enabled in the INIT subroutine with input "Enable". This takes care of initializing all alarms directly in the INIT subroutine.

**Counting alarms**

As soon as this function block is active, output "ActiveAlarms" will indicate how many alarms are currently active. This includes all mapp component alarms and user-defined alarms. Inactive, unacknowledged alarms are not included.

The number of all active/inactive yet unacknowledged alarms is indicated on output "PendingAlarms". These are exactly those alarms that are displayed in the HMI application if they require acknowledgment.

**Alarm management in machine groups**

There can only be one MpAlarmXCore per machine group (if not working with groups, then it follows that there is only one MpAlarmXCore in the project). For more information about machine groups, see section Creating modular applications. A use case demonstrating how MpAlarmX can look is already available: Use case 3: Advanced alarm management for a production line.

**1.5.1.9 MpAlarmXHistory**

This function block can be used to export the alarm history to a specified data storage device.
For a tutorial on the topic of alarm history, see our YouTube channel.

**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 MpAlarmXHistory configuration).</td>
</tr>
<tr>
<td>BOOL</td>
<td>Enable</td>
<td>The function block is active as long as this input is set.</td>
</tr>
<tr>
<td>BOOL</td>
<td>ErrorReset</td>
<td>Resets function block errors.</td>
</tr>
<tr>
<td>&amp;STRING[50]</td>
<td>DeviceName</td>
<td>File device (data storage medium) where the files are stored.</td>
</tr>
<tr>
<td>BOOL</td>
<td>Export</td>
<td>Saves the current alarm history to a file on the specified data storage device</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 Mp-</td>
<td>Connection to mapp (MpLink of an MpAlarmXHistory configuration).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ComIdentType</td>
<td></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>DeviceName</td>
<td>Pointer to STRING[50]</td>
<td>File device (data storage medium) where the files are stored.</td>
</tr>
<tr>
<td>IN</td>
<td>Language</td>
<td>STRING</td>
<td>Defines the language or unit used when exporting the alarm history.</td>
</tr>
<tr>
<td>IN</td>
<td>Export</td>
<td>BOOL</td>
<td>Saves the current alarm history to a file on the specified data storage device</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>MpAlarmXInfoType</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 be carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

### 1.5.1.9.1 Description

This function block uses the MpLink from an MpAlarmXHistory configuration.

MpAlarmXHistory is used to log and export the alarm history.

**Logging**

Every state change of an alarm can be logged in the alarm history. Which state changes should be logged can be specified for each alarm in the configuration. The following changes can be logged:
• Inactive --> Active: Moment at which the alarm was enabled
• Active --> Inactive: Moment at which the alarm was reset
• Unacknowledged --> Acknowledged: Moment at which the alarm was acknowledged

For more information about how to influence reporting behavior, see section Alarm properties (see History report at the bottom of the page).

Export
The "Export" command exports the alarm history as a CSV file to the specified data storage device ("DeviceName"). The export language or unit is defined with input "Language".

In order to define the language and unit system, input "Language" can be used in the following way: 'Language|UnitSystem'.

Examples would be "de|metric" or "en|imperial-us". For more information, see Unit management.

1.5.1.10 MpAlarmXHistoryUI
This function block is used to establish a connection between MpAlarmX and an HMI application. MpAlarmXHistoryUI creates a list of all alarms with status changes.

For a tutorial on the topic of alarm history, see our YouTube channel.

Function block

<table>
<thead>
<tr>
<th>Interface</th>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>UISetup</td>
<td>MpAlarmXHistoryUISetupType</td>
<td>Used to configure the elements connected to the HMI application.</td>
</tr>
<tr>
<td>IN</td>
<td>UIConnect</td>
<td>MpAlarmXHistoryUIConnectType</td>
<td>This structure contains the parameters needed for the connection to the HMI application.</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>Info</td>
<td>MpAlarmXInfoType</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 be carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

1.5.1.10.1 Description
The MpLink for this function block is the same that is used for function block MpAlarmXHistory. MpAlarmXHistoryUI represents a connection between the alarm history and an HMI application. For this, the MpAlarmXHistory and MpAlarmXCore components must be active. Data is exchanged between MpAlarmXHistoryUI and the HMI application using structure MpAlarmXHistoryUIConnectType.
**MpAlarmXHistoryUIConnectType**

Structure "UIConnect" is divided into the following areas:

- **Status**: Parameter "Status" provides information about the current activity of the alarm history.
- **AlarmList**: The alarm history is displayed using structure MpAlarmXHistoryUIAlarmListType.
- **Details**: Additional information about a selected alarm is provided by structure MpAlarmXHistoryUIDetailsType.
- **Language**: Defines the language or unit used to display the alarm history. In order to define the language and unit system, input "Language" can be used in the following way: 'Language|UnitSystem'. Examples would be "de|metric" or "en|imperial-us". For more information, see Unit management.

**MpAlarmXHistoryUISetupType**

"AlarmListSize" defines in this structure how many alarms should be displayed on one page of the HMI application. "AlarmListScrollWindow" determines how many entries from the list are displayed in advance when scrolling up and down.

**Integrating mapp View**

If the HMI application is based on mapp View, then the alarm page can be set up using the following components:

Integrating parameters is explained in the respective data types, for example with MpAlarmXHistoryUIConnectType.

**1.5.1.11 MpAlarmXListUI**

This function block is used to establish a connection between MpAlarmX and an HMI application. MpAlarmXListUI creates a list of active or unacknowledged alarms.
Function block

For a tutorial on the topic of alarm visualization, see our YouTube channel.

**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>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>UISetup</td>
<td>MpAlarmXListUISetupType</td>
<td>Used to configure the elements connected to the HMI application.</td>
</tr>
<tr>
<td>IN</td>
<td>UIConnect</td>
<td>Pointer to MpAlarmXListUIConnectType</td>
<td>This structure contains the parameters needed for the connection to the HMI application.</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>Info</td>
<td>MpAlarmXInfoType</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 be carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

**1.5.1.11.1 Description**

The MpLink for this function block is the same that is used for function block MpAlarmXCore. The component represents a connection between alarm management and an HMI application. Data is exchanged between MpAlarmXListUI and the HMI application using structure MpAlarmXListUIConnectType. An MpAlarmXCore component must be active to use MpAlarmXListUI.

**MpAlarmXListUIConnectType**

Structure "UIConnect" is divided into the following areas:

- **Status**: Parameter "Status" provides information about the current activity of the alarm list.
- **AlarmList**: Current alarms are displayed using structure MpAlarmXListUIAlarmListType.
- **AcknowledgeAll**: "AcknowledgeAll" acknowledges all displayed alarms.
- **Acknowledge**: "Acknowledge" acknowledges a selected alarm.
- **Details**: Additional information about a selected alarm is provided by structure MpAlarmXListUIDetailsType. The details also include the alarm backtrace. It searches and displays the error cause in the Logger if available.
- **Language**: Defines the language or unit used to display the alarms. In order to define the language and unit system, input "Language" can be used in the following way: 'Language|UnitSystem'. Examples would be "de|metric" or "en|imperial-us". For more information, see Unit management.
**mapp Services**

**MpAlarmXListUISetupType**

"AlarmListSize" defines in this structure how many alarms should be displayed on one page of the HMI application. "AlarmListScrollWindow" determines how many entries from the list are displayed in advance when scrolling up and down.

**Integrating mapp View**

If the HMI application is based on mapp View, then the alarm page can be set up using the following components:
Integrating parameters is explained in the respective data types, for example with `MpAlarmXListUIConnectType`.

### 1.5.1.12 MpAlarmXQuery

A query can be started using `MpAlarmXQuery`. This allows you to search for certain alarms in the alarm list.

**Function block**

```plaintext
## Optional parameters

<table>
<thead>
<tr>
<th>MpAlarmXQuery</th>
<th>&amp;MpComIdentType</th>
<th>MpAlarmXQueryModeEnum</th>
<th>&amp;STRING</th>
<th>&amp;STRING</th>
<th>MpAlarmXQueryInfoType</th>
</tr>
</thead>
<tbody>
<tr>
<td>MpXQuery</td>
<td>MpLink</td>
<td>Active</td>
<td>BOOL</td>
<td>DINT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enable</td>
<td>Error</td>
<td>BOOL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ErrorReset</td>
<td>StatusID</td>
<td>BOOL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mode</td>
<td>CommandDone</td>
<td>BOOL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Name</td>
<td>Info</td>
<td>MpAlarmXQueryInfoType</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Language</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Execute</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Next</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MpAlarmXQuery</th>
<th>BOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>MpComIdentType</td>
<td>MpLink</td>
</tr>
</tbody>
</table>
```

Data sheet V
mapp Services

The same MpLink used by MpAlarmXCore is used for this function block. With the help of MpAlarmXQuery, a query defined in the MpAlarmXCore configuration can be started. The desired query is specified on input parameter "Name".

"Mode" defines whether all existing entries should be searched or only the newest ones. In this case, the most recent alarms are those generated since the last "Update count" (see MpAlarmXCore configuration, section "Query"). Input parameter "Language" can define the language or unit in which the filtered alarms are displayed. In order to define the language and unit system, input "Language" can be used in the following way: 'Language|UnitSystem'. Examples would be "de|metric" or "en|imperial-us". For more information, see Displaying units for variables.

"Execute = TRUE" starts the request. The requested information can be seen in the process variable defined in the MpAlarmXCore configuration in section "Queries".

The following information can be taken from structure "Info":

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).

1.5.1.12.1 Description

IN MpLink Pointer to MpComIdentType Connection to mapp (MpLink of an MpAlarmXCore configuration).
IN Enable BOOL The function block is active as long as this input is set.
IN ErrorReset BOOL Resets function block errors.
IN Mode MpAlarmXQueryModeEnum Defines whether all or only the newest entries should be displayed.
IN Name STRING[50] Pointer to Name of the query. Defined in the MpAlarmXCore configuration.
IN Language STRING[20] Pointer to STRING[50] Defines the language and unit in which the filtered alarms should be displayed.
IN Execute BOOL Command to execute the request.
IN Next BOOL Command to display further entries of the request.
OUT Active BOOL Function block active.
OUT Error BOOL Error occurred during execution.
OUT StatusID DINT Status information.
OUT CommandBusy BOOL Function block currently executing command.
OUT CommandDone BOOL Execution successful. Function block is finished.
OUT Info MpAlarmXQueryInfoType Additional information about the component.
• **AvailableSpace**: Specifies how much space the process variable defined in the **MpAlarmXCore configuration** in section "Queries" offers. If the process variable is a STRING array of 100 elements, 100 is displayed.

• **RowsRead**: Indicates how many entries have been read by the query.

• **HasMoreRows**: With "HasMoreRows = FALSE", all entries of the query are visible in the process variable. With "HasMoreRows = TRUE", not all entries in the process variable could be displayed. If a STRING array of 6 elements was used as the process variable but there are 9 entries, "HasMoreRows = TRUE" is displayed. To get the last 3 entries, command "Next = TRUE" of MpAlarmXQuery must be used. The last 3 entries are written to the variable. The information is written to the first elements of the process variable:

![Diagram showing AvailableSpace and HasMoreRows](image)

1.5.1.13 **MpAlarmXReset**

This function resets user-defined alarms.

**Function**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MpComIdentType</td>
<td>MpLink</td>
<td>Connection to mapp (MpLink of an MpAlarmXCore configuration).</td>
</tr>
<tr>
<td>Name</td>
<td>STRING[255]</td>
<td>Name used to identify the alarm.</td>
</tr>
<tr>
<td>ReturnValue</td>
<td>DINT</td>
<td>Status information of function.</td>
</tr>
</tbody>
</table>

**Interface**

**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).

1.5.1.13.1 **Description**

This component uses the same mapp Link that was used by **MpAlarmXSet** to set the alarm.
MpAlarmXReset resets an alarm that was enabled by MpAlarmXSet. The name of alarm must be specified on input "Name".

1.5.1.14 MpAlarmXResetID

This function resets user-defined alarms.

**Function**

```plaintext
&MpComIdentType
MpLink
UDINT InstanceID
DINT Return value
```

**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>MpComIdentType</td>
<td>Connection to mapp (MpLink of an MpAlarmXCore configuration).</td>
</tr>
<tr>
<td>IN</td>
<td>InstanceID</td>
<td>UDINT</td>
<td>Instance ID used to identify the alarm.</td>
</tr>
<tr>
<td>OUT</td>
<td>ReturnValue</td>
<td>DINT</td>
<td>Status information of function.</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).

1.5.1.14.1 Description

This component uses the same mapp Link that was used by MpAlarmXSet to set the alarm.

MpAlarmXReset resets an alarm that was enabled by MpAlarmXSet. The instance ID of the alarm must be specified on input "InstanceID". The instance ID is the return value returned when setting the alarm using MpAlarmXSet.

- **Working with ID functions is only necessary using alarms with multiple instances.**

1.5.1.15 MpAlarmXSet

This function enables user-defined alarms.

**Function**

```plaintext
MpComIdentType
MpLink
STRING[255] Name
UDINT Return value
```

**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>MpComIdentType</td>
<td>Connection to mapp (MpLink of an MpAlarmXCore configuration).</td>
</tr>
<tr>
<td>IN</td>
<td>Name</td>
<td>STRING[255]</td>
<td>Name used to identify the alarm.</td>
</tr>
<tr>
<td>OUT</td>
<td>Return value</td>
<td>UDINT</td>
<td>Instance ID of the alarm that was enabled.</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).
1.5.1.15.1 Description
This function uses the MpLink of an MpAlarmXCore configuration.
Executing this function enables a user-defined alarm. User-defined alarms can be created and configured using the MpAlarmXCore configuration.
The “Name” of the alarm that should be enabled is specified. The return value is the instance ID of the alarm that is used to identify the alarm in the event of multiple instances.

If an alarm is enabled that does not exist in the configuration, then a new alarm is created. This alarm is created as a persistent alarm (behavior).

1.5.2 Data types and enumerators
1.5.2.1 Data types
1.5.2.1.1 MpAlarmXActionConfigType
This data type describes an action in the alarm mapping that can occur as the result of a trigger (e.g. severity >= 10).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>MpAlarmXActionEnum</td>
<td>Action that should be carried out (e.g. reaction, escalate, etc.)</td>
</tr>
</tbody>
</table>
| Name      | STRING[255] | This parameter has different meanings depending on the type:  
• Reaction: Defines the desired reaction by its name.  
• Replace alarm: Specifies the alarm name with which the triggering alarm should be replaced. If no specific meaning is listed for an action, then the parameter name is not necessary for this action type. |

1.5.2.1.2 MpAlarmXAlarmConfigType
This data type describes all the properties of an alarm.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message</td>
<td>STRING[255]</td>
<td>&quot;&quot;</td>
<td>Alarm text, either directly or as a reference to the text system. Referencing the text system looks something like this: “Test: ($Namespace/TextID)” It is important to note that the “$” characters must be specified as follows in the Structured Text programming language: “$$”. The specification thus changes to “Test: ($$Namespace/TextID)”</td>
</tr>
<tr>
<td>Code</td>
<td>UDINT</td>
<td>0</td>
<td>Alarm number (optional; identification in the application occurs using the name of the alarm)</td>
</tr>
<tr>
<td>Severity</td>
<td>UDINT</td>
<td>1</td>
<td>Severity of the alarm</td>
</tr>
<tr>
<td>Behavior</td>
<td>MpAlarmXBehaviorType</td>
<td></td>
<td>Behavior of the alarm (in the expanded view of the configuration)</td>
</tr>
<tr>
<td>Disable</td>
<td>BOOL</td>
<td>FALSE</td>
<td>Enables/disables the alarm</td>
</tr>
<tr>
<td>AdditionalInformation1</td>
<td>STRING[255]</td>
<td>&quot;&quot;</td>
<td>Additional alarm information, e.g. link to PDF, HTML, video, etc.</td>
</tr>
<tr>
<td>AdditionalInformation2</td>
<td>STRING[255]</td>
<td>&quot;&quot;</td>
<td>Additional alarm information, e.g. link to PDF, HTML, video, etc.</td>
</tr>
</tbody>
</table>

1.5.2.1.3 MpAlarmXBehaviorType
This data type describes the behavior of an alarm.
### Parameter Data type Value Description

**AutoReset**
- **BOOL**
  - **FALSE**
  - Defines whether the alarm should be reset immediately after it is set. Calling `MpAlarmXReset` is then no longer necessary.
    - **Edge alarm:** `TRUE`
    - **Persistent alarm:** `FALSE`

**Acknowledge**
- **MpAlarmXAcknowledgeConfigEnum**
  - **mpALARMX_ACK_REQ**
  - Defines if and when an alarm must be acknowledged
    - **Edge alarm:** `mpALARMX_ACK_REQ`
    - **Persistent alarm:** `mpALARMX_ACK_REQ`

**MultipleInstances**
- **BOOL**
  - **FALSE**
  - Defines whether multiple instances of an alarm are possible. For more info, see Alarm instances
    - **Edge alarm:** `TRUE`
    - **Persistent alarm:** `FALSE`

**ReactionUntilAcknowledged**
- **BOOL**
  - **TRUE**
  - Reactions to this alarm remain active until the alarm has been acknowledged.
    - **Edge alarm:** `TRUE`
    - **Persistent alarm:** `TRUE`

**HistoryReport**
- **MpAlarmXHistoryReportType**
  - Defines which state changes of the alarm should be documented in the alarm history.

---

#### 1.5.2.1.4 MpAlarmXDiagType

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><strong>MpAlarmXStatusIDType</strong></td>
<td>StatusID diagnostic structure</td>
</tr>
</tbody>
</table>

---

#### 1.5.2.1.5 MpAlarmXHistoryReportType

Defines which state changes of an alarm should be recorded in the alarm history.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| InactiveToActive | **BOOL** | **TRUE** | State change from inactive to active
  - **Edge alarm:** `TRUE`
  - **Persistent alarm:** `TRUE`
| ActiveToInactive | **BOOL** | **TRUE** | State change from active to inactive
  - **Edge alarm:** `FALSE`
  - **Persistent alarm:** `FALSE`
| UnacknowledgedToAcknowledged | **BOOL** | **TRUE** | State change from unacknowledged to acknowledged
  - **Edge alarm:** `TRUE`
  - **Persistent alarm:** `TRUE`

---

#### 1.5.2.1.6 MpAlarmXHistoryUIAlarmListType

List of all alarm status changes.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td><strong>ARRAY[0..49] of UDINT</strong></td>
<td>Level of the alarm</td>
</tr>
</tbody>
</table>
| Code | **ARRAY[0..49] of UDINT** | Code of the alarm
  - **mapp View connection:** Datapoint from `TableItem` / `Data: stringValue`
| Message | **ARRAY[0..49] of WCHAR[255]** | Description of the alarm
  - **mapp View connection:** Datapoint from `TableItem` / `Data: stringValue`
| AdditionalInformation1 | **ARRAY[0..49] of STRING[255]** | Link with additional information |
| AdditionalInformation2 | **ARRAY[0..49] of STRING[255]** | Link with additional information |
| Name | **ARRAY[0..49] of STRING[255]** | Name of the alarm
  - **mapp View connection:** Datapoint from `TableItem` / `Data: stringValue`
| Scope | **ARRAY[0..49] of STRING[255]** | Scope of the alarm |
| OldState | **ARRAY[0..49] of MpAlarmXStatusEnum** | State change of the alarm
  - **mapp View connection:** Datapoint from `TableItemImageList` / `Data: selectedIndex`
| NewState | **ARRAY[0..49] of MpAlarmXStatusEnum** | State change of the alarm
  - **mapp View connection:** Datapoint from `TableItemImageList` / `Data: selectedIndex` |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| Timestamp       | ARRAY[0..49] of STRING[50] | Timestamp at the moment of the status change  
* mapp View connection: Datapoint from TableItem / Data: stringValue |
| SelectedIndex   | UINT               | Index of the currently selected alarm                                                            |
| MaxSelection    | UINT               | Index of the last alarm                                                                           |
| PageUp          | BOOL               | Jumps to the start of the current page and then scrolls up one page at a time  
* VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0  
* mapp View connection: EventBinding, see below for more information |
| PageDown        | BOOL               | Jumps to the end of the current page and then scrolls down one page at a time  
* VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0  
* mapp View connection: EventBinding, see below for more information |
| StepUp          | BOOL               | Selects the previous entry in the list  
* VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0  
* mapp View connection: EventBinding, see below for more information |
| StepDown        | BOOL               | Selects the next entry in the list  
* VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0  
* mapp View connection: EventBinding, see below for more information |
| RangeStart      | REAL               | Shows a bar indicating which part of the list is currently being displayed. Used  
for the starting value of "Range" for a scaled element in the HMI application.  
This scaled element should correspond to the (possible) size of the list. "Range"  
should correspond to the number of entries that are displayed on one page.  
* VC4 connection: StartDatapoint from Scale |
| RangeEnd        | REAL               | Shows a bar indicating which part of the list is currently being displayed. Used  
for the ending value of "Range" for a scaled element in the HMI application.  
This scaled element should correspond to the (possible) size of the list. "Range"  
should correspond to the number of entries that are displayed on one page.  
* VC4 connection: EndDatapoint from Scale |

**VC4 connection:**

**mapp View connection:**
Event binding

In order to be able to navigate up and down in the table using the buttons, an event binding must be added for the respective button.

**PageUp**

```xml
<EventBinding>
  <Source xsi:type="widgets.brease.Button.Event" contentRefId="AlarmHistory" widgetRefId="PageUp" event="Click" />
  <EventHandler>
    <Action>
        <Method xsi:type="opcUa.NodeAction.SetValueBool" value="true" />
      </Target>
    </Action>
  </EventHandler>
</EventBinding>
```

**PageDown**

```xml
<EventBinding>
  <Source xsi:type="widgets.brease.Button.Event" contentRefId="AlarmHistory" widgetRefId="PageDown" event="Click" />
  <EventHandler>
    <Action>
        <Method xsi:type="opcUa.NodeAction.SetValueBool" value="true" />
      </Target>
    </Action>
  </EventHandler>
</EventBinding>
```

**StepUp**

```xml
<EventBinding>
  <Source xsi:type="widgets.brease.Button.Event" contentRefId="AlarmHistory" widgetRefId="StepUp" event="Click" />
  <EventHandler>
    <Action>
      <Target xsi:type="opcUa.NodeAction" refId="::AsGlobalPV:AlarmXHistoryUIConnect.AlarmList.StepUp" >
        <Method xsi:type="opcUa.NodeAction.SetValueBool" value="true" />
      </Target>
    </Action>
  </EventHandler>
</EventBinding>
```
1.5.2.1.7 MpAlarmXHistoryUIConnectType

This data type creates a connection between MpAlarmXHistoryUI and the HMI application. All alarms that have occurred and their states are listed.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>MpAlarmXHistoryUIStatusEnum</td>
<td>Current operation</td>
</tr>
<tr>
<td>AlarmList</td>
<td>MpAlarmXHistoryUIAlarmListType</td>
<td>List of all alarm status changes</td>
</tr>
<tr>
<td>Details</td>
<td>MpAlarmXHistoryUIDetailsType</td>
<td>Details about the currently selected alarm</td>
</tr>
<tr>
<td>Language</td>
<td>STRING [20]</td>
<td>Defines the language or unit used to display the alarms</td>
</tr>
</tbody>
</table>

1.5.2.1.8 MpAlarmXHistoryUIDetailsType

Structure with additional information for the alarm currently selected in the list

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>UDINT</td>
<td>Level of the alarm</td>
</tr>
<tr>
<td>Code</td>
<td>UDINT</td>
<td>Code of the alarm</td>
</tr>
<tr>
<td>Message</td>
<td>STRING [255]</td>
<td>Description of the alarm</td>
</tr>
<tr>
<td>AdditionalInformation1</td>
<td>STRING [255]</td>
<td>Link with additional information</td>
</tr>
<tr>
<td>AdditionalInformation2</td>
<td>STRING [255]</td>
<td>Link with additional information</td>
</tr>
<tr>
<td>Name</td>
<td>STRING [255]</td>
<td>Name of the alarm</td>
</tr>
<tr>
<td>Scope</td>
<td>STRING [255]</td>
<td>Scope of the alarm</td>
</tr>
<tr>
<td>OldState</td>
<td>MpAlarmXStateEnum</td>
<td>Old status of the alarm</td>
</tr>
<tr>
<td>NewState</td>
<td>MpAlarmXStateEnum</td>
<td>New status of the alarm</td>
</tr>
<tr>
<td>Timestamp</td>
<td>STRING [50]</td>
<td>Timestamp at status change</td>
</tr>
<tr>
<td>DataDescriptions</td>
<td>ARRAY [0..9] of STRING [50]</td>
<td>List with additional information about the alarm</td>
</tr>
<tr>
<td>DataValues</td>
<td>ARRAY [0..9] of STRING [255]</td>
<td>List with additional data about the alarm</td>
</tr>
<tr>
<td>TimestampDT</td>
<td>DATE_AND_TIME</td>
<td>Moment at which the alarm became active (format can be modified using MpAlarmXHistoryUISetupType)</td>
</tr>
</tbody>
</table>

1.5.2.1.9 MpAlarmXHistoryUISetupType

Additional configuration options for the HMI application.
Parameter "TimeStampPattern" specifies the format used to display the timestamp in the HMI application

- Default format: %Y %m %d %H:%M:%S:%L (e.g. "1990 25 03:13:27:253")
- Modified format (example): %d.%m.%Y %H:%M:%S:%L (e.g. "25.02.1991 13:27:03:561")

The following are additional options for adjusting the format:

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%H</td>
<td>Hour in 24-hour format. Leading zero if necessary: 00 - 23</td>
</tr>
<tr>
<td>%I</td>
<td>Hour in 12-hour format. Leading zero if necessary: 01 - 12</td>
</tr>
<tr>
<td>%k</td>
<td>Hour in 24-hour format: 0 - 23</td>
</tr>
<tr>
<td>%M</td>
<td>Minute within an hour. Leading zero if necessary: 00 - 59</td>
</tr>
<tr>
<td>%S</td>
<td>Seconds within a minute. Leading zero if necessary: 00 - 99</td>
</tr>
<tr>
<td>%L</td>
<td>Seconds since 1 January 1970</td>
</tr>
<tr>
<td>%C</td>
<td>Four-digit year divided by 100. Displayed with two digits. Leading zero if necessary: 00 - 99</td>
</tr>
<tr>
<td>%Y</td>
<td>Year as a four-digit number. Zero at the beginning if necessary</td>
</tr>
<tr>
<td>%y</td>
<td>The last two digits of the year. Leading zero if necessary: 00 - 99</td>
</tr>
<tr>
<td>%m</td>
<td>Month. Leading zero if necessary: 01 - 31</td>
</tr>
<tr>
<td>%d</td>
<td>Day of the month. Leading zero if necessary: 01 - 31</td>
</tr>
<tr>
<td>%e</td>
<td>Day of the month: 1 - 31</td>
</tr>
<tr>
<td>%H</td>
<td>Time in 24-hour format: &quot;%H:%M&quot;</td>
</tr>
<tr>
<td>%T</td>
<td>Time in 24-hour format: &quot;%H:%M:%S&quot;</td>
</tr>
<tr>
<td>%D</td>
<td>Date in the following format: &quot;%m/%d/%y&quot;</td>
</tr>
<tr>
<td>%F</td>
<td>Date in ISO 8601 format: &quot;%Y-%m-%d&quot;</td>
</tr>
<tr>
<td>%c</td>
<td>Date and time in the following format: &quot;%F %T.%L&quot;</td>
</tr>
</tbody>
</table>

1.5.2.1.10 MpAlarmXInfoType

This data type provides additional information for the component.

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

1.5.2.1.11 MpAlarmXListUILevelListType

List of alarms and associated navigation.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>ARRAY[0..49] of UDINT</td>
<td>Level of the alarm</td>
</tr>
<tr>
<td>Code</td>
<td>ARRAY[0..49] of UDINT</td>
<td>Code of the alarm</td>
</tr>
<tr>
<td>Message</td>
<td>ARRAY[0..49] of WSTRING[255]</td>
<td>Description of the alarm</td>
</tr>
<tr>
<td>InstanceID</td>
<td>ARRAY[0..49] of UDINT</td>
<td>More exact identification number of each alarm (helpful with multiple instances of the same alarm)</td>
</tr>
<tr>
<td>Name</td>
<td>ARRAY[0..49] of STRING[255]</td>
<td>Name of the alarm</td>
</tr>
<tr>
<td>Scope</td>
<td>ARRAY[0..49] of STRING[255]</td>
<td>Scope of the alarm</td>
</tr>
<tr>
<td>StateActive</td>
<td>ARRAY[0..49] of BOOL</td>
<td>State change of the alarm</td>
</tr>
<tr>
<td>Parameter</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>StateAcknowledged</td>
<td>ARRAY[0..49] of BOOL</td>
<td>State change of the alarm</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>VC4 connection:</strong> IndexDatapoint from Bitmap</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>mapp View connection:</strong> Datapoint from TableItemImageList / Data: select-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>edIndex</td>
</tr>
<tr>
<td>Timestamp</td>
<td>ARRAY[0..49] of STRING[50]</td>
<td>Timestamp at the moment of the status change</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>VC4 connection:</strong> TextDatapoint from Listbox</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>mapp View connection:</strong> Datapoint from TableItem / Data: stringValue</td>
</tr>
<tr>
<td>SelectedIndex</td>
<td>UINT</td>
<td>Index of the currently selected alarm</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>VC4 connection:</strong> IndexDatapoint from Listbox</td>
</tr>
<tr>
<td>MaxSelection</td>
<td>UINT</td>
<td>Index of the last alarm</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>VC4 connection:</strong> MaxDatapoint from Listbox</td>
</tr>
<tr>
<td>PageUp</td>
<td>BOOL</td>
<td>Jumps to the start of the current page and then scrolls up one page at a time</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>VC4 connection:</strong> Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>mapp View connection:</strong> EventBinding, see below for more information</td>
</tr>
<tr>
<td>PageDown</td>
<td>BOOL</td>
<td>Jumps to the end of the current page and then scrolls down one page at a time</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>VC4 connection:</strong> Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>mapp View connection:</strong> EventBinding, see below for more information</td>
</tr>
<tr>
<td>StepUp</td>
<td>BOOL</td>
<td>Selects the previous entry in the list</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>VC4 connection:</strong> Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>mapp View connection:</strong> EventBinding, see below for more information</td>
</tr>
<tr>
<td>StepDown</td>
<td>BOOL</td>
<td>Selects the next entry in the list</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>VC4 connection:</strong> Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>mapp View connection:</strong> EventBinding, see below for more information</td>
</tr>
<tr>
<td>RangeStart</td>
<td>REAL</td>
<td>Shows a bar indicating which part of the list is currently being displayed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Used for the starting value of &quot;Range&quot; for a scaled element in the HMI app</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lication. This scaled element should correspond to the (possible) size of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the list. &quot;Range&quot; should correspond to the number of entries that are</td>
</tr>
<tr>
<td></td>
<td></td>
<td>displayed on one page.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>VC4 connection:</strong> StartDatapoint from Scale</td>
</tr>
<tr>
<td>RangeEnd</td>
<td>REAL</td>
<td>Shows a bar indicating which part of the list is currently being displayed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Used for the ending value of &quot;Range&quot; for a scaled element in the HMI app</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lication. This scaled element should correspond to the (possible) size of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the list. &quot;Range&quot; should correspond to the number of entries that are</td>
</tr>
<tr>
<td></td>
<td></td>
<td>displayed on one page.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>VC4 connection:</strong> EndDatapoint from Scale</td>
</tr>
<tr>
<td>TimestampDT</td>
<td>ARRAY[0..49] of DATE_AND_TIME</td>
<td>Moment at which the alarm became active (format can be modified using</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MpAlarmXListUISetupType)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>VC4 connection:</strong> TextDatapoint from Listbox</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>mapp View connection:</strong> Datapoint from TableItem / Data: stringValue</td>
</tr>
</tbody>
</table>

**VC4 connection:**

![Image of VC4 connection setup]
Event binding

In order to be able to navigate up and down in the table using the buttons, an event binding must be added for the respective button.

PageUp

```xml
<EventBinding>
    <Source xsi:type="widgets.brease.Button.Event" contentRefId="AlarmPage" widgetRefId="PageUp" event="Click" />
    <EventHandler>
        <Action>
            <Method xsi:type="opcUa.NodeAction.SetValueBool" value="true" />
        </Action>
    </EventHandler>
</EventBinding>
```
<EventBinding>
  <Source xsi:type="widgets.brease.Button.Event" contentRefId="AlarmPage" widgetRefId="PageUp" event="Click" />
  <EventHandler>
    <Action>
      <Method xsi:type="opcUa.NodeAction.SetValueBool" value="true" />
    </Action>
  </EventHandler>
</EventBinding>

<EventBinding>
  <Source xsi:type="widgets.brease.Button.Event" contentRefId="AlarmPage" widgetRefId="PageDown" event="Click" />
  <EventHandler>
    <Action>
      <Method xsi:type="opcUa.NodeAction.SetValueBool" value="true" />
    </Action>
  </EventHandler>
</EventBinding>

<EventBinding>
  <Source xsi:type="widgets.brease.Button.Event" contentRefId="AlarmPage" widgetRefId="StepUp" event="Click" />
  <EventHandler>
    <Action>
      <Method xsi:type="opcUa.NodeAction.SetValueBool" value="true" />
    </Action>
  </EventHandler>
</EventBinding>

<EventBinding>
  <Source xsi:type="widgets.brease.Button.Event" contentRefId="AlarmPage" widgetRefId="StepDown" event="Click" />
  <EventHandler>
    <Action>
      <Method xsi:type="opcUa.NodeAction.SetValueBool" value="true" />
    </Action>
  </EventHandler>
</EventBinding>
1.5.2.1.12 MpAlarmXListUIBacktraceType

This data type is used to specify the error cause for the alarm as far as possible. The cause is read from the Logger. All logger entries involving the selected alarm are queried and displayed via "RecordID and "OriginID".

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecordID</td>
<td>ARRAY[0..4] of UDINT</td>
<td>Record IDs of the errors&lt;br&gt;<strong>VC4 connection:</strong> Datapoint from&lt;br&gt;text snippet / text group: MpAlarmXListRecordID TextIndexOffset from&lt;br&gt;listbox&lt;br&gt;mapp View connection: Datapoint from TableItem / Data: value</td>
</tr>
<tr>
<td>LogbookName</td>
<td>ARRAY[0..4] of STRING[100]</td>
<td>Logbook name where the error is listed&lt;br&gt;<strong>VC4 connection:</strong> Datapoint from listbox / TextDatapoint&lt;br&gt;mapp View connection: Datapoint from TableItem / Data: stringValue</td>
</tr>
<tr>
<td>EventID</td>
<td>ARRAY[0..4] of DINT</td>
<td>Event IDs of the errors&lt;br&gt;<strong>VC4 connection:</strong> Datapoint from text snippet / text group: MpAlarmXListEventID TextIndexOffset from ListBox&lt;br&gt;mapp View connection: Datapoint from TableItem / Data: value</td>
</tr>
<tr>
<td>Description</td>
<td>ARRAY[0..4] of STRING[255]</td>
<td>Descriptions of the errors&lt;br&gt;<strong>VC4 connection:</strong> Datapoint from ListBox / TextDatapoint&lt;br&gt;mapp View connection: Datapoint from TableItem / Data: stringValue</td>
</tr>
</tbody>
</table>
Information:

Figure 11: Text group: MpAlarmXListEventID

Figure 12: Text snippets in the text group

Figure 13: Text group: MpAlarmXListRecordID

mapp View connection:
1.5.2.1.13 MpAlarmXListUIConnectType

This data type can be used to create a connection between MpAlarmXListUI and the HMI application.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>MpAlarmXListUIStatusEnum</td>
<td>Current operation</td>
</tr>
<tr>
<td>AlarmList</td>
<td>MpAlarmXListUIAlarmListType</td>
<td>List of active or unacknowledged alarms</td>
</tr>
<tr>
<td>AcknowledgeAll</td>
<td>BOOL</td>
<td>Acknowledges all alarms</td>
</tr>
<tr>
<td>VC4 connection:</td>
<td></td>
<td>Datapoint from Button / Type: SetDatapoint / SetValue: 1</td>
</tr>
<tr>
<td>mapp View connection:</td>
<td>EventBinding, see example code below for more information.</td>
<td></td>
</tr>
<tr>
<td>AcknowledgeID</td>
<td>UDINT</td>
<td>Acknowledges the alarm specified by the instance ID</td>
</tr>
<tr>
<td>VC4 connection:</td>
<td></td>
<td>Datapoint from Button / Type: SetDatapoint / SetValue: Instance ID</td>
</tr>
<tr>
<td>Acknowledge</td>
<td>BOOL</td>
<td>Acknowledges the selected alarm</td>
</tr>
<tr>
<td>VC4 connection:</td>
<td></td>
<td>Datapoint from Button / Type: SetDatapoint / SetValue: 1</td>
</tr>
<tr>
<td>mapp View connection:</td>
<td>EventBinding, see example code below for more information.</td>
<td></td>
</tr>
<tr>
<td>Details</td>
<td>MpAlarmXListUIDetailsType</td>
<td>Details about the alarm selected in the list</td>
</tr>
<tr>
<td>Language</td>
<td>STRING[20]</td>
<td>Defines the language or unit used to display the alarms</td>
</tr>
</tbody>
</table>

VC4 connection:
Event binding for button "Acknowledge"

```xml
<EventBinding>
  <Source xsi:type="widgets.brease.Button.Event" contentRefId="AlarmPage" widgetRefId="Acknowledge" event="Click" />
  <EventHandler>
    <Action>
      <Target xsi:type="opcUa.NodeAction" refId="::AsGlobalPV:AlarmXListUIConnect.Acknowledge">
        <Method xsi:type="opcUa.NodeAction.SetValueBool" value="true" />
      </Target>
    </Action>
  </EventHandler>
</EventBinding>
```

Event binding for button "AcknowledgeAll"

```xml
<EventBinding>
  <Source xsi:type="widgets.brease.Button.Event" contentRefId="AlarmPage" widgetRefId="AcknowledgeAll" event="Click" />
  <EventHandler>
    <Action>
      <Target xsi:type="opcUa.NodeAction" refId="::AsGlobalPV:AlarmXListUIConnect.AcknowledgeAll">
        <Method xsi:type="opcUa.NodeAction.SetValueBool" value="true" />
      </Target>
    </Action>
  </EventHandler>
</EventBinding>
```

### 1.5.2.1.14 MpAlarmXListUIDetailsType

Structure with additional information for the alarm currently selected in the list

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity</td>
<td>UDINT</td>
<td>Level of the alarm</td>
</tr>
<tr>
<td>Code</td>
<td>UDINT</td>
<td>Code of the alarm</td>
</tr>
<tr>
<td>Message</td>
<td>STRING[255]</td>
<td>Description of the alarm</td>
</tr>
<tr>
<td>AdditionalInformation1</td>
<td>STRING[255]</td>
<td>Link with additional information</td>
</tr>
<tr>
<td>AdditionalInformation2</td>
<td>STRING[255]</td>
<td>Link with additional information</td>
</tr>
<tr>
<td>InstanceID</td>
<td>UDINT</td>
<td>More exact identification number of each alarm (helpful with multiple instances of the same alarm)</td>
</tr>
<tr>
<td>Name</td>
<td>STRING[255]</td>
<td>Name of the alarm</td>
</tr>
<tr>
<td>Scope</td>
<td>STRING[255]</td>
<td>Scope of the alarm</td>
</tr>
<tr>
<td>StateActive</td>
<td>BOOL</td>
<td>Indicates whether the alarm is active</td>
</tr>
</tbody>
</table>
### 1.5.2.1.15 MpAlarmXListUISetupType

Additional configuration options for the HMI application.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Default value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlarmListSize</td>
<td>UINT</td>
<td>20</td>
<td>Number of alarms to be displayed on one page of the list in the HMI application</td>
</tr>
<tr>
<td>AlarmListScrollWindow</td>
<td>USINT</td>
<td>1</td>
<td>Determines how many entries from the list are initially displayed when scrolling up and down</td>
</tr>
<tr>
<td>TimeStampPattern</td>
<td>STRING[50]</td>
<td>&quot;%Y-%m-%d %H:%M:%S:%L&quot;</td>
<td>Format used to display the timestamp in the HMI application</td>
</tr>
</tbody>
</table>

Parameter "TimeStampPattern" specifies the format used to display the timestamp in the HMI application

-  Default format: %Y %m %d %H:%M:%S:%L (e.g. "1990 02 03:13:27:253")
-  Modified format (example): %d.%m.%Y %H:%M:%S:%L (e.g. "25.02.1991 13:27:03:561")

The following are additional options for adjusting the format:

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%H</td>
<td>Hour in 24-hour format. Leading zero if necessary: 00 - 23</td>
</tr>
<tr>
<td>%I</td>
<td>Hour in 12-hour format. Leading zero if necessary: 01 - 12</td>
</tr>
<tr>
<td>%k</td>
<td>Hour in 24-hour format: 0 - 23</td>
</tr>
<tr>
<td>%M</td>
<td>Minute within an hour. Leading zero if necessary: 00 - 59</td>
</tr>
<tr>
<td>%S</td>
<td>Seconds within a minute. Leading zero if necessary: 00 - 99</td>
</tr>
<tr>
<td>%L</td>
<td>Milliseconds within a second. Leading zero if necessary: 000 - 999</td>
</tr>
<tr>
<td>%s</td>
<td>Seconds since 1 January 1970</td>
</tr>
<tr>
<td>%c</td>
<td>Four-digit year divided by 100. Displayed with two digits. Leading zero if necessary: 00 - 99</td>
</tr>
<tr>
<td>%y</td>
<td>Year as a four-digit number. Zero at the beginning if necessary</td>
</tr>
<tr>
<td>%m</td>
<td>Month. Leading zero if necessary: 01 - 31</td>
</tr>
<tr>
<td>%d</td>
<td>Day of the month. Leading zero if necessary: 01 - 31</td>
</tr>
<tr>
<td>%e</td>
<td>Day of the month: 1 - 31</td>
</tr>
<tr>
<td>%R</td>
<td>Time in 24-hour format: &quot;%H: %M&quot;</td>
</tr>
<tr>
<td>%T</td>
<td>Time in 24-hour format: &quot;%H:%M:%S&quot;</td>
</tr>
<tr>
<td>%D</td>
<td>Date in the following format: &quot;%m/%d/%y&quot;</td>
</tr>
<tr>
<td>%Y</td>
<td>Date in ISO 8601 format: &quot;%Y-%m-%d&quot;</td>
</tr>
<tr>
<td>%C</td>
<td>Date and time in the following format: &quot;%F %T.%L&quot;</td>
</tr>
</tbody>
</table>

### 1.5.2.1.16 MpAlarmXMappingConfigType

This data type describes all actions that can be carried out in response to a trigger (e.g. alarm becomes active).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>ARRAY[0..19] of MpAlarmXActionConfigType</td>
<td>List of actions that should be carried out after a defined trigger.</td>
</tr>
</tbody>
</table>

### 1.5.2.1.17 MpAlarmXQueryInfoType

This data type provides additional information for function block MpAlarmXQuery.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AvailableSpace</td>
<td>UDINT</td>
<td>Maximum number of entries that can be specified in the PV.</td>
</tr>
<tr>
<td>RowsRead</td>
<td>UDINT</td>
<td>Number of entries read.</td>
</tr>
<tr>
<td>HasMoreRows</td>
<td>BOOL</td>
<td>Specifies whether more information is available.</td>
</tr>
<tr>
<td>Diag</td>
<td>MpAlarmXDiagType</td>
<td>Diagnostic structure for the status ID</td>
</tr>
</tbody>
</table>

### 1.5.2.1.18 MpAlarmXStatusIDType

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.
### Data sheet V

#### Parameter | Description
--- | ---
ID | `MpAlarmXErrorEnum` Error code for mapp component
Severity | `MpComSeveritiesEnum` Severity of error
Code | `UINT` Code for the status ID. This error number can be used to search for additional information in the help system.

### 1.5.2.2 Enumerators

#### 1.5.2.2.1 MpAlarmXAcknowledgeConfigEnum

This enumerated data type indicates if and how an alarm can be acknowledged.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpALARMX_ACK_DISABLED</td>
<td>No acknowledgment required, inactive alarms automatically disappear from the alarm list.</td>
</tr>
<tr>
<td>mpALARMX_ACK_REQ</td>
<td>The alarm must be acknowledged. Acknowledgment is possible at any time.</td>
</tr>
<tr>
<td>mpALARMX_ACK_REQ_AFTER_ACTIVE</td>
<td>The alarm must be acknowledged. Acknowledgment can only occur when the alarm is no longer active.</td>
</tr>
</tbody>
</table>

#### 1.5.2.2.2 MpAlarmXActionEnum

This enumerated data type describes the different actions that can be defined in the alarm mapping as a reaction to a trigger.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpALARMX_ACT_NONE</td>
<td>No action</td>
</tr>
<tr>
<td>mpALARMX_ACT_REACTION</td>
<td>A reaction should be triggered.</td>
</tr>
<tr>
<td>mpALARMX_ACT_ESCALATE_ALARM</td>
<td>The alarm should be escalated to the next level. See Intelligent alarm handling, Mapping alarms</td>
</tr>
<tr>
<td>mpALARMX_ACT_ESCALATE_REACTION</td>
<td>The reaction should be escalated to the next level. See Intelligent alarm handling, Mapping alarms</td>
</tr>
<tr>
<td>mpALARMX_ACT_REPLACE_ALARM</td>
<td>The alarm should be escalated to the next level and replaced by another alarm in the process. See Intelligent alarm handling, Mapping alarms</td>
</tr>
<tr>
<td>mpALARMX_ACT_SEND_MESSAGE</td>
<td>The alarm should be sent using the MpTweet messaging system.</td>
</tr>
<tr>
<td>mpALARMX_ACT_REMAIN</td>
<td>The alarm should be handled locally. See Intelligent alarm handling, Mapping alarms</td>
</tr>
</tbody>
</table>

#### 1.5.2.2.3 MpAlarmXHistoryUIStatusEnum

This enumerated data type provides information about the current activity of a component.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpALARMX_HISTORY_UI_STATUS_IDLE</td>
<td>No process is currently active.</td>
</tr>
<tr>
<td>mpALARMX_HISTORY_UI_STATUS_ERROR</td>
<td>The last operation generated an error.</td>
</tr>
</tbody>
</table>

#### 1.5.2.2.4 MpAlarmXListUIStatusEnum

This enumerated data type provides information about the current activity of a component.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpALARMX_LIST_STATUS_IDLE</td>
<td>No process is currently active.</td>
</tr>
<tr>
<td>mpALARMX_LIST_STATUS_ERROR</td>
<td>The last operation generated an error.</td>
</tr>
</tbody>
</table>

#### 1.5.2.2.5 MpAlarmXMappingTypeEnum

This enumerator specifies which trigger causes a certain action to be carried out.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpALARMX_MAPPING_TYPE_ALARM_NAME</td>
<td>The trigger is an alarm.</td>
</tr>
<tr>
<td>mpALARMX_MAPPING_TYPE_SEVERITY</td>
<td>The trigger is the severity.</td>
</tr>
<tr>
<td>mpALARMX_MAPPING_TYPE_DEFAULT</td>
<td>The default action should be used.</td>
</tr>
</tbody>
</table>

#### 1.5.2.2.6 MpAlarmXQueryModeEnum

This enumerated data type specifies how the alarm list is to be searched.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpALARMX_QUERY_MODE_ALL</td>
<td>All entries in the alarm list are searched.</td>
</tr>
<tr>
<td>mpALARMX_QUERY_MODE_NEW</td>
<td>Only the newest entries in the alarm list are searched.</td>
</tr>
</tbody>
</table>
1.5.2.2.7 MpAlarmXStateEnum

This enumerated data type provides information about the current activity of a component.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpALARMX_STATE_NONE</td>
<td>Alarm is not enabled</td>
</tr>
<tr>
<td>mpALARMX_STATE_ACTIVE</td>
<td>Alarm active</td>
</tr>
<tr>
<td>mpALARMX_STATE_INACTIVE</td>
<td>Alarm inactive</td>
</tr>
<tr>
<td>mpALARMX_STATE_ACKNOWLEDGED</td>
<td>Alarm acknowledged</td>
</tr>
<tr>
<td>mpALARMX_STATE_UNACKNOWLEDGED</td>
<td>Alarm not acknowledged</td>
</tr>
</tbody>
</table>

1.5.3 Status numbers

1.5.3.1 1083367432: MpAlarmXCore not yet active

Description:
This component can only become active if MpAlarmXCore is active.

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

Cause/Solution:
• Enable MpAlarmXCore.

Constant:
mpALARMX_INF_WAIT_CORE_FB

These function blocks / functions can report this error:
• MpAlarmXHistory
• MpAlarmXListUI
• MpAlarmXHistoryUI

1.5.3.2 1083183915: Memory size changed

Description:
The memory size of the history data object was changed in the configuration. The data object has been adjusted.

Constant:
mpALARMX_INF_HISTORY_DO_SIZE_CHANGE

1.5.3.3 1083183914: Memory type changed

Description:
The memory type of the history data object was changed in the configuration. The data object has been relocated.

Constant:
mpALARMX_INF_HISTORY_DO_MEMORY_CHANGE

1.5.3.4 -2138041538: Configuration not found

Description:
A configuration could not be found for alarm {2:Name}. The default configuration will be used.

Additional information
• {2:Name}: Name of the alarm

Cause/Solution:
• Check whether the alarm was specified correctly.
Constant:

mpALARMx_WRN_ALARM_CONFIGURATION_MISSING

These function blocks / functions can report this error:

- MpAlarmXSet
- MpAlarmXReset
- MpAlarmXAcknowledge

1.5.3.5 -2138041537: MpAlarmBasic detected

Description:
An active MpAlarmBasic component was detected. MpAlarmBasic and MpAlarmX cannot be used in the same project.

Cause/Solution:
- Disable/Remove component MpAlarmBasic.

Constant:

mpALARMx_WRN_ALARM_BASIC_DETECTED

1.5.3.6 -2138041536: No messenger service found

Description:
No messenger service was found when attempting to execute action "SendMessage".

Cause/Solution:
- Add MpTweet - Messaging system.
- Check whether MpTweet is active and connected to a modem.

Constant:

mpALARMx_WRN_MESSENGER_SERVICE_NOT_FOUND

1.5.3.7 -2138041535: Error executing SendMessage

Description:
The messenger service is reporting an error when executing action "SendMessage".

Additional information
- \{1:ErrorNumber\}: Reason for the error

Constant:

mpALARMx_WRN_MESSENGER_ERROR

1.5.3.8 -1064299732: Error changing memory type

Description:
The memory type of the data object could not be changed.

Additional information
- \{1:ErrorNumber\}: Reason for the error. The error number comes from library DataObj.

Cause/Solution:
- Use MpComDump to create a file that contains all information about the components and services being used. Save the Logger information as well. Contact B&R Support with this information.
Constant:
mpALARMX_ERR_HISTORY_DO_MEMORY_CHANGE

1.5.3.9 -1064299731: Creation error

Description:
Error creating a new history data object.

Additional information
• {1:ErrorNumber}: Reason for the error. The error number comes from library DataObj.

Cause/Solution:
• Use MpComDump to create a file that contains all information about the components and services being used. Save the Logger information as well. Contact B&R Support with this information.

Constant:
mpALARMX_ERR_HISTORY_DO_CREATE

1.5.3.10 -1064299730: Error during open

Description:
Error changing the data object size. Cannot create temporary data object. Error cause {1:ErrorNumber}

Additional information
• {1:ErrorNumber}: Reason for the error.

Cause/Solution:
• Use MpComDump to create a file that contains all information about the components and services being used. Save the Logger information as well. Contact B&R Support with this information.

Constant:
mpALARMX_ERR_HISTORY_DO_RELOCATE_OPEN_EXISTING

1.5.3.11 -1064299729: Creation error

Description:
Error changing the data object size. Cannot create new data object.

Additional information
• {1:ErrorNumber}: Reason for the error

Cause/Solution:
• Use MpComDump to create a file that contains all information about the components and services being used. Save the Logger information as well. Contact B&R Support with this information.

Constant:
mpALARMX_ERR_HISTORY_DO_RELOCATE_CREATE_NEW

1.5.3.12 -1064299728: Initialization error

Description:
Error changing the data object size. Cannot format new data object.
Additional information
• \{1:ErrorNumber\}: Reason for the error

Cause/Solution:
• Use MpComDump to create a file that contains all information about the components and services being used. Save the Logger information as well. Contact B&R Support with this information.

Constant:
\texttt{mpALARMX\_ERR\_HISTORY\_DO\_RELOCATE\_FORMAT\_INIT}

1.5.3.13 -1064299727: Error during copying
Description:
Error changing the data object size. Cannot copy new data object.

Additional information
• \{1:ErrorNumber\}: Reason for the error

Cause/Solution:
• Use MpComDump to create a file that contains all information about the components and services being used. Save the Logger information as well. Contact B&R Support with this information.

Constant:
\texttt{mpALARMX\_ERR\_HISTORY\_DO\_RELOCATE\_COPY}

1.5.3.14 -1064299726: Error during deletion
Description:
Error changing the data object size. Cannot delete old data object.

Additional information
• \{1:ErrorNumber\}: Reason for the error

Cause/Solution:
• Use MpComDump to create a file that contains all information about the components and services being used. Save the Logger information as well. Contact B&R Support with this information.

Constant:
\texttt{mpALARMX\_ERR\_HISTORY\_DO\_RELOCATE\_REMOVE\_OLD}

1.5.3.15 -1064299725: Error during open or initialization error
Description:
Error during open or initializing the history file format.

Additional information
• \{1:ErrorNumber\}: Internal error number

Cause/Solution:
• Use MpComDump to create a file that contains all information about the components and services being used. Save the Logger information as well. Contact B&R Support with this information.
**Constant:**

`mpALARMX_ERR_HISTORY_FORMAT_INIT`

1.5.3.16 -1064299724: Error creating new export file

**Description:**

Error creating new export file `{2:File}` for `{3:Device}`

**Additional information**

- `{1:ErrorNumber}`: Reason for the error
- `{2:File}`: File to be created
- `{3:Device}`: Device that should use the file

**Cause/Solution:**

- Use MpComDump to create a file that contains all information about the components and services being used. Save the Logger information as well. Contact B&R Support with this information.

**Constant:**

`mpALARMX_ERR_HISTORY_EXPORT_CREATE`

1.5.3.17 -1064299723: Error writing export file

**Description:**

Error writing export file `{2:File}` for `{3:Device}`

**Additional information**

- `{1:ErrorNumber}`: Reason for the error
- `{2:File}`: File to be written
- `{3:Device}`: Device that should use the file

**Cause/Solution:**

- Use MpComDump to create a file that contains all information about the components and services being used. Save the Logger information as well. Contact B&R Support with this information.

**Constant:**

`mpALARMX_ERR_HISTORY_EXPORT_WRITE`

1.5.3.18 -1064299716: PV not found

**Description:**

Process variable `{2:PVName}` could not be found and is not available as an alarm text snippet.

**Additional information**

- `{2:PVName}`: Name of the process variable

**Cause/Solution:**

- Check how the text snippet is written.
- Check the specified variable.

**Constant:**

`mpALARMX_ERR_ATTACHED_PV_NOT_FOUND`
1.5.3.19 -1064299715: Unsupported data type of PV

Description:
Alarm text snippet with PV {2:PVName} is using an unsupported data type.

Additional information
- {2:PVName}: Name of the process variable

Cause/Solution:
- Change the data type of the process variable. For more information about unsupported data types, see "Integrating application data into the alarm text" in Alarm texts.

Constant:
mpALARMX_ERR_ATTACHED_PV_UNKNOWN_TYPE

1.5.3.20 -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:
- MpAlarmXCore
- MpAlarmXHistory

Constant:
mpALARMX_ERR_ACTIVATION

1.5.3.21 -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:
- MpAlarmXCore
- MpAlarmXCheckReaction
- MpAlarmXSet
mapp Services

- MpAlarmXReset
- MpAlarmXHistory

Constant:

mpALARM_ERR_MPLINK_NULL

1.5.3.22 -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:
- MpAlarmXCore
- MpAlarmXCheckReaction
- MpAlarmXSet
- MpAlarmXReset
- MpAlarmXHistory

Constant:

mpALARMX_ERR_MPLINK_INVALID

1.5.3.23 -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:
- MpAlarmXCore
- MpAlarmXHistory

Constant:

mpALARMX_ERR_MPLINK_CHANGED

1.5.3.24 -1064239099: Invalid MpLink contents

Description:
The value of variable "MpLink" 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:

These function blocks / functions can report this error:
- MpAlarmXCore
- MpAlarmXCheckReaction
- MpAlarmXSet
- MpAlarmXReset
- MpAlarmXHistory

Constant:
mpALARMX_ERR_MPLINK_CORRUPT

1.5.3.25 -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:

These function blocks / functions can report this error:
- MpAlarmXCore
- MpAlarmXHistory

Constant:
mpALARMX_ERR_MPLINK_IN_USE

1.5.3.26 -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.
- The function is not enabled in the configuration.

These function blocks / functions can report this error:

- MpAlarmXCore
- MpAlarmXHistory

Constant:

mpALARMX_ERR_CONFIG_INVALID

1.5.3.27 -1064116224: Specified name is NULL

Description:
The specified alarm name is NULL.

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

Cause/Solution:

- Specify the alarm name.

Constant:

mpALARMX_ERR_NAME_NULL

These function blocks / functions can report this error:

- MpAlarmXAcknowledge
- MpAlarmXSet
- MpAlarmXReset
- MpAlarmXCheckState

1.5.3.28 -1064116223: No name specified

Description:
Specified string for alarm name is blank

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 alarm name.

Constant:

mpALARMX_ERR_NAME_EMPTY

These function blocks / functions can report this error:

- MpAlarmXAcknowledge
- MpAlarmXSet
- MpAlarmXReset
- MpAlarmXCheckState
1.5.3.29 -1064116221: Missing value on UIConnect

Description:
NULL was appended to "UIConnect".

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

Cause/Solution:
• Input "UIConnect" forgotten

Constant:
mpALARMX_ERR_MISSING_UICONNECT

These function blocks / functions can report this error:
• MpAlarmXListUI
• MpAlarmXHistoryUI

1.5.3.30 -1064116220: Cannot disable/acknowledge alarm

Description:
The specified alarm is unknown / not active and therefore cannot be disabled/acknowledged.

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 alarm is active.
• Check the alarm name.
• Check the configuration.

Constant:
mpALARMX_ERR_ALARM_NOT_ACTIVE

These function blocks / functions can report this error:
• MpAlarmXReset
• MpAlarmXAcknowledge

1.5.3.31 -1064116219: No alarm selected

Description:
No alarm in the list was selected.

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

Cause/Solution:
• Select an alarm.

Constant:
mpALARMX_ERR_ALARM_NOT_SELECTED
These function blocks / functions can report this error:

- **MpAlarmXListUI**

**1.5.3.32 -1064116218: Alarm acknowledgment not permitted**

**Description:**
The alarm cannot be acknowledged using this MpAlarmXListUI.

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

**Cause/Solution:**
- If the alarm is displayed using multiple MpAlarmXListUI instances (e.g. MpAlarmXListUI is connected once directly to MpAxisBasic to only display the mapp alarms of this component and once to MpAlarmXCore to display all alarms), then acknowledgment can only take place at one place – on the MpAlarmXListUI that is connected to MpAlarmXCore via MpLink.

**Constant:**
```
mpALARMX_ERR_ACK_NOT_ALLOWED
```

These function blocks / functions can report this error:

- **MpAlarmXListUI**

**1.5.3.33 -1064116217: Invalid file device**

**Description:**
The specified file device on input "DeviceName" 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:**
- Nothing is attached to input "DeviceName".
- The attached file device does not exist.
- A connection to the desired file device could not be established.

**Constant:**
```
mpALARMX_ERR_INVALID_FILE_DEV
```

These function blocks / functions can report this error:

- **MpAlarmXHistory**

**1.5.3.34 -1064116215: MpAlarmXCore already exists**

**Description:**
There is already MpAlarmXCore in this application / application module / group.

**Reaction:**
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.
Cause/Solution:
  - Only one MpAlarmXCore is possible per application / application module.

Constant:
mpALARMX_ERR_CORE_INST_EXISTS

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

1.5.3.35 -1064116214: Error initializing memory

Description:
Error initializing memory for the alarm history. 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 specified memory medium.

Constant:
mpALARMX_ERR_MEMORY_INIT

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

1.5.3.36 -1064116213: Error during export

Description:
The file could not be exported.

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 specified memory medium.

Constant:
mpALARMX_ERR_WRITE_EXPORT_FILE

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

1.5.3.37 -1064116212: Alarm name not found

Description:
The alarm with the specified name was not found.

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

Cause/Solution:
• Can occur after command "Load" if there is still no mapping for the specified action (e.g. no entry exists in the alarm mapping for alarm "TempHigh")
• Check "Name"

Constant:
mpALARMX_ERR_NAME_NOT_FOUND

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

1.5.3.38 -1064116211: Request not found

Description:
The specified request could not be found.

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

Cause/Solution:
• Check if the request exists in the MpAlarmXCore configuration.

Constant:
mpALARMX_ERR_QUERY_NOT_FOUND

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

2 mapp Recipe: Recipe management

mapp Recipe provides all of the functions necessary for simple yet high-speed recipe management. This includes reading and writing as well as a connection to Visual Components 4.

This component is compatible with all other mapp components and therefore acts as a centralized recipe management system that consolidates the parameters from across the entire machine infrastructure.

On our YouTube channel, the tutorials mapp Recipe - Part 1 and mapp Recipe - Part 2 explain how recipe management can be implemented.

Our B&R online tutorial platform includes tutorials about mapp Recipe.
2.1 Concept

### User stories

**End user**
- As an end user, I would like to manage and use various recipes.

**Machine operator**
- As a machine operator, I would like to load the appropriate recipe at the beginning of a shift.

**Shift supervisor**
- As a shift supervisor, I would like to be able to modify and save recipes.

**Application engineer**
- As an application engineer, I would like to be able to save my recipe parameters to different recipes and load them from different recipes as well.
- As an application engineer, I would like to be able to assign my parameters to different recipes using different categories (PID control, machine parameters, etc.).
- As an application engineer, I would like an easy way to pack my PVs (strings, arrays, structures, etc.) into recipes.

#### 2.1.1 Categories

If necessary, PVs can be divided up into different recipes. There are categories set aside for this. The category of a recipe can be defined via input "Category". The name of the category can be freely chosen. As an example, PID parameters and axis parameters will be stored in different recipes. To do this, two MpRecipeXml function blocks with different categories are needed.

<table>
<thead>
<tr>
<th>mapp component</th>
<th>Recipe extension</th>
<th>Category</th>
<th>Registered PVs</th>
</tr>
</thead>
<tbody>
<tr>
<td>MpRecipeXml_PID</td>
<td>.rpid</td>
<td>&quot;PID&quot;</td>
<td>• kv</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• tr</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• tv</td>
</tr>
<tr>
<td>MpRecipeXml_Axis</td>
<td>.raxis</td>
<td>'Axis'</td>
<td>• Speed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Acceleration</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Target position</td>
</tr>
</tbody>
</table>

When registering the PVs above, the desired category must be specified on input "Category" of MpRecipeRegPar or MpRecipeRegParSync. To easily differentiate between recipes during loading, using different extensions during the creation process is recommended. This can be done by simply using different file types for the filename on input "FileName". In the example above, .rpid is used for the recipe with the PID parameters while .raxis is used for the recipe with the axis parameters. Filtering then makes it possible to quickly access the desired recipe in the HMI application.

#### 2.1.2 Auditing recipe management

Events for these components can be recorded using MpAudit. For more information about how to create an entry, see section Creating entries.

The identification number can be incorporated in the respective entry using token %act.

The following events can occur for these mapp components:

**MpRecipe event**

This event type allow events involving recipe management, such as which recipe was loaded, to be recorded.

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>MpRecipe components</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Events</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>op</td>
<td>Name of the user logged when the event occurred</td>
</tr>
<tr>
<td>ev</td>
<td>Event ID</td>
</tr>
</tbody>
</table>

Table 2: MpRecipe event
### Table 2: MpRecipe event

<table>
<thead>
<tr>
<th>Event</th>
<th>Identification number (returned via %act)</th>
<th>Additional information</th>
</tr>
</thead>
<tbody>
<tr>
<td>The loading of a recipe has been started.</td>
<td>0</td>
<td>[%op] - User&lt;br&gt;[%file] - Recipe&lt;br&gt;[%dev] - Data storage device</td>
</tr>
<tr>
<td>The loading of a recipe has been completed.</td>
<td>1</td>
<td>[%op] - User&lt;br&gt;[%file] - Recipe&lt;br&gt;[%dev] - Data storage device</td>
</tr>
<tr>
<td>Saving a recipe</td>
<td>2</td>
<td>[%op] - User&lt;br&gt;%[file] - Recipe&lt;br&gt;[%dev] - Data storage device</td>
</tr>
<tr>
<td>Renaming a recipe</td>
<td>3</td>
<td>[%op] - User&lt;br&gt;[%old] - Old name of recipe&lt;br&gt;[%new] - New name of recipe&lt;br&gt;[%dev] - Data storage device</td>
</tr>
<tr>
<td>Deleting a recipe</td>
<td>4</td>
<td>[%op] - User&lt;br&gt;%[file] - Recipe&lt;br&gt;[%dev] - Data storage device</td>
</tr>
</tbody>
</table>

#### 2.1.3 MpRecipe in a modular application

This section explains what to look out for when implementing MpRecipe in modular applications. For general information about using mapp in application modules or machine groups, see section Creating modular applications.

##### Registering PVs in an application module

If recipe management is located in an application module, it is important to remember to specify the application module name before the variable name when specifying the process variables (PV) of the MpRecipeRegPar component.

- SpecificationOfGlobalProcessVariable: ApplicationModuleName::ProcessVariable
- SpecificationOfLocalProcessVariable: ApplicationModuleName::ProcessVariable

##### Behavior in a modular application

This section explains how MpRecipe behaves when used in combination with application modules or in a machine group. The following image shows an example application in which MpRecipe is used in application modules:
The parameters stored in the recipe are changed depending on the position at which recipe handling is used. Recipe handling only takes into the recipe parameters that are in (or underneath) the same application module or the same machine group.

For the example above, this means that recipe handling in the **Mill** module and **Glue** module act completely independently of one another. Each parameter that is registered in the **Mill** application module using `MpRecipeRegPar` or `MpRecipeRegParSync` comes into a recipe that only contains parameters from this application module. Regardless of this, there is another recipe in which all parameters registered in the **Glue** application module are collected. The recipe handling in the framework project collects the parameters that are registered in the framework project, but also all parameters in the application modules. When working with machine groups (mapp hierarchy), then each machine group also collects all of the parameters from the groups underneath them.

### 2.2 Guides

#### 2.2.1 Getting started

**2.2.1.1 Recipe management in an HMI application**

This section provides a step-by-step explanation of how to create and edit a recipe using `MpRecipe`. It will also demonstrate how to display recipes in the HMI application and use web-based diagnostics.

**2.2.1.1.1 Creating a project**

You must first create a new project in Automation Studio.
2.2.1.2 Add mapp component

Add configuration for MpRecipe

The next step is to select the desired mapp component in the Toolbox and then drag-and-drop the configuration into the "mapp" folder.

Name the MpLink

When a configuration is added, the MpLink is automatically added along with it. When the configuration is opened, the MpLink is shown at the very top.
2.2.1.3 Adding a program

The next step is to add a program. In this case, we will add a Ladder Diagram program. We could use any of the other programming languages, however.

2.2.1.4 Adding MpRecipeXml to the program

Searching for MpRecipe in the Ladder Diagram Catalog

We can select the "mapp" option in the Ladder Diagram Catalog. This will display all of the available mapp components that are in the project. In our case, this includes recipe management.

Selecting a function block

MpRecipeXml forms the core of recipe management. The function block is added to the program using drag-and-drop.

Adding function block

A function block instance is generated after we have pulled the function block into the program.

2.2.1.5 Configuring MpRecipeXml

Connecting MpLink

First, connect input "MpLink" to the MpLink previously created in the Configuration View. Its address is passed on to the function block.
Enabling the component

The component is enabled by setting input "Enable". This is indicated by output "Active".

Determining the data storage device

The data storage device for recipe management is defined using input "DeviceName". An address is specified.

Naming the recipe

Input "FileName" is used to provide a name for the recipe. The filename is specified as an address.

2.2.1.1.6 Creating the data storage device

The next step is to create the data storage device we have just specified. 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.
2.2.1.1.7 Generating recipe data

Creating a data type
We will need to create a new data type with different values so that we will be able to save data in the recipe.

Creating a variable
We then create a variable, "testParameter", with this data type in the recipe management program.

Calling the variable
In order for the new variable to also exist in Automation Runtime, it must be called at least once in the software. In this case, it will be used in the initialization subroutine.

2.2.1.1.8 Adding MpRecipeRegPar to the program
Our variable must be registered with MpRecipeXml in order to include it in our recipe. This is done using function block MpRecipeRegPar.

Searching for MpRecipe in the Ladder Diagram Catalog
Navigating back to MpRecipe in the catalog

Selecting a function block
MpRecipeRegPar is used for registration. The function block is added to the program using drag-and-drop.
Adding function block
A function block instance is generated after we have pulled the function block into the program.

2.2.1.9 Configuring MpRecipeRegPar

Connecting MpLink
MpLink is also connected first for this component. We will use the same MpLink used by MpRecipeXml. It is specified by its address.

Enabling the component
The component is enabled by setting input "Enable". This is indicated by output "Active".

Registering the variable
Input "PVName" is used to register the created variable in the recipe. The PV is specified using its name. This is composed of the program and PV name.

ProgramName:PVName
The address of this string is passed on to MpRecipeRegPar.

2.2.1.10 Generating the file structure
Now generate a file structure for the memory card for ARsim.
2.2.1.11 Saving the recipe

As soon as the controller is in RUN mode, we can check both function blocks to see if they are active. If both are active, we can issue further commands.

Saving the recipe

To save, we use the "Save" command from MpRecipeXml. The recipe has been saved successfully once "CommandDone = TRUE". The "Save" command can then be reset again.

2.2.1.12 Opening and editing the recipe

Opening the recipe

The recipe in XML format can now be found on the data storage device we specified earlier. It contains the variable including all values.
Editing the recipe

It is now possible to edit the string externally, e.g. to change the contents of the string. The recipe is then saved (again externally).

2.2.1.13 Loading the recipe

The recipe can now be loaded again using the "Load" command from MpRecipeXml. That the parameters modified externally have been loaded successfully can be seen via output "CommandDone" as well as in the Watch window. The "Load" command can then be reset again.

2.2.1.14 Opening the WebXs for diagnostics

Opening the WebXs

For diagnostic purposes, we will next open the WebXs for the mapp components being used. To do so, we open Google Chrome with the following address:

- 127.0.0.1/mapp

Component view

We can then select a mapp component from the WebXs and view live data for all inputs and outputs. We can see also all connected structures.
2.2.1.15 Creating the HMI application

Duplicating recipes
In order to be able to display list with several recipes at a later time, we will make a few copies on our data storage device.

Adding a visualization object
To display the recipes in the HMI application, we have to create a VC4 object.

Configuring VNC server for the HMI application
The HMI application is then connected to a VNC server.

2.2.1.16 Adding MpRecipeUI to the program
The MpRecipeUI component can be used to help display recipe management in the HMI application.

Searching for MpRecipe in the Ladder Diagram Catalog
Navigating back to MpRecipe in the catalog

Selecting a function block
The HMI application is managed by MpRecipeUI. The function block is added to the program using drag-and-drop.
Adding function block
A function block instance is generated after we have pulled the function block into the program.

2.2.1.17 Configuring MpRecipeUI

Connecting MpLink
MpLink is also connected first for this component. We will use the same MpLink used by MpRecipeXml. It is specified by its address.

Enabling the component
The component is enabled by setting input "Enable". This is indicated by output "Active".

Appending UISetup
The UISetup structure (MpRecipeUISetupType) is then connected to the function block. This allows us to configure how many recipes we want to display on each page.
Configuring the size of the list
A list size of 3 for the UISetup structure can be configured directly in the initialization subroutine.

Appending UIConnect
The UIConnect structure (MpRecipeUIConnectType) is connected to MpRecipeUI via its address. The contents of the structure will be connected to the data points of the VC4-based HMI application later.

2.2.1.18 Creating recipe management in the HMI application
Next, an HMI application for the recipe list and keys for loading and saving are created.

Adding a recipe list
The recipes can be shown in the HMI application using a listbox. Because the list size has been limited to 3, only 3 recipes will be shown at a time. The listbox is connected to the UIConnect structure via RcpCtrl.UIConnect.Recipe.List.Names. The index of the listbox is connected to RcpCtrl.UIConnect.Recipe.List.SelectedIndex, with the maximum index connected to RcpCtrl.UIConnect.Recipe.List.MaxSelection.
Navigation

The two buttons above and below the listbox can be used to navigate through the list. They are connected to `RcpCtrl.UIConnect.Recipe.List.StepUp` and `-StepDown`.

Variable for displaying the recipes

A variable from the registered PV is connected to the HMI application so that it is possible to determine whether loading or saving a recipe was successful or not. In this case, we will connect a numeric field to `RcpCtrl.testParameter.testUSINT`.

Key for saving

We will add a key for saving the selected recipe. This key is connected to `RcpCtrl.UIConnect.Recipe.Save`. 
Key for loading
We will add a key for loading the selected recipe. This key is connected to `RcpCtrl.UIConnect.Recipe.Load`.

As soon as the HMI application is finished, we can transfer the current software to the controller.

2.2.1.19 Testing the HMI application

Opening the VNC viewer
The HMI application can be opened with VNC Viewer.

Scrolling, saving and loading
Now it is possible to scroll through the recipe list in the HMI application. Recipes can be loaded and saved as needed.
2.3 Configuration

2.3.1 MpRecipeXml configuration

General settings
Different storage settings can be defined in the first part of the configuration. "Save automatically" allows the recipe file to be saved automatically. How often the file should be saved must be defined.

The recipe file can be encrypted with "Encrypt". "Passphrase" is the key used to encrypt the file. A 256-bit AES key is used. The user cannot independently decrypt the file. "PassPhrase" is only needed to provide the application developer with the possibility of affecting the degree of security. "PassPhrase" must be changed in order to ensure encryption takes places sensibly.

"Auditing" defines whether events in recipe management should be recorded. For more information, see Auditing recipe management.

"Auto file extension" defines whether the file extension (.xml) is appended automatically.

Signing recipes
A recipe can be signed using parameter Signature. This ensures that it can only be loaded if the recipe has not been altered. The signature is generated from a combination of Passphrase and the contents of the recipe. When loading the recipe, the system now searches for a file in the same file device with the same name as the recipe but with extension .mprecipesgn. The signature it contains helps to check whether the content of the recipe is valid. If the contents are modified, then the signature must also be adjusted. This is possible in 2 ways:

- Saving using MpRecipe
- Creating the recipe and signature externally

Signing using MpRecipe
If the recipe is saved by, then a new signature is created. Its contents are defined via the application. Passphrase is specified in the configuration. When saved, the file containing the signature is adjusted based on Passphrase and the contents of the recipe.

Creating the recipe and signature externally
An HMAC-SHA256 hash algorithm is used to create the signature. For this, the exact contents of the recipe must be known as well as Passphrase. A file with the name of the recipe and extension .mprecipesgn is then created.

This extension is specified after the .xml extension, for example "MyRecipe.xml.mprecipesgn"!

The signature is copied to this file. The contents of this file are also in XML format. For more information, see Use case 6: Creating a signed recipe with an external tool. Both files – the recipe and signature file – are then copied to same folder on the controller.

2.3.2 MpRecipeCsv configuration
General settings

Different storage settings can be defined in the first part of the configuration. "Save automatically" allows the recipe file to be saved automatically. How often the file should be saved must be defined.

The recipe file can be encrypted with "Encrypt". "Passphrase" is the key used to encrypt the file. A 256-bit AES key is used. The user cannot independently decrypt the file. "PassPhrase" is only needed to provide the application developer with the possibility of affecting the degree of security. "PassPhrase" must be changed in order to ensure encryption takes places sensibly.

"Auditing" defines whether events in recipe management should be recorded. For more information, see Auditing recipe management.

"Auto file extension" defines whether the file extension (.xml) is appended automatically.

Signing recipes

A recipe can be signed using parameter Signature. This ensures that it can only be loaded if the recipe has not been altered. The signature is generated from a combination of Passphrase and the contents of the recipe. When loading the recipe, the system now searches for a file in the same file device with the same name as the recipe but with extension .mprecipesgn. The signature it contains helps to check whether the content of the recipe is valid.

If the contents are modified, then the signature must also be adjusted. This is possible in 2 ways:

- Saving using MpRecipe
- Creating the recipe and signature externally

Signing using MpRecipe

If the recipe is saved by, then a new signature is created. Its contents are defined via the application. Passphrase is specified in the configuration. When saved, the file containing the signature is adjusted based on Passphrase and the contents of the recipe.

Creating the recipe and signature externally

An HMAC-SHA256 hash algorithm is used to create the signature. For this, the exact contents of the recipe must be known as well as Passphrase. A file with the name of the recipe and extension .mprecipesgn is then created.

This extension is specified after the .csv extension, for example "MyRecipe.csv.mprecipesgn"!

The signature is copied to this file. The contents of this file are in XML format. For more information, see Use case 6: Creating a signed recipe with an external tool. Both files – the recipe and signature file – are then copied to same folder on the controller.

2.4 Use cases

2.4.1 Use case 1: Machine with simple recipe management

Requirement

A machine should store recipe data such as ingredients and quantities of a product (e.g. coffee) to a file. Each project (e.g. cappuccino, espresso, latte macchiato) should have its own recipe file. One of the three coffee types listed above can be selected and loaded in the application.

Solution

Component list

- MpRecipeXml (own MpLink): Saves and loads the recipe files
- MpRecipeRegPar (MpLink from MpRecipeXml): Registers the process variables. Individual variables or entire structures can be registered.

Connection diagram
Using the mapp components

A recipe can be loaded and saved with the **MpRecipeXml** component. The contents are defined using the **MpRecipeRegPar** component. **MpRecipeRegPar** is used to register the entire structure (ingredients and quantities) of the coffee.

**Setup/Settings**

Registration of a **local** variable:

![Registration of a local variable](image)

Registration of a **global** variable

![Registration of a global variable](image)

### 2.4.2 Use case 2: Machine with product and machine parameters (category)

**Requirement**

A machine should save product parameters such as ingredients, quantity and data as well as various machine parameters (PIDs). Both the product and machine parameters should be stored on two different data storage devices using XML files and then loaded independently of each other. In addition, it should be possible to work with the machine parameters on a special service screen in the HMI application.

**Solution**

**Component list**

- **MpRecipeXml** (own MpLink): Saves and loads the recipe file
- **MpRecipeRegPar** (MpLink from MpRecipeXml): Registers process variables (PVs). The recipe structure and settings are registered with this component.
- **MpRecipeUI** (MpLink from MpRecipeXml): Connects to the VC4-based HMI application.

**Connection diagram**

![Connection diagram](image)

**Using the mapp components**

**MpRecipeXML** is used to save and load process variables in XML files. Since the files should be saved to two separate recipes, two different **MpRecipeXML** instances are used. In order to separate the product and machine parameters, they are divided into two different structures.

The product parameter structure is registered using under the "Product" category using **MpRecipeRegPar**. The category is determined by input "Category". All parameters for which a category is specified are saved in the recipe for the same category.
The second structure for the machine parameters is registered under the "Machine" category with another \texttt{MpRecipeRegPar}.

The same category names are also used for the two \texttt{MpRecipeXml} instances. This means that the first \texttt{MpRecipeXml} instance saves the product parameters, while the second instance saves the machine parameters.

In order to see this separation of the two recipes in the HMI application, two different \texttt{MpRecipeUI} instances are used. They can manage both recipes independently of each other. One is connected to the \texttt{MpRecipeXml} for the "Product" category, and the other is connected to the \texttt{MpRecipeXml} for the "Machine" category (using the same \texttt{MpLink}).

2.4.3 Use case 3: Recipe management with synchronization when saving

\textbf{Requirement}

A machine should be able to save recipes in the form of XML files. Recipe parameters can be changed in the HMI application. Before a recipe with changed data is saved, synchronization should take place. This ensures that only complete recipes as well as a consistent data image is saved in the recipe files.

\textbf{Solution}

\textbf{Component list}

- \texttt{MpRecipeXml} (own \texttt{MpLink}): Saves and loads the recipe file
- \texttt{MpRecipeRegParSync} (MpLink from \texttt{MpRecipeXml}): Registration of process variables (PVs). Entries structures can also be registered.

\textbf{Connection diagram}

\textit{Using the mapp components}

With \texttt{MpRecipeXml}, PV data can be saved to an XML file. Since it should be possible to work on the parameters from the HMI application, \texttt{MpRecipeRegParSync} must be used to perform a synchronized save. This ensures that recipes are only saved as a complete set of parameter values.

One parameter structure is connected to the HMI application; another parameter structure is registered with \texttt{MpRecipeRegParSync}. If the save command is now called on \texttt{MpRecipeXml}, then "SaveNotification" is set on \texttt{MpRecipeRegParSync}. This is notification that a save procedure should begin. The application can now make sure of a consistent data image by copying the parameter structure connected to the HMI application and checking its plausibility.

As soon as the data is copied, the command is set to "ConfirmSave" on \texttt{MpRecipeRegParSync}.

ConfirmSave notifies \texttt{MpRecipeXml} that the parameters are secure and consistent and that the PV data can now be written to the recipe file.

2.4.4 Use case 4: Secure update of machine with new recipe data (UpdateNotification)

\textbf{Requirement}

An injection molding machine should be able to save recipes in the form of XML files. Recipe parameters can be changed in the HMI application and loaded at any time. Before a recipe with modified data is used by the machine, it should be ensured that the production cycle is located at the end. In this way, the new recipe data will not be applied during an injection procedure, which would result in faulty production, but rather when the product is finished and a new cycle can safely take place.

\textbf{Solution}

\textbf{Component list}

- \texttt{MpRecipeXml} (own \texttt{MpLink}): Saves and loads the recipe file
- \texttt{MpRecipeRegPar} (MpLink from \texttt{MpRecipeXml}): Registration of process variables (PVs). Entries structures can also be registered.
- \texttt{MpRecipeUI} (MpLink from \texttt{MpRecipeXml}): Connects to the VC4-based HMI application.
Using the mapp components

With **MpRecipeXml**, PV data can be saved to an XML file. The contents are defined using the **MpRecipeRegPar** component.

The entire structure of the product is registered using **MpRecipeRegPar**. In our case, we have 2 parameter structures. The first is for the current product parameters that are registered using MpRecipeRegPar. The second structure is directly connected to the machine, i.e. it can affect the injection procedure, for example.

A load command ("Load = TRUE") is possible at any time. Parameter "UpdateNotification" must be used, however, to ensure that the changed parameters are only applied at the end of a product cycle. This parameter must be set to TRUE as soon as output signal "CommandDone = TRUE".

Setting "UpdateNotification" indicates to the **MpRecipeRegPar** component that a new recipe is available. This can be read out on the output signal ("UpdateNotification = TRUE") of **MpRecipeRegPar**. If the production cycle is at an end (e.g. "ProductCycleEnd = TRUE") and the output signal "UpdateNotification = TRUE", then the recipe data can be used by the machine, for example, by exactly and only at this moment copying the recipe parameters (i.e. the structure with the current parameters) to the parameter structure that directly influences the injection procedure.

### 2.4.5 Use case 5: Protecting recipe data against tampering

#### Requirement

Recipe data should be protected against external tampering without having to encrypt the entire file. Only authorized users should be permitted to modify recipe data. If the recipe has been changed by an unauthorized user, the loading of the recipe must be prevented.

#### Solution

**Component list**

- **MpRecipeXml** (own MpLink): Saves and loads the recipe file
- **MpRecipeRegPar** (MpLink from MpRecipeXml): Registration of process variables (PVs). Entire structures can also be registered.

**Connection diagram**

![Connection Diagram](image-url)
Configuring components

Using the mapp components

The MpRecipe configuration permits a password to be specified to sign the recipe data. With this option, a signature file is created for each recipe. It contains an HMAC value for the recipe file. This HMAC value is used when loading the recipe file to ensure that it has not been modified. The signature is unique for each recipe file and defined password. For details about signing recipes, see here.

When loading the recipe, MpRecipe checks the signature against the contents of the recipe file. If the signature does not match with the contents (i.e. the contents of the file have been modified), then the data is not loaded and the function block indicates error mpRECIPE_ERR_SIGNATURE_INVALID.

If the signature file cannot be found, but the configuration specifies that a signature is needed, then the function block indicates error mpRECIPE_ERR_SIGNATURE_MISSING.

In addition, alarm mpRECIPE_ALM_LOAD_FAILED is generated for the above-mentioned errors if the mapp alarm system is used. The alarm contains a link to a Logger entry containing more information about the reason.

2.4.6 Use case 6: Creating a signed recipe with an external tool

Requirement

The recipe with the machine data is created by the machine manufacturer. The machine manufacturer would like to prevent the end customer from modifying the machine data and possibly damaging the machine. For this reason, the recipe should be signed with the machine data.

Solution

Component list

- **MpRecipeXml** (own MpLink): Saves and loads the recipe file.
- **MpRecipeRegPar** (MpLink from MpRecipeXml): Registration of process variables (PVs). Entire structures can also be registered.

Connection diagram

Configuring the component

The MpRecipe configuration permits a password to be specified to sign the recipe data.
Using the mapp components

If a recipe is now loaded, its contents and the defined password are checked to see whether the recipe contains valid data.

The recipe is created externally. The XML schema definition for all recipes with file format .xml is attached here: MpRecipeXml

The signature must then be generated and saved in a second file. The signature is a SHA256HMAC value. This signature can be generated as follows:

In our use case, the signature is generated using the tool HashCalc.

- Entering the passphrase:

  - To generate the SHA256 value, select "SHA256" and click on "Calculate":

  ![HashCalc](image)
• The calculated SHA256 value is then displayed here:

```
7371b0ba4f7e36f03e03b2d5f8e497db0337be26f3e37f708a23af52d6e355a2
```

• Selecting the HMAC key:
  First, copy the SHA256 value. Then select HMAC and add the key as a "Hex string".
Now specify the recipe file. Select data format "File" and then the recipe file.

After selecting the file, click on "Calculate".

The signature is then displayed here:
This value is then saved in a new file. This file must have the same name as the recipe file, but the file extension is .mprecipesgn (e.g. Recipe1.xml.mprecipesgn).

The structure of the file must look like this, with the previously generated SHA256 value entered under "Value".
If the recipe file and signature file have the same name and are saved on the same data storage device (defined with "DeviceName"), and the signature option is enabled in the configuration, then MpRecipe will check the signature when loading the file. If the recipe file has not been changed, then the check will be successful. Otherwise, an error will be indicated.

2.5 Libraries

2.5.1 Function blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MpRecipeXml</td>
<td>This function block makes it possible to load parameters from a certain file or write parameters to that file (XML format).</td>
</tr>
<tr>
<td>MpRecipeCsv</td>
<td>This function block makes it possible to load parameters from a certain file or write parameters to that file (CSV format).</td>
</tr>
<tr>
<td>MpRecipeRegPar</td>
<td>This function block can be used to declare a PV as a recipe parameter.</td>
</tr>
<tr>
<td>MpRecipeRegParSync</td>
<td>This function block can be used to declare a PV as a recipe parameter and perform synchronization when saving a recipe.</td>
</tr>
<tr>
<td>MpRecipeUI</td>
<td>This function block creates a connection to Visual Components 4 HMI applications in order to display a list of possible recipes.</td>
</tr>
</tbody>
</table>

2.5.1.1 MpRecipeCsv

This function block makes it possible to load parameters from a certain file or write parameters to that file.

**Function block**

```text
# Optional parameters
```

```markdown
<table>
<thead>
<tr>
<th>MpRecipeCsv</th>
<th>MpComIdentType</th>
<th>MpLink</th>
<th>Active</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BOOL</td>
<td>Enable</td>
<td>Error</td>
</tr>
<tr>
<td></td>
<td>BOOL</td>
<td>ErrorReset</td>
<td>StatusID</td>
</tr>
<tr>
<td>&amp;STRING</td>
<td>DeviceName</td>
<td>CommandDone</td>
<td></td>
</tr>
<tr>
<td>&amp;STRING</td>
<td>FileName</td>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>BOOL</td>
<td>Load</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BOOL</td>
<td>Save</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```markdown
MpRecipeCsvInfoType
```
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 MpRecipeCsv 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>DeviceName</td>
<td>STRING[50]</td>
<td>File device (data storage medium) where the files are stored.</td>
</tr>
<tr>
<td>IN</td>
<td>FileName</td>
<td>Pointer to STRING[255]</td>
<td>Name of the file that contains the recipe.</td>
</tr>
<tr>
<td>IN</td>
<td>Header</td>
<td>Pointer to MpRecipeCsvHeaderType</td>
<td>Header information that is either written or read (depending on the command). An header is not added to the file if NULL is appended or the header structure is empty.</td>
</tr>
<tr>
<td>IN</td>
<td>Category</td>
<td>STRING[50]</td>
<td>Name of the category used to organize the parameters.</td>
</tr>
<tr>
<td>IN</td>
<td>LoadType</td>
<td>MpRecipeCsvLoadEnum</td>
<td>Defines how the recipe is loaded. By default, the entire recipe is loaded.</td>
</tr>
<tr>
<td>IN</td>
<td>Load</td>
<td>BOOL</td>
<td>Loads the recipe parameters from the specified file (&quot;FileName&quot;)</td>
</tr>
<tr>
<td>IN</td>
<td>Save</td>
<td>BOOL</td>
<td>Saves the current recipe parameters to a defined file</td>
</tr>
<tr>
<td>IN</td>
<td>UpdateNotification</td>
<td>BOOL</td>
<td>A rising edge on this input induces all MpRecipeRegPar and MpRecipeRegParSync to have the value TRUE on their output &quot;UpdateNotification&quot; for one cycle. This makes it possible to inform the connected components of new recipe values.</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>MpRecipeCsvInfoType</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).

2.5.1.1.1 Description

The MpLink of an MpRecipeCsv configuration is used for this function block.

The following are the main functions of this function block:

- **Load**: Recipe data can be loaded to the respective process variables..
- **Save**: Current data from registered PVs can be written to recipes.
  
  If the specified recipe does not yet exist, it will be created. If no extension is specified for a new recipe, .csv will be added automatically. Automatic appending of the extension can be disabled in the MpRecipeCsv configuration under section Auto file extension.
  
  If multiple instances of MpRecipeCsv with different registered PVs exist in a group created by MpComGroup, then all PVs in the group are written to recipes if "Save" is set to TRUE.

- **Version control**: Reading the header information makes it possible to check which version of the recipe is currently being used.

The recipes processed by this mapp component are CSV files.

- **The CSV file format is open and can be manipulated with a basic text editor if not encrypted.**

Loading a recipe

Before a load command can be issued, a valid data storage medium ("DeviceName") and recipe ("FileName") must be specified. It is then necessary to use input "LoadType" to select whether the entire recipe should be loaded or just its header information. If a structure is connected to input "Header" (MpRecipeCsvHeaderType), the recipe's header information will be stored there. The following are the possible modes for "LoadType":

- **mpRECIPE_FILE_LOAD_HEADER**: Only the contents of the recipe header are read and then written to the structure above. If a variable is not connected to input "Header", then nothing will be written. This mode is especially helpful for version control (see further below).
- **mpRECIPE_FILE_LOAD_ALL**: Reads the entire content of the recipe. The header information is written to the structure on input "Header" (if present); the recipe data is written to the respective PVs.
The "Load" command is used to read the data from the recipe. A successful load procedure is confirmed by "CommandDone = TRUE".

If a recipe is loaded that contains unregistered parameters, only the parameters that are currently registered will be loaded. Assume we are loading recipe "Coffee", which contains parameters for sugar, milk, coffee grounds and water. Because only the parameters for sugar, milk and coffee grounds are currently registered, the parameter for water is not loaded. Warning mpRECIP_WRN_LOAD_WITH_WARN is reported.

If a recipe is loaded that does not contain all currently registered parameters, however, then the parameters that are not reloaded remain unchanged. For example, recipe "Coffee" is loaded with the parameters for sugar, milk and coffee grounds. However, the parameters for sugar, milk, coffee grounds and water are currently registered. The value currently being used for the water parameter remains unchanged in this case.

If a CSV file is loaded that uses a different delimiter (";" instead of ","), then the file cannot be read correctly. The recipe cannot therefore be loaded.

After the load procedure, all MpRecipeRegPar and MpRecipeRegParSync function blocks can be informed that new parameters have been read from the recipe. This command must be enabled manually on input "UpdateNotification". This procedure is useful if the current process is not being directly operated using the recipe data, but instead the data must first be applied. This can happen if new data has been loaded and the user passes on the "UpdateNotification" signal.

### Saving a recipe

When "Save" = TRUE, the current PV data is written to the recipe. This procedure takes place differently for function blocks MpRecipeRegPar and MpRecipeRegParSync:

- **MpRecipeRegPar**: If a process variable has been registered using this function block, then the new data is stored directly in the recipe when the save command is issued.

- **MpRecipeRegParSync**: When using this function block, the transfer of data from the PV to the recipe must be confirmed. When "Save" = TRUE, the status "SaveNotification = TRUE" can be found on all MpRecipeRegParSync function blocks. This indicates that the current PV data can be written to the recipe. This must first be acknowledged, however. The save procedure is confirmed via input "ConfirmSave" on MpRecipeRegParSync; only then is the data actually transferred from the PV to the recipe. This allows the user to manipulate the data or prevent potential data inconsistencies before the data is saved.

If the name of the file ("FileName") does not have an extension, .csv will be added automatically. Automatic appending of the extension can be disabled in the MpRecipeCsv configuration under section Auto file extension.

As soon as the save is completed, output "CommandDone" is set to TRUE. If saving is delayed due to active MpRecipeRegParSync function blocks (no "ConfirmSave" confirmation occurs), then the "Save" command remains active, which is indicated by "CommandBusy = TRUE". PV data will be saved only after the process has been confirmed by all stations.

Automatic saving can be enabled in the MpRecipeCsv configuration with "Save automatically". A save interval must then be specified. "Start immediately" defines whether the recipe should be saved immediately when MpRecipeCsv is enabled.

### Version control

Version control can be implemented using inputs "Header" and "LoadType". Since it is possible to only load the header of a recipe, it is easy to determine which version of the recipe is on hand. The user can then either load another recipe or load the complete recipe.

### Recipe categories

The different PVs can be split up into several recipes via input "Category". If the same category is specified when registering a PV (MpRecipeRegPar or MpRecipeRegParSync) and managing a recipe (MpRecipeCsv), then the PV value is stored in the recipe along with the same category. More information is available here: Categories.

A separate MpRecipeRegPar or MpRecipeRegParSync function block is necessary for each PV.

### 2.5.1.1.2 Timing diagrams

The timing diagrams for managing recipes encompass the saving, loading and synchronization of parameters.
Saving a recipe

Setting "Save" to TRUE on function block MpRecipeCsv initiates the save procedure. Because this example uses several MpRecipeRegParSync function blocks, the recipe will not be saved until these function blocks provide confirmation. As soon as the save command arrives, the "SaveNotification" status is set. This indicates that the current parameters should be saved to the recipe. The parameters are transferred if this status is confirmed by the "ConfirmSave" command.

Loading a recipe

The loading of a recipe is shown here first. The "UpdateNotification" command (which is triggered by the application ¹) then indicates that new values have been written to the PVs. These outputs are only active for one cycle and then reset automatically. In this case, the signal of the MpRecipeRegPar_1 function block is delayed since it is running in a task with a slower cycle time².
"UpdateNotification" functionality

The "UpdateNotification" command is triggered on MpRecipeCsv and generates a "UpdateNotification" status signal on all connected MpRecipeRegPar function blocks that indicates that new values have been written to the PVs. This output is only active for one cycle and then reset automatically. In this example, the signal of function block MpRecipeRegPar_1 is delayed since it is running in a task with a slower cycle time.

2.5.1.2 MpRecipeRegPar

This function block can be used to register a PV as a recipe parameter.

Derived data types, arrays that do not start with index 0 and multidimensional arrays can only be registered starting with AR D4.26 or AR 4.34.

Function block

## Optional parameters
The MpLink of an MpRecipeXml/MpRecipeCsv configuration is used. This function block can be used to include a process variable in recipe management. If a PV is registered, its contents are taken into account when the recipes are loaded and saved. The validity of the PV is evaluated on a rising edge on the enable input. The PV is registered as soon as the component is active ("Active" = TRUE). If the component is no longer active, then the PV is also no longer registered. A separate MPRecipeRegPar or MPRecipeRegParSync must be used for each PV that should be in the recipe.

Saving and loading take place without delay for this component. If an "UpdateNotification" is sent out by a MpRecipeXml or MpRecipeCsv function block, this is indicated on output "UpdateNotification".

Input "Category" can be used to assign the PV to a certain recipe (see Categories).

A new PV can be registered at any time (specifying the name of process variables). The following format is used:
mapp Services

- **Global PV**: `<PV_name>` (e.g. "gTestVariable", "gControl.Parameter.Distance")
- **Local PV**: `<task_name>:<PV_name>` (e.g. "Task1:LocalTestVariable", "AxisCtrl:Basic.Command.MoveVelocity")

**Registration limitations:**
- Arrays with more than 65535 elements cannot be registered.

### 2.5.1.3 MpRecipeRegParSync

This function block can be used to register a PV as a recipe parameter and perform synchronization when saving a recipe.

**Derived data types**, arrays that do not start with index 0 and multidimensional arrays can only be registered starting with AR D4.26 or AR 4.34.

**Function block**

#### ## Optional parameters

<table>
<thead>
<tr>
<th>&amp;MpComIdentTypeMpRecipeRegParSync</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;MpComIdentType MpLink Active BOOL</td>
</tr>
<tr>
<td>BOOL Enable Error BOOL</td>
</tr>
<tr>
<td>BOOL ErrorReset StatusID DINT</td>
</tr>
<tr>
<td>&amp;STRING PVName</td>
</tr>
<tr>
<td>BOOL ConfirmSave SaveNotification BOOL</td>
</tr>
<tr>
<td>BOOL UpdateNotification</td>
</tr>
<tr>
<td>MpRecipeInfoTypeInfo</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><strong>IN</strong></td>
<td>MpLink</td>
<td>Pointer to MpComIdentType</td>
<td>Connection to mapp (MpLink of an MpRecipeCsv/MpRecipeXml 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>PVName</td>
<td>Pointer to STRING(100)</td>
<td>Name of the PV that should be saved to the recipe.</td>
</tr>
<tr>
<td>IN</td>
<td>Category</td>
<td>Pointer to STRING(50)</td>
<td>Name of the category used to organize the parameters.</td>
</tr>
<tr>
<td>IN</td>
<td>ConfirmSave</td>
<td>BOOL</td>
<td>Confirms that the current data of the registered PV can be saved to the recipe. This is used in connection with &quot;SaveNotification&quot;.</td>
</tr>
<tr>
<td><strong>OUT</strong></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>
</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).

#### 2.5.1.3.1 Description

The MpLink of an MpRecipeXml/MpRecipeCsv configuration is used.

This function block can be used to include a process variable in recipe management. If a PV is registered, its contents are taken into account when the recipes are loaded and saved. The validity of the PV is evaluated on a rising edge on the enable input. The PV is registered as soon as the component is active ("Active" = TRUE). If the component is no longer active, then the PV is also no longer registered. A separate MPRecipeRegPar or MpRecipeRegParSync must be used for each PV that should be in the recipe.

Loading take place without delay for this component. If an "UpdateNotification" is sent out by a MpRecipeXml or MpRecipeUI function block, this is indicated on output "UpdateNotification".

Saving is delayed in order to allow the user to manipulate the data before it is saved in the recipe. If "Save = TRUE", this component will save the status "SaveNotification" on the output. It is then necessary to use input "ConfirmSave" to confirm that the PV data should be transferred to the recipe.

Input "Category" can be used to assign the PV to a certain recipe (see Categories).

A new PV can be registered at any time (specifying the name of process variables). The following format is used:

- **Global PV**: `<PV_name>` (e.g. "gTestVariable", "gControl.Parameter.Distance")
- **Local PV**: `<task_name>:<PV_name>` (e.g. "Task1:LocalTestVariable", "AxisCtrl:Basic.Command.MoveVelocity")

#### Registration limitations:

- Arrays with more than 65535 elements cannot be registered.

#### 2.5.1.4 MpRecipeUI

This function block creates a connection to Visual Components 4 HMI applications in order to display a list of possible recipes.

#### Information:

The MpFile library must be transferred to the controller in order to use this function block.

On our YouTube channel, the tutorials mapp Recipe - Part 1 and mapp Recipe - Part 2 explain how recipe management can be implemented.

Our B&R online tutorial platform includes tutorials about mapp Recipe.
mapp Services

Function block

**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 MpRecipeCsv/MpRecipeXml 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>UISetup</td>
<td>MpRecipeUISetupType</td>
<td>Used to configure the elements connected to the HMI application.</td>
</tr>
<tr>
<td>IN</td>
<td>UIConnect</td>
<td>Pointer to MpRecipeUIConnectType</td>
<td>This structure contains the parameters needed for the connection to the HMI application.</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>Info</td>
<td>MpRecipeInfoType</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 be carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

**2.5.1.4.1 Description**

The same MpLink used by MpRecipeXml is used for this component as well. This component represents a connection between recipe management and a Visual Components 4 HMI application. Data is exchanged between MpRecipeUI and the HMI application using structure MpRecipeUIConnectType.
**MpRecipeUIConnectType**

Structure "UIConnect" is divided into the following areas:

- **Status**: Parameter "Status" provides information about the current activity of recipe management.
- **Recipe**: Structure `MpRecipeUIRecipeType` contains a list of available recipes. The data storage medium (file device) from when the recipes are read is defined by input "DeviceName" of `MpRecipeXml`. This list can be sorted and filtered to make it easier to find a recipe. Loading, saving and deleting are also handled using this structure. The recipe list is automatically regenerated after a recipe is deleted. In addition, the header of the currently selected recipe is shown immediately using this structure, which allows the description and version of the recipe to be displayed without having to load the entire recipe.
- **New**: Structure `MpRecipeUINewType` can be used to create a new recipe. The recipe list is automatically regenerated after a recipe is created.

**Search function**

Parameter "Filter" in data type `MpRecipeUIRecipeType` can be used to search for certain files. It is possible to use the "*" wildcard in the search function. If "*Test*.xml" is used for filtering, for example, the files "Test01.xml", "MyTest.xml" and "MyText01.xml" will be displayed. If the "*" wildcard is not used, then it will only search for "Text.xml".

If no filter is specified or only the "*" symbol, then all existing files and folders will be displayed.

**MpRecipeUISetupType**

"RecipeListSize" defines in this structure how many entries from the file list should be displayed on one page of the HMI application. "RecipeListScrollWindow" determines how many entries from the list are displayed in advance when scrolling up and down. Parameter "Confirmation" can be used to enable or disable various dialog boxes.

The `MpFile` library must be transferred to the controller in order to use this function block.
This function block makes it possible to load parameters from a certain file or write parameters to that file.

On our [YouTube channel](https://www.youtube.com), the tutorials [mapp Recipe - Part 1](https://www.youtube.com/watch?v=12345) and [mapp Recipe - Part 2](https://www.youtube.com/watch?v=67890) explain how recipe management can be implemented.


### 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>Pointer to MpComIdentType</td>
<td>Connection to mapp (MpLink of an MpRecipeXml configuration)</td>
</tr>
<tr>
<td>MpLink</td>
<td>BOOL</td>
<td>Active</td>
</tr>
<tr>
<td>Enable</td>
<td>BOOL</td>
<td>The function block is active as long as this input is set.</td>
</tr>
<tr>
<td>ErrorReset</td>
<td>BOOL</td>
<td>Resets function block errors.</td>
</tr>
<tr>
<td>&amp;STRING DeviceName</td>
<td>STRING[50]</td>
<td>File device (data storage medium) where the files are stored.</td>
</tr>
<tr>
<td>FileName</td>
<td>STRING[255]</td>
<td>Name of the file that contains the recipe. The file can also be stored in a folder on the specified data storage device. In this case, the folder name must also be specified (e.g. &quot;FolderName/Filename&quot;).</td>
</tr>
<tr>
<td>Header</td>
<td>Pointer to MpRecipeXmlHeaderType</td>
<td>Header information that is either written or read (depending on the command). A header is not added to the file if NULL is appended or the header structure is empty.</td>
</tr>
<tr>
<td>Category</td>
<td>STRING[50]</td>
<td>Name of the category used to organize the parameters.</td>
</tr>
<tr>
<td>LoadType</td>
<td>MpRecipeXmlLoadEnum</td>
<td>Defines how the recipe is loaded.</td>
</tr>
<tr>
<td>Load</td>
<td>BOOL</td>
<td>Loads the recipe parameters from the specified file (&quot;FileName&quot;).</td>
</tr>
<tr>
<td>Save</td>
<td>BOOL</td>
<td>Saves the current recipe parameters to a defined file.</td>
</tr>
<tr>
<td>UpdateNotification</td>
<td>BOOL</td>
<td>A rising edge on this input induces all MpRecipeRegPar and MpRecipeRegParSync to have the value TRUE on their output &quot;UpdateNotification&quot; for one cycle. This makes it possible to inform the connected components of new recipe values.</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 MpRecipeXml 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>DeviceName</td>
<td>Pointer to STRING[50]</td>
<td>File device (data storage medium) where the files are stored.</td>
</tr>
<tr>
<td>IN</td>
<td>FileName</td>
<td>Pointer to STRING[255]</td>
<td>Name of the file that contains the recipe. The file can also be stored in a folder on the specified data storage device. In this case, the folder name must also be specified (e.g. &quot;FolderName/Filename&quot;).</td>
</tr>
<tr>
<td>IN</td>
<td>Header</td>
<td>Pointer to MpRecipeXmlHeaderType</td>
<td>Header information that is either written or read (depending on the command). A header is not added to the file if NULL is appended or the header structure is empty.</td>
</tr>
<tr>
<td>IN</td>
<td>Category</td>
<td>Pointer to STRING[50]</td>
<td>Name of the category used to organize the parameters.</td>
</tr>
<tr>
<td>IN</td>
<td>LoadType</td>
<td>MpRecipeXmlLoadEnum</td>
<td>Defines how the recipe is loaded.</td>
</tr>
<tr>
<td>IN</td>
<td>Load</td>
<td>BOOL</td>
<td>Loads the recipe parameters from the specified file (&quot;FileName&quot;).</td>
</tr>
<tr>
<td>IN</td>
<td>Save</td>
<td>BOOL</td>
<td>Saves the current recipe parameters to a defined file.</td>
</tr>
<tr>
<td>IN</td>
<td>UpdateNotification</td>
<td>BOOL</td>
<td>A rising edge on this input induces all MpRecipeRegPar and MpRecipeRegParSync to have the value TRUE on their output &quot;UpdateNotification&quot; for one cycle. This makes it possible to inform the connected components of new recipe values.</td>
</tr>
<tr>
<td>OUT</td>
<td>Active</td>
<td>BOOL</td>
<td>Function block active.</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).

2.5.1.5.1 Description

The MpLink of an MpRecipeXml configuration is used for this function block.

The following are the main functions of this function block:

- **Load**: Recipe data can be loaded to the respective process variables.
- **Save**: Current data from registered PVs can be written to recipes. If the specified recipe does not yet exist, it will be created. If no extension is specified for a new recipe, .xml will be added automatically. Automatic appending of the extension can be disabled in the MpRecipeXml configuration under section Auto file extension.

  If multiple instances of MpRecipeXml with different registered PVs exist in a group created by MpCom-Group, then all PVs in the group are written to recipes if "Save" is set to TRUE.

- **Version control**: Reading the header information makes it possible to check which version of the recipe is currently being used.

The recipes processed by this mapp component are XML files.

The XML file format is open and can be manipulated with a basic text editor if not encrypted.

Loading a recipe

Before a load command can be issued, a valid data storage medium ("DeviceName") and recipe ("FileName") must be specified. It is then necessary to use input "LoadType" to select whether the entire recipe should be loaded or just its header information. If a structure is connected to input "Header" (MpRecipeXmlHeaderType), the recipe’s header information will be stored there. The following are the possible modes for "LoadType":

- **mpRECIPE_FILE_LOAD_HEADER**: Only the contents of the recipe header are read and then written to the structure above. If a variable is not connected to input "Header", then nothing will be written. This mode is especially helpful for version control (see further below).
- **mpRECIPE_FILE_LOAD_ALL**: Reads the entire content of the recipe. The header information is written to the structure on input "Header" (if present); the recipe data is written to the respective PVs.

The "Load" command is used to read the data from the recipe. A successful load procedure is confirmed by "CommandDone = TRUE".

If a recipe is loaded that contains unregistered parameters, only the parameters that are currently registered will be loaded. Assume we are loading recipe "Coffee", which contains parameters for sugar, milk, coffee grounds and water. Because only the parameters for sugar, milk and coffee grounds are currently registered, the parameter for water is not loaded. Warning mpRECIPE_WRN_LOAD_WITH_WARN is reported.

If a recipe is loaded that does not contain all currently registered parameters, however, then the parameters that are not reloaded remain unchanged. For example, recipe "Coffee" is loaded with the parameters for sugar, milk, coffee grounds and water. However, the parameters for sugar, milk, coffee grounds and water are currently registered. The value currently being used for the water parameter remains unchanged in this case.

After the load procedure, all MpRecipeRegPar and MpRecipeRegParSync function blocks can be informed that new parameters have been read from the recipe. This command must be enabled manually on input "UpdateNotification". This procedure is useful if the current process is not being directly operated using the recipe data, but instead the data must first be applied. This can happen if new data has been loaded and the user passes on the "UpdateNotification" signal.
Saving a recipe

When "Save" = TRUE, the current PV data is written to the recipe. This procedure takes place differently for function blocks MpRecipeRegPar and MpRecipeRegParSync:

- **MpRecipeRegPar**: If a process variable has been registered using this function block, then the new data is stored directly in the recipe when the save command is issued.
- **MpRecipeRegParSync**: When using this function block, the transfer of data from the PV to the recipe must be confirmed. When "Save" = TRUE, the status "SaveNotification = TRUE" can be found on all MpRecipeRegParSync function blocks. This indicates that the current PV data can be written to the recipe. This must first be acknowledged, however. The save procedure is confirmed via input "ConfirmSave" on MpRecipeRegParSync; only then is the data actually transferred from the PV to the recipe. This allows the user to manipulate the data or prevent potential data inconsistencies before the data is saved.

If the name of the file ("FileName") does not have an extension, .xml will be added automatically. Automatic appending of the extension can be disabled in the MpRecipeXml configuration under section *Auto file extension*.

As soon as the save is completed, output "CommandDone" is set to TRUE. If saving is delayed due to active MpRecipeRegParSync function blocks (no "ConfirmSave" confirmation occurs), then the "Save" command remains active, which is indicated by "CommandBusy = TRUE". PV data will be saved only after the process has been confirmed by all stations.

Automatic saving can be enabled in the MpRecipeXml configuration with "Save automatically". A save interval must then be specified. "Start immediately" defines whether the recipe should be saved immediately when MpRecipeXml is enabled.

Version control

Version control can be implemented using inputs "Header" and "LoadType". Since it is possible to only load the header of a recipe, it is easy to determine which version of the recipe is on hand. The user can then either load another recipe or load the complete recipe.

Recipe categories

The different PVs can be split up into several recipes via input "Category". If the same category is specified when registering a PV (MpRecipeRegPar or MpRecipeRegParSync) and managing a recipe (MpRecipeXml), then the PV value is stored in the recipe along with the same category. More information is available here: Categories.

A separateMpRecipeRegPar or MpRecipeRegParSync function block is necessary for each PV.

XML schema for recipes

The XML schema definition for all recipes with file format .xml is attached here: MpRecipeXml

2.5.1.5.2 Timing diagrams

The timing diagrams for managing recipes encompass the saving, loading and synchronization of parameters.

Saving a recipe

Setting "Save" to TRUE on function block MpRecipeXml initiates the save procedure. Because this example uses several MpRecipeRegParSync function blocks, the recipe will not be saved until these function blocks provide confirmation. As soon as the save command arrives, the "SaveNotification" status is set. This indicates that the current parameters should be saved to the recipe. The parameters are transferred if this status is confirmed by the "ConfirmSave" command.
Loading a recipe

The loading of a recipe is shown here first. The "UpdateNotification" command (which is triggered by the application 1) then indicates that new values have been written to the PVs. These outputs are only active for one cycle and then reset automatically. In this case, the signal of the *MpRecipeRegPar_1* function block is delayed since it is running in a task with a slower cycle time\(^2\).
"UpdateNotification" functionality

The "UpdateNotification" command is triggered on MpRecipeXml and generates a "UpdateNotification" status signal on all connected MpRecipeRegPar function blocks that indicates that new values have been written to the PVs. This output is only active for one cycle and then reset automatically. In this example, the signal of function block MpRecipeRegPar_1 is delayed since it is running in a task with a slower cycle time.

2.5.2 Data types and enumerators

2.5.2.1 Data types

2.5.2.1.1 MpRecipeCsvHeaderType

The header information for recipes is stored in this structure. The header structure is used with the MpRecipeCsv component during creation, saving and reading.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>STRING[100]</td>
<td>Name of the recipe</td>
</tr>
<tr>
<td>Description</td>
<td>STRING[255]</td>
<td>Comment that describes the recipe</td>
</tr>
<tr>
<td>Version</td>
<td>STRING[20]</td>
<td>Version of the recipe</td>
</tr>
<tr>
<td>DateTime</td>
<td>DATE_AND_TIME</td>
<td>Date and time the recipe was created</td>
</tr>
</tbody>
</table>

2.5.2.1.2 MpRecipeCsvInfoType

This data type provides additional information for the MpRecipeCsv component.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileSize</td>
<td>UDINT</td>
<td>Size of the recipe stored on the respective data storage medium</td>
</tr>
<tr>
<td>PendingSync</td>
<td>UDINT</td>
<td>Number of pending save notifications (&quot;SaveNotification&quot;) that have not yet been confirmed</td>
</tr>
<tr>
<td>LastLoadedRecipe</td>
<td>STRING[255]</td>
<td>The last recipe loaded</td>
</tr>
<tr>
<td>Diag</td>
<td>MpRecipeDiagType</td>
<td>Diagnostic structure for the status ID</td>
</tr>
</tbody>
</table>

2.5.2.1.3 MpRecipeDiagType

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>MpRecipeStatusIDType</td>
<td>StatusID diagnostic structure</td>
</tr>
</tbody>
</table>

2.5.2.1.4 MpRecipeInfoType

This data type provides additional information for the MpRecipe component.

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

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>MpRecipeErrorCode</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>

### 2.5.2.1.6 MpRecipeUIConnectType

This data type can be used to create a connection between the MpRecipeUI component and the Visual Components 4 HMI application. It includes a list of possible recipes as well as the ability to create new ones.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>MpRecipeUIStatusEnum</td>
<td>Current operation</td>
</tr>
<tr>
<td>Recipe</td>
<td>MpRecipeUIRecipeType</td>
<td>Access to existing recipes</td>
</tr>
<tr>
<td>New</td>
<td>MpRecipeUINewType</td>
<td>To create new recipes</td>
</tr>
<tr>
<td>Message box</td>
<td>MpRecipeUIMessageBoxType</td>
<td>Controls dialog boxes</td>
</tr>
<tr>
<td>DefaultLayerStatus</td>
<td>UINT</td>
<td>Status data point for the default layer of the visualization page where the recipe management system is being displayed</td>
</tr>
<tr>
<td>VC4 connection: StatusDatapoint from Layer</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 2.5.2.1.7 MpRecipeUIHeaderType

The header information for recipes is displayed in this structure. The header structure is loaded automatically when a recipe is selected in the HMI application.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>STRING[100]</td>
<td>Name of the recipe</td>
</tr>
<tr>
<td>Description</td>
<td>STRING[255]</td>
<td>Comment that describes the recipe</td>
</tr>
<tr>
<td>Version</td>
<td>STRING[20]</td>
<td>Version of the recipe</td>
</tr>
<tr>
<td>DateTime</td>
<td>DATE_AND_TIME</td>
<td>Date and time the recipe was created</td>
</tr>
</tbody>
</table>

### 2.5.2.1.8 MpRecipeUIMessageBoxType

Data type used to control the display of a dialog box.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LayerStatus</td>
<td>UINT</td>
<td>Visibility of the dialog box (status data point for the dialog box layer)</td>
</tr>
<tr>
<td>VC4 connection: StatusDatapoint from Layer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>MpRecipeUIMessageType</td>
<td>Type of dialog box</td>
</tr>
<tr>
<td>Confirm</td>
<td>BOOL</td>
<td>Confirm the operation</td>
</tr>
<tr>
<td>VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cancel</td>
<td>BOOL</td>
<td>Cancels the operation</td>
</tr>
<tr>
<td>VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 2.5.2.1.9 MpRecipeUINewType

All necessary items for creating a new recipe can be found here. The recipe list is automatically regenerated after a recipe is created. If no extension is specified for a recipe, .xml will be added automatically.
### Parameter | Data type | Description
--- | --- | ---
FileName | STRING[255] | Name of the new recipe. VC4 connection: Datapoint from String.
Create | BOOL | Creates a new recipe. VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0.

#### 2.5.2.1.10 MpRecipeUIRecipeListType

List of possible recipes and associated navigation.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Names</td>
<td>ARRAY[0..19] of STRING[255]</td>
<td>List of all available recipes (filenames). VC4 connection: TextDatapoint from ListBox.</td>
</tr>
<tr>
<td>SelectedIndex</td>
<td>UINT</td>
<td>Index of the entry currently selected in the list. VC4 connection: IndexDatapoint from ListBox.</td>
</tr>
<tr>
<td>MaxSelection</td>
<td>UINT</td>
<td>Index of the last entry in the list. VC4 connection: MaxDatapoint from ListBox.</td>
</tr>
<tr>
<td>PageUp</td>
<td>BOOL</td>
<td>Jumps to the start of the current page and then scrolls up one page at a time. The size of the page is defined using parameter &quot;RecipeListSize&quot; in structure MpRecipeUISetupType. VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0.</td>
</tr>
<tr>
<td>PageDown</td>
<td>BOOL</td>
<td>Jumps to the end of the current page and then scrolls down one page at a time. The size of the page is defined using parameter &quot;RecipeListSize&quot; in structure MpRecipeUISetupType. VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>StepUp</td>
<td>BOOL</td>
<td>Selects the previous entry in the list. VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0.</td>
</tr>
<tr>
<td>StepDown</td>
<td>BOOL</td>
<td>Selects the next entry in the list. VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0.</td>
</tr>
<tr>
<td>RangeStart</td>
<td>REAL</td>
<td>Shows a bar indicating which part of the list is currently being displayed. Used for the starting value of &quot;Range&quot; for a scaled element in the HMI application. This scaled element should correspond to the (possible) size of the list. &quot;Range&quot; should correspond to the number of entries that are displayed on one page. VC4 connection: StartDatapoint from Scale.</td>
</tr>
<tr>
<td>RangeEnd</td>
<td>REAL</td>
<td>Shows a bar indicating which part of the list is currently being displayed. Used for the ending value of &quot;Range&quot; for a scaled element in the HMI application. This scaled element should correspond to the (possible) size of the list. &quot;Range&quot; should correspond to the number of entries that are displayed on one page. VC4 connection: EndDatapoint from Scale.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sizes</th>
<th>ARRAY[0..19] of UDINT</th>
<th>Size of the files</th>
</tr>
</thead>
<tbody>
<tr>
<td>LastModified</td>
<td>ARRAY[0..19] of STRING[50]</td>
<td>The last time the files were modified/edited</td>
</tr>
</tbody>
</table>
All data necessary to manage recipes. The recipe list is automatically regenerated after a recipe is created.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>List</td>
<td>MpRecipeUIRecipeListType</td>
<td>List of all available recipes (filenames), as well as the navigation of the list.</td>
</tr>
<tr>
<td>Load</td>
<td>BOOL</td>
<td>Loads the selected recipe. VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0.</td>
</tr>
<tr>
<td>Save</td>
<td>BOOL</td>
<td>Saves the selected recipe. VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0.</td>
</tr>
<tr>
<td>Filter</td>
<td>STRING[255]</td>
<td>Filter used to sort the recipes. The filter string refers to the recipe name and indicates the recipes that have the string in its name. It is possible to use the \* wildcard in the search function. For more information about the search function, see section &quot;Search function&quot; in the description of MpRecipeUI.Description. VC4 connection: Datapoint from String.</td>
</tr>
<tr>
<td>SortOrder</td>
<td>MpRecipeUISortOrderEnum</td>
<td>Sorts the recipe list. VC4 connection: Datapoint from Button / Type: ToggleDatapoint / SetValue: 1 / ResetValue: 0.</td>
</tr>
<tr>
<td>Refresh</td>
<td>BOOL</td>
<td>Updates the recipe list. VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0.</td>
</tr>
<tr>
<td>UpdateNotification</td>
<td>BOOL</td>
<td>Request to all mapp components to enter the current values of the recipe parameters. VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1.</td>
</tr>
<tr>
<td>Delete</td>
<td>BOOL</td>
<td>Deletes the selected recipe. VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0.</td>
</tr>
<tr>
<td>Rename</td>
<td>BOOL</td>
<td>Renames the selected recipe. VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0.</td>
</tr>
<tr>
<td>NewFileName</td>
<td>STRING[255]</td>
<td>New name of the recipe. VC4 connection: Datapoint from String.</td>
</tr>
<tr>
<td>Header</td>
<td>MpRecipeUIHeaderType</td>
<td>Header data of the currently selected recipe.</td>
</tr>
</tbody>
</table>
### 2.5.2.1.12 MpRecipeUISetupConfirmType

This data type can be used to define when confirmation windows should be displayed.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecipeLoad</td>
<td>BOOL</td>
<td></td>
<td>Enables a confirmation window when loading a recipe</td>
</tr>
<tr>
<td>RecipeSave</td>
<td>BOOL</td>
<td></td>
<td>Enables a confirmation window when overwriting a recipe:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• This dialog box is displayed if saving a recipe that already exists.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• This dialog box is displayed if attempting to generate a new recipe that actually already exists. In order for the dialog box to be displayed during the above cases, &quot;RecipeCreate&quot; must equal TRUE.</td>
</tr>
<tr>
<td>RecipeCreate</td>
<td>BOOL</td>
<td></td>
<td>Enables a confirmation window when creating a recipe</td>
</tr>
<tr>
<td>RecipeDelete</td>
<td>BOOL</td>
<td></td>
<td>Enables a confirmation window when deleting a recipe</td>
</tr>
<tr>
<td>RecipeRename</td>
<td>BOOL</td>
<td></td>
<td>Enables a confirmation window when renaming a recipe</td>
</tr>
</tbody>
</table>

### 2.5.2.1.13 MpRecipeUISetupType

Additional configuration options for the HMI application.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecipeListSize</td>
<td>UINT</td>
<td>20</td>
<td>Number of recipes to be displayed on one page of the list in the HMI application. Up to 20 recipes can be displayed.</td>
</tr>
<tr>
<td>RecipeListScrollWindow</td>
<td>USINT</td>
<td>1</td>
<td>Determines how many entries from the list are initially displayed when scrolling up and down</td>
</tr>
<tr>
<td>Confirmation</td>
<td>MpRecipeUISetupConfirmType</td>
<td></td>
<td>Displays the confirmation window</td>
</tr>
<tr>
<td>AutoLoadHeader</td>
<td>BOOL</td>
<td></td>
<td>Defines whether the header of a recipe should be loaded automatically when selected in the recipe list (header data is found in structure MpRecipeUIRecipeType)</td>
</tr>
</tbody>
</table>
RecipeListSize

RecipeListSize specifies how many files are displayed on one side of a listbox in VC4. The listbox in the HMI application displays a varied number of elements depending on how big it is. The number of elements shown in the listbox can be defined by a simulation value (SimulationValue) in the properties of the listbox (see image above: "Any File"). The number of these elements must match the value of RecipeListSize.

RecipeListScrollWindow

RecipeListScrollWindow specifies how many entries should be "previewed" when scrolling. In this example, RecipeListScrollWindow = 3 is defined. If the current selection is fourth from the bottom, then pressing the Step-Down button will select the next entry (Any File 7) and preview the next three entries (Any File 8, Any File 9 and Any File A).

2.5.2.1.14 MpRecipeXmlHeaderType

The header information for recipes is stored in this structure. The header structure is used with the MpRecipeXml component during creation, saving and reading.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>STRING[100]</td>
<td>Name of the recipe</td>
</tr>
<tr>
<td>Description</td>
<td>STRING[255]</td>
<td>Comment that describes the recipe</td>
</tr>
<tr>
<td>Version</td>
<td>STRING[20]</td>
<td>Version of the recipe</td>
</tr>
<tr>
<td>DateTime</td>
<td>DATE_AND_TIME</td>
<td>Date and time the recipe was created</td>
</tr>
</tbody>
</table>

2.5.2.1.15 MpRecipeXmlInfoType

This data type provides additional information for the MpRecipeXml component.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileSize</td>
<td>UDINT</td>
<td>Size of the recipe stored on the respective data storage medium</td>
</tr>
<tr>
<td>PendingSync</td>
<td>UDINT</td>
<td>Number of pending save notifications (&quot;SaveNotification&quot;) that have not yet been confirmed</td>
</tr>
<tr>
<td>LastLoadedRecipe</td>
<td>STRING[255]</td>
<td>The last recipe loaded</td>
</tr>
<tr>
<td>Diag</td>
<td>MpRecipeDiagType</td>
<td>Diagnostic structure for the status ID</td>
</tr>
</tbody>
</table>

2.5.2.2 Enumerators

2.5.2.2.1 MpRecipeCsvLoadEnum

This enumerated data type determines how the recipe is loaded, i.e. only the header or the entire recipe.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpRECIPE_CSV_LOAD_ALL</td>
<td>Loads the entire content of the recipe</td>
</tr>
<tr>
<td>mpRECIPE_CSV_LOAD_HEADER</td>
<td>Loads only the recipe's header</td>
</tr>
</tbody>
</table>

2.5.2.2.2 MpRecipeUIMessageEnum

This enumerated data type provides information about the type of dialog box. Dialog boxes are only displayed if they have been enabled in the MpRecipeUISetupConfirmType structure.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpRECIPE_UI_MSG_NONE</td>
<td>No dialog box</td>
</tr>
<tr>
<td>mpRECIPE_UI_MSG_CONFIRM_LOAD</td>
<td>Dialog box for confirming the loading procedure</td>
</tr>
<tr>
<td>mpRECIPE_UI_MSG_CONFIRM_SAVE</td>
<td>Dialog box for confirming the saving procedure</td>
</tr>
<tr>
<td>mpRECIPE_UI_MSG_CONFIRM_CREATE</td>
<td>Dialog box for confirming the creation of a new recipe</td>
</tr>
<tr>
<td>mpRECIPE_UI_MSG_CONFIRM_DELETE</td>
<td>Dialog box for confirming the deletion of a new recipe</td>
</tr>
<tr>
<td>mpRECIPE_UI_MSG_CONFIRM_RENAME</td>
<td>Dialog box for confirming the renaming of a recipe</td>
</tr>
</tbody>
</table>
2.5.2.2.3 MpRecipeUISortOrderEnum

This enumerated data type determines how sorting takes place. It is possible to sort in either ascending or descending alphabetical order.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpRECIPE_UI_SORT_ASCENDING</td>
<td>Sorts in ascending (alphabetic) order</td>
</tr>
<tr>
<td>mpRECIPE_UI_SORT_DESCENDING</td>
<td>Sorts in descending (alphabetic) order</td>
</tr>
<tr>
<td>mpRECIPE_UI_SORT_DATE_ASCENDING</td>
<td>Sorts in ascending order by date last modified</td>
</tr>
<tr>
<td>mpRECIPE_UI_SORT_DATE_DESCENDING</td>
<td>Sorts in descending order by date last modified</td>
</tr>
</tbody>
</table>

2.5.2.2.4 MpRecipeUIStatusEnum

This enumerated data type provides information about the current activity of a component.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpRECIPE_UI_STATUS_IDLE</td>
<td>No process is currently active.</td>
</tr>
<tr>
<td>mpRECIPE_UI_STATUS_LOAD</td>
<td>A recipe is being loaded.</td>
</tr>
<tr>
<td>mpRECIPE_UI_STATUS_SAVE</td>
<td>A recipe is being saved.</td>
</tr>
<tr>
<td>mpRECIPE_UI_STATUS_CREATE</td>
<td>A recipe is being created.</td>
</tr>
<tr>
<td>mpRECIPE_UI_STATUS_REFRESH</td>
<td>The recipe list is being refreshed.</td>
</tr>
<tr>
<td>mpRECIPE_UI_STATUS_NOTIFY</td>
<td>mapp components are being informed that new parameters are to be used.</td>
</tr>
<tr>
<td>mpRECIPE_UI_STATUS_DELETE</td>
<td>A recipe is being deleted.</td>
</tr>
<tr>
<td>mpRECIPE_UI_STATUS RENAME</td>
<td>A recipe is being renamed.</td>
</tr>
<tr>
<td>mpRECIPE_UI_STATUS_ERROR</td>
<td>The last operation generated an error.</td>
</tr>
</tbody>
</table>

2.5.2.2.5 MpRecipeXmlLoadEnum

This enumerated data type determines how the recipe is loaded, i.e. only the header or the entire recipe.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpRECIPE_XML_LOAD_ALL</td>
<td>Loads the entire content of the recipe</td>
</tr>
<tr>
<td>mpRECIPE_XML_LOAD_HEADER</td>
<td>Loads only the recipe’s header</td>
</tr>
</tbody>
</table>

2.5.3 Status numbers

2.5.3.1 1083342870: MpRecipeXml not yet active

Description:
This component can only become active if MpRecipeXml / MpRecipeCsv is active.

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

Cause/Solution:
- Enable MpRecipeXml.
- Enable MpRecipeCsv.

Constant:
mpRECIPE_INF_WAIT_RECIPE_FB

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

2.5.3.2 -2138044264: Synchronous saving not supported

Description:
Synchronized saving is not supported by the current parameters.

Constant:
mpRECIPE_WRN_SYNC_SAVE_NOT_REQUIRED
2.5.3.3 -2138044263: Synchronized saving executed

Description:
Synchronized saving was executed with a combination of synchronized and unsynchronized parameters.

Constant:
mpRECIPE_WRN_SYNC_SAVE_MIXED

2.5.3.4 -2138044262: Unexpected number of columns in CSV file

Description:
A row was found in the CSV file that did not have the expected number of columns.

Cause/Solution:
• Check the CSV file.

Constant:
mpRECIPE_WRN_CSV_INVALID_COLUMN_COUNT

2.5.3.5 -2138044261: Cannot parse contents to 32-bit integer

Description:
Content {2:Content} in row {3:Line} and column {4:Column} of file {5:FileName} cannot be parsed to a 32-bit integer.

Additional information
• {2:Content}: Content that should be parsed.
• {3:Line}: Row containing the content.
• {4:Column}: Column containing the content.
• {5:FileName}: File containing the content.

Cause
• The file was edited externally.
• Any string not represented by a number cannot be parsed.

Constant:
mpRECIPE_WRN_CSV_INVALID_INT32

2.5.3.6 -2138044260: Cannot parse contents to 32-bit UINT

Description:
Content {2:Content} in row {3:Line} and column {4:Column} of file {5:FileName} cannot be parsed to a 32-bit UINT.

Additional information
• {2:Content}: Content that should be parsed.
• {3:Line}: Row containing the content.
• {4:Column}: Column containing the content.
• {5:FileName}: File containing the content.

Cause
• The file was edited externally.
• Any string not represented by a number cannot be parsed.

Constant:
mpRECIPE_WRN_CSV_INVALID_UINT32
2.5.3.7 -2138044259: Cannot parse contents to Boolean

Description:
Content (2:Content) in row (3:Line) and column (4:Column) of file (5:FileName) cannot be parsed to Boolean.

Additional information
- {2:Content}: Content that should be parsed.
- {3:Line}: Row containing the content.
- {4:Column}: Column containing the content.
- {5:FileName}: File containing the content.

Cause
- The file was edited externally.
- Any string not represented by a Boolean value cannot be parsed.

Constant:
mpRECIPE_WRN_CSV_INVALID_BOOLEAN

2.5.3.8 -2138044258: Could not parse contents to 64-bit Float

Description:
Content (2:Content) in row (3:Line) and column (4:Column) of file (5:FileName) could not be parsed to a 64-bit float.

Additional information
- {2:Content}: Content that should be parsed.
- {3:Line}: Row containing the content.
- {4:Column}: Column containing the content.
- {5:FileName}: File containing the content.

Cause
- The file was edited externally.
- Any string not represented by a number cannot be parsed.

Constant:
mpRECIPE_WRN_CSV_INVALID_FLOAT64

2.5.3.9 -2138044255: Could not locate PV

Description:
Process variable (2:PVName) could not be located. Error source: {1:ErrorNumber}

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

Cause/Solution:
- Check the process variable.

Constant:
mpRECIPE_WRN_PV_NOT_FOUND
2.5.3.10 -2138044254: Data type not supported

Description:
Data type \(2:\text{DataType}\) of variable \(3:\text{PVName}\) is not a known IEC data type and therefore not supported.

Additional information
- \(2:\text{DataType}\): Specified data type.
- \(3:\text{PVName}\): Name of the process variable.

Cause/Solution:
- Change the data type.
- For more information about unsupported data types, see \[here\].

Constant:
mpRECIPE_WRN_PV_TYPE_UNKNOWN

2.5.3.11 -2138044253: PV data type different on target system

Description:
The data type of variable \(2:\text{PVName}\) is not the same as the data type used on the target system.

Additional information
- \(2:\text{PVName}\): Name of the affected process variable.

Cause/Solution:
- Change the data type to the original data type.

Constant:
mpRECIPE_WRN_PV_TYPE_MISMATCH

2.5.3.12 -2138044252: Value of variable outside range of values

Description:
The value of variable \(2:\text{PVName}\) is outside the data type's range of values.

Additional information
- \(2:\text{PVName}\): Name of the affected process variable.

Cause/Solution:
- Change the range of values.

Constant:
mpRECIPE_WRN_PV_VALUE_OUT_OF_RANGE

2.5.3.13 -2138044251: PV string value too large

Description:
The string value of variable \(2:\text{PVName}\) does not fit in the memory reserved for it on the target system.

Additional information
- \(2:\text{PVName}\): Name of the process variable.
Cause/Solution:

- Allocate a larger memory block.

Constant:

```
mpRECIPE_WRN_PV_STRING_OUT_OF_RANGE
```

**2.5.3.14 -2138044250: Could not make PV part of recipe**

Description:
The attempt to make process variable {2:PVName} part of the recipe failed. Error source: {1:ErrorNumber}

Additional information

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

Cause/Solution:

- Check that the process variable has the correct data type.
- Check that the process variable is specified correctly.

Constant:

```
mpRECIPE_WRN_PV_TRAVERSAL
```

**2.5.3.15 -2138044248: Illegal data type**

Description:
The data type is not supported as part of a recipe.

Cause/Solution:

- Change the data type.
- For more information about unsupported data types, see here.

Constant:

```
mpRECIPE_WRN_PV_TYPE_NOT_SUPPORTED
```

**2.5.3.16 -2138044247: Recipe format error**

Description:
Recipe format error in {2:Context}.

Additional information

- {2:Context}: Text containing format errors

Constant:

```
mpRECIPE_WRN_RECIPE_FORMAT
```

**2.5.3.17 -2138044246: Invalid recipe contents**

Description:
Invalid recipe contents in {2:Context}.

Additional information

- {2:Context}: Text containing invalid recipe contents
Cause/Solution:

- If the file was edited externally, check whether the contents of the file have been damaged. This can happen by specifying an incorrect XML tag, for example.

Constant:

```plaintext
mpRECIPE_WRN_RECIPE_FORMAT_CONTENT
```

**2.5.3.18 -2138044244: Expected attribute missing in element**

**Description:**
Expected attribute `{2:Attribute}` missing in element `{3:Element}`.

**Additional information**
- `{2:Attribute}`: Expected attribute
- `{3:Element}`: Element

Constant:

```plaintext
mpRECIPE_WRN_DATA_MODEL_MISSING_ATTRIBUTE
```

**2.5.3.19 -2138044242: Expected element missing**

**Description:**
Expected element `{2:Element}` missing.

**Additional information**
- `{2:Element}`: Element

Constant:

```plaintext
mpRECIPE_WRN_DATA_MODEL_MISSING_ELEMENT
```

**2.5.3.20 -2138044241: Unexpected element found**

**Description:**
Unexpected element `{2:Element}` found.

**Additional information**
- `{2:Element}`: Element

Constant:

```plaintext
mpRECIPE_WRN_DATA_MODEL_UNEXPECTED_ELEMENT
```

**2.5.3.21 -2138044240: Unknown parameter content type**

**Description:**
Parameter content type `{2:Type}` was unrecognized. Could not parse content.

**Additional information**
- `{2:Type}`: Type of parameter.
Cause

- This error occurs if the file was edited externally.
- Only HeaderData and PvParameter are permitted to be used as types in the file:

```
<Element Name="MpRecipe.Header" Type="HeaderData">
  <Property ID="Name" Type="STRING" Value="TestHeader" />
  <Property ID="Description" Type="STRING" Value="TestDescription" />
  <Property ID="Version" Type="STRING" Value="0.0.99" />
  <Property ID="Created" Type="UDINT" Value="0" />
</Element>
<Element Name="MpRecipe.Xn:UserNestedStruct" Type="PvParameter">
  <Group ID="MpRecipe.Xn:UserNestedStruct">
    <Property ID="DeviceName" Data type="USINT" Value="0" />
    <Group ID="open">
      <Property ID="CommandName" Data type="STRING" Value="" />
    </Group>
  </Group>
</Element>
```

Constant:

mpRECIPE_WRN_DATA_MODEL_UNKNOWN_CONTENT

2.5.3.22 -2138044233: Invalid data type of header property

Description:
The data type of the header property {2:Property} is invalid. Expected {3:Expected}, actual {4:Actual}.

Additional information

- {2:Property}: Header property.
- {3:Expected}: Expected data type.
- {4:Actual}: Actual data type.

Cause/Solution:

- Check the specified header.
- Change the header.

Constant:

mpRECIPE_WRN_HEADER_TYPE_MISMATCH

2.5.3.23 -2138044232: Invalid header content

Description:
Invalid header content in {2:Context} detected when loading file {3:File}. Error source: {1:ErrorNumber}

Additional information

- {1:ErrorNumber}: Error source
- {2:Context}: Content containing invalid information
- {3:File}: Affected file

Constant:

mpRECIPE_WRN_HEADER_FORMAT

2.5.3.24 -2137882618: Warnings while saving recipe

Description:
Warnings were triggered while saving a recipe. See the Logger for additional information.
Reaction:
The function block indicates an active warning on output "StatusID". The save procedure resumes and as many registered PVs as possible are saved to the recipe.

Cause/Solution:
- Check the parameters being saved.
- Arrays that do not start with index 0 cannot be registered.
- Multidimensional arrays cannot be registered.
- Arrays with more than 65535 elements cannot be registered.
- Derived data types
- Variables of data type TIME_OF_DAY cannot be registered.
- Variables of data type BYTE cannot be registered.
- Variables of data type WORD cannot be registered.
- Variables of data type DWORD cannot be registered.
- Variables of data type LWORD cannot be registered.
- Variables of data type WSTRING cannot be registered.
- Variables of data type LINT cannot be registered.

Constant:
mpRECIPE_WRN_SAVE_WITH_WARN

These function blocks / functions can report this error:
- MpRecipeXml
- MpRecipeCsv
- MpRecipeUI

2.5.3.25 -2137882617: Warnings while loading recipe

Description:
Warnings were triggered while loading a recipe. For additional details, see the Logger (PV name, for example).

Reaction:
The function block indicates an active warning on output "StatusID". The load procedure resumes and as many registered PVs are loaded from the recipe to the registered PVs as possible.

Cause/Solution:
- Check the parameters in the recipe (data type, damaged XML structure in recipe, etc.).

Constant:
mpRECIPE_WRN_LOAD_WITH_WARN

These function blocks / functions can report this error:
- MpRecipeXml
- MpRecipeCsv
- MpRecipeUI

2.5.3.26 -2137882611: Invalid value for "UIConnect"

Description:
An invalid value (NULL pointer) is connected on input "UIConnect" of function block MpRecipeUI.

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

- Verify the structure on input "UIConnect" of function block MpRecipeUI (NULL appended).

Constant:

mpRECIPE_WRN_MISSING_UICONNECT

These function blocks / functions can report this error:

- MpRecipeUI

2.5.3.27 -2137882606: No PV registered

Description:

No process variable was registered.

Reaction:

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

Cause/Solution:

- Neither function block MpRecipeRegPar or MpRecipeRegParSync was found.

Constant:

mpRECIPE_WRN_NO_PV_REGISTERED

These function blocks / functions can report this error:

- MpRecipeXml
- MpRecipeCsv

2.5.3.28 -2137882603: Empty recipe created

Description:

An empty recipe was saved.

Reaction:

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

Cause/Solution:

- Parameters registered, but incorrect category specified

Constant:

mpRECIPE_WRN_EMPTY_RECIPE

These function blocks / functions can report this error:

- MpRecipeXml
- MpRecipeCsv
- MpRecipeUI

2.5.3.29 -2137882600: No PV present

Description:

An attempt was made to load a recipe that did not contain any of the registered process variables.

Reaction:

The function block indicates an active warning on output "StatusID".
Cause/Solution:

- The recipe does not contain any valid parameters.
- If function block `MpRecipeCsv` is used and option "Encrypt" = `TRUE` in the `MpRecipeCsv` configuration, then it must be ensured that a recipe with the same passphrase is loaded that was used to save/create it. If the passphrase was changed in the meantime (passphrase = 3), the recipe created with passphrase = 1 cannot be opened.

**Constant**

`mpRECIPE_NO_PV_FOUND`

These function blocks / functions can report this error:

- `MpRecipeCsv`
- `MpRecipeXml`

2.5.3.30 -1064302433: A file with the same name already exists.

Description:

A file with the same name already exists.

Cause/Solution:

- Specify another filename.

**Constant**

`mpRECIPE_ERR_FILE_EXISTS`

2.5.3.31 -1064302432: Could not locate information about PV

Description:

Information about process variable {2:PVName} could not be located. Error source: {1:ErrorNumber}

Additional information

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

Cause/Solution:

- Check the process variable.
- The process variable was specified incorrectly.

**Constant**

`mpRECIPE_ERR_PV_NOT_FOUND`

2.5.3.32 -1064302421: Required attribute missing

Description:

Required attribute {2:Attribute} missing in element {3:Element}.

Additional information

- {2:Attribute}: Required attribute
- {3:Element}: Element in which the attribute is missing
Cause

- This error occurs if the file was edited externally.
- The file must include attributes Name and Type.
  Example: <Element Name="MpRecipeXml:UserNestedStruct" Type="PvParameter">

Constant:

mpRECIPE_ERR_DATA_MODEL_MISSING_ATTRIBUTE

2.5.3.33 -1064302419: Required attribute missing

Description:
Required element {2:Element} missing.

Additional information

- {2:Element}: Element

Cause/Solution:

- Check if the process variable was registered.
- Check whether input parameter "Category" was used and whether the correct category was specified.

Constant:

mpRECIPE_ERR_DATA_MODEL_MISSING_ELEMENT

2.5.3.34 -1064302415: Error creating data model

Description:
Error creating data model for loading/saving a recipe. Error source: {1:ErrorNumber}

Additional information

- {1:ErrorNumber}: Reason for the error.

Constant:

mpRECIPE_ERR_DATA_MODEL_CREATE_NEW

2.5.3.35 -1064302414: Error closing data model

Description:
Error closing data model after loading/saving recipe. Error source: {1:ErrorNumber}

Additional information

- {1:ErrorNumber}: Reason for the error.

Constant:

mpRECIPE_ERR_DATA_MODEL_CLOSE

2.5.3.36 -1064302413: Error initializing data model

Description:
Error initializing data object being used to create a new recipe. Error source: {1:ErrorNumber}

Additional information

- {1:ErrorNumber}: Reason for the error.
Constant:
mpRECIPE_ERR_DATA_MODEL_INIT

2.5.3.37 -1064302412: Parameter with undefined name found

Description:
Parameter with undefined name found.

Cause/Solution:
• Assign a name.

Constant:
mpRECIPE_ERR_PARAMETER_NAME

2.5.3.38 -1064302411: MpFile unable to list files

Description:
MpFile could not list files for the selected drive. Error source: {1:ErrorNumber}

Additional information
• {1:ErrorNumber}: Reason for the error.

Cause/Solution:
• Add library MpFile.
• Check the specified data storage device.

Constant:
mpRECIPE_ERR_MPFILE_LISTING_FILES

2.5.3.39 -1064302410: Could not add service

Description:
Adding a service for the recipe system failed. Error source: {1:ErrorNumber}

Additional information
• {1:ErrorNumber}: Error source

Constant:
mpRECIPE_ERR_CORE_SERVICE

2.5.3.40 -1064302407: Recipe or signature modified

Description:
Verification of the signature failed. See the Logger for additional information.

Cause/Solution:
• The recipe file or signature file was modified externally.

Constant
mpRECIPE_ERR_SIGNATURE_INVALID
**2.5.3.41 -1064302406: Signature file missing**

**Description:**
The signature file required for the recipe was not found. See the Logger for additional information.

**Cause/Solution:**
- Check whether the signature file exists.

**Constant**
mpRECIPE_ERR_SIGNATURE_MISSING

**2.5.3.42 -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:**
- MpRecipeXml
- MpRecipeCsv
- MpRecipeRegPar
- MpRecipeRegParSync
- MpRecipeUI

**Constant:**
mpRECIPE_ERR_ACTIVATION

**2.5.3.43 -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:**
- MpRecipeXml
- MpRecipeCsv
- MpRecipeRegPar
- MpRecipeRegParSync
- MpRecipeUI
Constant:
mpRECIPE_ERR_MPLINK_NULL

2.5.3.44 -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:
• MpRecipeXml
• MpRecipeCsv
• MpRecipeRegPar
• MpRecipeRegParSync
• MpRecipeUI

Constant:
mpRECIPE_ERR_MPLINK_INVALID

2.5.3.45 -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:
• MpRecipeXml
• MpRecipeCsv
• MpRecipeRegPar
• MpRecipeRegParSync
• MpRecipeUI

Constant:
mpRECIPE_ERR_MPLINK_CHANGED

2.5.3.46 -1064239099: Invalid MpLink contents

Description:
The value of variable "MpLink" 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:
- MpRecipeXml
- MpRecipeCsv
- MpRecipeRegPar
- MpRecipeRegParSync
- MpRecipeUI

Constant:
mpRECIPE_ERR_MPLINK_CORRUPT

2.5.3.47 -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:
- MpRecipeXml
- MpRecipeCsv
- MpRecipeRegPar
- MpRecipeRegParSync
- MpRecipeUI

Constant:
mpRECIPE_ERR_MPLINK_IN_USE

2.5.3.48 -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.
- The function is not enabled in the configuration.

These function blocks / functions can report this error:

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

- MpRecipeXml
- MpRecipeCsv

Constant:

mpRECIPE_ERR_CONFIG_INVALID

2.5.3.49 -1064140799: Error saving recipe

Description:
An error occurred while saving a recipe. This points to a problem with the file system.

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 filename on input "FileName".
- Check the file device on input "DeviceName".
- The following characters are not permitted in the filename: \ / : * ? < > |
- Check whether the specified data storage device has sufficient memory available.
- Check whether an external program is accessing the file (e.g. Excel).

Constant:

mpRECIPE_ERR_SAVE_DATA

These function blocks / functions can report this error:

- MpRecipeXml
- MpRecipeCsv
- MpRecipeUI

2.5.3.50 -1064140798: Error loading recipe

Description:
An error occurred while loading a recipe. The parameters were not transferred.

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

Cause/Solution:

- Make sure that the XML structure of the recipe file is valid.
- Check whether the file device on input "DeviceName" can be accessed.

Constant:

mpRECIPE_ERR_LOAD_DATA
These function blocks / functions can report this error:

- MpRecipeXml
- MpRecipeCsv
- MpRecipeUI

2.5.3.51 -1064140797: Invalid file device

Description:
The specified file device on input "DeviceName" 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:
- Nothing is attached to input "DeviceName".
- The attached file device does not exist.
- A connection to the desired file device could not be established.

Constant:

mpRECIPE_ERR_INVALID_FILE_DEV

These function blocks / functions can report this error:

- MpRecipeXml
- MpRecipeCsv
- MpRecipeUI

2.5.3.52 -1064140796: Invalid filename

Description:
The filename on input "FileName" 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:
- Nothing is attached to input "FileName".
- The attached file does not exist.
- No access rights

Constant:

mpRECIPE_ERR_INVALID_FILE_NAME

These function blocks / functions can report this error:

- MpRecipeXml
- MpRecipeCsv
- MpRecipeUI

2.5.3.53 -1064140795: Component currently executing command

Description:
This occurs if the component is being used by multiple sources at the same time.
Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- This can occur if the function block receives commands from multiple sources at the same time (for example, a new read command in the MpRecipeUIConnectType structure from MpRecipeUI while a "Load" command from component MpRecipeXml is active).

Constant:
mpRECIPE_ERR_CMD_IN_PROGRESS

These function blocks / functions can report this error:
- MpRecipeXml
- MpRecipeCsv
- MpRecipeUI

2.5.3.54 -1064140792: Error converting recipe parameters

Description:
An error occurred while converting the parameters of a recipe. No recipe has been saved. 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. The save procedure is canceled. No data is saved to the recipe.

Cause/Solution:
- Check the parameters being saved.
- Arrays that do not start with index 0 cannot be registered.
- Multi-dimensional arrays cannot be registered.
- Arrays with more than 65535 elements cannot be registered.
- Derived data types
- Variables of data type TIME_OF_DAY cannot be registered.
- Variables of data type BYTE cannot be registered.
- Variables of data type WORD cannot be registered.
- Variables of data type DWORD cannot be registered.
- Variables of data type LWORD cannot be registered.
- Variables of data type WSTRING cannot be registered.
- Variables of data type LINT cannot be registered.

Constant:
mpRECIPE_ERR_SAVE_WITH_ERRORS

These function blocks / functions can report this error:
- MpRecipeXml
- MpRecipeCsv
- MpRecipeUI

2.5.3.55 -1064140791: Error converting recipe to parameters

Description:
An error occurred while converting the recipe to parameters. No recipe has been read. For addition details, see the Logger (PV name, for example).
Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time. The load procedure is canceled. No data is saved to the PVs.

Cause/Solution:
- Check the parameters in the recipe (data type, damaged XML structure in recipe, etc.).

Constant:
mpRECIPE_ERR_LOAD_WITH_ERRORS

These function blocks / functions can report this error:
- MpRecipeXml
- MpRecipeCsv
- MpRecipeUI

2.5.3.56 -1064140790: No connection to parent component

Description:
No MpRecipeXml / MpRecipeCsv component was found.

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 component causing the error has a link to the MpRecipeXml / MpRecipeCsv component.
- Check whether MpRecipeXml / MpRecipeCsv is working correctly.

Constant:
mpRECIPE_ERR_MISSING_RECIPE

These function blocks / functions can report this error:
- MpRecipeRegPar
- MpRecipeRegParSync
- MpRecipeUI

2.5.3.57 -1064140789: No connection to file manager

Description:
No MpFile component was found.

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 MpFile library has been transferred to the controller.

Constant:
mpRECIPE_ERR_MISSING_MPFILE

These function blocks / functions can report this error:
- MpRecipeUI
2.5.3.58 -1064140788: Invalid sort order selected

Description:
The selected sort order on input "SortOrder" 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:
- Check the value on input "SortOrder" in structure MpRecipeUIConnectType.

Constant:
mpRECIPE_ERR_INVALID_SORT_ORDER

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

2.5.3.59 -1064140786: Invalid PV

Description:
No PV could be found with the specified name.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time. The PV is re-evaluated on a rising edge on the enable input.

Cause/Solution:
- PV with this name not used by the system; check the name/address of the PV.

Constant:
mpRECIPE_ERR_INVALID_PV_NAME

These function blocks / functions can report this error:
- MpRecipeRegPar
- MpRecipeRegParSync

2.5.3.60 -1064140785: Invalid LoadType

Description:
The value on input "LoadType" 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:
- Check the value on input "LoadType"; possible options are listed in section MpRecipeXmlLoadEnum.

These function blocks / functions can report this error:
- MpRecipeXml
- MpRecipeCsv
2.5.3.61 -1064140784: Error creating file list

Description:
No new recipe list could be created.

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 connection to the data storage device ("DeviceName" of function block MpRecipeXml).

Constant:
mpRECIPE_ERR_LISTING_FILES

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

2.5.3.62 -1064140783: PV name is NULL.

Description:
An invalid PV name (NULL) was passed to the function.

Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time. The PV is re-evaluated on a rising edge on the enable input.

Cause/Solution:
• PV name is NULL.

Constant:
mpRECIPE_ERR_PV_NAME_NULL

These function blocks / functions can report this error:
• MpRecipeRegPar
• MpRecipeRegParSync

2.5.3.63 -1064140781: Saving active

Description:
A synchronized save procedure is already active.

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

Cause/Solution:
• Wait until the current save procedure is completed.
• One of the parameters that should be saved has not yet sent a "SaveNotification".

Constant:
mpRECIPE_ERR_SYNC_SAVE_ACTIVE
mapp Services

These function blocks / functions can report this error:

  • MpRecipeXml
  • MpRecipeCsv

2.5.3.64 -1064140780: Error during deletion

Description:
An error occurred while deleting the file.

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

Cause/Solution:

  • The file to be deleted could not be found.
  • Deleting the file is not permitted (write-protected).

Constant:

mpRECIPE_ERR_DELETING_FILE

These function blocks / functions can report this error:

  • MpRecipeUI

2.5.3.65 -1064140777: Cannot rename file

Description:
The selected file cannot be renamed.

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

Cause/Solution:

  • Specify the filename.

Constant:

mpRECIPE_ERR_RENAMING_FILE

These function blocks / functions can report this error:

  • MpRecipeUI

2.5.4 Alarms

2.5.4.1 mpRECIPE_ALM_SAVE_FAILED: Recipe save failed

Description:
An error occurred during saving. No recipe has been written.

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

Cause/Solution:

  • One possible reason is an incorrect file device on input "DeviceName".
  • Check that the file device (USB flash drive, memory card) is properly connected.
Behavior
Edge alarm

2.5.4.2 mpRECIPE_ALM_SAVE_FAILED_PART: Recipe saved partially failed

Description:
The recipe could only partially be saved successfully. Not all parameters were saved correctly. Additional details can be found in the logger.

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

Cause/Solution:
- Check the parameters being saved.

Behavior
Edge alarm

2.5.4.3 mpRECIPE_ALM_LOAD_FAILED: Recipe load failed

Description:
An error occurred during loading. No data was loaded.

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

Cause/Solution:
- One possible reason is an incorrect file device on input "DeviceName".
- Check that the file format is correct.

Behavior
Edge alarm

2.5.4.4 mpRECIPE_ALM_LOAD_FAILED_PART: Recipe load partially failed

Description:
The recipe could only partially be loaded successfully. Not all parameters were loaded correctly. See the Logger for additional information.

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

Cause/Solution:
- Check the parameters in the recipe (data type, damaged XML structure in recipe, etc.).
mapp Backup can store backup files of the target system on a data storage device such as a USB flash drive or install an existing backup. Updates can also be installed using mapp Backup.

3.1 Concept

User stories

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OEM</td>
</tr>
<tr>
<td>• As an OEM, I would like to automatically create a backup of my controller every day and store it on a USB flash drive.</td>
</tr>
<tr>
<td>• As a service technician, I would like to update the machine quickly and easily to the latest software version.</td>
</tr>
<tr>
<td>• As a service technician, if a controller is damaged, I would like to replace the controller and load the last saved backup to the controller.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Service technician</th>
</tr>
</thead>
<tbody>
<tr>
<td>• As a service technician, I would like to update the machine quickly and easily to the latest software version.</td>
</tr>
<tr>
<td>• As a service technician, if a controller is damaged, I would like to replace the controller and load the last saved backup to the controller.</td>
</tr>
</tbody>
</table>

3.1.1 Structure of a backup

A backup is a folder containing the files listed below. The folder name also defines the name of the backup. If a backup is installed or created, exactly this name must be used.

A backup consists of the following files:

- **Main folder:** In this example, the main folder is named "myBackupPrefix_2017_12_08_06_30". If a backup is created, a main folder with all files and subfolders is saved. The name of the folder is defined using parameter "Name" on MpBackupCore or parameter "NamePrefix" of the MpBackupCore configuration. If a backup is to be installed, the name of the main folder must be specified.

- **Target identification file:** This file determines on which target system the backup is to be installed. By default, mapp Backup allows installation on any target system. However, if required, a limitation can be defined in this file so that the backup can only be installed on target systems that have the same ConfigID, MAC address, station number or hostname, for example. For more information, see Structure of the project installation package.
- **Project installation package**: There are two files in this package: the configuration file (pipconfig.xml) and the transfer module to be installed. The configuration file specifies how the project is to be installed. The transfer module contains a consistent project status with all necessary information to install the project on the target system. It contains all application modules, system modules and user files (if any) located on the user partition. This includes all programs, libraries, data objects or AR version, for example:

![Diagram of project installation package]

The data created by MpBackupCore is identical to the data of a project installation package that can be created using Automation Studio or the Runtime Utility Center.

For more information about the backup and how it can be configured, see section Structure of the project installation package.

MpBackupCore can create and install backups. For more information on how to use a backup, see here.

However, an update can be installed using MpBackupCore. For more information, see section Installing an update.

### 3.1.2 Using a backup

A backup is a backup file of the target system used. It contains application modules, system modules and user files (if any) located on the user partition. For more information, see Structure of a backup.

This section explains how to use a backup:

**Creating/Installing a backup manually**

A backup can be created manually by MpBackupCore. A name and a data storage device must be specified for the backup.

A backup can also be installed using function block MpBackupCore. The desired backup and the data storage device where the backup is located must be specified.

For more information, see here.

**Creating a backup automatically**

MpBackupCore also offers the possibility to create a backup automatically. If a backup is to be created automatically, this can be defined via the MpBackupCore configuration. The configuration can be adjusted by function block MpBackupCoreConfig at runtime. For an automatically created backup, the timestamp such as "myPrefix_%Y_%m_%d_%H_%M_%S" is automatically appended to the name of the backup.

### 3.1.3 Installing an update

MpBackupCore can also be used to install an update. For example, if the machine application has been modified, this change can be installed in the form of an update on the target system.

This can be done using a project installation package:

**Using a project installation package as an update**

An application is also available on this subject. Application 2: Installing a new software version on the machine
As explained in section Structure of a backup, the backup data generated by MpBackupCore is identical to the data generated in the project installation package in Automation Studio or the Runtime Utility Center (RUC). This means the data in the project installation package can be installed via MpBackupCore. For this to work, the project installation package files must be saved in a folder. The name of the folder ("My_Backup") can then be specified on MpBackupCore. This allows an update to be performed on the target system.

3.1.4 Configuration during runtime

The configuration parameters can be modified at runtime using MpBackupCoreConfig. Nevertheless, it is important to note the state that the function block must be in for the different parameters to allow the change to take place. The following states are possible:

- ![Icon](image.png)
  This icon means that this parameter can be changed in any state.

- ![Icon](image.png)
  This icon means that this parameter is applied when a new positive edge occurs on input "Enable" of MpBackupCore.

Each parameter in the configuration structure (MpBackupCoreConfigType) is shown with one of these icons.

3.2 Guides

3.2.1 Getting started

3.2.1.1 Creating a backup

This section describes step by step how to create a backup using MpBackupCore.

The goal is to back up the data of the machine application.

3.2.1.1.1 Creating a project

You must first create a new project in Automation Studio. For more information, see here.

3.2.1.1.2 Adding the mapp component

Adding the MpBackupCore configuration

To create a backup, the MpBackupCore configuration is added.

It contains the mapp Link, which uniquely identifies the mapp component within the mapp environment. The mapp Link establishes the connection between programming and configuration.

3.2.1.1.3 Adding a program

The next step is to add a Ladder Diagram program. For more information, see here.

3.2.1.1.4 Adding MpBackupCore

In the next step, function block MpBackupCore is added. Backups can be created or installed using MpBackupCore.
3.2.1.5 Configuring MpBackupCore

Connecting MpLink
Using input "MpLink", we establish a connection to the configuration we created earlier named "gBackupCore".

Enabling the function block
The function block is enabled using input "Enable".

Defining the data storage device
A data storage device must be specified so that the backup can be saved. This is created in the hardware configuration and called "HD".
Defining a name for the backup

Finally, the name of the backup to be created must be defined. "MyBackup" is specified.

3.2.1.6 Generating the file structure

Now generate a file structure for the memory card for ARsim. For more information, see here.

3.2.1.7 Testing the program

After the changes are downloaded, the program can be tested. Monitor mode is enabled. Command "Create = TRUE" creates a backup.

The backup is stored on the specified data storage device. Which data the backup contains is listed in section Structure of a backup.

3.3 Configuration

3.3.1 MpBackupCore configuration

In the MpBackupCore configuration, you can define via "Automatic backup" whether a backup should be created automatically.

If a backup is to be created automatically, the following parameters can be configured:

- **Name prefix**: Defines the name of the backup. A timestamp is automatically appended after the name. For example, this could look like this: "myPrefix_%Y_%m_%d_%H_%M_%S".
- **Device name**: Specification of the data storage device where the backup is to be stored.
- **Mode**: Defines the time interval in which the backup is to be saved. The backup can be saved weekly or daily. If the file is to be saved weekly, the day is defined via "Day". "Time" defines the time at which the backup is to be created. The time is specified in seconds starting at 12:00 AM. If the file is to be saved at 6:00 AM, a time of 21600 s must be specified.
- **Overwrite oldest**: Defines whether older backups should be deleted or not. The maximum number of files is defined using parameter "Maximum number of backups".

The configuration can be modified at runtime using function block MpBackupCoreConfig.
If option "Override oldest" option is enabled at runtime by `MpBackupCoreConfig` and there are 30 backups on the specified data storage device but "MaximumNumberOfBackups" only allows 10, the oldest 20 backups are deleted.

3.4 Applications

3.4.1 Application 1: Creating a weekly backup

Requirement
For safety reasons, the current application including the user data of a machine should be stored weekly on a data storage device such as a USB flash drive. The backup file should be created and saved automatically every Friday.

Solution

Component list
- `MpBackupCore` (own MpLink): Creates the backup file and saves it to a specified data storage device.

Connection diagram

Configuration
The MpBackupCore configuration is added to create a backup file. Parameter "Automatic backup" is set to `TRUE` so that the backup is created weekly. The name of the backup file is defined using "Name prefix". When the file is created, a timestamp is automatically appended to its name. To define the storage location of the file, the desired data storage device is specified on "Device name". This must first be specified in the configuration of the target system. Parameter "Mode" must be set to "Weekly" so that the backup is created automatically on a weekly basis. "Day" and "Time" can then be used to select the desired day and time. The time is specified in seconds starting at 12:00 AM. If the backup is to be created at 6:00 AM, time 21600 must be specified.

Using the mapp components
Function block `MpBackupCore` is added and configured. A backup file is stored automatically on the specified data storage device using the settings made in the `MpBackupCore` configuration.

3.4.2 Application 2: Installing a new software version on the machine

Requirement
A software update is to be performed for a machine that is already in use for several months. The software update is to be installed on the machine using a USB flash drive.
Solution

Component list

- **MpBackupCore** (own MpLink): Creates and installs backups and updates.

Connection diagram

![Connection diagram](image)

Configuration

To perform a software update, the MpBackupCore configuration must be added.

Software update

The project is adjusted via Automation Studio. So that the new software can be transferred to the machine, a project installation package is created. For more information on how to create and use a project installation package, see here.

If the project installation package was created successfully, the data is located on a data storage device such as a USB flash drive:

![USB flash drive](image)

In order for the data to be used by mapp Backup, the data must be copied to a folder. The data was copied to folder "My_Backup". The data is transferred to a USB flash drive.
Using the mapp components

Function block **MpBackupCore** is added and configured.

An **HMI application** can be used to interact with MpBackupCore.

The name of the previously created "My_Backup" folder is specified on input parameter "Name". The file is located on a USB flash drive connected to the machine.

The update for the machine is installed using command "Install = TRUE".

3.4.3 Application 3: Restoring the machine software on a new controller

**Requirement**

The controller of a machine has been damaged and must be replaced. There is a backup of the previous controller. This was generated as explained here. This backup is to be transferred to the new controller using a USB flash drive.

**Restoring the new controller from a backup**

The existing backup of the previous controller is to be transferred to the new controller. There are different solutions depending on the target system used and the runtime version of the target system. Three different solutions are listed here. For the listed solutions, a controller **without** a CompactFlash card is used.

- Control with Boot AR ≥4.33
- Controller with Boot AR <4.33
- Controller with BIOS and without Boot AR

To find out which Boot AR version the target system uses, you can connect to the controller via Automation Studio, for example. For more information, see here.

3.4.3.1 Control with Boot AR ≥4.33

**Preparing the USB flash drive with backup:**

For the backup to be installed on the target system, the contents of the backup must be on a USB flash drive. The backup files are not permitted to be located in subfolders. This means that the "target identification file" file must be located in the root directory of the USB flash drive.
Installing a backup

The USB flash drive is connected to the target system

In order to install the backup from the USB flash drive, the controller must be in operating mode BOOT. This is done with the operating mode switch or reset button of the controller. The backup is automatically installed on the controller.

3.4.3.2 Controller with Boot AR <4.33

If the controller has a Boot AR version <4.33, the following preparations must be made before the backup can be installed:

Preparing the USB flash drive with backup:

For the backup to be installed on the target system, the contents of the backup must be on a USB flash drive. The backup files are not permitted to be located in subfolders. This means that the "target identification file" file must be located in the root directory of the USB flash drive.
Generating the project installation package

To install the backup, an AR version ≥4.33 is required.

A project installation package is created so that the target system has the necessary runtime version. This means that a project is created in Automation Studio that has the hardware of the target system used. In addition, a runtime version ≥4.33 must be set in the project.

Finally, option "Activate USB stick install" in the target system configuration is enabled so that the backup can be installed later via a USB flash drive.
A project installation package can now be created using "Generate project installation package". For more information, see here.

It is important to use option "Generate PIP incl. support of AR < 4.33". This is the only way to install the project installation package on the target system.
In the transfer settings, you can specify how the package is to be installed on the target system. It is recommended to select option "Force initial transfer". 

The data created by the project installation package is transferred to a USB flash drive and connected to the target system. It is important that the data is located in the root directory of the USB flash drive, i.e. the data is not permitted to located in subfolders.

In order to install the data from the USB flash drive, the controller must be in operating mode BOOT. This is done with the operating mode switch of the controller. The project installation package is then automatically installed on the controller. Once the package is installed, the USB flash drive can be removed from the target system.

Installing a backup

If the target system now has AR version ≥4.33, the USB flash drive containing the backup to be installed can be connected to the target system. To install the data, the controller must be restarted. If option "Activate USB install" was enabled previously, the backup on the USB flash drive is installed when the controller is started up.

3.4.3.3 Controller with BIOS and without Boot AR

If there is BIOS and no Boot AR on the controller, the following preparations must be made:

Preparing the USB flash drive with backup:

For the backup to be installed on the target system, the contents of the backup must be on a USB flash drive. The backup files are not permitted to be located in subfolders. This means that the "target identification file" file must be located in the root directory of the USB flash drive.
Creating an offline installation

To install the backup, an AR version ≥4.33 is required.

This means that a project is created in Automation Studio that has the hardware of the target system used. In addition, a runtime version ≥4.33 must be set in the project.

Finally, option "Activate USB stick install" in the target system configuration is enabled so that the backup can be installed later via a USB flash drive.
For the new runtime version to be transferred to the target system, an offline installation must be performed. The data generated by the offline installation must be stored on a USB flash drive:

The USB flash drive with the data is then connected to the target system. The BIOS on the target system must then be set so that the boot process is started from the connected USB flash drive. This installs the project located on the USB flash drive. Once the project is installed, the USB flash drive can be removed from the target system. The setting in the BIOS system can be reset.

**Installing a backup**

If the target system now has AR version ≥4.33, the USB flash drive containing the backup to be installed can be connected to the target system. To install the data, the controller must be restarted. If option "Activate USB install" was enabled previously, the backup on the USB flash drive is installed when the controller is started up.
3.5 Libraries

3.5.1 Data types and enumerators

3.5.1.1 MpBackupAutoBackupType

This data type determines when and how the backup is to be saved.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Configuration update</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>BOOL</td>
<td>Specifies whether a backup should be created automatically (&quot;Enabled = TRUE&quot;) or not (&quot;Enabled = FALSE&quot;)</td>
<td></td>
</tr>
<tr>
<td>NamePrefix</td>
<td>STRING[50]</td>
<td>Defines the name of the backup. A timestamp is automatically appended after the name. For example, this could look like this: &quot;myPrefix_%Y_%m_%d_%H_%M_%S&quot;.</td>
<td></td>
</tr>
<tr>
<td>DeviceName</td>
<td>STRING[255]</td>
<td>Specification of the data storage device where the backup is to be stored.</td>
<td></td>
</tr>
<tr>
<td>Mode</td>
<td>MpBackupIntervalType</td>
<td>Defines the time interval in which the backup is to be saved.</td>
<td></td>
</tr>
<tr>
<td>OverwriteOldest</td>
<td>MpBackupOverwriteOldestType</td>
<td>Defines whether older backups should be deleted or not.</td>
<td></td>
</tr>
</tbody>
</table>

3.5.1.2 MpBackupCoreConfigType

This data type defines whether the backup is to be saved automatically.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AutomaticBackup</td>
<td>MpBackupAutoBackupType</td>
<td>This structure can be used to define whether a backup is to be created automatically.</td>
<td></td>
</tr>
</tbody>
</table>

3.5.1.3 MpBackupCoreInfoType

This data type provides additional information for the function block.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LastBackup</td>
<td>MpBackupLastBackupType</td>
<td>Information about the last backup created.</td>
<td></td>
</tr>
<tr>
<td>Project</td>
<td>MpBackupProjectInfoType</td>
<td>Information about the backup to be installed or requested.</td>
<td></td>
</tr>
<tr>
<td>Diag</td>
<td>MpBackupDiagType</td>
<td>Diagnostic structure for the status ID.</td>
<td></td>
</tr>
</tbody>
</table>

3.5.1.4 MpBackupDiagType

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>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>StatusID</td>
<td>MpBackupStatusIDType</td>
<td>StatusID diagnostic structure.</td>
<td></td>
</tr>
</tbody>
</table>

3.5.1.5 MpBackupInfoType

This data type provides additional information for the function block.

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

3.5.1.6 MpBackupIntervalType

This data type describes the interval at which the backup is to be created automatically.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
<th>Configuration update</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interval</td>
<td>MpBackupIntervalEnum</td>
<td>Parameter &quot;Interval&quot; defines the interval at which the file is to be saved. The file can be saved weekly or daily.</td>
<td></td>
</tr>
<tr>
<td>Day</td>
<td>MpBackupDayEnum</td>
<td>Defines the day on which the backup is to be created.</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>TIME_OF_DAY</td>
<td>Defines the time at which the backup is to be created. The time is specified in seconds starting at 12:00 AM. If the file is to be saved at 6:00 AM, a time of 21600 s must be specified.</td>
<td></td>
</tr>
</tbody>
</table>
3.5.1.7 MpBackupLastBackupType

This data type contains information about the last backup created.

<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 last backup created.</td>
</tr>
<tr>
<td>DateTime</td>
<td>DATE_AND_TIME</td>
<td>Time at which the backup was created.</td>
</tr>
<tr>
<td>GenerationType</td>
<td>MpBackupGenerationTypeEnum</td>
<td>Specifies whether the last backup was created manually or automatically.</td>
</tr>
</tbody>
</table>

3.5.1.8 MpBackupOverwriteOldestType

This data type specifies whether the oldest backup is to be overwritten.

If this option is enabled and there are 30 backups on the specified data storage device but "MaximumNumberOfBackups" only allows 10, the oldest 20 backups are deleted.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>BOOL</td>
<td>Defines whether older backups should be overwritten.</td>
</tr>
<tr>
<td>MaximumNumberOfBackups</td>
<td>UINT</td>
<td>Defines the number of possible backups. Instead of exceeding this, the oldest existing backup is overwritten.</td>
</tr>
</tbody>
</table>

3.5.1.9 MpBackupProjectInfoCurrentType

This data type provides information about the active backup on the target system. For more information about these two parameters, see here.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConfigurationID</td>
<td>STRING[255]</td>
<td>Configuration ID of the installed project.</td>
</tr>
<tr>
<td>ConfigurationVersion</td>
<td>STRING[64]</td>
<td>Configuration version of the installed project.</td>
</tr>
</tbody>
</table>

3.5.1.10 MpBackupProjectInfoRequestType

This data type provides information about the last backup installed on the target system.

<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 backup from which information was requested.</td>
</tr>
<tr>
<td>ConfigurationID</td>
<td>STRING[255]</td>
<td>Configuration ID of the project.</td>
</tr>
<tr>
<td>ConfigurationVersion</td>
<td>STRING[64]</td>
<td>Configuration version of the project.</td>
</tr>
</tbody>
</table>

3.5.1.11 MpBackupProjectInfoType

This data type provides information about backups.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>MpBackupProjectInfoCurrentType</td>
<td>Project information of the current project.</td>
</tr>
<tr>
<td>Requested</td>
<td>MpBackupProjectInfoRequestType</td>
<td>Project information of the requested backup.</td>
</tr>
</tbody>
</table>

3.5.1.12 MpBackupStatusIDType

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>MpBackupErrorEnum</td>
<td>Error code for the function block.</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 documentation.</td>
</tr>
</tbody>
</table>

3.5.1.2 Enumerators

3.5.1.2.1 MpBackupDayEnum

This enumerated data type specifies the day on which an automatic backup is to be created.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpBACKUP_DAY_MONDAY</td>
<td>The file is created on a Monday.</td>
</tr>
<tr>
<td>mpBACKUP_DAY_TUESDAY</td>
<td>The file is created on a Tuesday.</td>
</tr>
<tr>
<td>mpBACKUP_DAY_WEDNESDAY</td>
<td>The file is created on a Wednesday.</td>
</tr>
</tbody>
</table>
### 3.5.1.2.2 MpBackupGenerationTypeEnum

This enumerated data type specifies how the backup was created.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpBACKUP_GENERATION_AUTO</td>
<td>The backup was created automatically.</td>
</tr>
<tr>
<td>mpBACKUP_GENERATION_MANUAL</td>
<td>The backup was created manually.</td>
</tr>
</tbody>
</table>

### 3.5.1.2.3 MpBackupIntervalEnum

This enumerated data type specifies the interval at which a backup is to be created.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpBACKUP_INTERVAL_DAILY</td>
<td>The backup is created daily.</td>
</tr>
<tr>
<td>mpBACKUP_INTERVAL_WEEKLY</td>
<td>The backup is created weekly.</td>
</tr>
</tbody>
</table>

### 3.5.2 Function blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MpBackupCore</td>
<td>This function block can be used to store or install a backup or update on a data storage device.</td>
</tr>
<tr>
<td>MpBackupCoreConfig</td>
<td>This function block can be used to configure the automatic creation of a backup.</td>
</tr>
</tbody>
</table>

### 3.5.2.1 MpBackupCore

This function block can be used to store or install a backup or update on a data storage device.

#### Function block

```
## Optional parameters

MpBackupCore

BOOL Enable
BOOL ErrorReset
&STRING DeviceName
&STRING Name
BOOL Create
BOOL Install
BOOL RequestInfo

MpBackupCoreInfoType

MpBackupKern

MpBackupKern

MpBackupKern
```
## 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 Mp-ComIdentType</td>
<td>Connection to mapp (MpLink of an MpBackup 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>DeviceName</td>
<td>STRING[50] to STRING[255]</td>
<td>File device (data storage medium) where the files are stored. Name of the backup/update. The file can also be saved/found in a folder on the specified data storage device. In this case, the folder name must also be specified (e.g. &quot;FolderName/Filename&quot;).</td>
</tr>
<tr>
<td>IN</td>
<td>Name</td>
<td>STRING[255]</td>
<td>File device (data storage medium) where the files are stored. Name of the backup/update. The file can also be saved/found in a folder on the specified data storage device. In this case, the folder name must also be specified (e.g. &quot;FolderName/Filename&quot;).</td>
</tr>
<tr>
<td>IN</td>
<td>Create</td>
<td>BOOL</td>
<td>Command to create a backup.</td>
</tr>
<tr>
<td>IN</td>
<td>Install</td>
<td>BOOL</td>
<td>Command to install a backup.</td>
</tr>
<tr>
<td>IN</td>
<td>RequestInfo</td>
<td>BOOL</td>
<td>This command can be used to request information about the backup. The information is visible in output parameter Info.</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>MpBackupCoreInfoType</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 be carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

#### 3.5.2.1.1 Description

The MpLink of an MpBackupCore configuration is used for this function block.

This function block can be used to create or install a backup. How a backup is structured is explained in section Structure of a backup. How a backup is used is explained in section Using a backup.

With MpBackupCore, an update can also be installed. For more information, see section Installing an update.

### Creating a backup

A backup can be created using command "Create = TRUE". The file is stored on the data storage device specified on "DeviceName". The name of the backup is determined by input parameter "Name". The file can also be stored in a folder on the specified data storage device. In this case, the folder name must also be specified (e.g. "FolderName/Filename"). If a backup is created automatically, a timestamp is automatically appended to the name of the backup. No timestamp is appended to a manually created backup. For more information, see Using a backup.

### Installing a backup/update

An existing backup/update can be installed on the target system with command "Install = TRUE". The data storage device in which the desired backup/update is stored is determined via parameter "DeviceName". The backup/update to be loaded is specified on parameter "Name". The backup/update can also be stored in a folder on the specified data storage device. In this case, the folder name must also be specified (e.g. "FolderName/Filename").

### Requesting information about a backup

The configuration ID and configuration version of a backup can be requested with command "RequestInfo = TRUE". For more information about these two parameters, see here.

The name of the desired backup and location of the backup ("DeviceName") must be specified on parameter "Name". The requested information is displayed in structure MpBackupProjectInfoRequestType of the info structure. This makes it possible to check whether the configuration ID of the requested backup is identical to the configuration ID of the currently installed backup.

### Info structure

The info structure contains additional information about the created backups:

- **LastBackup**: Parameter "LastBackup" specifies the last backup created using command "Create".
- **Project**: Parameter "Project" specifies which project is installed on the controller. In addition, the information requested via "RequestInfo" is displayed in the info structure.
The data created by MpBackupCore is identical to the data of a project installation package that can be created using Automation Studio or the Runtime Utility Center. For more information, see section Structure of the project installation package.

3.5.2.2 MpBackupCoreConfig

This function block can be used to configure the automatic creation of a backup.

**Function block**

## Optional parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MpComIdentType</td>
<td>BOOL</td>
<td>Connection to mapp (MpLink of an MpBackUpCore configuration)</td>
</tr>
<tr>
<td>Enable</td>
<td>BOOL</td>
<td>The function block is active as long as this input is set.</td>
</tr>
<tr>
<td>ErrorReset</td>
<td>BOOL</td>
<td>Resets function block errors.</td>
</tr>
<tr>
<td>Configuration</td>
<td>MpBackupCoreConfigType</td>
<td>Structure used to specify the configuration.</td>
</tr>
<tr>
<td>Load</td>
<td>BOOL</td>
<td>Loads the configuration of the component.</td>
</tr>
<tr>
<td>Save</td>
<td>BOOL</td>
<td>Saves the configuration of the component.</td>
</tr>
<tr>
<td>Active</td>
<td>BOOL</td>
<td>Function block active.</td>
</tr>
<tr>
<td>Error</td>
<td>BOOL</td>
<td>Error occurred during execution.</td>
</tr>
<tr>
<td>StatusID</td>
<td>DINT</td>
<td>Status information.</td>
</tr>
<tr>
<td>CommandBusy</td>
<td>BOOL</td>
<td>Function block currently executing command.</td>
</tr>
<tr>
<td>CommandDone</td>
<td>BOOL</td>
<td>Execution successful. Function block is finished.</td>
</tr>
<tr>
<td>Info</td>
<td>MpBackupInfoType</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 MpBackUpCore 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>Configuration</td>
<td>MpBackupCoreConfigType</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>MpBackupInfoType</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 be carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

3.5.2.2.1 Description

The same MpLink used by MpBackupCore is used for this function block.
This function block reads and writes the configuration. The complete configuration 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 respective configuration data types for the function block explains when parameters in the configuration are applied.

**Configuring structure MpBackupCoreConfigType**

This configuration defines whether the backup is created automatically.

If the backup is created automatically, the following settings can be made:

- **NamePrefix**: Defines the name of the backup. A timestamp is automatically appended after the name. For example, this could look like this: "myPrefix_%Y_%m_%d_%H_%M_%S".
- **DeviceName**: Specification of the data storage device where the backup is to be stored.
- **Mode**: Defines the time interval in which the backup is to be saved. The backup can be saved weekly or daily.
- **OverwriteOldest**: Defines whether older backups should be deleted or not.

### 3.5.3 Status numbers

#### 3.5.3.1 -1064159488: Installation failed

**Description:**
Specified backup/update {2:Name} could not be installed. Error number: {1:ErrorNumber}

**Additional information**
- {2:Name}: Name of the backup/update
- {1:ErrorNumber}: Error number

**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 backup was specified on input parameter "Name".
- Check whether the specified data storage device exists.

**Constant:**

mpBACKUP_ERR_INSTALL_FAILED

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

#### 3.5.3.2 -1064159487: Creation failed

**Description:**
Specified backup/update {2:Name} could not be created. Cause of error: {1:ErrorNumber}
mapp Services

Additional information
- {2:Name}: Name of the backup/update
- {1:ErrorNumber}: Error number

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 backup was specified on input parameter "Name".
- Check whether the specified data storage device exists.
- Check whether write permissions exist on the specified data storage device.

Constant:
mpBACKUP_ERR_CREATE_FAILED

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

3.5.3.3 -1064159486: Could not query information

Description:
The information of desired backup/update {2:Name} could not be queried. Cause of error: {1:ErrorNumber}

Additional information
- {2:Name}: Name of the backup/update
- {1:ErrorNumber}: Error number

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 backup was specified on input parameter "Name".
- Check whether the specified data storage device exists.

Constant:
mpBACKUP_ERR_REQUEST_INFO_FAILED

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

3.5.3.4 -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:
- MpBackupCoreConfig

3.5.3.5 -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:
- MpBackupCore
- MpBackupCoreConfig

3.5.3.6 -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:
- MpBackupCore

3.5.3.7 -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:
3.5.3.8 -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:
- MpBackupCore
- MpBackupCoreConfig

3.5.3.9 -1064239099: Invalid MpLink contents

Description:
The value of variable "MpLink" 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:
- MpBackupCore
- MpBackupCoreConfig

3.5.3.10 -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:
3.5.3.11 -1064239096: Configuration structure is 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:

- MpBackupCoreConfig

3.5.3.12 -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:

- MpBackupCoreConfig

3.5.3.13 -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:

- MpBackupCoreConfig

3.5.3.14 -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:
- MpBackupCoreConfig

3.5.3.15 -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.
- The function is not enabled in the configuration.

These function blocks / functions can report this error:
- MpBackupCore
- MpBackupCoreConfig

3.5.3.16 -1064289140: Failed to read backup/update

Description:
Backup/Update {3:Name} could not be installed from data storage device {2:DeviceName}. The file could not be read.

Additional information
- {2:DeviceName}: Specified data storage device
- {3:Name}: Name of the backup/update

Cause/Solution:
- Check whether the specified data storage device exists.

3.5.3.17 -1064289139: File open/read failed

Description:
Opening and reading file {3:FileName} from data storage device {2:DeviceName} failed.

Additional information
- {3:FileName}: Name of the file
- {2:DeviceName}: Specified data storage device
3.5.3.18 -1064289138: Invalid target identification file

Description:
Target identification file {2:FileName} is invalid.

Additional information
- {3:FileName}: Name of the file

Cause/Solution:
- Check whether the specified data storage device exists.
- Check the specified filename.
- Check the file contents.

3.5.3.19 -1064289137: Cannot install update/backup on target system

Description:
The update/backup cannot be installed since it does not match the target system used.

Cause/Solution:
- Use a different backup/update.
- Use a different target system.

3.5.3.20 -1064289136: Cannot install project

Description:
The project could not be installed. Cause of error: {2:FailureCause}

Additional information
- {2:FailureCause}: Cause of error

Cause/Solution:
- Check whether the specified data storage device exists.
- Check the specified filename.

3.5.3.21 -1064289135: Data storage device for PIP not found

Description:
The specified data storage device for the project installation package (PIP) was not found.

Cause/Solution:
- Check whether the specified data storage device exists.

3.5.3.22 -1064289134: Invalid PIP

Description:
The project installation package (PIP) cannot be installed or does not exist on the data storage device.

Cause/Solution:
- Check the contents of the PIP.
- Check if the PIP exists.
3.5.3.23 -1064289133: Cannot install PIP

Description:
The project installation package (PIP) cannot be installed.

Cause/Solution:
- Check whether the specified data storage device exists.
- Check the specified filename.

3.5.3.24 -1064289132: PIP not usable on target system

Description:
The project installation package (PIP) cannot be installed since it is not supported by the target system.

Cause/Solution:
- The runtime version on the target system must be ≥4.33.

3.5.4 Alarms

3.5.4.1 mpBACKUP_ALM_CREATE_FAILED: Creation of backup failed

Description:
The backup could not be created.

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

Cause/Solution:
- Check whether a name was assigned for the backup.
- Check whether the specified data storage device exists.
- Check whether write permissions exist on the specified data storage device.

Behavior
Edge alarm

3.5.4.2 mpBACKUP_ALM_INSTALL_FAILED: Failed to install backup

Description:
The desired backup could not be installed.

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

Cause/Solution:
- Check whether a name was assigned for the backup.
- Check whether the specified data storage device exists.
4 mapp IO: Managing hardware configuration

mapp IO makes it possible to modify a hardware configuration at runtime.

Our B&R online tutorial platform includes tutorials about mapp IO.

4.1 Concept

<table>
<thead>
<tr>
<th>User stories</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine manufacturer</td>
<td>• As a machine manufacturer, I would like the flexibility of being able to create the hardware configuration in an external tool in an established format and to import this directly on the machine.</td>
</tr>
<tr>
<td></td>
<td>• As a machine manufacturer, I would like to make sure that replacement hardware modules come from me.</td>
</tr>
<tr>
<td></td>
<td>• As a service technician, I would like to reassign an I/O data point to another I/O data point at runtime without having to restart.</td>
</tr>
<tr>
<td></td>
<td>• As a service technician, I would like to be able to replace a defective hardware module with another module (in stock) without having to use Automation Studio.</td>
</tr>
<tr>
<td>Service technician</td>
<td>• As an application engineer, I would like to only have to maintain a single piece of software for all possible machine configurations.</td>
</tr>
<tr>
<td></td>
<td>• As an application engineer, I do not want to have to check whether an alarm must be set for each individual module when the module is not functioning correctly.</td>
</tr>
</tbody>
</table>

4.1.1 Basic principle

3 things are necessary to operate a hardware module:

- **AR configuration**: Contains information about how the module works as well as possible parameters and settings (e.g. type of temperature sensor or whether an analog input pays attention to voltage or current). The AR configuration of a module can be changed after the module has been added to the hardware tree.

- **I/O mapping**: The I/O mapping defines the connection between software and hardware (e.g. which process variable receives that current value from an analog input). The I/O mapping of a module can be changed after the module has been added to the hardware tree.

- **Firmware**: Software embedded into the module that receives the AR configuration data and operates the module according to the configured settings. The firmware is included in the transfer to the controller as soon as the module has been added to the hardware tree.

If the hardware configuration should be changed (e.g. adding a new module), this normally has to be configured in Automation Studio and then transferred to the controller.
MpIO offers a flexible alternative. The hardware configuration can be modified using a simple import file in XML format. Specifically, this can be done to modify the AR configuration as well as the I/O mapping.

The import file defines the position of the hardware module in the hardware structure as well as the data points in the software to which the hardware module is connected. The import file can be created in an external tool such as SAP or EPLAN. For information about the format of the import file, see section Structure of the import file. The import file (yellow) references modules that existed in the original AR configuration (orange) and uses this information to generate a new AR configuration. The exact process is described in section Managing the hardware configuration.

The following steps are necessary to work with MpIO and an import file:

1. Create an Automation Studio project with all hardware modules in all variants that should be possible later on the machine.
   Not only are all necessary modules added one time, but also each of their variants. For example, X20AT2402 is added with sensor type J, while another X20AT2402 is added with sensor type K.
2. Transfer the project to the controller.
   This puts the AR configuration used later as a reference onto the controller. In addition, the firmware for all necessary modules is transferred to the controller.
3. Create the import file based on the actual machine structure.
4. Import the file using MpIOImport.

During the import procedure, a new AR configuration is created based on the import file. This then references certain module configurations from the original AR configuration that was transferred to the controller via Automation Runtime in step 2. When the new AR configuration is generated, it is enabled immediately. This may result in a restart of the controller.

Part of the import file is the I/O mapping, which is made immediately for the respective module.

For a detailed explanation of the format shown in the image, see section Structure of the import file.

**4.1.2 Managing the hardware configuration**

When an Automation Studio project is transferred to the controller, it includes the AR configuration (orange), I/O mapping (blue) and the NC mapping table (brown). In addition to the firmware, this information is necessary for the proper configuration of the hardware and to establish the connection to the software. All data necessary for this is stored in ROM.
The files are stored in ROM, where they are then loaded to DRAM during each restart.

**Using MpIO**

MpIO can be used to modify the configuration at runtime without using Automation Studio. If function block MpIOImport is enabled, then the very first thing that happens is the active configuration files are backed up.

The configuration files (AR configuration, I/O mapping(s) and NC mapping table(s)) are copied to a backup file device defined in the MpIO configuration.

**Command "Import"**

With command "Import", MpIOImport additionally copies the file defined by input "FileName" (yellow) to the backup file device.

At this point, the first hardware configuration defined using Automation Studio is active. A new hardware configuration will be generated in the next step from the data present on the backup file device, however. Depending on parameter "Master configuration" in the MpIO configuration, it will be based on either the Automation Studio hardware configuration or the import file:

- **None**: The new hardware configuration is based on the import file. The data provided by Automation Studio serves only as a reference.
- **Automation Runtime**: The hardware configuration provided by Automation Studio remains the basis, and the import file is used to extend a certain interface (e.g. IF6, X2X on X20CP1586).

If the new hardware configuration is generated from the data on the backup file device, then it will be applied immediately.

Depending on whether the new hardware configuration only changes the I/O mapping or the AR configuration as well, then the controller will restart.
The hardware configuration continues to be loaded from ROM each time the controller is restarted, but it corre-
sponds no longer to the hardware configuration transferred using Automation Studio, but instead the hardware
configuration that was imported using MpiOImport.

Restoring the hardware configuration

MpiO provides a mechanism that monitors whether the hardware configuration has been modified externally (i.e.
not using MpiO), for example via an Automation Studio download or using Runtime Utility Center. If the hardware
configuration is changed in this way while MpiO is active, then an Alarm is triggered.

During the activation process, MpiO additionally checks whether the hardware configuration taken from the data
on the backup file device corresponds to the hardware configuration that was loaded from ROM. If this is not the
case, then the hardware configuration was modified externally (i.e. not using MpiO). MpiO then immediately re-
imports the hardware configuration on the backup file device. This may cause the controller to restart; once this is
completed, the imported hardware configuration is active again. If this automatic restore procedure is not desired,
it can be prevented using input "ImportAllowed" on MpiOImport.

4.1.3 Structure of the import file

This section describes the structure of the import file. The format of the import file is compatible with the I/O mapping
table, the AR configuration and the NC mapping table that are created using Automation Studio.

Overview of XML elements and attributes

<table>
<thead>
<tr>
<th>XML element</th>
<th>Sub-element / Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I OCFG</td>
<td></td>
<td>Root node</td>
</tr>
<tr>
<td></td>
<td>Version</td>
<td>Version of the configuration</td>
</tr>
<tr>
<td>Module</td>
<td>ID</td>
<td>Defines a new module.</td>
</tr>
<tr>
<td></td>
<td>Hardware</td>
<td>Defines the address/interface to which the module is connected (e.g. &quot;IF6.ST1&quot;)</td>
</tr>
<tr>
<td></td>
<td>SourceName</td>
<td>Unique identification string of the I/O module (e.g. &quot;X20DI9371&quot;)</td>
</tr>
<tr>
<td></td>
<td>SourceID</td>
<td>Name of the module in the referenced configuration (e.g. &quot;MyModule&quot;)</td>
</tr>
<tr>
<td></td>
<td>Source</td>
<td>ID of the module in the referenced configuration (e.g. &quot;IF6.ST7&quot;)</td>
</tr>
<tr>
<td></td>
<td>ImportMode</td>
<td>Optional: Defines where the module configuration can be found (e.g. &quot;AR&quot;)</td>
</tr>
<tr>
<td></td>
<td>LN</td>
<td>Optional: Defines how the module should be imported (e.g. &quot;ModuleOnly&quot;)</td>
</tr>
<tr>
<td></td>
<td>ID</td>
<td>The configuration of a module can be modified with these.</td>
</tr>
<tr>
<td></td>
<td>Value</td>
<td>Unique identification of an element.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New value of the parameter.</td>
</tr>
<tr>
<td>I OCFG</td>
<td>ID</td>
<td>Name of the data point to be connected (e.g. &quot;AnalogInput01&quot;)</td>
</tr>
<tr>
<td></td>
<td>D PName</td>
<td>Fully qualified name of the variable to be connected (e.g. &quot;Program:MyDatapoint&quot;)</td>
</tr>
<tr>
<td></td>
<td>Device</td>
<td>Defines task class in which the I/O value is transferred.</td>
</tr>
<tr>
<td></td>
<td>Mode</td>
<td>Defines how pre-existing entries are handled (e.g. &quot;Remove&quot;)</td>
</tr>
<tr>
<td>Nbject</td>
<td>Name</td>
<td>NC mapping Information for the module.</td>
</tr>
<tr>
<td></td>
<td>Channel</td>
<td>Fully qualified name of the variable to be connected (e.g. &quot;gAxis01&quot;)</td>
</tr>
<tr>
<td></td>
<td>Channel number</td>
<td>Channel number to which the connection should take place (e.g. &quot;2&quot;)</td>
</tr>
</tbody>
</table>

I OCFG

Node "I OCFG" is the root node of the XML file and therefore required in every import file.
Module

Node "Module" defines a new module for the configuration. This can include a module configuration as well as the I/O mapping for that module.

- **ID:**
  
  The ID defines the position in the hardware topology where the module is added. The position of the module is defined as follows:

  ```xml
  [SL<slotnr>][.SS<sslnr>] [.IF<ifnr>.ST<stanr>][.SL<staslotnr>][.SS<stasslnr>].<chnr>
  ```

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slotnr</td>
<td>Slot number</td>
</tr>
<tr>
<td>sslnr</td>
<td>Subslot number</td>
</tr>
<tr>
<td>ifnr</td>
<td>Interface number</td>
</tr>
<tr>
<td>stanr</td>
<td>Station number</td>
</tr>
<tr>
<td>staslotnr</td>
<td>Station slot number</td>
</tr>
<tr>
<td>stasslnr</td>
<td>Station subslot number</td>
</tr>
<tr>
<td>chnr</td>
<td>Channel number</td>
</tr>
</tbody>
</table>

  **Example:**

  ```xml
  <Module ID="IF6.ST1" Hardware="X20DI9371" />
  <Module ID="IF3.ST1.IF1.ST2" Hardware="X20DO9322" />
  ```

  The following modules marked in red are added in this example:

  ![Diagram showing hardware setup]

- **Hardware:**
  Unique identification of the module. For the identification string, see the help documentation for the module.

**Order data**

<table>
<thead>
<tr>
<th>Model number</th>
</tr>
</thead>
<tbody>
<tr>
<td>X20DI9371</td>
</tr>
<tr>
<td>X20cDI9371</td>
</tr>
</tbody>
</table>

- **SourceName:**
  Name of the hardware module in the referenced configuration. This can be used if the module is named in the Automation Studio configuration. Replaces property "Hardware".

  ```xml
  <Module ID="IF3.ST1.IF1.ST2" SourceName="MOutput" />
  ```
• **SourceID:**
  Defines the position of the module being added in the current configuration.
  Example:
  ```xml
  <Module ID="IF6.ST1" SourceID="IF6.ST13" />
  ```

• **Source:**
  This parameter is optional. It defines where the configuration of the modules can be found. The default value is "AR".
  - **AR:** The configuration from the master configuration is used.
  - **Template:** The configuration from the Template files is used.

• **ImportMode:**
  This parameter is optional. The following example imports module X20BC0083, which already exists in the AR configuration, using different import modes. The import mode can be defined as follows:
  - **ModuleOnly:** Only the specified module is added.
  - **ModuleWithSubslots:** The specified module is added with all of its subslots.
  - **Branch:** The specified module is added with all of its subslots and submodules.
If "ImportMode" is not specified, then ModuleWithSubslots will be used by default.

**Parameter**
This node allows the configuration of each module to be modified individually.

- **ID:** Unique identification of a parameter in a group or the configuration.
- **Value:** New value of the parameter.

**Example:**

```xml
<Parameter ID="Cyclic1Duration" Value="2000" />
```

**LN**
This node allows an I/O mapping to be defined for a module.

- **ID:** Name of the data point for which an I/O mapping is created (e.g. "IF6.ST1")
- **DPName:** Name of the process variable that is connected (e.g. "Program:MyPV")
- **Device:** Defines the task class in which the value is updated. If no value is specified, "TC#1-CPYDEV" is used as the task class by default.
- **Mode:** Defines how pre-existing entries should be handled (e.g. "replace")

**Mode**
The following possibilities are available for "Mode":

- **Add:** Adds the defined mapping without removing any pre-existing mappings. This mode is only possible for inputs and is the default value for these.
- **Replace:** Replaces pre-existing mappings with those defined in the import file. This is the default mode for outputs.
- **Clear:** Deletes an existing mapping without defining a new one.

The I/O mapping for a module is defined as follows:

```xml
<Module ID="IF3.ST1.IF1.ST2" Hardware="X20DO9322" >
  <LN ID="DigitalOutput03" DPName="doOutput2" />
</Module>
```

The closing tag for node "Module" must come after all "LN" nodes!

**NcObject**
This node imports a new NC mapping for the module.

- **Name:** Name of the axis variable connected to the module.
mapp Services

- **Channel:**
The channel of the module to which the variable is connected.

```
<Module ID="IF3.ST1" Source="AR" SourceName="MAcopos" />
<NcObject Name="myAxis" Channel="1" />
</Module>
```

**The closing tag for node "Module" must come after all "NcObject" nodes!**

**Example of a complete import file:**

```
<IOCFG xmlns="http://www.br-automation.com/AR/IO" Version="2.0">
  <Module ID="$root" Source="AR" SourceID="$root" />
  <Module ID="IF6.ST1" Source="AR" SourceName="MyInputModule" />
  <Module ID="IF6.ST1.IO">
    <LN ID="DigitalInput01" DPName="diInput1" />
  </Module>
  <Module ID="IF6.ST2" Source="AR" SourceName="MyOutputModule">
    <LN ID="DigitalOutput01" DPName="doOutput1" />
  </Module>
</IOCFG>
```

The CPU from the current configuration is applied via node "Module" with ID "$root".

After importing this file, the hardware configuration will look like this:

![Hardware Tree](image)

The XML schema definition for the import file is attached here: Import

**4.1.4 Template files**

Template files are templates for certain hardware modules and provide another way to integrate hardware modules into the hardware configuration.

Section **Basic principle** describes the method of transferring to the controller a hardware configuration from Automation Studio with all possible modules. The import file then determines which modules should be located at which position in the hardware configuration.

Template files provide another way to reference hardware modules with more flexibility. The main difference is that Template is selected for parameter "Source" in the configuration instead of AR, i.e. the hardware configuration transferred using Automation Studio. The system then searches for a template file on the template file device ("Template device name" in the MpIO configuration). There must be 2 template files per module:

- **MyTemplate.ar** (e.g. X20DI9371.ar) – This file contains the entire module configuration in the same way that it is contained in the AR configuration.
- **MyTemplate.io** (e.g. X20DI9371.io) – This file contains the entire I/O mapping in the same way that it is contained in the I/O mapping table.
Except for the extension, the name of template files can be selected as needed. The template is specified in the import file by its filename without the extension.

```
<IOCFG xmlns="http://www.br-automation.com/AR/IO" Version="2.0">
  <Module ID="$root" Source="AR" SourceID="$root" />  <!-- ... -->
  <Module ID="IF6.ST1" Source="Template" SourceName="MyTemplate" />
</IOCFG>
```

Only one hardware module is permitted to be defined in each template file. Interface modules and bus controllers are exceptions to this. For these modules, all submodules/subslots can be also added in the template file. They are likewise imported depending on \textit{ImportMode}.

**Creating template files**

Template files are created by opening file "arconfig.br" for an existing configuration. All modules being used as well as their properties are entered in this file. Example for module X20DI9371:

```
<Module ID="IF6.ST1" Hardware="X20DI9371">
  <Parameter ID="HardwareModuleName" Value="X20DI9371" />
</Module>

<Module ID="IF6.ST1.IO" Hardware="X20DI9371">
  <Parameter ID="FunctionModel" Value="Standard" />
  <Parameter ID="FunctionModelNumber" Value="0" Type="USINT" />
  <Parameter ID="Supervision" Value="on" />
  <Parameter ID="InputFilter" Value="10" Type="USINT" />
  <Parameter ID="DigitalInputsPacked" Value="off" />
</Module>
```

These parameters are copied to a file called "DI9371.ar". The structure of this file must appear as follows:

```
<?xml version="1.0"?><!DOCTYPE AutomationRuntimeIOSystem SYSTEM "version.xml" PUBLIC "-//AutomationRuntime//DTD version.dtd" "version.dtd">
<IOCFG xmlns="http://www.br-automation.com/AR/IO" Version="2.0">
  <!-- Add Module Konfiguration here. -->
</IOCFG>
```

The content shown above is added inside node \texttt{<IOCFG>}. This completes the template file with the module configuration. The template file with the I/O mapping is created next.

"iomap.br" contains all of the information needed to pack the module's I/O mapping into the second template file. Small excerpt for the example above:

```
<LN ID="%IX.IF6.ST1.ModuleOk" Type="BOOL">
  <Prod Device="IF6.ST1" DPName="ModuleOk" Kind="io" />
</LN>

<LN ID="%ID.IF6.ST1.SerialNumber" Type="UDINT">
  <Prod Device="IF6.ST1" DPName="SerialNumber" Kind="io" />
</LN>

<LN ID="%IW.IF6.ST1.ModuleID" Type="UINT">
  <Prod Device="IF6.ST1" DPName="ModuleID" Kind="io" />
</LN>

<LN ID="%IX.IF6.ST1.DigitalInput01" Type="BOOL">
  <Prod Device="IF6.ST1" DPName="DigitalInput01" Kind="io"/>
  <Cons Device="TC#4-CPYDEV" DPName="gdiVariable" Kind="pv"/>
</LN>
```

All of the module's I/O data points must be copied to template file "DI9371.io". Structure of this file:

```
<?xml version="1.0" encoding="utf-8"?><!DOCTYPE AutomationRuntimeIOSystem SYSTEM "version.xml" PUBLIC "-//AutomationRuntime//DTD version.dtd" "version.dtd">
<IO xmlns="http://www.br-automation.com/AR/IO">
  <Links>
    <!-- Add Module IO-Information here. -->
  </Links>
</IO>
```

The content shown above is added inside node \texttt{<IO>}. This completes the template file with the I/O mapping. Once these 2 files are completed, there is a template for hardware module X20DI9371.
Module firmware in template files

To ensure that the firmware for the modules referenced in the template files is also on the controller, function "Additionally supported hardware" in Automation Studio is used. This option is located in the configuration for the controller:

4.1.5 Working with special hardware

Working with safety modules

Safety modules are handled by MpIO like gray I/O modules. The only difference to the gray I/O modules has to do with the "crosslinks", i.e. cross-references between the safety module and the SafeLOGIC controller that are stored in the hardware configuration of the safety modules.

If a safety module is moved, added or deleted using MpIO, then the crosslinks are obsolete. This can cause errors during import. Property "Remove unresolved crosslinks" can be used in the MpIO configuration to avoid these errors. If this property is set to TRUE, then all outdated crosslinks are automatically deleted during import.

MpIO cannot modify the program or functionality of the SafeLOGIC controller! If the hardware configuration is modified, it must also be ensured that the SafeLOGIC controller can handle the modified conditions.

Working with motion control (ACP10)

MpIO also makes it possible to assign axis objects to a certain drive. Section Structure of the import file describes how to connect an axis object with a drive in the import file. Which axis object is located at which hardware position can be defined in the import file. The axis object as well as the target position in the hardware tree are changed here. The NC Init parameter module and ACOPOS parameter tables are not modified. These are applied from the source (defined by SourceID or SourceName in the import file).

The image shows an NC mapping table with 6 axes. During import, axes A2 and A3 as well as A5 and A6 should be switched out. In addition, axis A4 does not exist in the configuration for the customer. The associated import file could look like this:

```xml
<IOCFG xmlns="http://www.br-automation.com/AR/IO" Version="2.0">
  <Module ID="$root" Source="AR" SourceID="$root" />
  <Module ID="IF3.ST1" Source="AR" SourceName="A1">
    <NcObject Name="gAxis01" Channel="1" />
  </Module>
  <Module ID="IF3.ST2" Source="AR" SourceName="A3">
    <NcObject Name="gAxis03" Channel="1" />
  </Module>
  <Module ID="IF3.ST3" Source="AR" SourceName="A2">
    <NcObject Name="gAxis02" Channel="1" />
  </Module>
</IOCFG>
```
• In the import file, axis A1 remains unchanged at interface address IF3.ST1, the POWERLINK interface with node number 1.

• Axes A2 and A3 exchange their positions on the POWERLINK interface. Address IF3.ST2 is specified for Module ID for axis A3, which is identified by SourceName. This changes the position of the drive in the hardware tree. The NC Init parameter module used for axis A3 and the ACOPOS parameter table are not changed.

• This is reversed for axis A2, where Module ID is defined as IF3.ST3.

• Axis A4 is omitted entirely, possibly because the end customer does not need a certain option when buying the machine.

• Axes A5 and A6 also exchange their positions on the POWERLINK interface. The difference with the exchange of A2 and A3 is that the axes are identified by SourceID instead of SourceName – in other words, the original address in the hardware configuration.

**Working with 3rd-party hardware**

3rd-party hardware is handled by MpiIO like B&R modules. If the hardware is configured using Automation Studio, then the I/O mapping and configuration of the 3rd-party hardware can be modified with the import file.

**4.1.6 General conditions**

This section lists the general conditions of mapp IO:

• Interface cards directly on the controller cannot be changed via MpiIO.
• CNC channels cannot be configured.
• Module supervision ("Module supervised") should be disabled for all modules. Otherwise, the controller starts in service mode if the hardware modules are not present in the specified slots.

**4.2 Guides**

**4.2.1 Getting started**

**4.2.1.1 Managing the hardware configuration at runtime**

This section explains how to modify the hardware configuration at runtime using MpiIOImport.

**4.2.1.1.1 Creating a project**

You must first create a new project in Automation Studio. For more information, see here.

**4.2.1.2 Hardware configuration**

The hardware with all possible usable modules was added to the project.
This allows different machine types to be implemented. At runtime, the modules needed for the desired machine type should be imported:

4.2.1.1.3 New hardware configuration

Next, the hardware configuration to be imported is defined. The goal is to create a hardware configuration for machine type A at runtime. This means that an input and output module must be imported at runtime. The import file has already been created for this purpose. The CPU from the current configuration is applied via node "Module" with ID "$root". Module X20DI9372 should then be used on slot 1 of the controller and module X20DO8232 on slot 2. "Source" defines the location of the module configuration.
4.2.1.4 Adding the mapp component

Adding the MpIO configuration

To modify the hardware configuration at runtime, the MpIO configuration is added.

It contains the mapp Link, which uniquely identifies the mapp component within the mapp environment. The mapp Link establishes the connection between programming and configuration. "Backup device name" specifies a data storage device to which backup files of the original configuration are stored during an import. The data storage device was defined earlier in the hardware configuration.
4.2.1.5 Adding a program

The next step is to add a Ladder Diagram program. For more information, see here.

4.2.1.6 Adding MpIOImport to the program

Searching for MpIO in the Ladder Diagram Catalog

MpIO can be looked for in the Ladder Diagram Catalog. All MpIO components are listed.

Selecting the function block

Function block MpIOImport is then added.
4.2.1.7 Configuring MpIOImport

Connecting MpLink
Using input "MpLink", we establish a connection to the configuration we created earlier named "gIOConfig".

Enabling the function block
The function block is enabled using input "Enable".
Defining the hardware configuration to be imported
"FileName" can be used to define the name of the file to be imported.

Defining the data storage device
"DeviceName" defines the data storage device containing the import file. The data storage device was created earlier in the hardware configuration.
4.2.1.8 Generating the file structure

Now generate a file structure for the memory card for ARsim. For more information, see here.

4.2.1.9 Testing the program

After the changes are downloaded, the program can be tested. Monitor mode is enabled. Function block MpiOImport was added to the Watch window. The import of hardware configurations must be permitted with "ImportAllowed".

The Import command can be used to import configuration "NewConfig". The hardware configuration is imported after the controller is restarted.

This can be checked using the System Diagnostics Manager. The previously imported hardware configuration is used.
4.3 Configuration

4.3.1 MpI/O configuration

"Show other advanced parameters" must be enabled in order to display the advanced parameters in the configuration:

Master configuration

Parameter "Master configuration" defines the basis used when setting up the hardware configuration:
None: If "None" is selected, the import file defined the entire hardware configuration starting with the controller.

Automation Runtime: In this case, the hardware configuration on the controller serves as the basis. Parameter "Insert Position" defines the interface to which the modules in the import file should be added (e.g. "IF6" for the X2X interface on X20CP1586).

Certain areas of the master configuration can also be ignored using parameter "Exclude". The interfaces defined here are removed from the hardware configuration during import.

Preserve PV mapping
This item ensures that variable mappings are left untouched during import.

Remove unresolved links
Only affects safety modules. If a safety module is removed during import by MpiOImport, then the links in the SafeLOGIC controller are not updated automatically, which can lead to errors. Property "Remove unresolved links" can define whether the import should look for and delete unused or invalid links for the SafeLOGIC controller.

For more information, see section Working with special hardware.

Backup device name
Parameter "Backup device name" specifies the file device to which backup files of the original configuration are saved during import. The file device should be created on the CompactFlash card or onboard flash memory.

Template device name
Parameter "Template device name" specifies the file device on which template files are looked for during import.

Plausibility checks
The plausibility checks verify whether the process variables specified in the import file are valid for the I/O mapping. For example, it is possible to check whether the data type of a process variable is suitable for the data type on a hardware module. If all plausibility checks are disabled, then whether the mappings make sense is not checked.

Allow narrowing: Defines whether mapping a smaller data type (e.g. INT) to a larger data type (e.g. DINT) is permitted. This has no effect if "Check Types = FALSE".

Check types: Defines whether data types for a mapping must match during import.

Check PVs: Defines whether the existence of the specified variables should be checked during import.

Check axis PVs: Defines whether the existence of the specified axis references should be checked during import.

Replace
"Replace" can be used to remove a certain prefix in the names of hardware modules, for example. The configuration in which the action should take place is also specified. The action defines what should be changed in this file.

"Pattern" defines the prefix to be removed, for example.

This can be used to configure the hardware configurations for different machines in AS (e.g. modules for machine 1 receive prefix "1_", modules for machine 2 receive prefix "2_"). This makes it possible to use 2 different modules with the same name but different prefixes in a project. After import, only the name of the module is displayed.

Module "1_X20DO9322" is used in this example:
Using "Replace", prefix "1_" is removed from the module name.

After import, the hardware configuration looks like this:

**I/O datapoint filters**

Item "I/O data point filters" defines which variables are permitted for I/O mapping. If only variables with prefix "di" or "ai" are permitted for "Input variables", for example, then MpIO checks whether the import file follows this rule during import. An error is output for input variables with a different prefix.

### 4.4 Use cases

#### 4.4.1 Use case 1: Selecting the hardware configuration at runtime

**Requirement**

A machine manufacturer sells a machine with 3 different expansion levels – machine types A, B and C. When buying, the customer decides which variant he needs. However, it should be possible to retrofit at any time and to change the expansion level directly via the HMI application. The application adapts to the respective expansion level.

It should be possible to decide during runtime which machine type should be used. This means that depending on which machine was purchased, the corresponding hardware configuration can be set.

The machine builder only wants to maintain one project with one hardware setup.
Solution

Basic idea
- A project with a configuration that contains all the hardware for all machine types.
- Import of respective expansion level via XML file.
- Adjustment of the application using library SYS_Lib.

Component list
- **MpIOImport** (own MpLink): Manages the import of the new hardware configuration.
- **MpIOImportUI** (MpLink from MpIOImport): Contains all necessary parameters and structures for establishing a connection to an HMI application and importing a hardware configuration.

Connection diagram

Configuration
A "backup device name" is specified in the configuration. The original files of the configuration are stored on this data storage device, which is defined in the configuration of the target system. For more information about configuration, see MpIO configuration.
Using the mapp components

Function block **MpIOImport** is added to manage the hardware configuration.

**MpIOImport** is then configured. This requires "FileDevice", "FileName" as well as "ImportAllowed" and "Import".

Function block **MpIOImportUI** is added and configured to change the hardware configuration at runtime via the HMI application.

Depending on which machine type is used, the application is adjusted using library SYS_Lib.

In order to be able to import the respective hardware configuration with mapp IO, all modules required for the different machine types are added in Automation Studio. For more information, see [here](#).

It is important to note that "ModuleSupervised" must be set to "off"! Otherwise, the controller starts in service mode if the hardware modules are not present in the specified slots.
The import file for machine type A could look like this:

```xml
<IOCFG xmlns="http://www.br-automation.com/AR/IO" Version="2.0">

  <Module ID="$root" Source="AR" SourceID="$root" />

  <Module ID="IF6.ST1" Source="AR" SourceName="A_AnalogInputModule">
    <Parameter ID="Supervision" Value="on" />
    <LN ID="AnalogInput01" DPName="Program:aiInput1" />
    <LN ID="AnalogInput02" DPName="Program:aiInput2" />
  </Module>

  <Module ID="IF6.ST2" Source="AR" SourceName="A_DigitalOutputModule">
    <Parameter ID="Supervision" Value="on" />
    <LN ID="DigitalOutput01" DPName="Program:doOutput1" />
    <LN ID="DigitalOutput02" DPName="Program:doOutput2" />
  </Module>

  <Module ID="IF6.ST3" Source="AR" SourceName="A_SafeOutputModule">
    <Parameter ID="Supervision" Value="on" />
  </Module>

  <Module ID="IF6.ST4" Source="AR" SourceName="AB_DigitalInputModule">
    <Parameter ID="Supervision" Value="on" />
  </Module>

</IOCFG>
```

For more information about the structure of an import file, see [here](#).

This import file is transferred to the controller in data storage device "HD":

This file is imported using MpIOMap. As soon as the import is completed, all modules must be present on the specified slots since "ModuleSupervised" is re-enabled using the import file. For example, the final hardware configuration for machine type A could look like this:
4.5 Libraries

4.5.1 Data types and enumerators

4.5.1.1 Data types

4.5.1.1.1 MpIODiagType
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>MpIOStatusIDType</td>
<td>StatusID diagnostic structure.</td>
</tr>
</tbody>
</table>

4.5.1.1.2 MpIOImportInfoType
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>ImportFileActive</td>
<td>BOOL</td>
<td>The configuration has been imported. The hardware configuration has been changed.</td>
</tr>
<tr>
<td>Diag</td>
<td>MpIODiagType</td>
<td>Diagnostic structure for the status ID.</td>
</tr>
</tbody>
</table>

4.5.1.1.3 MpIOImportUIConnectType
This data type creates a connection between component MpIOImportUI and the VC4-based HMI application.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultLayerStatus</td>
<td>UINT</td>
<td>Status data point for the default layer of the page where the list of possible import files is being displayed VC4 connection: StatusDatapoint from Layer</td>
</tr>
<tr>
<td>Status</td>
<td>MpIOImportUIStatusEnum</td>
<td>Current operation</td>
</tr>
<tr>
<td>FileList</td>
<td>MpIOUIFileListType</td>
<td>Contains a list of all files available for import as well as list navigation</td>
</tr>
<tr>
<td>Message box</td>
<td>MpIOUIMessageBoxType</td>
<td>Controls dialog boxes.</td>
</tr>
<tr>
<td>Import</td>
<td>BOOL</td>
<td>Command for starting an import</td>
</tr>
</tbody>
</table>

4.5.1.1.4 MpIOImportUISetupType
Additional configuration options to define the HMI application.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileListSize</td>
<td>UINT</td>
<td>Number of files to be displayed on one page of the list in the HMI application. Up to 20 files can be displayed.</td>
</tr>
<tr>
<td>FileListScrollWindow</td>
<td>USINT</td>
<td>Indicates how many entries from the list are initially displayed when scrolling up and down.</td>
</tr>
</tbody>
</table>

4.5.1.1.5 MpIOInfoType
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>MpIODiagType</td>
<td>Diagnostic structure for the status ID.</td>
</tr>
</tbody>
</table>
4.5.1.1.6 MpIOTstatusIDType

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>MpI0ErrorEnum</td>
<td>Error code for the function block.</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 documentation.</td>
</tr>
</tbody>
</table>

4.5.1.1.7 MpIOUIFileListType

This data type contains a list of files and associated navigation.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Names</td>
<td>ARRAY[0..19] of STRING[100]</td>
<td>List of all files available for import.</td>
</tr>
<tr>
<td>Scroll</td>
<td>MpIOUIScrollSelectType</td>
<td>Contains all data points used in the HMI application for scrolling the list.</td>
</tr>
<tr>
<td>Refresh</td>
<td>BOOL</td>
<td>Command to reload the list.</td>
</tr>
</tbody>
</table>

4.5.1.1.8 MpIOUIMessageBoxType

Data type used to control the display of a dialog box.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LayerStatus</td>
<td>UINT</td>
<td>Visibility of the dialog box (status data point for the dialog box layer)</td>
</tr>
<tr>
<td>Type</td>
<td>MpIOUIMessageEnum</td>
<td>Type of dialog box.</td>
</tr>
<tr>
<td>StatusID</td>
<td>DINT</td>
<td>Current status ID for display.</td>
</tr>
<tr>
<td>ErrorNumber</td>
<td>UINT</td>
<td>Current error number to be displayed (corresponds to the contents of “StatusID”)</td>
</tr>
<tr>
<td>Confirm</td>
<td>BOOL</td>
<td>Confirms the operation.</td>
</tr>
<tr>
<td>Cancel</td>
<td>BOOL</td>
<td>Cancels the operation.</td>
</tr>
</tbody>
</table>

4.5.1.1.9 MpIOUIScrollSelectType

This data type contains all of the data points necessary to scroll a list.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxSelection</td>
<td>UINT</td>
<td>Index of the last entry in the list.</td>
</tr>
<tr>
<td>SelectedIndex</td>
<td>UINT</td>
<td>Index of the entry currently selected in the list.</td>
</tr>
<tr>
<td>PageUp</td>
<td>BOOL</td>
<td>Jumps to the start of the current page and then scrolls up one page at a time. The size of the page is defined using parameter “FileListSize” of structure MpIOImportUISetupType.</td>
</tr>
<tr>
<td>StepUp</td>
<td>BOOL</td>
<td>Selects the previous entry in the list.</td>
</tr>
<tr>
<td>StepDown</td>
<td>BOOL</td>
<td>Selects the next entry in the list.</td>
</tr>
</tbody>
</table>
4.5.1.2 Enumerators

4.5.1.2.1 MpiOUIMessageEnum

This enumerated data type defines the messages that can be displayed in a dialog box.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpIO_UI_MSG_NONE</td>
<td>The dialog box is not displayed.</td>
</tr>
<tr>
<td>mpIO_UI_MSG_ERROR</td>
<td>The dialog box displays an error.</td>
</tr>
<tr>
<td>mpIO_UI_MSG_CONFIRM_IMPORT</td>
<td>The dialog box displays a confirmation dialog box for the import.</td>
</tr>
<tr>
<td>mpIO_UI_MSG_CONFIRM_REBOOT</td>
<td>The dialog box displays a confirmation dialog box for the import with subsequent restart.</td>
</tr>
</tbody>
</table>

4.5.1.2.2 MpiOImportUIStatusEnum

This enumerated data type provides information about the current activity of a component.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpIO_IMPORTUI_STATUS_IDLE</td>
<td>No process is currently active.</td>
</tr>
<tr>
<td>mpIO_IMPORTUI_STATUS_REFRESH</td>
<td>The file list is reloaded.</td>
</tr>
<tr>
<td>mpIO_IMPORTUI_STATUS_BUILD</td>
<td>The import data is created by the import file.</td>
</tr>
<tr>
<td>mpIO_IMPORTUI_STATUS_DIALOG</td>
<td>A dialog box is active. Waiting for user input.</td>
</tr>
<tr>
<td>mpIO_IMPORTUI_STATUS_DISPOSE</td>
<td>Import data is discarded (after cancelation by the user).</td>
</tr>
<tr>
<td>mpIO_IMPORTUI_STATUS_IMPORT</td>
<td>The hardware configuration is being imported.</td>
</tr>
<tr>
<td>mpIO_IMPORTUI_STATUS_ERROR</td>
<td>An error occurred during the last operation.</td>
</tr>
</tbody>
</table>

4.5.2 Function blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MpiImport</td>
<td>This function block imports a new hardware configuration.</td>
</tr>
<tr>
<td>MpiImportUI</td>
<td>This function block provides a user interface for the import.</td>
</tr>
</tbody>
</table>

4.5.2.1 MpiImport

This function block imports a new hardware configuration.

Our B&R online tutorial platform includes tutorials about mapp IO.

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 &amp;MpComIdentType</td>
<td>Connection to mapp (MpLink of an MpIO 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>ImportAllowed</td>
<td>BOOL</td>
<td>Defines whether an import is permitted.</td>
</tr>
<tr>
<td>IN</td>
<td>FileName</td>
<td>Pointer to STRING[255]</td>
<td>Name of the file to be imported.</td>
</tr>
<tr>
<td>IN</td>
<td>DeviceName</td>
<td>Pointer to STRING[50]</td>
<td>Name of the file device containing the file.</td>
</tr>
<tr>
<td>IN</td>
<td>Import</td>
<td>BOOL</td>
<td>Command for importing the configuration.</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 finished.</td>
</tr>
<tr>
<td>OUT</td>
<td>Info</td>
<td>MpIOImportInfoType</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 be carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

4.5.2.1.1 Description

This function block requires the MpLink of an MpIO configuration.

Function block MpIOImport manages the import of hardware configurations. In addition to modified hardware configurations, the NC mapping or I/O mapping can be imported via a new configuration.

The hardware configuration defined by input "FileName" is imported using input "Import". Input "ImportAllowed" can be used as an additional security mechanism to prevent a configuration from being imported at the wrong time.

During the upload itself, it is possible to check whether the hardware configuration has been modified and to respond accordingly. For details, see section Restoring the hardware configuration.
Import file

The import file is compatible with the arconfig, iomap and NC mapping formats used by AR. This makes it possible to import the data generated by Automation Studio.

In addition, existing data from the original AR configuration or template files can be referenced. During import, all of these files are combined to create the hardware configuration.

For more information about the structure of the import file, see here.

Plausibility checks during import

During import, the mapping between process variable and hardware module can be checked for certain characteristics. Specifically, the following can be checked:

- Whether the specified process variables exist on the controller.
- Whether the data types of the process variables match the data types of the inputs/outputs.
- Whether a PV was connected to multiple inputs.
- Whether a module specified in the import file does not exist.
- Whether a module was added to a non-existing position or a position that is already occupied.

If one of these checks fails, the data is still imported as much as possible and an error is output on the function block.

If the function block is called during a project download, some PVs may not yet exist on the controller. The function block will indicate an error in this case.

For more information, see section MpIO configuration.

4.5.2.2 MpIOImportUI

This function block provides a user interface for the import.

Function block

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN MpLink</td>
<td>Pointer to MpComIdentType</td>
</tr>
<tr>
<td>IN Enable</td>
<td>BOOL</td>
</tr>
<tr>
<td>IN ErrorReset</td>
<td>BOOL</td>
</tr>
<tr>
<td>IN UISetup</td>
<td>MpIOImportUISetupType</td>
</tr>
<tr>
<td>IN UIConnect</td>
<td>Pointer to MpIOImportUIConnectType</td>
</tr>
<tr>
<td>OUT Active</td>
<td>BOOL</td>
</tr>
<tr>
<td>OUT Error</td>
<td>BOOL</td>
</tr>
<tr>
<td>OUT StatusID</td>
<td>DINT</td>
</tr>
<tr>
<td>OUT Info</td>
<td>MpIOInfoType</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).
4.5.2.2.1 Description
The MpLink for this function block is the same that is used for function block MpiOImport. The component represents a connection between MpiOImport and an HMI application. Component MpiOImport must be active to use MpiOImportUI.

MpiOImportUIConnectType
Structure "UIConnect" is divided into the following areas:
- **DefaultLayerStatus**: Indicates whether the layer with the list of possible import files is displayed.
- **Status**: Current operation.
- **FileList**: Contains a list of all files available for import.
- **MessageBox**: The different dialog boxes for MpiO can be configured in structure MpiOUIMessageBoxType.
- **Import**: This command starts an import.

MpiOImportIOSetupType
"FileListSize" defines in this structure how many import files should be displayed on one page of the HMI application. "FileListScrollWindow" determines how many entries from the list are displayed in advance when scrolling up and down.

4.5.3 Status numbers
4.5.3.1 -1064120312: Import blocked

**Description:**
Command "Import" is blocked.

**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 "ImportAllowed" on MpiOImport.

**Constant**
mpIO_ERR_IMPORT_NOT_ALLOWED

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

4.5.3.2 -1064120314: MpiOImport not yet active

**Description:**
Function block MpiOImport is not active.

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

**Cause/Solution:**
- Enable function block MpiOImport.

**Constant**
mpIO_ERR_MISSING_IMPORT
These function blocks / functions can report this error:

- MpIOImportUI

4.5.3.3 -1064120315: Missing value on UIConnect

**Description:**
NULL was appended to "UIConnect".

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

**Cause/Solution:**
- Input "UIConnect" forgotten

**Constant**
mpIO_ERR_MISSING_UICONNECT

These function blocks / functions can report this error:

- MpIOImportUI

4.5.3.4 -1064120316: Error creating backup files

**Description:**
An error occurred while creating the backup files. 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 "Backup device name" as specified in the MpIO configuration.

**Constant**
mpIO_ERR_CREATE_BACKUP

These function blocks / functions can report this error:

- MpIOImport

4.5.3.5 -1064120317: Error enabling imported hardware configuration

**Description:**
An error occurred while enabling the import on the controller. 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 data specified in the import file is valid.

**Constant**
mpIO_ERR_ACTIVATE_IMPORT_DATA
These function blocks / functions can report this error:

- **MpIOImport**

4.5.3.6 -1064120318: Error creating import data

**Description:**
An error occurred while creating the import data. 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 import file.

**Constant**
mpIO_ERR_BUILD_IMPORT_DATA

These function blocks / functions can report this error:

- **MpIOImport**

4.5.3.7 -1064120319: Invalid filename

**Description:**
The filename on input "FileName" is invalid. 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:**
- Nothing is attached to input "FileName".
- The attached file does not exist.
- No access rights

**Constant**
mpIO_ERR_INVALID_FILE_NAME

These function blocks / functions can report this error:

- **MpIOImport**

4.5.3.8 -1064120320: Invalid file device

**Description:**
The specified file device on input "DeviceName" is invalid. 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:**
- Nothing is attached to input "DeviceName".
- The attached file device does not exist.
- A connection to the desired file device could not be established.
Constant
mpIO_ERR_INVALID_FILE_DEV

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

4.5.3.9 -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:
- MpIOImport

4.5.3.10 -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:
- MpIOImport
  - MpIOImportUI

4.5.3.11 -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:

- **MpiOImport**
- **MpiOImportUI**

### 4.5.3.12 -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:

- **MpiOImport**
- **MpiOImportUI**

### 4.5.3.13 -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:

- **MpiOImport**
- **MpiOImportUI**

### 4.5.3.14 -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:

- **MpiOImport**
- **MpiOImportUI**
4.5.3.15 -1064293112: Invalid import mode

**Description:**
An invalid mode was defined for the import on {2:ID}.

**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 mode is invalid for this configuration.

**Additional information:**
- {2:ID}: ID of the affected "Module" or "LN" node in the import file.

**Constant**
mpIO_ERR_INVALID_MODE

4.5.3.16 -1064293113: Error copying module data

**Description:**
An error occurred while copying import data to module {2:ModuleID}.

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

**Additional information:**
- {2:ModuleID}: ID of the affected module.

**Constant**
mpIO_ERR_COPY_MODULE_DATA

4.5.3.17 -1064293114: NC mapping missing

**Description:**
The specified NC mapping is missing in the master configuration. It is not possible to create an NC mapping on channel {2:ChannelNumber}.

**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 channel exists on the module.

**Additional information:**
- {2:ChannelNumber}: Number of the affected channel on the module.

**Constant**
mpIO_ERR_NCMAPPING_MISSING
4.5.3.18 -1064293115: Invalid interface

Description:
Module {2:Module} was connected to an invalid interface.

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 module can be connected to the interface.

Additional information:
• {2:Module}: ID of the affected module.

Constant
mpIO_ERR_INVALID_INTERFACE

4.5.3.19 -1064293116: Invalid axis reference

Description:
Invalid axis reference {2:ID} was defined in an NC mapping. 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 axis reference in the import file is correct.

Additional information:
• {1:ErrorNumber}: Reason for the error.
• {2:ID}: ID of the affected axis reference.

Constant
mpIO_ERR_NC_MAPPING

4.5.3.20 -1064293118: Unresolved crosslinks found

Description:
Unresolved crosslinks were found in the import file.

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

Cause/Solution:
• Enable configuration parameter "Remove unresolved links".

Constant
mpIO_ERR_UNRESOLVED_LINKS
4.5.3.21 -1064293119: Error changing configuration

Description:
An error occurred while changing configuration {2:AR|IO|NC}.

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 configuration parameter can be changed.

Additional information:
- {1:ErrorNumber}: Reason for the error.
- {2:AR|IO|NC}: Indicates the configuration on which the error occurred.

Constant
mpIO_ERR_CHANGE_CONFIG

4.5.3.22 -1064293120: Multiple variable connections

Description:
PV {2:PvName} was connected to multiple inputs.

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

Cause/Solution:
- A PV is not permitted to be connected to multiple inputs.

Additional information:
- {2:PvName}: Name of the affected process variable.

Constant
mpIO_ERR_MULTIPLE_MAPPING

4.5.3.23 -1064293121: Invalid module

Description:
The module specified in the import file does not exist in the master configuration.

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 module exists in the master configuration.
- Check whether the module is specified correctly in the import file.

Constant
mpIO_ERR_MODULE_INVALID
4.5.3.24 -1064293122: Error copying import file

Description:
An error occurred while copying import file {2:FileName} to the backup data storage device.

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 there is enough memory.
- Check whether access to the data storage device is permitted/possible.

Additional information:
- {1:ErrorNumber}: Reason for the error.
- {2:FileName}: Name of the affected file.

Constant
mpIO_ERR_COPY_IMPORT_FILE

4.5.3.25 -1064293123: Node not defined correctly

Description:
A node in the import file is defined incorrectly.

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 import file.

Additional information:
- {1:ErrorNumber}: Reason for the error.
- {2:ID}: ID of the affected "Module" node.

Constant
mpIO_ERR_LINK_NODE

4.5.3.26 -1064293124: Unable to link PV

Description:
Specified PV {2:PvName} cannot be linked since it does not contain a prefix specified as a filter in the configuration.

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 filter is configured correctly.
- Check whether the name of the PV is correct.
Additional information:

- `{2:PvName}`: Name of the affected process variable.

Constant

mpIO_ERR_PV_FILTER

4.5.3.27 -1064293125: PV not found

Description:
PV `{2:PvName}` specified for the I/O mapping cannot be found on the controller.

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

Cause/Solution:

- If an import is started in the initialization subroutine of a task, it is possible that some PVs do not yet exist.
- Check whether the specified PV exists on the controller.

Additional information:

- `{2:PvName}`: Specified name of the affected process variable.

Constant

mpIO_ERR_PV_MISSING

4.5.3.28 -1064293126: Invalid data type

Description:
Data type `{3:PvType}` of process variable `{2:PvName}` does not match the data type of I/O data point `{4:IoType}`.

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 data type of the PV matches that of the I/O data point.

Additional information:

- `{2:PvName}`: Name of the affected process variable.
- `{3:PvType}`: Data type of the affected process variable.
- `{4:IoType}`: Data type of the I/O data point.

Constant

mpIO_ERR_PV_TYPE_MISMATCH

4.5.3.29 -1064293127: Unknown data type

Description:
Data type `{3:Type}` specified in the import file for process variable `{2:PvName}` was not recognized.

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 data type is known.

Additional information:
- {2:PvName}: Name of the affected process variable.
- {3:Type}: Unrecognized data type.

Constant
mpIO_ERR_PV_TYPE_UNKNOWN

4.5.3.30 -1064293128: Invalid file contents

Description:
The contents of import file {2:FileName} on data storage device {3:DeviceName} are invalid. The nodes required for import could not be found.

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 structure of the import file corresponds to the required structure.

Additional information:
- {2:FileName}: Name of the affected file.
- {3:DeviceName}: Name of the data storage device containing the file.

Constant
mpIO_ERR_INVALID_FILE_FORMAT

4.5.3.31 -1064293129: Data point missing

Description:
The specified data point {2:DatapointID} is missing for the module. It is not possible to create an I/O mapping on this data point.

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

Cause/Solution:
- Make sure that the specified data point exists for the module.

Additional information:
- {2:DatapointID}: ID of the desired data point.

Constant
mpIO_ERR_IOMAPPING_MISSING

4.5.3.32 -1064293130: Import conflict

Description:
A conflict occurred while importing data {2:ImportSource}. Properties that cannot be modified cannot be overwritten.
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 configuration properties that cannot be modified have are being overwritten.

Additional information:

• {2:ImportSource}: Indicates whether the NC mapping table or I/O mapping table is affected.
• {3:NodeID}: Node in the import file affected by the error.
• {4:AffectedSubNode}: Affected child node.

Constant

mpIO_ERR_IMPORT_CONFLICT

4.5.3.33 -1064293131: Error linking module

Description:
An error occurred while linking module {2:ModuleID} from the master configuration with {3:ModuleSourceID} {4:SourceID}.

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 module exists in the master configuration.

Additional information:

• {2:ModuleID}: ID of the affected module.
• {3:ModuleSourceID}: Specified "SourceName" of the module being linked.
• {4:SourceID}: Specified "SourceID" of the module being linked.

Constant

mpIO_ERR_LINK_SOURCE

4.5.3.34 -1064293132: Module already exists

Description:
Specified module {2:ModuleID} has already been linked and cannot be used twice.

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

Cause/Solution:

• Remove one of the module's links.
• Change the master configuration to "None".

Additional information:

• {2:ModuleID}: ID of the affected module.
Constant
mpIO_ERR_MODULE_EXISTS

4.5.3.35 -1064293133: SourceID missing

Description:
"SourceID" was not specified for the module being added {2:ModuleID}.

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 ID was specified.

Additional information:
• {2:ModuleID}: ID of the module being added.

Constant
mpIO_ERR_SOURCE_ID_MISSING

4.5.3.36 -1064293134: Module not found

Description:
Module {2:ModuleID} could not be found in specified source {3:SourceID}.

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 module exists in the original AR configuration.
• Check whether template files exist for the module.

Additional information:
• {1:ErrorNumber}: Reason for the error.
• {2:ModuleID}: Module being added.
• {3:SourceID}: Specified position of the module in the original AR configuration.

Constant
mpIO_ERR_MODULE_NOT_FOUND_IN_SOURCE

4.5.3.37 -1064293135: Invalid import source

Description:
The specified "SourceID" {2:SourceID} could not be found in the AR configuration.

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 "AR" or "Template" was specified as "Source".
mapp Services

Additional information:
- {2:SourceID}: Position of the module being imported.
- {3:ModuleID}: Module ID of the module being imported.
- {4:Offset}: Position in the import file at which the error occurred.

Constant
mpIO_ERR_IMPORT_SOURCE_INVALID

4.5.3.38 -1064293136: Attribute "ID" missing

Description:
"ID" was not defined for a module.

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

Cause/Solution:
- Define an ID for the module.

Additional information:
- {2:DebugOffset}: Position at which the error occurred.

Constant
mpIO_ERR_IMPORT_ID_MISSING

4.5.3.39 -1064293137: AR module not found

Description:
Module {2:ModuleName} specified in the import file cannot be found in the AR configuration.

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

Cause/Solution:
- Add the module in the AR configuration.
- Use template files.

Additional information:
- {2:ModuleName}: Name of the affected module.

Constant
mpIO_ERR_MODULE_NOT_FOUND

4.5.3.40 -1064293138: Error parsing import file

Description:
The following error occurred while parsing the XML data from import file {2:FileName} on data storage device {3:DeviceName} at position {5:OffsetInFile}: {4:ErrorDescription}.

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

- Check that the syntax of the XML data is correct.

Additional information:

- {2:FileName}: Name of the affected import file.
- {3:DeviceName}: Data storage device containing the import file.
- {4:ErrorDescription}: Description of the error in the import file.
- {5:OffsetInFile}: Indicates the position in the import file at which the error occurred.

Constant

mpIO_ERR_PARSE_XML_FILE

4.5.3.41 -1064293139: Error reading file

Description:
An error occurred while reading the import file {2:FileName} on data storage device {3:DeviceName}.

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 file is permitted.

Additional information:

- {1:ErrorCause}: Reason for the error.
- {2:FileName}: Name of the affected import file.
- {3:DeviceName}: Data storage device containing the import file.

Constant

mpIO_ERR_READING_FILE

4.5.3.42 -1064293140: Import file not found

Description:
The import file (.ar/.xml) could not be found.

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 import file exists.

Constant

mpIO_ERR_IMPORT_FILE_NOT_FOUND

4.5.3.43 1083190531: Restart required

Description:
The controller must be restarted to enable the hardware configuration.

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

• A restart is required to enable the hardware configuration.

Constant

(mpIO_INF_REBOOT_NECESSARY)

4.5.3.44 1083363335: Waiting on import function block

Description:
The function block can only become active if MpIOImport is active. See the Logger for additional information.

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

Cause/Solution:

• Enable MpIOImport.

Constant

(mpIO_INF_WAIT_IMPORT_FB)

These function blocks / functions can report this error:

• MpIOImportUI

4.5.4 Alarms

4.5.4.1 mpIO_ALM_IMPORT_FAILED: Import of hardware configuration failed

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

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

Cause/Solution:

• The contents of the import file are invalid.

4.5.4.2 mpIO_ALM_IMPORT_DONE: Import successful

Description:
The import of the new hardware configuration was successful.

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

Cause/Solution:

• A hardware configuration was successfully imported.

4.5.4.3 mpIO_ALM_CONFIG_CHANGED: Hardware configuration modified

Description:
The hardware configuration was modified (externally). See the Logger for additional information.

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

- A different hardware configuration was downloaded using Automation Studio.
- A different hardware configuration was downloaded using Runtime Utility Center.
- The hardware configuration was manipulated using libraries such as AsIOMMan or AsIOAcc.

5 mapp J1939: SAE J1939 network protocol

mapp J1939 follows the SAE J1939 network protocol. SAE J1939 describes the communication on a CAN bus system in commercial vehicles used to transmit diagnostic data (e.g. engine speed, temperature) and control information.

The component makes it possible to send and receive data via the network protocol. For more detailed information about the SAE J1939 network protocol, see here:

- SAE J1939 at Wikipedia

The component cannot be used in simulation.

5.1 Concept

User stories

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OEM</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

5.1.1 Transmitting and receiving information

Using the mapp component, data can be transmitted and received via a CAN interface that uses the J1939 network protocol. This can be diagnostic data or control information, for example. For information about how the J1939 CAN message is structured, see here. See this link for how the network protocol J1939 is structured and works:

- SAE J1939 at Wikipedia

Data is transmitted and received using function block MpJ1939Generic.

Which information is transferred is determined by the "parameter group number" (PGN). The PGNs to be transmitted or received are defined in the MpJ1939Generic configuration. Some of the PGNs are available in the configuration. Some PGNs can be queried cyclically; others are available on request.

PGNs that are defined by the configuration

For each PGN to be used, a process variable of a certain data type must be created. This process variable contains the desired information to be transmitted or received. For more information, see Working with PGNs in the configuration.
PGNs that are not defined by the configuration

Due to the variety of possible PGNs, not all are directly available in the configuration. For those PGNs that are not found in the configuration, function blocks \texttt{MpJ1939Receive}, \texttt{MpJ1939Transmit} or \texttt{MpJ1939SpecificRequest} can be used.

Function blocks \texttt{MpJ1939Receive} and \texttt{MpJ1939Transmit} can be used for PGNs that are read/transmitted cyclically. For those PGNs that can only be queried once, function block \texttt{MpJ1939SpecificRequest} is used. Whether a PGN is queried cyclically or once can be taken from the J1939 standard. For more information, see here.

The desired PGN must be specified on the function block. In this case, the user is responsible for managing the data. For more information, see here.

5.1.2 Structure of the J1939 CAN message

The J1939 CAN message is structured as follows:

<table>
<thead>
<tr>
<th>SAE J1939 CAN-Message</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>29 bit CAN-Identifier</strong></td>
</tr>
<tr>
<td>28...26</td>
</tr>
<tr>
<td><strong>Priority</strong></td>
</tr>
</tbody>
</table>

The CAN identifier consists of 3 parts: the priority, parameter group number (PGN) and source address. For more information about the PGN, see here. The priority of a PGN can be taken from the J1939 standard. For more information, see SAE J1939 at Wikipedia.
Resolving a CAN identifier

A CAN identifier can be 18FF72E0, for example. To find out which PGN and priority the CAN identifier has, proceed as follows:

1. Convert

The CAN identifier is specified in hex format. To find out the PGN number and priority, the hex format must be converted to binary format.

18FF72E0 in binary is 0001 1000 1111 1111 0111 0010 1110 0000.

2. Determine the PGN and priority

From the binary number, the PGN, priority and source address can be determined again in hex format.

- **Priority**: 6
- **PGN**: FF72
- **Source address**: E0

5.1.3 Timing

PGNs must be received or transmitted in a specific cycle time so that information can be transmitted or received.

For example, PGN DM01 must be received or transmitted at least every 1000 ms. If the PGN is received in a slower cycle time such as 3000 ms, the PGN is received, but the status of the PGN indicates that the PGN was received in the wrong cycle time (mpJ1939_PGN_ERROR_CYCLE_TIME).

If PGN DM01 is queried in a faster cycle time such as 10 ms, this is no problem. No error occurs during reception. PGN DM01 is nevertheless only received/transmitted in 1000 ms.

If PGN DM01 is received via function block MpJ1939Generic, this function block must be in a task class ≤1000 ms.

The status of the received/transmitted PGN can be read from parameter Status of the provided variable.

If different PGNs are to be received/transmitted, they can be called in a task class that contains the fastest cycle time for a PGN. If PGNs TSC1, DM1 and CCVS1 are used, all PGNs necessary for PGN TSC1 can be queried with a cycle time of 10 ms. It is important here that PGN DM1 is nevertheless only automatically received/transmitted every 1000 ms, for example, since this is the cycle time of the PGN.

The following cycle times must be observed for PGNs that can be defined in the MpJ1939Generic configuration:
### PGN Services

<table>
<thead>
<tr>
<th>PGN</th>
<th>Cycle time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSC1</td>
<td>10</td>
</tr>
<tr>
<td>DM1</td>
<td>1000</td>
</tr>
<tr>
<td>AMB</td>
<td>100</td>
</tr>
<tr>
<td>CCVS1</td>
<td>1000</td>
</tr>
<tr>
<td>EC1</td>
<td>5000</td>
</tr>
<tr>
<td>EEC1</td>
<td>20</td>
</tr>
<tr>
<td>EEC2</td>
<td>50</td>
</tr>
<tr>
<td>EEC3</td>
<td>250</td>
</tr>
<tr>
<td>EFL_P1</td>
<td>500</td>
</tr>
<tr>
<td>ET1</td>
<td>1000</td>
</tr>
<tr>
<td>IC1</td>
<td>500</td>
</tr>
<tr>
<td>LFE</td>
<td>100</td>
</tr>
<tr>
<td>SHUTDN</td>
<td>1000</td>
</tr>
<tr>
<td>VEP1</td>
<td>1000</td>
</tr>
<tr>
<td>CI</td>
<td>PGN is requested in a specified cycle.</td>
</tr>
<tr>
<td>LFC</td>
<td>PGN is requested in a specified cycle.</td>
</tr>
<tr>
<td>SOFT</td>
<td>PGN is requested in a specified cycle.</td>
</tr>
<tr>
<td>CM1</td>
<td>1000</td>
</tr>
<tr>
<td>AT1IG1</td>
<td>50</td>
</tr>
<tr>
<td>AT1IG2</td>
<td>500</td>
</tr>
<tr>
<td>AT1IMG</td>
<td>500</td>
</tr>
<tr>
<td>AT1OG1</td>
<td>50</td>
</tr>
<tr>
<td>AT1OG2</td>
<td>500</td>
</tr>
<tr>
<td>AT1TI</td>
<td>1000</td>
</tr>
<tr>
<td>DPFC1</td>
<td>1000</td>
</tr>
<tr>
<td>AT1S1</td>
<td>1000</td>
</tr>
<tr>
<td>DM01</td>
<td>1000</td>
</tr>
</tbody>
</table>

For PGNs that are not provided in the MpJ1939Generic configuration, information regarding cycle time can be taken from the J1939 standard. For more information, see [here](#).

#### 5.1.4 Memory management

A PGN that is not available via the configuration can be transmitted or received using function blocks MpJ1939Receive, MpJ1939Transmit and MpJ1939SpecificRequest.

To work with PGNs that are not available via the configuration, it is recommended to take the necessary information from the J1939 standard. For more information, see [here](#).

The desired PGN is defined via input parameter "PGN". The numeric code of the PGN is specified. Which numerical value a PGN identifies can be taken from the J1939 standard. There are two ways to manage the information of a PGN.

- Managing a PGN using a PV
- Managing PGN in memory

Depending on how the signals are in the PGN, a decision can be made between the two variants. The signals of a PGN always have a start bit and maximum length.

For example, if you have a PGN containing signals "Test_A" and "Test_B", these signals have a start bit and defined length.

For variant Managing a PGN using a PV, the signals must follow each other directly. There are no bits between the signals:

<table>
<thead>
<tr>
<th>Signal</th>
<th>Start bit</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test_A</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Test_B</td>
<td>8</td>
<td>16</td>
</tr>
</tbody>
</table>

For variant Managing PGN in memory, the signals do not follow each other directly. In the example below, signal "Test_A" has a length of 8 bits. Signal "Test_B" only starts at bit position 16, however.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Start bit</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test_A</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>Test_B</td>
<td>16</td>
<td>8</td>
</tr>
</tbody>
</table>

#### 5.1.4.1 Managing a PGN using a PV

This section explains how to manage the information of a PGN using a PV.

**Example**
For example, the numeric value of PGN DTBI1 is 351342334. PGN DTBI1 can return or transmit the following signals:

![Diagram of DTBI1 signals]

These signals have a defined length. For example, signal "BodyRollAngle" has a length of 16 bits.

Input parameter "Data" of the function blocks must be used to provide a buffer that can receive and display the values of PGN DTBI1.

To receive PGN DTBI1, a structure must be created that contains the exact length of the signals. It must be known exactly how PGN DTBI1 is structured. Each signal has a start bit and a fixed bit length. This means it is exactly defined from which bit the signal starts and from which bit it ends. This looks as follows for PGN DTBI1:

<table>
<thead>
<tr>
<th>Signal</th>
<th>Start bit</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>BodyRollAngle</td>
<td>0</td>
<td>16-bit</td>
</tr>
<tr>
<td>BodyRaiseAngle</td>
<td>16</td>
<td>16-bit</td>
</tr>
<tr>
<td>BodyRaisePosRelToRecommendMax</td>
<td>32</td>
<td>16-bit</td>
</tr>
<tr>
<td>BodyEjectorBladePos</td>
<td>48</td>
<td>8-bit</td>
</tr>
</tbody>
</table>

In order to obtain the information for this PGN, a structure must be created that has exactly the same bit length. It can look like this, for example:

Variable "My_DTBI1" was created that is of user-defined data type "My_DTBI1_Type". This data type contains variables that correspond to the same bit length of PGN DTBI1.

The alignment of the user-defined data type must be strictly observed. It must correspond exactly to the desired PGN.

The user-defined data type is not permitted to contain any stuff bytes!

### 5.1.4.2 Managing PGN in memory

This section explains how to manage the information of a PGN in memory.

**Example**

PGN CCVS3 has numerical value 419224830. PGN CCVS3 can return or transmit the following signals:
These signals have a defined length. For example, signal "CruiseCtrlSpeed" has a length of 16 bits.

A buffer must be provided via input parameter "Data" of function blocks MpJ1939Receive, MpJ1939SpecificRequest and MpJ1939Transmit that can receive and display the values of PGN CCVS3.

To receive PGN CCVS3, a variable would have to be created that can receive the information from the PGN. It must be known exactly how PGN CCVS3 is structured. Each signal has a start bit and a fixed bit length. This means it is exactly defined from which bit the signal starts and from which bit it ends. The signals of the PGN do not all follow each other directly. Some signals have unused bits in between. For PGN CCVS3, this is what it looks like:

<table>
<thead>
<tr>
<th>Signal</th>
<th>Start bit</th>
<th>Length [Bit]</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaptiveCruiseCtrlReadinessStatus</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>CruiseCtrlSystemCmdStatus</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>PrdctvCrsCtrlStSpeedOffsetStatus</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>SrcAddrssOfCntrlngDvcFrDsblngCr</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>SrcAddrssOfCntrlngDvcFrPnsngCrsC</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>AEBSReadinessState</td>
<td>24</td>
<td>2</td>
</tr>
<tr>
<td>DriverCancellationStatus</td>
<td>26</td>
<td>2</td>
</tr>
<tr>
<td>CruiseCtrlSetSpeed</td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>CruiseCtrlSpeed</td>
<td>48</td>
<td>16</td>
</tr>
</tbody>
</table>

To receive the information, variable "DataBuffer" of data type "BYTE[0..19]" is created, for example. Variable "DataBuffer" is specified later on input "Data" of function block MpJ1939Receive, for example.
If the value of signal "SrcAddrssOfCntrlIngDvcFrPsngCrsC" should now be evaluated, for example, "DataBuffer[1]" must be accessed. If signal "CruiseCtrlSystemCmdState" should be accessed, this can be done by bitwise access. This can be implemented using "DataBuffer[0].2", for example.

5.2 Guides

5.2.1 Getting started

5.2.1.1 Receiving engine data

This section explains how to receive engine data using MpJ1939Generic.

5.2.1.1.1 Creating a project

You must first create a new project in Automation Studio. For more information, see here.

Hardware configuration

APC2100.BY44.0000 can be added as the hardware to be used, for example. Plug-in card 5ACCIF01.FPCC-0000 is added so that the CAN interface can be used.

5.2.1.1.2 Adding the mapp component

Adding the MpJ1939Generic configuration

The MpJ1939Generic configuration is added in order to receive engine data.

It contains the mapp Link, which uniquely identifies the mapp component within the mapp environment. The mapp Link establishes the connection between programming and configuration.

The CAN interface is specified on parameter "CAN interface". This information can be taken from the Physical View.

The address used for unique identification in the CAN network must be specified under "Own device address". The remote station in the CAN network must be specified under "Device address". This is defined when the CAN network is set up.

The PGNs shown below can be received or transmitted. PGN EC1 should be received.
In order to receive the information, a process variable must be specified. Process variable My_EC1 of data type MpJ1939EC1Type is specified.

The process variable was defined in the global variable declaration file:

5.2.1.3 Adding a program
The next step is to add a Ladder Diagram program. For more information, see here.

5.2.1.4 Adding MpJ1939Generic
Function block MpJ1939Generic is added.
5.2.1.5 Configuring MpJ1939Generic

Connecting MpLink

Using input "MpLink", we establish a connection to the configuration we created earlier named "gJ1939Generic".

Enabling the function block

The function block is enabled using input "Enable".
5.2.1.1.6 Generating the file structure

Now generate a file structure for the memory card for ARsim. For more information, see here.

5.2.1.1.7 Testing the program

After the changes are downloaded, the program can be tested.

If an engine is connected to the CAN interface, PGN EC1 defined in the configuration is automatically received using MpJ1939Generic and the values are displayed on variable My_EC1.

5.3 Configuration

5.3.1 MpJ1939Generic configuration
**General settings**

The general settings specify information about the CAN interface.

The address used for unique identification in the CAN network must be specified under "Own device address". The CAN interface is specified under "CAN interface". This can be SS1.IF1, for example.

**PGN settings**

The remote station in the CAN network must be specified under "Device address". This is defined when the CAN network is set up.

PGNs are parameter group numbers. They can be used to send and receive information. For more information, see [Transmitting and receiving information](#).

For information about how to work with PGNs in the configuration, see [here](#).

**5.3.1.1 Working with PGNs in the configuration**

The **MpJ1939Generic configuration** defines which PGNs should be enabled and which should not.

If a PGN is enabled, it can be defined whether the PGN is to be transmitted ("Send") or received ("Receive").

If a PGN cannot be read or transmitted cyclically, the "Request" selection appears.
Finally, a process variable must be specified that receives or sends the information. The data type of the process variable differs from PGN to PGN.

For information about how to transmit or receive the data correctly, see section Timing.

The following table explains which data type must be used for which PGN.

<table>
<thead>
<tr>
<th>PGN</th>
<th>Data type to be used</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMB</td>
<td>MpJ1939AMBType</td>
</tr>
<tr>
<td>CCVS1</td>
<td>MpJ1939CCVS1Type</td>
</tr>
<tr>
<td>CI</td>
<td>MpJ1939CIType</td>
</tr>
<tr>
<td>DM01</td>
<td>MpJ1939DM01Type</td>
</tr>
<tr>
<td>EC1</td>
<td>MpJ1939EC1Type</td>
</tr>
<tr>
<td>EEC1</td>
<td>MpJ1939EEC1Type</td>
</tr>
<tr>
<td>EEC2</td>
<td>MpJ1939EEC2Type</td>
</tr>
<tr>
<td>EEC3</td>
<td>MpJ1939EEC3Type</td>
</tr>
<tr>
<td>EFL_P1</td>
<td>MpJ1939EFL_P1Type</td>
</tr>
<tr>
<td>ET1</td>
<td>MpJ1939ET1Type</td>
</tr>
<tr>
<td>HOURS</td>
<td>MpJ1939HOURSType</td>
</tr>
<tr>
<td>IC1</td>
<td>MpJ1939IC1Type</td>
</tr>
<tr>
<td>LFC1</td>
<td>MpJ1939LFC1Type</td>
</tr>
<tr>
<td>LFE1</td>
<td>MpJ1939LFE1Type</td>
</tr>
<tr>
<td>SHUTDN</td>
<td>MpJ1939SHUTDNType</td>
</tr>
<tr>
<td>SOFT</td>
<td>MpJ1939SOFTType</td>
</tr>
<tr>
<td>TSC1</td>
<td>MpJ1939TSC1Type</td>
</tr>
<tr>
<td>VEP1</td>
<td>MpJ1939VEP1Type</td>
</tr>
<tr>
<td>AT1T1I1</td>
<td>MpJ1939AT1T1I1Type</td>
</tr>
<tr>
<td>AT1IG1</td>
<td>MpJ1939AT1IG1Type</td>
</tr>
<tr>
<td>AT1IG2</td>
<td>MpJ1939AT1IG2Type</td>
</tr>
<tr>
<td>AT1IMG</td>
<td>MpJ1939AT1IMGType</td>
</tr>
<tr>
<td>AT1OG1</td>
<td>MpJ1939AT1OG1Type</td>
</tr>
<tr>
<td>AT1OG2</td>
<td>MpJ1939AT1OG2Type</td>
</tr>
<tr>
<td>AT1S1</td>
<td>MpJ1939AT1S1Type</td>
</tr>
<tr>
<td>CM1</td>
<td>MpJ1939CM1Type</td>
</tr>
<tr>
<td>DPFC1</td>
<td>MpJ1939DPFC1Type</td>
</tr>
</tbody>
</table>

5.3.2 MpJ1939Receive configuration

General settings
The address used for unique identification in the CAN network must be specified under "Own device address".

The path of the CAN interface is specified under "CAN interface". This can be SS1.IF1, for example.

The remote station in the CAN network must be specified under "Source address". This is defined when the CAN network is set up.

5.3.3 MpJ1939SpecificRequest configuration

General settings
The address used for unique identification in the CAN network must be specified under "Own device address".
The path of the CAN interface is specified under "CAN interface". This can be SS1.IF1, for example.

The remote station in the CAN network must be specified under "Source address". This is defined when the CAN network is set up.

5.3.4 MpJ1939Transmit configuration

General settings

The node number of the device used must be specified under "Own device address".

The path of the CAN interface is specified under "CAN interface". This can be SS1.IF1, for example.

The remote station in the CAN network must be specified under "Destination address". This is defined when the CAN network is set up.

5.4 Libraries

5.4.1 Data types and enumerators

5.4.1.1 Data types

5.4.1.1.1 MpJ1939AMBType

This data type describes PGN AMB (environmental conditions).
5.4.1.1.2 MpJ1939AT1IG1Type

This data type describes PGN AT1IG1 (aftertreatment 1 / suction gas 1).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOxSnsr1SelDiagStatus</td>
<td>MpJ1939NOS1SDSEnum</td>
<td>Status of NOx sensor AT1 (aftertreatment memory bank 1).</td>
</tr>
<tr>
<td>O2Snsr1PreliminaryFMI</td>
<td>INT</td>
<td>Value of the aftertreatment oxygen suction sensor.</td>
</tr>
<tr>
<td>SCRInNOxSnsr1PreliminaryFMI</td>
<td>INT</td>
<td>Value of the aftertreatment intake NOx sensor.</td>
</tr>
<tr>
<td>Snsr1HtrCtrl</td>
<td>MpJ1939S1HCEnum</td>
<td>Indicates the heating status during the warm-up process.</td>
</tr>
<tr>
<td>Snsr1HtrPreliminaryFMI</td>
<td>INT</td>
<td>Value measured in the intake exhaust gas sensor heater – either NOx or O2.</td>
</tr>
<tr>
<td>WRPPercentO21ReadingStable</td>
<td>MpJ1939WRPO21RSEnum</td>
<td>Indicates whether the percentage of the oxygen measured value of the aftertreatment intake gas sensor – either NOx or O2 – is stable as determined by the manufacturer's control software in exhaust bank 1.</td>
</tr>
<tr>
<td>SCRInNOx1ReadingStable</td>
<td>MpJ1939SCRINO1RSEnum</td>
<td>Indicates whether the NOx measured value of the aftertreatment intake NOx sensor is stable as determined by the manufacturer's control software in exhaust bank 1.</td>
</tr>
<tr>
<td>Snsr1AtTemp</td>
<td>MpJ1939S1ATEnum</td>
<td>Indicates whether the heating element of the aftertreatment suction gas sensor – either NOx or O2 – for accurate measurements in exhaust bank 1 is within the range specified by the manufacturer.</td>
</tr>
<tr>
<td>Snsr1PwrInRange</td>
<td>MpJ1939S1PIREnum</td>
<td>Indicates whether the power supplied to the aftertreatment intake gas sensor – either NOx or O2 – is within the manufacturer's specification in exhaust bank 1.</td>
</tr>
<tr>
<td>PercentO21</td>
<td>REAL</td>
<td>The actual oxidation factor of the gas in the exhaust gas flow.</td>
</tr>
<tr>
<td>SCRInNOx1</td>
<td>REAL</td>
<td>The amount of combined NO and NO2 in the exhaust gas entering the aftertreatment system.</td>
</tr>
<tr>
<td>Status</td>
<td>MpJ1939PGNStatusEnum</td>
<td>State of the parameter group number.</td>
</tr>
</tbody>
</table>

5.4.1.1.3 MpJ1939AT1IG2Type

This data type describes PGN AT1IG2 (aftertreatment 1 / suction gas 2).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPFInTempPreliminaryFMI</td>
<td>INT</td>
<td>Value measured by the intake temperature sensor of the diesel particulate filter.</td>
</tr>
<tr>
<td>ExhTemp1PreliminaryFMI</td>
<td>INT</td>
<td>Value measured by the sensor for exhaust gas temperature 1.</td>
</tr>
<tr>
<td>DPFInTemp</td>
<td>REAL</td>
<td>Temperature of byproducts of engine combustion entering the diesel particulate filter in exhaust bank 1.</td>
</tr>
<tr>
<td>ExhTemp1</td>
<td>REAL</td>
<td>Value measured by the exhaust gas temperature sensor.</td>
</tr>
<tr>
<td>Status</td>
<td>MpJ1939PGNStatusEnum</td>
<td>State of the parameter group number.</td>
</tr>
</tbody>
</table>

5.4.1.1.4 MpJ1939AT1IMGType

This data type describes PGN AT1IMG (aftertreatment 1 / intermediate gas).
### 5.4.1.1.5 MpJ1939AT1OG1Type

This data type describes PGN AT1OG1 (aftertreatment 1 / output gas 1).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O2Snsr1PreliminaryFMI</td>
<td>INT</td>
<td>Value measured by the oxygen sensor for the aftertreatment outlet.</td>
</tr>
<tr>
<td>NOxSnsr1SelfDiagStatus</td>
<td>MpJ1939NOS1SDSEnum</td>
<td>Status of NOx sensor AT01 (aftertreatment output bank 1).</td>
</tr>
<tr>
<td>NOxSnsr1PreliminaryFMI</td>
<td>INT</td>
<td>Value measured by the NOx sensor for the aftertreatment outlet.</td>
</tr>
<tr>
<td>Snsr1HtrCtrl</td>
<td>MpJ1939S1HCEnum</td>
<td>Indicates the heating status during the warm-up process. After receiving a switch-on command, the gas sensor is started up according to a profile defined by the manufacturer.</td>
</tr>
<tr>
<td>Snsr1HtrPreliminaryFMI</td>
<td>INT</td>
<td>Specifies the value measured in the outlet exhaust gas sensor heater – either NOx or O2.</td>
</tr>
<tr>
<td>WRPercentO21ReadingStable</td>
<td>MpJ1939WRPO21RSEnum</td>
<td>Specifies whether the percentage of the oxygen measured value of the aftertreatment outlet gas sensor – either NOx or O2 is stable as determined by the manufacturer's control software in exhaust bank 1.</td>
</tr>
<tr>
<td>NOx1ReadingStable</td>
<td>MpJ1939NO1RSEnum</td>
<td>Specifies whether the NOx measured value of the aftertreatment outlet NOx sensor is stable as determined by the manufacturer's control software in exhaust bank 1.</td>
</tr>
<tr>
<td>Snsr1AtTemp</td>
<td>MpJ1939S1ATEnum</td>
<td>Specifies whether the heating element of the aftertreatment outlet gas sensor – either NOx or O2 – for accurate measurements in exhaust bank 1 is within the range specified by the manufacturer.</td>
</tr>
<tr>
<td>Snsr1PwrInRange</td>
<td>MpJ1939S1PIREnum</td>
<td>Specifies whether the power supplied to the aftertreatment outlet gas sensor – either NOx or O2 – is within the manufacturer's specification in exhaust bank 1.</td>
</tr>
<tr>
<td>PercentO21</td>
<td>REAL</td>
<td>The actual oxidation factor (oxygen percentage) of the gas in the exhaust gas flow.</td>
</tr>
<tr>
<td>NOx1</td>
<td>REAL</td>
<td>The amount of combined NO and NO2 in the exhaust gas leaving the aftertreatment system.</td>
</tr>
</tbody>
</table>

### 5.4.1.6 MpJ1939AT1OG2Type

This data type describes PGN AT1OG2 (aftertreatment 1 / output gas 2).
## Parameter Table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPFOutExhTempPreliminaryFMI</td>
<td>INT</td>
<td>Value measured by the outlet temperature sensor of the diesel particulate filter.</td>
</tr>
<tr>
<td>ExhTemp3PreliminaryFMI</td>
<td>INT</td>
<td>Value measured by exhaust gas temperature sensor 3.</td>
</tr>
<tr>
<td>DPFOutTemp</td>
<td>REAL</td>
<td>Temperature of the engine combustion byproducts leaving the diesel particulate filter exhaust in exhaust bank 1.</td>
</tr>
<tr>
<td>ExhTemp3</td>
<td>REAL</td>
<td>Value measured by the exhaust gas temperature sensor.</td>
</tr>
<tr>
<td>Status</td>
<td>MpJ1939PGNStatusEnum</td>
<td>State of the parameter group number.</td>
</tr>
</tbody>
</table>

### 5.4.1.1.7 MpJ1939AT1S1Type

This data type describes PGN AT1S1 (aftertreatment 1 / service 1).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPFSoilLoadRegenThreshold</td>
<td>REAL</td>
<td>This parameter specifies the value that first causes DPF regeneration in aftertreatment 1.</td>
</tr>
<tr>
<td>DPFTimeSinceLastActiveRegen</td>
<td>INT</td>
<td>Specifies the time since the last active regeneration event of diesel particulate filter 1.</td>
</tr>
<tr>
<td>DPFAshLoadPercent</td>
<td>INT</td>
<td>Specifies the ash load percentage of diesel particulate filter 1.</td>
</tr>
<tr>
<td>DPFSoilLoadPercent</td>
<td>INT</td>
<td>Displays the soil load percentage of diesel particulate filter 1.</td>
</tr>
<tr>
<td>Status</td>
<td>MpJ1939PGNStatusEnum</td>
<td>State of the parameter group number.</td>
</tr>
</tbody>
</table>

### 5.4.1.1.8 MpJ1939AT1T1I1Type

This data type describes PGN AT1T1I1 (aftertreatment 1 / diesel exhaust gas tank 1 / information 1).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Volume</td>
<td>REAL</td>
<td>Number of software identification identifiers represented in the software identification parameter group.</td>
</tr>
<tr>
<td>Temp1</td>
<td>INT</td>
<td>Software identification of an electronic module.</td>
</tr>
<tr>
<td>Lvl</td>
<td>REAL</td>
<td>Specifies the level of the diesel exhaust gas fluid level in mm in the diesel exhaust gas fluid reservoir for aftertreatment system 1.</td>
</tr>
<tr>
<td>LvlVolPreliminaryFMI</td>
<td>SINT</td>
<td>Value measured by the level sensor of the diesel exhaust tank.</td>
</tr>
<tr>
<td>TempPreliminaryFMI</td>
<td>SINT</td>
<td>Specifies the diesel exhaust gas tank temperature for aftertreatment system 1.</td>
</tr>
<tr>
<td>SCROpIndSev</td>
<td>MpJ1939DEFTLLIEnum</td>
<td>Specifies the state of the operator prompting system for anomalies with the SCR system.</td>
</tr>
<tr>
<td>Htr</td>
<td>REAL</td>
<td>Percentage of heating applied to the aftertreatment equipment for diesel exhaust gas fluid tanks.</td>
</tr>
<tr>
<td>HtrPreliminaryFMI</td>
<td>SINT</td>
<td>Value measured for the heater of the diesel exhaust gas liquid tank for aftertreatment system 1.</td>
</tr>
<tr>
<td>Status</td>
<td>MpJ1939PGNStatusEnum</td>
<td>State of the parameter group number.</td>
</tr>
</tbody>
</table>

### 5.4.1.1.9 MpJ1939CCVS1Type

This data type describes PGN CCVS1 (speed control / vehicle speed 1).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EngShutdownOverrideSwitch</td>
<td>MpJ1939ESOSENumb</td>
<td>Switching signal indicating the position of the overload switch of the engine shutdown. This switch function allows the operator to override an impending engine shutdown.</td>
</tr>
<tr>
<td>EngIdleDecrementSwitch</td>
<td>MpJ1939ETMSEnum</td>
<td>Switching signal indicating the position of the engine diagnostic test mode switch.</td>
</tr>
<tr>
<td>EngIdleIncrementSwitch</td>
<td>MpJ1939EIISENumb</td>
<td>Switching signal indicating the position of the idle decrement switch.</td>
</tr>
<tr>
<td>CruiseCtrlStates</td>
<td>MpJ1939CCSEnum</td>
<td>State of the cruise control.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PTOGovernorState</td>
<td>MpJ1939PTOGEEnum</td>
<td>This parameter is used to indicate the current state or operating mode of the power take-off governor.</td>
</tr>
<tr>
<td>CruiseCtrlSetSpeed</td>
<td>INT</td>
<td>Speed of the cruise control.</td>
</tr>
<tr>
<td>CruiseCtrlAccelerateSwitch</td>
<td>MpJ1939CCASENum</td>
<td>Specifies whether cruise control is in position &quot;Accelerate&quot;.</td>
</tr>
<tr>
<td>CruiseCtrlResumeSwitch</td>
<td>MpJ1939CCRSEnum</td>
<td>Specifies whether cruise control is in position &quot;Resume&quot;.</td>
</tr>
<tr>
<td>CruiseCtrlDecelerateSwitch</td>
<td>MpJ1939CCDEDEnum</td>
<td>Specifies whether cruise control is in position &quot;Decelerate&quot;.</td>
</tr>
<tr>
<td>CruiseCtrlSetSwitch</td>
<td>MpJ1939CCSEnum</td>
<td>Specifies whether cruise control is in position &quot;Set&quot;.</td>
</tr>
<tr>
<td>ClutchSwitch</td>
<td>MpJ1939CSEnum</td>
<td>Specifies whether the clutch pedal is depressed.</td>
</tr>
<tr>
<td>BrakeSwitch</td>
<td>MpJ1939BSEnum</td>
<td>Specifies whether the brake pedal is actuated.</td>
</tr>
<tr>
<td>CruiseCtrlResumeSwitch</td>
<td>MpJ1939CCRSEnum</td>
<td>Specifies whether cruise control is in position &quot;Resume&quot;.</td>
</tr>
<tr>
<td>CruiseCtrlCoastSwitch</td>
<td>MpJ1939CCCSEnum</td>
<td>Specifies whether cruise control is in position &quot;Decelerate&quot;.</td>
</tr>
<tr>
<td>CruiseCtrlSetSwitch</td>
<td>MpJ1939CCSEnum</td>
<td>Specifies whether cruise control is in position &quot;Set&quot;.</td>
</tr>
<tr>
<td>WheelBasedVehicleSpeed</td>
<td>REAL</td>
<td>Speed of the vehicle calculated from the wheel or stern shaft speed.</td>
</tr>
<tr>
<td>ParkBrakeReleaseInhibitRq</td>
<td>MpJ1939PBRIREnum</td>
<td>The parking brake release inhibitor request specifies whether the parking brake remains actuated.</td>
</tr>
<tr>
<td>CruiseCtrlPauseSwitch</td>
<td>MpJ1939CCPSEnum</td>
<td>Specifies the position of the cruise control interrupt switch.</td>
</tr>
<tr>
<td>ParkingBrakeSwitch</td>
<td>MpJ1939PBSEnum</td>
<td>Specifies the status of the parking brake.</td>
</tr>
<tr>
<td>TwoSpeedAxleSwitch</td>
<td>MpJ1939TIASEnum</td>
<td>Specifies the status of the current axis range.</td>
</tr>
<tr>
<td>Status</td>
<td>MpJ1939PGNstatusEnum</td>
<td>State of the parameter group number.</td>
</tr>
</tbody>
</table>

### 5.4.1.10 MpJ1939CIType

This data type describes PGN CI (component identification).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UnitNumber</td>
<td>INT</td>
<td>Specifies the device number for the vehicle's power supply.</td>
</tr>
<tr>
<td>SerialNumber</td>
<td>INT</td>
<td>Serial number of the component.</td>
</tr>
<tr>
<td>Model</td>
<td>INT</td>
<td>Model of the component.</td>
</tr>
<tr>
<td>Make</td>
<td>INT</td>
<td>Number of the component that was created in accordance with the guidelines of the &quot;American Trucking Association Vehicle Maintenance Reporting Standard (ATA/VMRS)&quot;.</td>
</tr>
<tr>
<td>Status</td>
<td>MpJ1939PGNstatusEnum</td>
<td>State of the parameter group number.</td>
</tr>
</tbody>
</table>

### 5.4.1.11 MpJ1939CM1Type

This data type describes PGN CM1 (cabin message 1).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SelectedMaxVehicleSpeedLimit</td>
<td>INT</td>
<td>Specifies the maximum vehicle speed selected by the user.</td>
</tr>
<tr>
<td>RqCabZoneHeating</td>
<td>MpJ1939RCEHEnum</td>
<td>Specifies the state of the cabin heater.</td>
</tr>
<tr>
<td>RqEngZoneHeating</td>
<td>MpJ1939RZEHEnum</td>
<td>Specifies the state of the engine zone heater.</td>
</tr>
<tr>
<td>AuxHeaterModeRq</td>
<td>MpJ1939AHRSEnum</td>
<td>Specifies the state of the auxiliary heater.</td>
</tr>
<tr>
<td>EngAutomaticStartEnableSwitch</td>
<td>MpJ1939ESASEEnum</td>
<td>Specifies the state of the idle management system.</td>
</tr>
<tr>
<td>AutomaticGearShiftingEnableSwitch</td>
<td>MpJ1939AGSEEnum</td>
<td>Specifies the state of the automatic transmission.</td>
</tr>
<tr>
<td>AFTRegenForceSw</td>
<td>MpJ1939AFTRFSEnum</td>
<td>Specifies the state of the switch that forces aftertreatment regeneration.</td>
</tr>
<tr>
<td>AFTRegenInhSw</td>
<td>MpJ1939AFTRISEnum</td>
<td>Specifies the state of the switch that prevents aftertreatment regeneration.</td>
</tr>
<tr>
<td>VhclLmtngSpcGovernorEnableSwitch</td>
<td>MpJ1939VLSGESEnum</td>
<td>Specifies whether the vehicle limiting speed governor (VLSG) is switched on.</td>
</tr>
<tr>
<td>VhclLmtngSpcGmrnIncrementSwitch</td>
<td>MpJ1939VLSGISEnum</td>
<td>Switching signal that increments the vehicle limiting speed governor (VLSG).</td>
</tr>
<tr>
<td>VhclLmtngSpcGmrnDecrementSwitch</td>
<td>MpJ1939VLSGDSEnum</td>
<td>Switching signal that decrements the vehicle limiting speed governor (VLSG).</td>
</tr>
<tr>
<td>ParkBrakeCmd</td>
<td>MpJ1939PBCEnum</td>
<td>Specifies whether the parking brake is controlled by a secondary device.</td>
</tr>
<tr>
<td>SeatBeltSwitch</td>
<td>MpJ1939SBEEnum</td>
<td>Specifies the state of the switch used to determine whether the seat belt is fastened.</td>
</tr>
</tbody>
</table>
Parameter | Data type | Description
--- | --- | ---
OperatorSeatDirectionSwitch | MpJ1939OSDSEnum | Specifies whether the driver's seat is in the front driving position.
BatteryMainSwitchHoldRq | MpJ1939BMSHREnum | Specifies whether the main switch of the battery must be held.
AuxHeaterCoolantPumpRq | MpJ1939AHCPREnum | Specifies whether the cooling water pump of the auxiliary heater should be switched on.
CabInteriorTempCmd | REAL | Parameters for controlling a specific cabin interior temperature.
RequestedPercentFanSpeed | REAL | Fan speed of the first or only fan in the system as a ratio of the actual fan drive (current speed) to the fully used fan drive (maximum fan speed).
Status | MpJ1939PGNStatusEnum | State of the parameter group number.

5.4.1.12 MpJ1939DM01Type

This data type describes PGN DM01 (active diagnostic error codes).

Parameter | Data type | Description
--- | --- | ---
FlashRedStopLamp | MpJ1939FRSLEnum | Makes it possible to let the red brake lamp blink.
FlashProtectLamp | MpJ1939FPLEnum | Makes it possible to let the engine protection lamp blink.
FlashMalfuncIndicatorLamp | MpJ1939FMILEnum | Makes it possible to let the fault indicator lamp blink.
FlashAmberWarningLamp | MpJ1939FAWLEnum | Makes it possible to let the yellow warning light blink.
AmberWarningLampStatus | MpJ1939AWLSEnum | State of the yellow warning lamp. In the event of a fault, the vehicle does not have to be stopped immediately.
RedStopLampState | MpJ1939RSLSEnum | Status of the red brake lamp. In the event of a fault, the vehicle must be stopped immediately.
ProtectLampStatus | MpJ1939PLSEnum | Status of the engine protection lamp.
DTC5 | INT | Diagnostic error code 5
DTC4 | INT | Diagnostic error code 4
DTC3 | INT | Diagnostic error code 3
DTC2 | INT | Diagnostic error code 2
DTC1 | INT | Diagnostic error code 1
Status | MpJ1939PGNStatusEnum | State of the parameter group number.

5.4.1.13 MpJ1939DPFC1Type

This data type describes PGN DPFC1 (diesel particulate filter - controller 1).

Parameter | Data type | Description
--- | --- | ---
DslPrtcltFltrActvRgnrtnInhbt5629 | MpJ1939DPFARI5629Enum | Indicates the state of the diesel particulate filter's active regeneration inhibition due to the low exhaust gas pressure.
AFT1DPFCondNotMetForActiveRegen | MpJ1939AFT1DPFCNMFAREnum | Specifies whether regeneration of diesel particulate filter 1 is inhibited or not under current engine operating conditions.
HydrocarbonDoserPurgingEnable | MpJ1939HDPEEnum | Defines the flush release of the hydrocarbon doser.
DslPrtcltFltrActvRgnrtnFrcdSttus | MpJ1939DPFARFSEnum | Displays the forced execution status of regeneration of the diesel particulate filter.
ExhaustSystemHighTempLampCmd | MpJ1939ESHTLCEnum | Specifies the state of the exhaust gas system high-temperature lamp.
DslPrtcltFltrActvRgnrtnInhbt3715 | MpJ1939DPFARI3715Enum | Indicates the state of the active regeneration inhibition of the diesel particulate filter due to a temporary system lock.
DslPrtcltFltrActvRgnrtnInhbt3717 | MpJ1939DPFARI3717Enum | Indicates the state of the active regeneration inhibition of the diesel particulate filter since the vehicle speed is less than the permissible vehicle speed.
DslPrtcltFltrActvRgnrtnInhbtDTE | MpJ1939DPFARDTEEnum | Indicates the state of the diesel particulate filter's active regeneration inhibition since the engine is not warmed up.
DslPrtcltFltrActvRgnrtnInhbtDTE | MpJ1939DPFARDTEEnum | Indicates the state of the diesel particulate filter's active regeneration inhibition since the engine is not warmed up.
DslPrtcltFltrActvRgnrtnInhbt3715 | MpJ1939DPFARI3715Enum | Indicates the state of the active regeneration inhibition of the diesel particulate filter due to a permanent system lock.
DslPrtcltFltrActvRgnrtnInhbt3717 | MpJ1939DPFARI3717Enum | Indicates the state of the active regeneration inhibition of the diesel particulate filter since the vehicle speed is less than the permissible vehicle speed.
### Parameter Data type Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDTL</td>
<td>MpJ1939DPFARIDTLEnum</td>
<td>Indicates the state of the diesel particulate filter's active regeneration inhibition since the exhaust gas temperature is too low.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDTL</td>
<td>MpJ1939DPFARIDTLEnum</td>
<td>Indicates the state of the diesel particulate filter's active regeneration inhibition since the gearbox is not idling.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDT</td>
<td>MpJ1939DPFARIDTEnum</td>
<td>Indicates the state of the active regeneration inhibition of the diesel particulate filter caused by the diesel particulate filter regeneration suppression switch.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDT</td>
<td>MpJ1939DPFARIDTEnum</td>
<td>Indicates the state of the active regeneration inhibition of the diesel particulate filter.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDTS</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition of the diesel particulate filter since the PTO shaft is active.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition of the diesel particulate filter since the service brake is activated.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition of the diesel particulate filter since the clutch is disengaged.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition of the diesel particulate filter due to an active system fault.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition of the diesel particulate filter due to a system timeout.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition of the diesel particulate filter since the parking brake is not set.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition of the diesel particulate filter since the vehicle speed is above a permissible limit.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition of the diesel particulate filter since the accelerator pedal is idle.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the exhaust gas temperature is too low.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the gearbox is not idling.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the parking brake is not set.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the vehicle speed is above a permissible limit.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the accelerator pedal is idle.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the clutch is disengaged.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the PTO shaft is active.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the service brake is activated.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the clutch is disengaged.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the gearbox is not idling.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the parking brake is not set.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the vehicle speed is above a permissible limit.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the accelerator pedal is idle.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the clutch is disengaged.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the PTO shaft is active.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the service brake is activated.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the clutch is disengaged.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the gearbox is not idling.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the parking brake is not set.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the vehicle speed is above a permissible limit.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the accelerator pedal is idle.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the clutch is disengaged.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the PTO shaft is active.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the service brake is activated.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the clutch is disengaged.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the gearbox is not idling.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the parking brake is not set.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the vehicle speed is above a permissible limit.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the accelerator pedal is idle.</td>
</tr>
<tr>
<td>DslPrtcltFltrActvRgnrtnInhbtdDST</td>
<td>MpJ1939DPFARIDTSEnum</td>
<td>Indicates the state of the active regeneration inhibition since the clutch is disengaged.</td>
</tr>
</tbody>
</table>

### 5.4.1.1.14 MpJ1939EC1Type

This data type describes PGN EC1 (engine configuration 1).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SupportTSC1CtrlPurposeGroup4</td>
<td>INT</td>
<td>This parameter specifies which TSC1 control purposes are supported in group 4.</td>
</tr>
<tr>
<td>SupportTSC1CtrlPurposeGroup3</td>
<td>INT</td>
<td>This parameter specifies which TSC1 control purposes are supported in group 3.</td>
</tr>
<tr>
<td>SupportTSC1CtrlPurposeGroup2</td>
<td>INT</td>
<td>This parameter specifies which TSC1 control purposes are supported in group 2.</td>
</tr>
<tr>
<td>SupportTSC1CtrlPurposeGroup1</td>
<td>INT</td>
<td>This parameter specifies which TSC1 control purposes are supported in group 1.</td>
</tr>
<tr>
<td>SupportVariableRateTSC1Message</td>
<td>INT</td>
<td>This parameter specifies which TSC1 transfer rates are supported by the engine ECU in addition to the required 10 ms transfer rate for temporary power train control purposes.</td>
</tr>
<tr>
<td>EngDefaultTorqueLimit</td>
<td>DINT</td>
<td>Specifies the accompanying parameter to the transmission torque limit.</td>
</tr>
<tr>
<td>EngMomentOfInertia</td>
<td>REAL</td>
<td>Moment of inertia of the engine.</td>
</tr>
<tr>
<td>EngExRngRqstdSpdCtrlRngUpprLimit</td>
<td>REAL</td>
<td>The maximum engine speed.</td>
</tr>
<tr>
<td>EngRqstdTrqueCtrlRangeUpperLimit</td>
<td>INT</td>
<td>The maximum engine torque that the engine permits when operating in a torque control / limit mode.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EngRqstdTrqueCtrlRangeLowerLimit</td>
<td>INT</td>
<td>The minimum engine torque that the engine permits when operating in a torque control / limit mode.</td>
</tr>
<tr>
<td>EngRqstdSpeedCtrlRangeUpperLimit</td>
<td>INT</td>
<td>The maximum engine speed regardless of the load the engine permits when operating in a speed control / limit mode.</td>
</tr>
<tr>
<td>EngMaxMomentaryOverrideTimeLimit</td>
<td>INT</td>
<td>The minimum engine speed that the engine permits when operating in a speed control / limit mode.</td>
</tr>
<tr>
<td>EngMomenrtyOverrideSpeedPoint7</td>
<td>REAL</td>
<td>The maximum engine speed above idling permitted by the engine controller during a momentary high idling override.</td>
</tr>
<tr>
<td>EngReferenceTorque</td>
<td>REAL</td>
<td>This parameter is the 100% reference value for all specified engine torque parameters. It is defined only once and does not change if another engine torque characteristic diagram becomes valid.</td>
</tr>
<tr>
<td>EngGain</td>
<td>DINT</td>
<td>Value of the end speed controller.</td>
</tr>
<tr>
<td>EngSpeedAtHighIdlePoint6</td>
<td>REAL</td>
<td>Engine speed of the high idling speed (point 6) of the engine torque characteristic diagram.</td>
</tr>
<tr>
<td>EngPercentTorqueAtPoint5</td>
<td>REAL</td>
<td>The torque limit indicating the available engine torque that can be provided by the engine at points 5 of the engine characteristic diagram.</td>
</tr>
<tr>
<td>EngSpeedAtPoint5</td>
<td>INT</td>
<td>Engine speed from point 5 of the engine torque characteristic diagram.</td>
</tr>
<tr>
<td>EngPercentTorqueAtPoint4</td>
<td>REAL</td>
<td>The torque limit indicating the available engine torque that can be provided by the engine at points 4 of the engine characteristic diagram.</td>
</tr>
<tr>
<td>EngSpeedAtPoint4</td>
<td>INT</td>
<td>Engine speed from point 4 of the engine torque characteristic diagram.</td>
</tr>
<tr>
<td>EngPercentTorqueAtPoint3</td>
<td>INT</td>
<td>The torque limit indicating the available engine torque that can be provided by the engine at point 3 of the engine characteristic diagram.</td>
</tr>
<tr>
<td>EngSpeedAtPoint3</td>
<td>REAL</td>
<td>Engine speed from point 3 of the engine torque characteristic diagram.</td>
</tr>
<tr>
<td>EngPercentTorqueAtPoint2</td>
<td>INT</td>
<td>The torque limit indicating the available engine torque that can be provided by the engine at point 2 of the engine characteristic diagram.</td>
</tr>
<tr>
<td>EngSpeedAtPoint2</td>
<td>REAL</td>
<td>Engine speed from point 2 of the engine torque characteristic diagram.</td>
</tr>
<tr>
<td>EngPercentTorqueAtIdlePoint1</td>
<td>INT</td>
<td>The torque limit specifies the available engine torque that the engine can deliver when idling.</td>
</tr>
<tr>
<td>EngSpeedAtIdlePoint1</td>
<td>REAL</td>
<td>Stationary low idling speed of the engine, which contains influences due to the engine temperature (after switching on) and other stationary changes (calibration offsets, sensor failures, etc.).</td>
</tr>
<tr>
<td>Status</td>
<td></td>
<td>State of the parameter group number.</td>
</tr>
</tbody>
</table>

5.4.1.1.15 MpJ1939EEC1Type

This data type describes PGN EEC1 (electronic engine control 1).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EngDemandPercentTorque</td>
<td>SINT</td>
<td>The required torque output of the engine through all dynamic internal inputs.</td>
</tr>
<tr>
<td>EngStarterMode</td>
<td>MpJ1939ESMEnum</td>
<td>Specifies why a starter did not work.</td>
</tr>
<tr>
<td>SrcAddressOfCtrlngDvcForEngCtrl</td>
<td>USINT</td>
<td>The source address of the SAE J1939 device that is currently controlling the engine.</td>
</tr>
<tr>
<td>EngSpeed</td>
<td>REAL</td>
<td>Actual engine speed calculated over a minimum crankshaft angle of 720 degrees divided by the number of cylinders.</td>
</tr>
<tr>
<td>ActualEngPercentTorque</td>
<td>SINT</td>
<td>The calculated output torque of the engine. The data is transmitted in the specified torque as a percentage of the reference engine torque.</td>
</tr>
</tbody>
</table>
### Parameter Details

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DriversDemandEngPercentTorque</td>
<td>SINT</td>
<td>The requested torque output of the engine by the driver.</td>
</tr>
<tr>
<td>ActEngPcntnTorqueHighResolution</td>
<td>REAL</td>
<td>This parameter indicates an additional torque as a percentage of the reference engine torque.</td>
</tr>
<tr>
<td>EngTorqueMode</td>
<td>SINT</td>
<td>State signal indicating which engine torque mode is currently generating, limiting or controlling the torque.</td>
</tr>
<tr>
<td>Status</td>
<td>MpJ1939PGNStatusEnum</td>
<td>State of the parameter group number.</td>
</tr>
</tbody>
</table>

---

#### 5.4.1.1.16 MpJ1939EEC2Type

This data type describes PGN EEC2 (electronic engine control 2).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EstPumpingPercentTorque</td>
<td>SINT</td>
<td>The calculated torque specifies the estimated amount of torque loss due to the engine ventilation system.</td>
</tr>
<tr>
<td>ActMaxAvailableEngPercentTorque</td>
<td>REAL</td>
<td>This is the maximum amount of torque that the engine can deliver immediately as a percentage of the reference engine torque.</td>
</tr>
<tr>
<td>SCRThermalManagementActive</td>
<td>MpJ1939SCRTMAEnum</td>
<td>Specifies whether the exhaust gas temperatures have been increased to regenerate the SCR aftertreatment system or to prepare for regeneration of the SCR aftertreatment system.</td>
</tr>
<tr>
<td>DPFThermalManagementActive</td>
<td>MpJ1939DPFTMAEnum</td>
<td>Indicates that the exhaust gas temperatures have been increased for regeneration of the diesel particulate filter aftertreatment system or for preparation of regeneration of the diesel particulate aftertreatment system.</td>
</tr>
<tr>
<td>MnntaryEngMaxPowerEnableFeedback</td>
<td>MpJ1939MEMPEFEEnum</td>
<td>Specifies whether the maximum engine power is currently activated.</td>
</tr>
<tr>
<td>VhcleAccelerationRateLimitStatus</td>
<td>MpJ1939VARLSEnum</td>
<td>State of the system that limits the maximum vehicle forward acceleration.</td>
</tr>
<tr>
<td>AccelPedalPos2</td>
<td>REAL</td>
<td>The ratio of the actual position of the second analog engine speed/torque request input device (such as an accelerator pedal or throttle lever) to the maximum position of the input device.</td>
</tr>
<tr>
<td>RemoteAccelPedalPos</td>
<td>REAL</td>
<td>The ratio of the actual position of the remote analog engine speed/torque request input device (such as an accelerator pedal or throttle lever) to the maximum position of the input device.</td>
</tr>
<tr>
<td>EngPercentLoadAtCurrentSpeed</td>
<td>SINT</td>
<td>The ratio of the actual engine percentage torque (indicated) to the maximum indicated torque available at the current engine speed is limited to zero torque during engine braking.</td>
</tr>
<tr>
<td>AccelPedalPos1</td>
<td>REAL</td>
<td>The ratio of the actual position of the analog engine speed/torque request input device (such as an accelerator pedal or throttle lever) to the maximum position of the input device.</td>
</tr>
<tr>
<td>AccelPedal2LowIdleSwitch</td>
<td>MpJ1939AP2LISEnum</td>
<td>Switching signal indicating the state of the low idle switch on accelerator pedal 2.</td>
</tr>
<tr>
<td>RoadSpeedLimitStatus</td>
<td>MpJ1939RSLiSEnum</td>
<td>State of the system for limiting the maximum vehicle speed.</td>
</tr>
<tr>
<td>AccelPedalKickdownSwitch</td>
<td>MpJ1939APKSEnum</td>
<td>Switching signal indicating whether the accelerator pedal kick-down switch is open or closed.</td>
</tr>
<tr>
<td>AccelPedal1LowIdleSwitch</td>
<td>MpJ1939AP1LISEnum</td>
<td>Switching signal indicating the state of the low idle switch on accelerator pedal 1.</td>
</tr>
<tr>
<td>Status</td>
<td>MpJ1939PGNStatusEnum</td>
<td>State of the parameter group number.</td>
</tr>
</tbody>
</table>

---

#### 5.4.1.1.17 MpJ1939EEC3Type

This data type describes PGN EEC3 (electronic engine control 3).
## mapp Services

### Parameter Data type Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFT2ExhDewPoint</td>
<td>MpJ1939AFT2EDPEnum</td>
<td>Specifies whether the temperature on the outlet side of the aftertreatment system has exceeded the dew point.</td>
</tr>
<tr>
<td>AFT2InDewPoint</td>
<td>MpJ1939AFT2IDPEnum</td>
<td>Specifies whether the temperature on the intake side of the aftertreatment system has exceeded the dew point.</td>
</tr>
<tr>
<td>AFT1ExhDewPoint</td>
<td>MpJ1939AFT1EDPEnum</td>
<td>Specifies whether the temperature on the outlet side of the aftertreatment system has exceeded the dew point.</td>
</tr>
<tr>
<td>AFT1InDewPoint</td>
<td>MpJ1939AFT1IDPEnum</td>
<td>Specifies whether the temperature on the intake side of the aftertreatment system has exceeded the dew point.</td>
</tr>
<tr>
<td>AFT1ExhGasMFR</td>
<td>REAL</td>
<td>Measured exhaust gas mass before the aftertreatment system in exhaust bank 1 and 2.</td>
</tr>
<tr>
<td>EstEngPrslctLossesPercentTorque</td>
<td>INT</td>
<td>The calculated torque specifies the estimated amount of torque loss due to wear and tear, such as cooling fan, air compressor, air conditioner, etc. It is expressed as a percentage of the engine reference torque.</td>
</tr>
<tr>
<td>EnginesDesiredOperatingSpeed</td>
<td>REAL</td>
<td>An indication of the engine for the optimal operating speed of the engine for the current conditions.</td>
</tr>
<tr>
<td>NominalFrictionPercentTorque</td>
<td>INT</td>
<td>The torque specifies how much torque is required by the basic engine itself, by the loss torque of the accessory.</td>
</tr>
<tr>
<td>Status</td>
<td>MpJ1939PGNStatusEnum</td>
<td>State of the parameter group number.</td>
</tr>
</tbody>
</table>

#### 5.4.1.1.18 MpJ1939EFL_P1Type

This data type describes PGN EFL_P1 (engine oil level / pressure 1).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EngCoolantLevel</td>
<td>REAL</td>
<td>Ratio of the liquid volume in the engine cooling system to the total cooling system volume.</td>
</tr>
<tr>
<td>EngCoolantPress</td>
<td>INT</td>
<td>Specifies the measuring pressure of the liquid in the engine cooling system.</td>
</tr>
<tr>
<td>EngCrankcasePress</td>
<td>REAL</td>
<td>First instance of the measuring pressure in the engine crankcase.</td>
</tr>
<tr>
<td>EngOilPress</td>
<td>INT</td>
<td>Pressure of the oil in the engine lubrication system. Provided by the oil pump.</td>
</tr>
<tr>
<td>EngOilLevel</td>
<td>REAL</td>
<td>Ratio of the current volume of engine oil to the maximum required volume.</td>
</tr>
<tr>
<td>EngExCrankcaseBlowbyPress</td>
<td>REAL</td>
<td>Specifies the crankcase blow-by pressure.</td>
</tr>
<tr>
<td>EngFuelDeliveryPress</td>
<td>INT</td>
<td>Fuel tank pressure of the fuel in the system as supplied by the supply pump to the injection pump.</td>
</tr>
<tr>
<td>Status</td>
<td>MpJ1939PGNStatusEnum</td>
<td>State of the parameter group number.</td>
</tr>
</tbody>
</table>

#### 5.4.1.1.19 MpJ1939ET1Type

This data type describes PGN ET1 (engine temperature 1).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EngIntercoolerThermostatOpening</td>
<td>REAL</td>
<td>Position of the thermostat. Used to regulate the temperature of the engine charge air cooler.</td>
</tr>
<tr>
<td>EngIntercoolerTemp</td>
<td>INT</td>
<td>The temperature of the liquid in the charge air cooler.</td>
</tr>
<tr>
<td>EngTurboOilTemp</td>
<td>REAL</td>
<td>Temperature of the turbocharger lubricant.</td>
</tr>
<tr>
<td>EngOilTemp1</td>
<td>REAL</td>
<td>Temperature of the engine lubricant.</td>
</tr>
<tr>
<td>EngFuelTemp1</td>
<td>INT</td>
<td>Temperature of the fuel.</td>
</tr>
<tr>
<td>EngCoolantTemp</td>
<td>INT</td>
<td>The temperature of the liquid in the engine cooling system.</td>
</tr>
<tr>
<td>Status</td>
<td>MpJ1939PGNStatusEnum</td>
<td>State of the parameter group number.</td>
</tr>
</tbody>
</table>

#### 5.4.1.20 MpJ1939HOURSType

This data type describes PGN HOURS (engine hours).
Parameter | Data type | Description
--- | --- | ---
EngTotalRevolutions | REAL | Number of revolutions of the engine crankshaft during operation.
EngTotalHoursOfOperation | REAL | Operating time of the engine.
Status | MpJ1939PGNStatusEnum | State of the parameter group number.

### 5.4.1.1.21 MpJ1939IC1Type
This data type describes PGN IC1 (suction/exhaust gas conditions 1).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EngCoolantFilterDiffPress</td>
<td>REAL</td>
<td>Specifies the change in coolant pressure.</td>
</tr>
<tr>
<td>EngExhaustGasTemp</td>
<td>REAL</td>
<td>Temperature of the combustion byproducts leaving the engine.</td>
</tr>
<tr>
<td>EngAirFilter1DiffPress</td>
<td>REAL</td>
<td>Specifies the change in engine air system pressure.</td>
</tr>
<tr>
<td>EngAirIntakePress</td>
<td>INT</td>
<td>Absolute air pressure at the inlet connection to the intake manifold or air box.</td>
</tr>
<tr>
<td>EngIntakeManifold1Temp</td>
<td>INT</td>
<td>Temperature of the pre-combustion air in the intake manifold of the engine air supply system.</td>
</tr>
<tr>
<td>EngIntakeManifold1Press</td>
<td>INT</td>
<td>Result of the measuring pressure measurement of the intake manifold.</td>
</tr>
<tr>
<td>AFT1DPFInPress</td>
<td>REAL</td>
<td>Exhaust gas pressure as a result of particle accumulation on the filter in the exhaust gas flow.</td>
</tr>
</tbody>
</table>

### 5.4.1.1.22 MpJ1939LFC1Type
This data type describes PGN LFC1 (fuel consumption (liquid) 1).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EngTotalFuelUsed</td>
<td>REAL</td>
<td>Quantity of fuel consumed during vehicle operation.</td>
</tr>
<tr>
<td>EngTripFuel</td>
<td>REAL</td>
<td>Fuel consumed during the whole or part of a journey.</td>
</tr>
</tbody>
</table>

### 5.4.1.1.23 MpJ1939LFE1Type
This data type describes PGN LFE1 (fuel consumption 1).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EngThrottleValve2Pos</td>
<td>REAL</td>
<td>Specifies the acquired position feedback of a sensor measuring the value of a fuel/air mixture.</td>
</tr>
<tr>
<td>EngThrottleValve1Pos</td>
<td>REAL</td>
<td>Specifies the acquired position feedback of a sensor measuring the value of a fuel/air mixture.</td>
</tr>
<tr>
<td>EngAverageFuelEconomy</td>
<td>REAL</td>
<td>Average of the current fuel consumption.</td>
</tr>
<tr>
<td>EngInstantaneousFuelEconomy</td>
<td>REAL</td>
<td>Current fuel savings at current vehicle speed.</td>
</tr>
<tr>
<td>EngFuelRate</td>
<td>REAL</td>
<td>Fuel consumption per unit of time.</td>
</tr>
</tbody>
</table>

### 5.4.1.1.24 MpJ1939SHUTDNType
This data type describes PGN SHUTDN (shutdown).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClntLvlEngPrtectionShutdownStatus</td>
<td>MpJ1939CLEPSSEnum</td>
<td>Specifies the configuration of the engine protection shutdown of the coolant level.</td>
</tr>
<tr>
<td>PTOShutdownHasShutdownEng</td>
<td>MpJ1939PTOSHSEEnum</td>
<td>State signal specifying whether or not the engine has been shut down by the PTO shutdown system.</td>
</tr>
<tr>
<td>EngAirShutoffStatus</td>
<td>MpJ1939EASSEnum</td>
<td>State signal indicating the actual measured position of the air cutoff.</td>
</tr>
<tr>
<td>EngOverspeedTest</td>
<td>MpJ1939EOTEnum</td>
<td>Specifies the state of the engine overspeed test signal as measured by the reporting ECM.</td>
</tr>
<tr>
<td>EngAirShutoffCmdStatus</td>
<td>MpJ1939EASCSEnum</td>
<td>Indicates if the air shutoff driver output is activated.</td>
</tr>
<tr>
<td>EngAlarmOutputCmdStatus</td>
<td>MpJ1939EAOCSEnum</td>
<td>Indicates when the alarm driver output is activated.</td>
</tr>
<tr>
<td>EngAlarmAcknowledge</td>
<td>MpJ1939EAAEnum</td>
<td>Indicates whether engine alarm acknowledgment is active.</td>
</tr>
</tbody>
</table>
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EngProtectionSystemConfig</td>
<td>MpJ1939EPSCEnum</td>
<td>Parameter that specifies the configuration of the engine shutdown system.</td>
</tr>
<tr>
<td>EngProtectionSystemTimerState</td>
<td>MpJ1939EPSTSEnum</td>
<td>Specifies the mode of the engine protection system's timer system.</td>
</tr>
<tr>
<td>EngProtectionSystemTimerOverride</td>
<td>MpJ1939EPSTOEnum</td>
<td>Specifies the state of the override function of the engine protection system timer.</td>
</tr>
<tr>
<td>EngPrctnSystmApproachingShutdown</td>
<td>MpJ1939EPSSAEEnum</td>
<td>State signal indicating whether engine shutdown is imminent.</td>
</tr>
<tr>
<td>EngPrtectionSystemHasShutdownEng</td>
<td>MpJ1939EPSHSEEnum</td>
<td>State signal indicating whether the engine protection system has shut down the engine.</td>
</tr>
<tr>
<td>EngWaitToStartLamp</td>
<td>MpJ1939EWTSLEnum</td>
<td>Indicates whether the engine is too cold to start.</td>
</tr>
<tr>
<td>RefrigerantHighPressSwitch</td>
<td>MpJ1939RHPSEnum</td>
<td>Specifies the position of the high pressure switch in the coolant circuit of an air conditioner.</td>
</tr>
<tr>
<td>RefrigerantLowPressSwitch</td>
<td>MpJ1939RLPSEnum</td>
<td>Specifies the position of the low pressure switch in the coolant circuit of an air conditioner.</td>
</tr>
<tr>
<td>A_CHighPressFanSwitch</td>
<td>MpJ1939ACHRPFSEnum</td>
<td>Switching signal indicating whether the pressure in the coolant circuit of an air conditioner is too high and the fan must be switched on.</td>
</tr>
<tr>
<td>EngIdleShutdownTimerFunction</td>
<td>MpJ1939EISTFEnum</td>
<td>Parameter specifying the configuration of the idle shutdown timer system.</td>
</tr>
<tr>
<td>EngIdleShutdownTimerState</td>
<td>MpJ1939EISTSEnum</td>
<td>State signal indicating the current operating mode of the idle shutdown timer system.</td>
</tr>
<tr>
<td>EngIdleShutdownTimerOverride</td>
<td>MpJ1939EISTOEnum</td>
<td>State signal indicating the state of the override function of the idle shutdown timer system.</td>
</tr>
<tr>
<td>EngIdleShutdownDriverAlertMode</td>
<td>MpJ1939EISDAMEnum</td>
<td>State signal indicating the state of the driver alert mode of the idle shutdown timer system.</td>
</tr>
<tr>
<td>EngIdleShutdownHasShutdownEng</td>
<td>MpJ1939EISHSEEnum</td>
<td>State signal that identifies whether or not the engine has been shut down by the idle shutdown timer system.</td>
</tr>
<tr>
<td>Status</td>
<td>MpJ1939PGNStatusEnum</td>
<td>State of the parameter group number.</td>
</tr>
</tbody>
</table>

### 5.4.1.1.25 MpJ1939SOFTType

This data type describes PGN SOFT (software identification).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NmbrOfSftwreIdentificationFields</td>
<td>INT</td>
<td>Number of software identification identifiers represented in the software identification parameter group.</td>
</tr>
<tr>
<td>SoftwareIdentification</td>
<td>INT</td>
<td>Software identification of an electronic module.</td>
</tr>
<tr>
<td>Status</td>
<td>MpJ1939PGNStatusEnum</td>
<td>State of the parameter group number.</td>
</tr>
</tbody>
</table>

### 5.4.1.1.26 MpJ1939TSC1Type

This data type describes PGN TSC1 (torque / speed control 1).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MessageChecksum</td>
<td>INT</td>
<td>Message checksum. The message checksum is used to verify the signal path from the transmitting device to the receiving device.</td>
</tr>
<tr>
<td>MessageCounter</td>
<td>INT</td>
<td>Message counter. The message counter is used to record situations in which the transmitting ECU malfunction repeats the same time period.</td>
</tr>
<tr>
<td>EngRequestedTorqueHighResolution</td>
<td>MpJ1939ERTHREnum</td>
<td>This parameter indicates an additional torque as a percentage of the reference engine torque.</td>
</tr>
<tr>
<td>TSC1CtrlPurpose</td>
<td>MpJ1939TSC1CPEnum</td>
<td>Specifies which control mode the transmitting device uses to generate the TSC1 command.</td>
</tr>
<tr>
<td>TSC1TransRate</td>
<td>MpJ1939TSC1TREnum</td>
<td>Specifies the transfer rate at which the transmitting device transmits the TSC1 message.</td>
</tr>
<tr>
<td>EngRequestedTorque_TorqueLimit</td>
<td>INT</td>
<td>Parameter provided to the engine in the torque/speed control message for controlling or limiting the output torque.</td>
</tr>
<tr>
<td>EngRequestedSpeed_SpeedLimit</td>
<td>REAL</td>
<td>Parameters provided to the engine by external sources in the torque/speed control message.</td>
</tr>
</tbody>
</table>
### 5.4.1.1.27 MpJ1939VEP1Type

This data type describes PGN VEP1 (vehicle current 1).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KeyswitchBatteryPotential</td>
<td>REAL</td>
<td>Battery potential measured at the input of the electronic control unit that is supplied by a key switch or similar switching device.</td>
</tr>
<tr>
<td>BatteryPotential_PowerInput1</td>
<td>REAL</td>
<td>This parameter measures the first source of the battery potential, measured at the input of the ECU/actuator, etc., coming from one or more batteries, regardless of the distance between the component and the battery.</td>
</tr>
<tr>
<td>ChargingSystemPotential</td>
<td>REAL</td>
<td>Electrical potential measured at the charging system output.</td>
</tr>
<tr>
<td>AltCurrent</td>
<td>INT</td>
<td>Value of the electrical current flow from the alternator.</td>
</tr>
<tr>
<td>NetBatteryCurrent</td>
<td>INT</td>
<td>Net current in/out of the first battery or first set of batteries used to start the engine, ignite and ignite (SLI).</td>
</tr>
<tr>
<td>Enable</td>
<td>BOOL</td>
<td>Enables the vehicle electrical power.</td>
</tr>
<tr>
<td>Status</td>
<td>MpJ1939PGNStatusEnum</td>
<td>State of the parameter group number.</td>
</tr>
</tbody>
</table>

### 5.4.1.1.28 MpJ1939GenericInfoType

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>Diag</td>
<td>MpJ1939DiagType</td>
<td>Diagnostic structure for the status ID.</td>
</tr>
</tbody>
</table>

### 5.4.1.1.29 MpJ1939ReceiveInfoType

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>MpJ1939DiagType</td>
<td>Diagnostic structure for the status ID.</td>
</tr>
<tr>
<td>ReceivedBytes</td>
<td>UINT</td>
<td>Number of bytes received.</td>
</tr>
</tbody>
</table>

### 5.4.1.1.30 MpJ1939SpecificRequestInfoType

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>MpJ1939DiagType</td>
<td>Diagnostic structure for the status ID.</td>
</tr>
<tr>
<td>ReceivedBytes</td>
<td>UINT</td>
<td>Number of bytes received.</td>
</tr>
</tbody>
</table>

### 5.4.1.1.31 MpJ1939TransmitInfoType

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>MpJ1939DiagType</td>
<td>Diagnostic structure for the status ID.</td>
</tr>
<tr>
<td>ReceivedBytes</td>
<td>UINT</td>
<td>Number of bytes received.</td>
</tr>
</tbody>
</table>

### 5.4.1.1.32 MpJ1939StatusIDType

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>MpJ1939ErrorEnum</td>
<td>Error code for the function block.</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 documentation.</td>
</tr>
</tbody>
</table>

5.4.1.1.33 MpJ1939DiagType

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.

5.4.1.2 Enumerators

5.4.1.2.1 MpJ1939ACHPFSEnum

This enumerated data type indicates whether the pressure in the coolant circuit of an air conditioner is too high and the fan must be switched on.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_ACHPFS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_ACHPFS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_ACHPFS_PRESSURE_HIGH</td>
<td>Pressure is too high.</td>
</tr>
<tr>
<td>mpJ1939_ACHPFS_PRESSURE_NORMAL</td>
<td>Pressure is normal.</td>
</tr>
</tbody>
</table>

5.4.1.2.2 MpJ1939AFT1DPFCNMFAREnum

This enumerated data type specifies the state of diesel particulate filter 1 regeneration under the current engine operating conditions.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_AFT1DPFCNMFAR_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_AFT1DPFCNMFAR_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_AFT1DPFCNMFAR_INHIBIT</td>
<td>Specifies that the active diesel particulate filter regeneration is inhibited.</td>
</tr>
<tr>
<td>mpJ1939_AFT1DPFCNMFAR_NOT_INHIBIT</td>
<td>Specifies that the active diesel particulate filter regeneration is not inhibited.</td>
</tr>
</tbody>
</table>

5.4.1.2.3 MpJ1939AFT1IDPEnum

This enumerated data type specifies whether the temperature on the intake side of the aftertreatment system has exceeded the dew point.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_AFT1IDP_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_AFT1IDP_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_AFT1IDP_EXCEEDED</td>
<td>Dew point exceeded.</td>
</tr>
<tr>
<td>mpJ1939_AFT1IDP_NOT_EXCEED</td>
<td>The dew point is not exceeded.</td>
</tr>
</tbody>
</table>

5.4.1.2.4 MpJ1939AFT1IDPEnum

This enumerated data type specifies whether the temperature on the outlet side of the aftertreatment system has exceeded the dew point.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_AFT1DP_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_AFT1DP_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_AFT1DP_EXCEEDED</td>
<td>The dew point is exceeded.</td>
</tr>
<tr>
<td>mpJ1939_AFT1IDP_NOT_EXCEED</td>
<td>The dew point is not exceeded.</td>
</tr>
</tbody>
</table>

5.4.1.2.5 MpJ1939AFT2EDPEnum

This enumerated data type specifies whether the temperature on the outlet side of the aftertreatment system has exceeded the dew point.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_AFT2EDP_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_AFT2EDP_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_AFT2EDP_EXCEEDED</td>
<td>The dew point was exceeded.</td>
</tr>
<tr>
<td>mpJ1939_AFT2EDP_NOT_EXCEED</td>
<td>The dew point is not exceeded.</td>
</tr>
</tbody>
</table>
5.4.1.2.6 MpJ1939AFT2IDPEnum
This enumerated data type specifies whether the temperature on the intake side of the aftertreatment system has exceeded the dew point.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_AFT2IDP_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_AFT2IDP_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_AFT2IDP_EXCEEDED</td>
<td>The dew point is exceeded.</td>
</tr>
<tr>
<td>mpJ1939_AFT2IDP_NOT_EXCEED</td>
<td>The dew point is not exceeded.</td>
</tr>
</tbody>
</table>

5.4.1.2.7 MpJ1939AFTDPFARSEnum
This enumerated data type indicates the state of the active regeneration of the diesel particulate filter (AFTDP-FARS).

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_AFTDPFARS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_AFTDPFARS_NEEDED</td>
<td>AFTDPFARS is needed.</td>
</tr>
<tr>
<td>mpJ1939_AFTDPFARS_ACTIVE</td>
<td>AFTDPFARS active.</td>
</tr>
<tr>
<td>mpJ1939_AFTDPFARS_NOT_ACTIVE</td>
<td>AFTDPFARS not active.</td>
</tr>
</tbody>
</table>

5.4.1.2.8 MpJ1939AFTDPFPRSSenum
This enumerated data type indicates the state of the passive regeneration of the diesel particulate filter (AFTDPF-PRS).

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_AFTDPFPPRS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_AFTDPFPPRS_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_AFTDPFPPRS_ACTIVE</td>
<td>AFTDPFPPRS active.</td>
</tr>
<tr>
<td>mpJ1939_AFTDPFPPRS_NOT_ACTIVE</td>
<td>AFTDPFPPRS not active.</td>
</tr>
</tbody>
</table>

5.4.1.2.9 MpJ1939AFTDPFSEnum
This enumerated data type indicates the state of regeneration demand and the urgency of the diesel particulate filter (AFTDPFS).

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_AFTDPFS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_AFTDPFS_RESERVED6</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_AFTDPFS_RESERVED5</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_AFTDPFS_RESERVED4</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_AFTDPFS_HIGHEST_LEVEL</td>
<td>AFTDPFS is needed (highest level).</td>
</tr>
<tr>
<td>mpJ1939_AFTDPFS_MODERATE_LEVEL</td>
<td>AFTDPFS is needed (medium level).</td>
</tr>
<tr>
<td>mpJ1939_AFTDPFS_LOWEST_LEVEL</td>
<td>AFTDPFS is needed (low level).</td>
</tr>
<tr>
<td>mpJ1939_AFTDPFS_NOT_NEEDED</td>
<td>AFTDPFS is not needed.</td>
</tr>
</tbody>
</table>

5.4.1.2.10 MpJ1939AFTRFSEnum
This enumerated data type specifies the state of the switch that forces aftertreatment regeneration.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_AFTRFs_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_AFTRFs_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_AFTRFs_ACTIVE</td>
<td>The switch is active.</td>
</tr>
</tbody>
</table>

5.4.1.2.11 MpJ1939AFTRISEnum
This enumerated data type specifies the state of the switch that prevents aftertreatment regeneration.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_AFTRIS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_AFTRIS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_AFTRIS_ACTIVE</td>
<td>The switch is active.</td>
</tr>
</tbody>
</table>

5.4.1.2.12 MpJ1939AGSEEnum
This enumerated data type specifies the state of the automatic transmission.
<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_AGSES_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_AGSES_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_AGSES_AUTO_ENABLED</td>
<td>The gearshift is enabled.</td>
</tr>
<tr>
<td>mpJ1939_AGSES_AUTO_DISABLED</td>
<td>The gearshift is disabled.</td>
</tr>
</tbody>
</table>

### 5.4.1.2.13 MpJ1939AHCPREnum

This enumerated data type specifies whether the cooling water pump of the auxiliary heater should be switched on.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_AHCPR_DONT_CARE</td>
<td>The cooling water pump is waiting.</td>
</tr>
<tr>
<td>mpJ1939_AHCPR_RESERVED</td>
<td>Reserved.</td>
</tr>
<tr>
<td>mpJ1939_AHCPR_ACTIVATE</td>
<td>The cooling water pump must be activated.</td>
</tr>
<tr>
<td>mpJ1939_AHCPR_DEACTIVATE</td>
<td>The cooling water pump must be deactivated.</td>
</tr>
</tbody>
</table>

### 5.4.1.2.14 MpJ1939AHMREnum

This enumerated data type specifies the state of the auxiliary heater.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_AHMR_DONT_CARE</td>
<td>The auxiliary heater is waiting.</td>
</tr>
<tr>
<td>mpJ1939_AHMR_RESERVED</td>
<td>Reserved.</td>
</tr>
<tr>
<td>mpJ1939_AHMR_HEATER_PUMP_UPKEEP</td>
<td>The heating pump is running.</td>
</tr>
<tr>
<td>mpJ1939_AHMR_NORMAL_MODE</td>
<td>The auxiliary heater is in &quot;normal mode&quot;.</td>
</tr>
<tr>
<td>mpJ1939_AHMR_ECONOMY_MODE</td>
<td>The auxiliary heater is in &quot;economy mode&quot;.</td>
</tr>
<tr>
<td>mpJ1939_AHMR_OFF_HAZARDOUS</td>
<td>The auxiliary heater is switched off due to hazardous materials.</td>
</tr>
<tr>
<td>mpJ1939_AHMR_DEACTIVATE_AUX</td>
<td>The auxiliary heater is deactivated.</td>
</tr>
</tbody>
</table>

### 5.4.1.2.15 MpJ1939AP1LISEnum

This enumerated data type specifies the state of the low idle switch on accelerator pedal 1.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_AP1LIS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_AP1LIS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_AP1LIS_IN_LOW_IDLE</td>
<td>The idle switch is in the low idle state.</td>
</tr>
<tr>
<td>mpJ1939_AP1LIS_NOT_IN_LOW_IDLE</td>
<td>The idle switch is not in the low idle state.</td>
</tr>
</tbody>
</table>

### 5.4.1.2.16 MpJ1939AP2LISEnum

This enumerated data type specifies the state of the low idle switch on accelerator pedal 2.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_AP2LIS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_AP2LIS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_AP2LIS_IN_LOW_IDLE</td>
<td>The idle switch is in the low idle state.</td>
</tr>
<tr>
<td>mpJ1939_AP2LIS_NOT_IN_LOW_IDLE</td>
<td>The idle switch is not in the low idle state.</td>
</tr>
</tbody>
</table>

### 5.4.1.2.17 MpJ1939APKSEnum

This enumerated data type specifies whether the accelerator pedal kick-down switch is open or closed.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_APKS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_APKS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_APKS_KICKDOWN_ACTIVE</td>
<td>The accelerator pedal kick-down switch is active.</td>
</tr>
<tr>
<td>mpJ1939_APKS_KICKDOWN_PASSIVE</td>
<td>The accelerator pedal kick-down switch is not active.</td>
</tr>
</tbody>
</table>

### 5.4.1.2.18 MpJ1939AWLSEnum

This enumerated data type indicates the state of the yellow warning lamp.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_AWLS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_AWLS_RESERVED</td>
<td>The yellow warning lamp is reserved.</td>
</tr>
<tr>
<td>mpJ1939_AWLS_LAMP_ON</td>
<td>The yellow warning lamp is on.</td>
</tr>
<tr>
<td>mpJ1939_AWLS_LAMP_OFF</td>
<td>The yellow warning lamp is switched off.</td>
</tr>
</tbody>
</table>

### 5.4.1.2.19 MpJ1939BMSHREnum

This enumerated data type specifies whether the main switch of the battery must be held.
This enumerated data type specifies the state of the brake pedal.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_BS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_BS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_BS_DEPRESSED</td>
<td>The brake pedal is pressed.</td>
</tr>
<tr>
<td>mpJ1939_BS_RELEASED</td>
<td>The brake pedal is not pressed.</td>
</tr>
</tbody>
</table>

This enumerated data type specifies the state of the cruise control switch.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_CCAS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_CCAS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_CCAS_IN_POS_ACCEL</td>
<td>The switch is in position &quot;Accelerate&quot;.</td>
</tr>
<tr>
<td>mpJ1939_CCAS_NOT_IN_POS_ACCEL</td>
<td>The switch is not in position &quot;Accelerate&quot;.</td>
</tr>
</tbody>
</table>

This enumerated data type specifies whether cruise control should be controlled by a source other than the driver.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_CCDC_DONT_CARE</td>
<td>Cruise control is waiting.</td>
</tr>
<tr>
<td>mpJ1939_CCDC_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_CCDC_NOT_ALLOWED</td>
<td>Cruise control is not controlled by another source.</td>
</tr>
<tr>
<td>mpJ1939_CCDC_IS_ALLOWED</td>
<td>Cruise control is controlled by another source.</td>
</tr>
</tbody>
</table>

This enumerated data type specifies how cruise control can be modified.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_CCES_NOTAVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_CCES_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_CCES_ENABLED</td>
<td>Cruise control can be modified.</td>
</tr>
<tr>
<td>mpJ1939_CCES_DISABLED</td>
<td>Cruise control modification is disabled.</td>
</tr>
</tbody>
</table>

This enumerated data type specifies whether the possibility of controlling cruise control from a source other than the driver should be temporarily disabled.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_CCPC_DONT_CARE</td>
<td>Cruise control is waiting.</td>
</tr>
<tr>
<td>mpJ1939_CCPC_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_CCPC_NOT_ALLOWED</td>
<td>Cruise control is not controlled by another source.</td>
</tr>
<tr>
<td>mpJ1939_CCPC_IS_ALLOWED</td>
<td>Cruise control is controlled by another source.</td>
</tr>
</tbody>
</table>
5.4.1.2.27 MpJ1939CCPSEnum

This enumerated data type specifies the state of the cruise control interrupt switch.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_CCPS_TAKE_NO_ACTION</td>
<td>Cruise control interrupt switch</td>
</tr>
<tr>
<td>mpJ1939_CCPS_ERROR_INDICATOR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_CCPS_ON</td>
<td>The cruise control interrupt switch is switched on.</td>
</tr>
<tr>
<td>mpJ1939_CCPS_OFF</td>
<td>The cruise control interrupt switch is switched off.</td>
</tr>
</tbody>
</table>

5.4.1.2.28 MpJ1939CCRCEnum

This enumerated data type specifies whether cruise control should be controlled by a source other than the driver.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_CCRC_DONT_CARE</td>
<td>Cruise control is waiting.</td>
</tr>
<tr>
<td>mpJ1939_CCRC_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_CCRC_REQUESTED</td>
<td>Cruise control is controlled by another source.</td>
</tr>
<tr>
<td>mpJ1939_CCRC_NOT_REQUESTED</td>
<td>Cruise control is not controlled by another source.</td>
</tr>
</tbody>
</table>

5.4.1.2.29 MpJ1939CCRSEnum

This enumerated data type specifies the state of the cruise control switch.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_CCRS_NOT_AVAILABLE</td>
<td>The switch is not available.</td>
</tr>
<tr>
<td>mpJ1939_CCRS_ERROR</td>
<td>The switch is in an error state.</td>
</tr>
<tr>
<td>mpJ1939_CCRS_IN_POS_RESUME</td>
<td>The switch is in position &quot;Resume&quot;.</td>
</tr>
<tr>
<td>mpJ1939_CCRS_NOT_IN_POS_RESUME</td>
<td>The switch is not in position &quot;Resume&quot;.</td>
</tr>
</tbody>
</table>

5.4.1.2.30 MpJ1939CCSEnum

This enumerated data type specifies the state of the cruise control.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_CCS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_CCS_ACCELERATOR_OVERRIDE</td>
<td>Cruise control is in acceleration override mode.</td>
</tr>
<tr>
<td>mpJ1939_CCS_SET</td>
<td>Cruise control is in position &quot;Set&quot;.</td>
</tr>
<tr>
<td>mpJ1939_CCS_RESUME</td>
<td>Cruise control is in position &quot;Resume&quot;.</td>
</tr>
<tr>
<td>mpJ1939_CCS_DECELERATE</td>
<td>Cruise control is in position &quot;Decelerate&quot;.</td>
</tr>
<tr>
<td>mpJ1939_CCS_ACCELERATE</td>
<td>Cruise control is in position &quot;Accelerate&quot;.</td>
</tr>
<tr>
<td>mpJ1939_CCS_HOLD</td>
<td>Cruise control is in position &quot;Hold&quot;.</td>
</tr>
<tr>
<td>mpJ1939_CCS_OFF</td>
<td>Cruise control is switched off.</td>
</tr>
</tbody>
</table>

5.4.1.2.31 MpJ1939CCSSEnum

This enumerated data type specifies the state of the cruise control switch.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_CCSS_NOT_AVAILABLE</td>
<td>The switch is not available.</td>
</tr>
<tr>
<td>mpJ1939_CCSS_ERROR</td>
<td>The switch is in an error state.</td>
</tr>
<tr>
<td>mpJ1939_CCSS_IN_POS_SET</td>
<td>Switch is in position &quot;Set&quot;.</td>
</tr>
<tr>
<td>mpJ1939_CCSS_NOT_IN_POS_SET</td>
<td>Switch is not in position &quot;Set&quot;.</td>
</tr>
</tbody>
</table>

5.4.1.2.32 MpJ1939CLEPSSEnum

This enumerated data type specifies the configuration of the engine protection shutdown of the coolant level.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_CLEPSS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_CLEPSS_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_CLEPSS_ENABLED</td>
<td>Engine protection shutdown of the coolant level enabled.</td>
</tr>
<tr>
<td>mpJ1939_CLEPSS_DISABLED</td>
<td>Engine protection shutdown of the coolant level not enabled.</td>
</tr>
</tbody>
</table>

5.4.1.2.33 MpJ1939CSEnum

This enumerated data type specifies the state of the clutch pedal.
### 5.4.1.2.34 MpJ1939DPFAARIEnum

This enumerated data type specifies the state of the active regeneration inhibition of the diesel particulate filter (DPFAARI) since the vehicle speed is less than the permissible vehicle speed.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_DPFAARI_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_DPFAARI_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DPFAARI_ENABLED</td>
<td>DPFAARI enabled.</td>
</tr>
<tr>
<td>mpJ1939_DPFAARI_NOT_ENABLED</td>
<td>DPFAARI not enabled.</td>
</tr>
</tbody>
</table>

### 5.4.1.2.35 MpJ1939DPFARFSEnum

This enumerated data type specifies the execution state of the diesel particle filter regeneration (DPFARFS).

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_DPFARFS_RESERVED3</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DPFARFS_SERVICE_TOOL</td>
<td>DPFARFS enabled by service tool.</td>
</tr>
<tr>
<td>mpJ1939_DPFARFS_SWITCH</td>
<td>DPFARFS enabled by switch.</td>
</tr>
<tr>
<td>mpJ1939_DPFARFS_NOT_ACTIVE</td>
<td>DPFARFS not enabled.</td>
</tr>
<tr>
<td>mpJ1939_DPFARFS_NOT_AVAILABLE</td>
<td>DPFARFS not available.</td>
</tr>
<tr>
<td>mpJ1939_DPFARFS_RESERVED6</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DPFARFS_RESERVED5</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DPFARFS_RESERVED4</td>
<td>Reserved for SAE assignment.</td>
</tr>
</tbody>
</table>

### 5.4.1.2.36 MpJ1939DPFARI3710Enum

This enumerated data type indicates the state of the diesel particulate filter's active regeneration inhibition (DPFARI3710) since the parking brake is not set.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_DPFARI3710_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_DPFARI3710_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DPFARI3710_INHIBITED</td>
<td>DPFARI3710 is inhibited.</td>
</tr>
<tr>
<td>mpJ1939_DPFARI3710_NOT_INHIBITED</td>
<td>DPFARI3710 is not inhibited.</td>
</tr>
</tbody>
</table>

### 5.4.1.2.37 MpJ1939DPFARI3712Enum

This enumerated data type indicates the state of the active regeneration inhibition of the diesel particulate filter (DPFARI3712) due to an active system fault.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_DPFARI3712_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_DPFARI3712_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DPFARI3712_INHIBITED</td>
<td>DPFARI3712 is inhibited.</td>
</tr>
<tr>
<td>mpJ1939_DPFARI3712_NOT_INHIBITED</td>
<td>DPFARI3712 is not inhibited.</td>
</tr>
</tbody>
</table>

### 5.4.1.2.38 MpJ1939DPFARI3713Enum

This enumerated data type indicates the state of the active regeneration inhibition of the diesel particulate filter (DPFARI3713) due to a system timeout.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_DPFARI3713_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_DPFARI3713_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DPFARI3713_INHIBITED</td>
<td>DPFARI3713 is inhibited.</td>
</tr>
<tr>
<td>mpJ1939_DPFARI3713_NOT_INHIBITED</td>
<td>DPFARI3713 is not inhibited.</td>
</tr>
</tbody>
</table>

### 5.4.1.2.39 MpJ1939DPFARI3715Enum

This enumerated data type indicates the state of the active regeneration inhibition of the diesel particulate filter (DPFARI3715) due to a permanent system lock.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_DPFARI3715_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_DPFARI3715_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DPFARI3715_INHIBITED</td>
<td>DPFARI3715 inhibited.</td>
</tr>
<tr>
<td>mpJ1939_DPFARI3715_NOT_INHIBITED</td>
<td>DPFARI3715 not inhibited.</td>
</tr>
</tbody>
</table>
5.4.1.2.40 MpJ1939DPFARI3717Enum

This enumerated data type specifies the state of the active regeneration inhibition of the diesel particulate filter (DPFARI3717) since the vehicle speed is less than the permissible vehicle speed.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_DPFARI3717_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_DPFARI3717_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DPFARI3717_INHIBITED</td>
<td>DPFARI3717 active.</td>
</tr>
<tr>
<td>mpJ1939_DPFARI3717_NOT_INHIBITED</td>
<td>DPFARI3717 not active.</td>
</tr>
</tbody>
</table>

5.4.1.2.41 MpJ1939DPFARI5629Enum

This enumerated data type specifies the state of the active regeneration inhibition of the diesel particulate filter due to the low exhaust gas pressure (DPFARI).

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_DPFARI5629_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_DPFARI5629_RESERVED</td>
<td>DPFARI is reserved.</td>
</tr>
<tr>
<td>mpJ1939_DPFARI5629_INHIBIT</td>
<td>DPFARI inhibited.</td>
</tr>
<tr>
<td>mpJ1939_DPFARI5629_NOT_INHIBIT</td>
<td>DPFARI not inhibited.</td>
</tr>
</tbody>
</table>

5.4.1.2.42 MpJ1939DPFARIDTAEnum

This enumerated data type indicates the state of the diesel particulate filter's active regeneration inhibition (DPFARIDTA) since the accelerator pedal is idle.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_DPFARIDTA_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTA_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTA_INHIBITED</td>
<td>DPFARIDTA is inhibited.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTA_NOT_INHIBITED</td>
<td>DPFARIDTA is not inhibited.</td>
</tr>
</tbody>
</table>

5.4.1.2.43 MpJ1939DPFARIDTCEnum

This enumerated data type indicates the state of the diesel particulate filter's active regeneration inhibition (DPFARIDTC) since the clutch is disengaged.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_DPFARIDTC_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTC_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTC_INHIBITED</td>
<td>DPFARIDTC is inhibited.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTC_NOT_INHIBITED</td>
<td>DPFARIDTC is not inhibited.</td>
</tr>
</tbody>
</table>

5.4.1.2.44 MpJ1939DPFARIDTEEEnum

This enumerated data type indicates the state of the diesel particulate filter's active regeneration inhibition (DPFARIDTE) since the engine is not warmed up.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_DPFARIDTE_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTE_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTE_INHIBITED</td>
<td>DPFARIDTE is inhibited.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTE_NOT_INHIBITED</td>
<td>DPFARIDTE is not inhibited.</td>
</tr>
</tbody>
</table>

5.4.1.2.45 MpJ1939DPFARIDTIEEnum

This enumerated data type indicates the state of the active regeneration inhibition of the diesel particulate filter caused by the diesel particulate filter regeneration suppression switch (DPFARIDTI).

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_DPFARIDTI_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTI_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTI_INHIBITED</td>
<td>DPFARIDTI is inhibited.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTI_NOT_INHIBITED</td>
<td>DPFARIDTI is not inhibited.</td>
</tr>
</tbody>
</table>

5.4.1.2.46 MpJ1939DPFARIDTLEEnum

This enumerated data type indicates the state of the diesel particulate filter's active regeneration inhibition (DPFARIDTL) since the exhaust gas temperature is too low.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_DPFARIDTL_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTL_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTL_INHIBITED</td>
<td>DPFARIDTL is inhibited.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTL_NOT_INHIBITED</td>
<td>DPFARIDTL is not inhibited.</td>
</tr>
</tbody>
</table>
5.4.1.2.47 MpJ1939DPFARIDTEnum
This enumerated data type indicates the state of the diesel particulate filter’s active regeneration inhibition (DPFARIDTO) since the gearbox is not idling.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_DPFARIDTO_NOT_AVAIL</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTO_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTO_INHIBITED</td>
<td>DPFARIDTO is inhibited.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTO_NOT_INHIBITED</td>
<td>DPFARIDTO is not inhibited.</td>
</tr>
</tbody>
</table>

5.4.1.2.48 MpJ1939DPFARIDTPEnum
This enumerated data type indicates the state of the diesel particulate filter’s active regeneration inhibition (DPFARIDTP) since the PTO shaft is active.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_DPFARIDTP_NOT_AVAIL</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTP_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTP_INHIBITED</td>
<td>DPFARIDTP is inhibited.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTP_NOT_INHIBITED</td>
<td>DPFARIDTP is not inhibited.</td>
</tr>
</tbody>
</table>

5.4.1.2.49 MpJ1939DPFARIDTSEnum
This enumerated data type indicates the state of the diesel particulate filter’s active regeneration inhibition (DPFARIDTS) since the service brake is activated.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_DPFARIDTS_NOT_AVAIL</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTS_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTS_INHIBITED</td>
<td>DPFARIDTS is inhibited.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTS_NOT_INHIBITED</td>
<td>DPFARIDTS is not inhibited.</td>
</tr>
</tbody>
</table>

5.4.1.2.50 MpJ1939DPFARIDTTEnum
This enumerated data type indicates the state of the active regeneration inhibition of the diesel particulate filter (DPFARIDTT) due to a temporary system lock.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_DPFARIDTT_NOT_AVAIL</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTT_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTT_INHIBITED</td>
<td>DPFARIDTT is inhibited.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTT_NOT_INHIBITED</td>
<td>DPFARIDTT is not inhibited.</td>
</tr>
</tbody>
</table>

5.4.1.2.51 MpJ1939DPFARIDTVEnum
This enumerated data type indicates the state of the diesel particulate filter’s active regeneration inhibition (DPFARIDTV) since the vehicle speed is above a permissible limit.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_DPFARIDTV_NOT_AVAIL</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTV_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTV_INHIBITED</td>
<td>DPFARIDTV is inhibited.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIDTV_NOT_INHIBITED</td>
<td>DPFARIDTV is not inhibited.</td>
</tr>
</tbody>
</table>

5.4.1.2.52 MpJ1939DPFARISEnum
This enumerated data type indicates the state of the active regeneration inhibition of the diesel particulate filter (DPFARIS).

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_DPFARIS_NOT_AVAIL</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIS_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIS_INHIBITED</td>
<td>DPFARIS is inhibited.</td>
</tr>
<tr>
<td>mpJ1939_DPFARIS_NOT_INHIBITED</td>
<td>DPFARIS is not inhibited.</td>
</tr>
</tbody>
</table>

5.4.1.2.53 MpJ1939DPFLECEnum
This enumerated data type indicates the state of the diesel particulate filter lamp.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_DPFLC_NOT_AVAIL</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_DPFLC_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DPFLC_INHIBITED</td>
<td>The diesel particulate filter lamp is blinking.</td>
</tr>
<tr>
<td>mpJ1939_DPFLC_NOT_INHIBITED</td>
<td></td>
</tr>
</tbody>
</table>
### 5.4.1.2.54 MpJ1939DPFTMAEnum

This enumerated data type indicates the state of exhaust gas temperatures for regeneration of the diesel particulate filter aftertreatment system or for preparation of regeneration of the diesel particulate treatment system (DPFTMA).

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mpJ1939_DPFTMA_DONT_CARE</code></td>
<td>DPFTMA is waiting.</td>
</tr>
<tr>
<td><code>mpJ1939_DPFTMA_RESERVED</code></td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td><code>mpJ1939_DPFTMA_ACTIVE</code></td>
<td>DPFTMA active.</td>
</tr>
<tr>
<td><code>mpJ1939_DPFTMA_NOT_ACTIVE</code></td>
<td>DPFTMA not active.</td>
</tr>
</tbody>
</table>

### 5.4.1.2.55 MpJ1939EAAEnum

This enumerated data type indicates whether engine alarm acknowledgment is active.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mpJ1939_EAA_NOT_AVAILABLE</code></td>
<td>State not available.</td>
</tr>
<tr>
<td><code>mpJ1939_EAA_ERROR</code></td>
<td>Engine alarm acknowledgment is in an error state.</td>
</tr>
<tr>
<td><code>mpJ1939_EAA_ACTIVE</code></td>
<td>Engine alarm acknowledgment active.</td>
</tr>
<tr>
<td><code>mpJ1939_EAA_NOT_ACTIVE</code></td>
<td>Engine alarm acknowledgment not active.</td>
</tr>
</tbody>
</table>

### 5.4.1.2.56 MpJ1939EAOCSEnum

This enumerated data type indicates when the alarm driver output is activated.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mpJ1939_EAOCS_NOT_AVAILABLE</code></td>
<td>State not available.</td>
</tr>
<tr>
<td><code>mpJ1939_EAOCS_RESERVED</code></td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td><code>mpJ1939_EAOCS_ACTIVE</code></td>
<td>Alarm driver output active.</td>
</tr>
<tr>
<td><code>mpJ1939_EAOCS_NOT_ACTIVE</code></td>
<td>Alarm driver output not active.</td>
</tr>
</tbody>
</table>

### 5.4.1.2.57 MpJ1939EASCSEnum

This enumerated data type indicates if the air shutoff driver output is activated.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mpJ1939_EASCS_NOT_AVAILABLE</code></td>
<td>State not available.</td>
</tr>
<tr>
<td><code>mpJ1939_EASCS_RESERVED</code></td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td><code>mpJ1939_EASCS_ENABLED</code></td>
<td>Air shutoff driver output enabled.</td>
</tr>
<tr>
<td><code>mpJ1939_EASCS_DISABLED</code></td>
<td>Air shutoff driver output not enabled.</td>
</tr>
</tbody>
</table>

### 5.4.1.2.58 MpJ1939EASESEnum

This enumerated data type specifies the state of the idle management system.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mpJ1939_EASES_NOT_AVAILABLE</code></td>
<td>State not available.</td>
</tr>
<tr>
<td><code>mpJ1939_EASES_ERROR</code></td>
<td>Error state.</td>
</tr>
<tr>
<td><code>mpJ1939_EASES_ON_STATE</code></td>
<td>The idle management system is switched on.</td>
</tr>
<tr>
<td><code>mpJ1939_EASES_OFF_STATE</code></td>
<td>The idle management system is switched off.</td>
</tr>
</tbody>
</table>

### 5.4.1.2.59 MpJ1939EASSEEnum

This enumerated data type specifies whether the actual measured position of the air cutoff is indicated or not.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mpJ1939_EASS_ERROR</code></td>
<td>Error state.</td>
</tr>
<tr>
<td><code>mpJ1939_EASS_RESTRICTED</code></td>
<td>The position is not permitted.</td>
</tr>
<tr>
<td><code>mpJ1939_EASS_ALLOWED</code></td>
<td>The position is permitted.</td>
</tr>
<tr>
<td><code>mpJ1939_EASS_NOT_AVAILABLE</code></td>
<td>Position not available.</td>
</tr>
</tbody>
</table>

### 5.4.1.2.60 MpJ1939EIDSEnum

This enumerated data type specifies the state of the idle decrement switch.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mpJ1939_EIDS_NOT_AVAILABLE</code></td>
<td>State not available.</td>
</tr>
<tr>
<td><code>mpJ1939_EIDS_ERROR</code></td>
<td>Error state.</td>
</tr>
<tr>
<td><code>mpJ1939_EIDS_ON</code></td>
<td>The switch is enabled.</td>
</tr>
<tr>
<td><code>mpJ1939_EIDS_OFF</code></td>
<td>The switch is disabled.</td>
</tr>
</tbody>
</table>
5.4.1.2.61 MpJ1939EIIEnum
This enumerated data type specifies the state of the idle increment switch.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_EIIS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_EIIS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_EIIS_ON</td>
<td>The switch is enabled.</td>
</tr>
<tr>
<td>mpJ1939_EIIS_OFF</td>
<td>The switch is disabled.</td>
</tr>
</tbody>
</table>

5.4.1.2.62 MpJ1939EISDAMEnum
This enumerated data type indicates the state of the driver alert mode of the idle shutdown timer system.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_EISDAM_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_EISDAM_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_EISDAM_ACTIVE</td>
<td>The driver alert mode of the idle shutdown timer system is active.</td>
</tr>
<tr>
<td>mpJ1939_EISDAM_INACTIVE</td>
<td>The driver alert mode of the idle shutdown timer system is not active.</td>
</tr>
</tbody>
</table>

5.4.1.2.63 MpJ1939EISHSEEnum
This enumerated data type specifies whether or not the engine has been shut down by the idle shutdown timer system.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_EISHSE_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_EISHSE_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_EISHSE_YES</td>
<td>The engine was shut down.</td>
</tr>
<tr>
<td>mpJ1939_EISHSE_NO</td>
<td>The engine was not shut down.</td>
</tr>
</tbody>
</table>

5.4.1.2.64 MpJ1939EISTFEnum
This enumerated data type specifies the configuration of the idle shutdown timer system.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_EISTF_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_EISTF_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_EISTF_ENABLED</td>
<td>The idle shutdown timer system is enabled.</td>
</tr>
<tr>
<td>mpJ1939_EISTF_DISABLED</td>
<td>The idle shutdown timer system is disabled.</td>
</tr>
</tbody>
</table>

5.4.1.2.65 MpJ1939EISTOEnum
This enumerated data type indicates the state of the override function of the idle shutdown timer system.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_EISTO_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_EISTO_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_EISTO_ACTIVE</td>
<td>The override function of the idle shutdown timer system is active.</td>
</tr>
<tr>
<td>mpJ1939_EISTO_INACTIVE</td>
<td>The override function of the idle shutdown timer system is not active.</td>
</tr>
</tbody>
</table>

5.4.1.2.66 MpJ1939EISTSEnum
This enumerated data type indicates the current operating mode of the idle shutdown timer system.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_EISTS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_EISTS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_EISTS_ACTIVE</td>
<td>Idle shutdown timer system active.</td>
</tr>
<tr>
<td>mpJ1939_EISTS_INACTIVE</td>
<td>Idle shutdown timer system not active.</td>
</tr>
</tbody>
</table>

5.4.1.2.67 MpJ1939EOCMEnum
This enumerated data type defines which type of command is used for the override control mode.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_EOCM_SPEED</td>
<td>Speed control</td>
</tr>
<tr>
<td>mpJ1939_EOCM_TORQUE_CONTROL</td>
<td>Torque control</td>
</tr>
<tr>
<td>mpJ1939_EOCM_SPEED_CONTROL</td>
<td>Speed/Torque limiting</td>
</tr>
<tr>
<td>mpJ1939_EOCM_OVERRIDE_DISABLED</td>
<td>Override control mode disabled.</td>
</tr>
</tbody>
</table>
5.4.1.2.68 MpJ1939EOTEnum

This enumerated data type specifies the state of the engine overspeed test signal as measured by the reporting ECM.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_EOT_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_EOT_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_EOT_ACTIVE</td>
<td>Engine overspeed test signal active.</td>
</tr>
<tr>
<td>mpJ1939_EOT_NOT_ACTIVE</td>
<td>Engine overspeed test signal not active.</td>
</tr>
</tbody>
</table>

5.4.1.2.69 MpJ1939EPSASEnum

This enumerated data type indicates whether engine shutdown is imminent.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_EPSAS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_EPSAS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_EPSAS_APPROACHING</td>
<td>Engine shutdown is imminent.</td>
</tr>
<tr>
<td>mpJ1939_EPSAS_NOT_APPROACHING</td>
<td>Engine shutdown is not imminent.</td>
</tr>
</tbody>
</table>

5.4.1.2.70 MpJ1939EPSCEnum

This enumerated data type specifies the configuration of the engine shutdown system.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_EPSC_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_EPSC_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_EPSC_ENABLED</td>
<td>The engine shutdown system is enabled.</td>
</tr>
<tr>
<td>mpJ1939_EPSC_DISABLED</td>
<td>The engine shutdown system is disabled.</td>
</tr>
</tbody>
</table>

5.4.1.2.71 MpJ1939EPSHSEEnum

This enumerated data type indicates whether the engine protection system has shut down the engine.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_EPSHSE_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_EPSHSE_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_EPSHSE_YES</td>
<td>The engine protection system has shut down the engine.</td>
</tr>
<tr>
<td>mpJ1939_EPSHSE_NO</td>
<td>The engine protection system has not shut down the engine.</td>
</tr>
</tbody>
</table>

5.4.1.2.72 MpJ1939EPSTOEnum

This enumerated data type specifies the state of the override function of the engine protection system timer.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_EPSTO_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_EPSTO_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_EPSTO_ACTIVE</td>
<td>Override function active.</td>
</tr>
<tr>
<td>mpJ1939_EPSTO_INACTIVE</td>
<td>Override function not active.</td>
</tr>
</tbody>
</table>

5.4.1.2.73 MpJ1939EPSTSEnum

This enumerated data type specifies the mode of the engine protection system's timer system.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_EPSTS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_EPSTS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_EPSTS_ACTIVE</td>
<td>Timer system active.</td>
</tr>
<tr>
<td>mpJ1939_EPSTS_INACTIVE</td>
<td>Timer system not active.</td>
</tr>
</tbody>
</table>

5.4.1.2.74 MpJ1939ERSCCEnum

This enumerated data type defines the controller characteristic curves that should be used during speed control (ERSCC).

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_ERSCC_ENGAGED_AND3</td>
<td>The drive train is engaged and in lockup state 2.</td>
</tr>
<tr>
<td>mpJ1939_ERSCC_ENGAGED_AND2</td>
<td>The drive train is engaged and in lockup state 1.</td>
</tr>
<tr>
<td>mpJ1939_ERSCC_DISENGA_NONLOCKUP1</td>
<td>The drive train is not engaged and not in lockup state 1.</td>
</tr>
<tr>
<td>mpJ1939_ERSCC_DISENGA_NONLOCKUP2</td>
<td>The drive train is not engaged and not in lockup state 2.</td>
</tr>
</tbody>
</table>
5.4.1.2.75 MpJ1939ERTHREnum
This enumerated data type specifies the additional torque that should be used as a percentage of the reference engine torque.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_ERTHR_0875</td>
<td>0.875% is used.</td>
</tr>
<tr>
<td>mpJ1939_ERTHR_0125</td>
<td>0.125% is used.</td>
</tr>
<tr>
<td>mpJ1939_ERTHR_0000</td>
<td>The torque should not be increased.</td>
</tr>
</tbody>
</table>

5.4.1.2.76 MpJ1939ESHTLCEnum
This enumerated data type specifies the state of the high-temperature lamp in the exhaust system.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_ESHTLC_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_ESHTLC_RESERVED6</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_ESHTLC_RESERVED5</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_ESHTLC_RESERVED4</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_ESHTLC_RESERVED3</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_ESHTLC_RESERVED2</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_ESHTLC_ON</td>
<td>The lamp is switched on.</td>
</tr>
<tr>
<td>mpJ1939_ESHTLC_OFF</td>
<td>The lamp is switched off.</td>
</tr>
</tbody>
</table>

5.4.1.2.77 MpJ1939ESMEnum
This enumerated data type specifies why a starter did not work.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_ESM_INHIBIT_OVERTEMP</td>
<td>Started inhibited due to overheating.</td>
</tr>
<tr>
<td>mpJ1939_ESM_INHIBIT_IMMobilIZER</td>
<td>Starter inhibited due to immobilizer.</td>
</tr>
<tr>
<td>mpJ1939_ESM_INHIBIT_ENGAGED</td>
<td>Starter inhibited due to an indented drive train or another transmission inhibitor.</td>
</tr>
<tr>
<td>mpJ1939_ESM_INHIBIT_ENGNOTRDY</td>
<td>Starter inhibited since the engine is not yet ready.</td>
</tr>
<tr>
<td>mpJ1939_ESM_INHIBIT_ENGINGURING</td>
<td>Starter inhibited due to running engine</td>
</tr>
<tr>
<td>mpJ1939_ESM_FINISHED</td>
<td>Start completed.</td>
</tr>
<tr>
<td>mpJ1939_ESM_ACTIVE_GEAR_ENGAGED</td>
<td>Starter active, gear is engaged.</td>
</tr>
<tr>
<td>mpJ1939_ESM_ACTIVE_GEARNOTENGA</td>
<td>Starter active, gear is not engaged.</td>
</tr>
<tr>
<td>mpJ1939_ESM_NOT_REQUESTED</td>
<td>Starter was not requested.</td>
</tr>
<tr>
<td>mpJ1939_ESM_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_ESM_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_ESM_ERROR_LEGACY</td>
<td>Starter is in an old error state.</td>
</tr>
<tr>
<td>mpJ1939_ESM_INHIBITED</td>
<td>Starter is inhibited.</td>
</tr>
<tr>
<td>mpJ1939_ESM_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_ESM_INHIBIT_ACTIVEINDUC</td>
<td>Starter inhibited due to active SCR triggering.</td>
</tr>
<tr>
<td>mpJ1939_ESM_INHIBIT_AIR_SHTVALV</td>
<td>Starter inhibited because the intake air shutoff valve is active.</td>
</tr>
</tbody>
</table>

5.4.1.2.78 MpJ1939ESOSEnum
This enumerated data type specifies the state of the switch.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_ESOS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_ESOS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_ESOS_ON</td>
<td>The switch is enabled.</td>
</tr>
<tr>
<td>mpJ1939_ESOS_OFF</td>
<td>The switch is disabled.</td>
</tr>
</tbody>
</table>

5.4.1.2.79 MpJ1939ETMSEnum
This enumerated data type specifies the state of the engine diagnostic test mode switch.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_ETMS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_ETMS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_ETMS_ON</td>
<td>The switch is enabled.</td>
</tr>
<tr>
<td>mpJ1939_ETMS_OFF</td>
<td>The switch is disabled.</td>
</tr>
</tbody>
</table>

5.4.1.2.80 MpJ1939EWTSLEnum
This enumerated data type indicates whether the engine is too cold to start.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_EWTSL_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_EWTSL_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_EWTSL_ON</td>
<td>The engine can be started.</td>
</tr>
<tr>
<td>mpJ1939_EWTSL_OFF</td>
<td>The engine cannot be started.</td>
</tr>
</tbody>
</table>
5.4.1.2.81 MpJ1939FAWLEnum
This enumerated data type defines the behavior of the yellow warning lamp.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_FAWL_UNAVAILABLE</td>
<td>The yellow warning lamp is not blinking.</td>
</tr>
<tr>
<td>mpJ1939_FAWL_RESERVED</td>
<td>The yellow warning lamp is reserved.</td>
</tr>
<tr>
<td>mpJ1939_FAWL_FAST_FLASH</td>
<td>The yellow warning lamp blinks at ≥2 Hz (50% duty cycle)</td>
</tr>
<tr>
<td>mpJ1939_FAWL_SLOW_FLASH</td>
<td>The yellow warning lamp blinks at 1 Hz (50% duty cycle)</td>
</tr>
</tbody>
</table>

5.4.1.2.82 MpJ1939FMILEnum
This enumerated data type defines the behavior of the fault indicator lamp.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_FMIL_UNAVAILABLE</td>
<td>The fault indicator lamp is not blinking.</td>
</tr>
<tr>
<td>mpJ1939_FMIL_CLASS_C_DTC</td>
<td>Fault indicator lamp is operated in &quot;Class C DTC&quot;.</td>
</tr>
<tr>
<td>mpJ1939_FMIL_FAST_FLASH</td>
<td>The fault indicator lamp blinks at ≥2 Hz (50% duty cycle)</td>
</tr>
<tr>
<td>mpJ1939_FMIL_SLOW_FLASH</td>
<td>The fault indicator lamp blinks at 1 Hz (50% duty cycle)</td>
</tr>
</tbody>
</table>

5.4.1.2.83 MpJ1939FPLEnum
This enumerated data type defines the behavior of the engine protection lamp.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_FPL_UNAVAILABLE</td>
<td>The engine protection lamp is not blinking.</td>
</tr>
<tr>
<td>mpJ1939_FPL_RESERVED</td>
<td>The engine protection lamp is reserved.</td>
</tr>
<tr>
<td>mpJ1939_FPL_FAST_FLASH</td>
<td>The engine protection lamp blinks at ≥2 Hz (50% duty cycle)</td>
</tr>
<tr>
<td>mpJ1939_FPL_SLOW_FLASH</td>
<td>The engine protection lamp blinks at 1 Hz (50% duty cycle)</td>
</tr>
</tbody>
</table>

5.4.1.2.84 MpJ1939FRSLEnum
This enumerated data type defines the behavior of the red brake lamp.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_FRSL_UNAVAILABLE</td>
<td>Red brake lamp not available</td>
</tr>
<tr>
<td>mpJ1939_FRSL_RESERVED</td>
<td>The red brake lamp is reserved.</td>
</tr>
<tr>
<td>mpJ1939_FRSL_FAST_FLASH</td>
<td>The red brake lamp blinks at ≥2 Hz (50% duty cycle)</td>
</tr>
<tr>
<td>mpJ1939_FRSL_SLOW_FLASH</td>
<td>The red brake lamp blinks at 1 Hz (50% duty cycle)</td>
</tr>
</tbody>
</table>

5.4.1.2.85 MpJ1939HDPEEnum
This enumerated data type defines the flush release of the hydrocarbon doser (HDPE).

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_HDPE_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_HDPE_URGENT</td>
<td>Flushing activated (urgent).</td>
</tr>
<tr>
<td>mpJ1939_HDPE_LESS_URGENT</td>
<td>Flushing activated (less urgent).</td>
</tr>
<tr>
<td>mpJ1939_HDPE_NOT_ENABLED</td>
<td>HDPE not enabled.</td>
</tr>
</tbody>
</table>

5.4.1.2.86 MpJ1939MEMPEFEnum
This enumerated data type specifies whether the maximum engine power is currently activated.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_MEMPEF_DONT_CARE</td>
<td>The engine power is waiting.</td>
</tr>
<tr>
<td>mpJ1939_MEMPEF_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_MEMPEF_SUPPORTED</td>
<td>The engine power is activated.</td>
</tr>
<tr>
<td>mpJ1939_MEMPEF_DISABLED</td>
<td>The engine power is disabled.</td>
</tr>
</tbody>
</table>

5.4.1.2.87 MpJ1939MILSEnum
This enumerated data type indicates the state of the fault indicator lamp.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_MILS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_MILS_SHORT_FOR_WWH_OBD</td>
<td>Fault indicator lamp not compliant with WWH-OBD.</td>
</tr>
<tr>
<td>mpJ1939_MILS_LAMP_ON</td>
<td>The fault indicator lamp is switched on.</td>
</tr>
<tr>
<td>mpJ1939_MILS_LAMP_OFF</td>
<td>The fault indicator lamp is switched off.</td>
</tr>
</tbody>
</table>
5.4.1.2.88 MpJ1939NO1RSEnum
This enumerated data type specifies the state of the NOx sensor.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_NO1RS_NOTAVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_NO1RS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_NO1RS_STABLE</td>
<td>The measured value of the sensor is stable.</td>
</tr>
<tr>
<td>mpJ1939_NO1RS_NOT_STABLE</td>
<td>The measured value of the sensor is not stable.</td>
</tr>
</tbody>
</table>

5.4.1.2.89 MpJ1939NOS1SDSEnum
This enumerated data type specifies the state of the NOx sensor.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_NOS1SDS_NOT_SUPPORTED</td>
<td>The sensor is not supported.</td>
</tr>
<tr>
<td>mpJ1939_NOS1SDS_NOT_POSSIBLE</td>
<td>Check by sensor not possible.</td>
</tr>
<tr>
<td>mpJ1939_NOS1SDS_ABORTED</td>
<td>Check of the sensor aborted.</td>
</tr>
<tr>
<td>mpJ1939_NOS1SDS_COMPLETE</td>
<td>Check of the sensor completed.</td>
</tr>
<tr>
<td>mpJ1939_NOS1SDS_ACTIVE</td>
<td>Sensor active.</td>
</tr>
<tr>
<td>mpJ1939_NOS1SDS_NOT_ACTIVE</td>
<td>Sensor not active.</td>
</tr>
</tbody>
</table>

5.4.1.2.90 MpJ1939OCMPEnum
This enumerated data type defines the priority of the override control mode received in the torque message.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_OCMP_LOW_PRIORITY</td>
<td>Low priority.</td>
</tr>
<tr>
<td>mpJ1939_OCMP_MEDIUM_PRIORITY</td>
<td>Medium priority.</td>
</tr>
<tr>
<td>mpJ1939_OCMP_HIGH_PRIORITY</td>
<td>High priority.</td>
</tr>
<tr>
<td>mpJ1939_OCMP_HIGHEST_PRIORITY</td>
<td>Very high priority.</td>
</tr>
</tbody>
</table>

5.4.1.2.91 MpJ1939OSDSEnum
This enumerated data type specifies whether the driver's seat is in the front driving position.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_OSDS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_OSDS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_OSDS_FACING_FORWARD</td>
<td>The driver's seat is in the front position.</td>
</tr>
<tr>
<td>mpJ1939_OSDS_NOT_FACING_FORWARD</td>
<td>The driver's seat is not in the front position.</td>
</tr>
</tbody>
</table>

5.4.1.2.92 MpJ1939PBCEnum
This enumerated data type specifies whether the parking brake is controlled by a secondary device.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_PBC_DONT_CARE</td>
<td>The parking brake is waiting.</td>
</tr>
<tr>
<td>mpJ1939_PBC_SAE_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_PBC_REQUESTED</td>
<td>The parking brake is requested.</td>
</tr>
<tr>
<td>mpJ1939_PBC_NOT_REQUESTED</td>
<td>The parking brake is not requested.</td>
</tr>
</tbody>
</table>

5.4.1.2.93 MpJ1939PBRIREnum
This enumerated data type specifies the state of the parking brake release inhibitor request (PBRIR).

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_PBRIR_UNAVAILABLE</td>
<td>PBRIR not available.</td>
</tr>
<tr>
<td>mpJ1939_PBRIR_SAE_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_PBRIR_REQUESTED</td>
<td>PBRIR was requested.</td>
</tr>
<tr>
<td>mpJ1939_PBRIR_NOT_REQUESTED</td>
<td>PBRIR was not requested.</td>
</tr>
</tbody>
</table>

5.4.1.2.94 MpJ1939PBSEnum
This enumerated data type specifies the state of the parking brake.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_PBS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_PBS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_PBS_SET</td>
<td>The parking brake is enabled.</td>
</tr>
<tr>
<td>mpJ1939_PBS_NOT_SET</td>
<td>The parking brake is not enabled.</td>
</tr>
</tbody>
</table>

5.4.1.2.95 MpJ1939PGNStatusEnum
This enumerated data type specifies the state of the parameter group number (PGN).
mapp Services

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_PGN_IDLE</td>
<td>No PGN is currently active.</td>
</tr>
<tr>
<td>mpJ1939_PGN_NEW_DATA</td>
<td>PGN data is available.</td>
</tr>
<tr>
<td>mpJ1939_PGN_ERROR_CYCLE_TIME</td>
<td>PGN was called in an incorrect cycle time.</td>
</tr>
<tr>
<td>mpJ1939_PGN_ERROR_LIMITS</td>
<td>The limit was exceeded.</td>
</tr>
<tr>
<td>mpJ1939_PGN_ERROR_TRANSMIT</td>
<td>Transmitted PGN invalid.</td>
</tr>
<tr>
<td>mpJ1939_PGN_ERROR_RECEIVE</td>
<td>Received PGN invalid.</td>
</tr>
<tr>
<td>mpJ1939_PGN_NOT_ACTIVE</td>
<td>PGN is not active.</td>
</tr>
</tbody>
</table>

5.4.1.2.96 MpJ1939PLSEnum

This enumerated data type specifies the state of the engine protection lamp.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_PLS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_PLS_RESERVED</td>
<td>Reserved.</td>
</tr>
<tr>
<td>mpJ1939_PLS_LAMP_ON</td>
<td>The engine protection lamp is on.</td>
</tr>
<tr>
<td>mpJ1939_PLS_LAMP_OFF</td>
<td>The engine protection lamp is off.</td>
</tr>
</tbody>
</table>

5.4.1.2.97 MpJ1939PTOGSEnum

This enumerated data type specifies the state of the power take-off governor.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_PTOGS_SPEED_MEMORY_2</td>
<td>The power take-off governor is set to speed memory 2.</td>
</tr>
<tr>
<td>mpJ1939_PTOGS_SPEED_MEMORY_1</td>
<td>The power take-off governor is set to speed memory 1.</td>
</tr>
<tr>
<td>mpJ1939_PTOGS_SET_SPEED_8</td>
<td>The power take-off governor is set to predefined speed 8.</td>
</tr>
<tr>
<td>mpJ1939_PTOGS_SET_SPEED_7</td>
<td>The power take-off governor is set to predefined speed 7.</td>
</tr>
<tr>
<td>mpJ1939_PTOGS_SET_SPEED_6</td>
<td>The power take-off governor is set to predefined speed 6.</td>
</tr>
<tr>
<td>mpJ1939_PTOGS_SET_SPEED_5</td>
<td>The power take-off governor is set to predefined speed 5.</td>
</tr>
<tr>
<td>mpJ1939_PTOGS_SET_SPEED_4</td>
<td>The power take-off governor is set to predefined speed 4.</td>
</tr>
<tr>
<td>mpJ1939_PTOGS_SET_SPEED_3</td>
<td>The power take-off governor is set to predefined speed 3.</td>
</tr>
<tr>
<td>mpJ1939_PTOGS_SET_SPEED_2</td>
<td>The power take-off governor is set to predefined speed 2.</td>
</tr>
<tr>
<td>mpJ1939_PTOGS_SET_SPEED_1</td>
<td>The power take-off governor is set to predefined speed 1.</td>
</tr>
<tr>
<td>mpJ1939_PTOGS_ACCEL_OVERRIDE</td>
<td>The power take-off governor is set to &quot;Acceleration override&quot;.</td>
</tr>
<tr>
<td>mpJ1939_PTOGS_ACCELERATE</td>
<td>The power take-off governor is set to &quot;Accelerate&quot;.</td>
</tr>
<tr>
<td>mpJ1939_PTOGS_RESUME</td>
<td>The power take-off governor is set to &quot;Resume&quot;.</td>
</tr>
<tr>
<td>mpJ1939_PTOGS_DECELERATE</td>
<td>The power take-off governor is set to &quot;Decelerate&quot;.</td>
</tr>
<tr>
<td>mpJ1939_PTOGS_SET</td>
<td>The power take-off governor is set to &quot;Set&quot;.</td>
</tr>
<tr>
<td>mpJ1939_PTOGS_REMOTE_STANDBY</td>
<td>The power take-off governor is set to &quot;Remote stand- by&quot;.</td>
</tr>
<tr>
<td>mpJ1939_PTOGS_STANDBY</td>
<td>The power take-off governor is set to &quot;Remote stand- by&quot;.</td>
</tr>
<tr>
<td>mpJ1939_PTOGS_REMOTE_HOLD</td>
<td>The power take-off governor is set to &quot;Remote hold&quot;.</td>
</tr>
<tr>
<td>mpJ1939_PTOGS_HOLD</td>
<td>The power take-off governor is set to &quot;Hold&quot;.</td>
</tr>
<tr>
<td>mpJ1939_PTOGS_OFF</td>
<td>The power take-off governor is switched off.</td>
</tr>
</tbody>
</table>

5.4.1.2.98 MpJ1939PTOSHSEEnum

This enumerated data type specifies whether or not the engine has been shut down by the PTO shutdown system.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_PTOSHSE_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_PTOSHSE_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_PTOSHSE_YES</td>
<td>The engine was shut down.</td>
</tr>
<tr>
<td>mpJ1939_PTOSHSE_NO</td>
<td>The engine was not shut down.</td>
</tr>
</tbody>
</table>

5.4.1.2.99 MpJ1939RCZHEnum

This enumerated data type specifies the state of the cabin heater.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_RCZH_DONT_CARE</td>
<td>The cabin heater is waiting.</td>
</tr>
<tr>
<td>mpJ1939_RCZH_RESERVED</td>
<td>Reserved.</td>
</tr>
<tr>
<td>mpJ1939_RCZH_HEAT_CAB_ZONE</td>
<td>The cabin heater is heating.</td>
</tr>
<tr>
<td>mpJ1939_RCZH_DO_NOT_HEAT</td>
<td>The cabin heater is not heating.</td>
</tr>
</tbody>
</table>
5.4.1.2.100 MpJ1939REZHEnum
This enumerated data type specifies the state of the engine zone heater.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_REZH_DONT_CARE</td>
<td>The engine heater is waiting.</td>
</tr>
<tr>
<td>mpJ1939_REZH_RESERVED</td>
<td>Reserved.</td>
</tr>
<tr>
<td>mpJ1939_REZH_HEAT_ENGINE_ZONE</td>
<td>The engine heater is heating.</td>
</tr>
<tr>
<td>mpJ1939_REZH_DO_NOT_HEAT</td>
<td>The engine heater is not heating.</td>
</tr>
</tbody>
</table>

5.4.1.2.101 MpJ1939RHPSEnum
This enumerated data type specifies the position of the high pressure switch in the coolant circuit of an air conditioner.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_RHPS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_RHPS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_RHPS_PRESSURE_TOO_HIGH</td>
<td>Pressure is too high.</td>
</tr>
<tr>
<td>mpJ1939_RHPS_PRESSURE_NORMAL</td>
<td>Pressure is normal.</td>
</tr>
</tbody>
</table>

5.4.1.2.102 MpJ1939RLPSEnum
This enumerated data type specifies the position of the low pressure switch in the coolant circuit of an air conditioner.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_RLPS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_RLPS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_RLPS_PRESSURE_TOO_LOW</td>
<td>Pressure too low.</td>
</tr>
<tr>
<td>mpJ1939_RLPS_PRESSURE_NORMAL</td>
<td>Pressure is normal.</td>
</tr>
</tbody>
</table>

5.4.1.2.103 MpJ1939RSLiSEnum
This enumerated data type specifies the state of the system for limiting the maximum vehicle speed.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_RSLIS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_RSLIS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_RSLIS_NOT_ACTIVE</td>
<td>Limiting not active.</td>
</tr>
<tr>
<td>mpJ1939_RSLIS_ACTIVE</td>
<td>Limiting is active.</td>
</tr>
</tbody>
</table>

5.4.1.2.104 MpJ1939RSLSEnum
This enumerated data type indicates the state of the red brake lamp.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_RSLS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_RSLS_RESERVED</td>
<td>The red brake lamp is reserved.</td>
</tr>
<tr>
<td>mpJ1939_RSLS_LAMP_ON</td>
<td>The red brake lamp is on.</td>
</tr>
<tr>
<td>mpJ1939_RSLS_LAMP_OFF</td>
<td>The red brake lamp is off.</td>
</tr>
</tbody>
</table>

5.4.1.2.105 MpJ1939S1ATEnum
This enumerated data type specifies the state of the heating element.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_S1AT_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_S1AT_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_S1AT_IN_RANGE</td>
<td>Value of the heating element in the permissible range.</td>
</tr>
<tr>
<td>mpJ1939_S1AT_NOT_IN_RANGE</td>
<td>Value of the heating element not in the permissible range.</td>
</tr>
</tbody>
</table>

5.4.1.2.106 MpJ1939S1HCEnum
This enumerated data type specifies the state of the warm-up process.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_S1HC_HEATER_OFF</td>
<td>The heater is off.</td>
</tr>
<tr>
<td>mpJ1939_S1HC_PREHEAT_1</td>
<td>Preheat level 1.</td>
</tr>
<tr>
<td>mpJ1939_S1HC_PREHEAT_2</td>
<td>Preheat level 2.</td>
</tr>
<tr>
<td>mpJ1939_S1HC_AUTOMATIC</td>
<td>Automatic heating.</td>
</tr>
</tbody>
</table>

5.4.1.2.107 MpJ1939S1PIREnum
This enumerated data type specifies the state of the aftertreatment intake gas sensor.
mapp Services

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_S1PIR_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_S1PIR_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_S1PIR_IN_RANGE</td>
<td>Value in permitted range.</td>
</tr>
<tr>
<td>mpJ1939_S1PIR_NOT_IN_RANGE</td>
<td>Value not in permitted range.</td>
</tr>
</tbody>
</table>

5.4.1.2.108 MpJ1939SBSEnum

This enumerated data type specifies the state of the switch used to determine whether the seat belt is fastened.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_SBS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_SBS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_SBS_OK</td>
<td>The safety belt is fastened.</td>
</tr>
<tr>
<td>mpJ1939_SBS_NOT_BUCKLED</td>
<td>The safety belt is not fastened.</td>
</tr>
</tbody>
</table>

5.4.1.2.109 MpJ1939SCRINO1RSEnum

This enumerated data type specifies the state of the NOx measured value.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_SCRINO1RS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_SCRINO1RS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_SCRINO1RS_STABLE</td>
<td>Measured value stable.</td>
</tr>
<tr>
<td>mpJ1939_SCRINO1RS_NOT_STABLE</td>
<td>Measured value not stable.</td>
</tr>
</tbody>
</table>

5.4.1.2.110 MpJ1939SCRTMAEnum

This enumerated data type specifies whether the exhaust gas temperatures have been increased to regenerate the SCR aftertreatment system or to prepare for regeneration of the SCR aftertreatment system.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_SCRTMA_DONT_CARE</td>
<td>SCRTMA is waiting.</td>
</tr>
<tr>
<td>mpJ1939_SCRTMA_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_SCRTMA_ACTIVE</td>
<td>SCRTMA active.</td>
</tr>
<tr>
<td>mpJ1939_SCRTMA_NOT_ACTIVE</td>
<td>SCRTMA not active.</td>
</tr>
</tbody>
</table>

5.4.1.2.111 MpJ1939TSASEnum

This enumerated data type specifies the state of the current axis range.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_TSAS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_TSAS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_TSAS_HIGH_SPEED_RANGE</td>
<td>The axis range is in the high speed range.</td>
</tr>
<tr>
<td>mpJ1939_TSAS_LOW_SPEED_RANGE</td>
<td>The axis range is in the low speed range.</td>
</tr>
</tbody>
</table>

5.4.1.2.112 MpJ1939TSC1CPEnum

This enumerated data type specifies which control mode the transmitting device uses to generate the TSC1 command.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_TSC1CP_P32_TEMP_PWR_CTRL</td>
<td>Mode &quot;Temporary drive control&quot;.</td>
</tr>
<tr>
<td>mpJ1939_TSC1CP_P5_ENG_PROTECTION</td>
<td>Mode &quot;Engine protection&quot;.</td>
</tr>
<tr>
<td>mpJ1939_TSC1CP_P4 ROAD_SPEED_GVN</td>
<td>Mode &quot;Speed limiting&quot;.</td>
</tr>
<tr>
<td>mpJ1939_TSC1CP_P3_PTO_GOVERNOR</td>
<td>Mode &quot;Power take-off governor&quot;.</td>
</tr>
<tr>
<td>mpJ1939_TSC1CP_P2 CRUISE_CONTROL</td>
<td>Mode &quot;Cruise control&quot;.</td>
</tr>
<tr>
<td>mpJ1939_TSC1CP_P1 ACCEL_PEDAL</td>
<td>Mode &quot;Accelerator pedal&quot;.</td>
</tr>
</tbody>
</table>

5.4.1.2.113 MpJ1939TSC1TREnum

This enumerated data type specifies the transfer rate at which the transmitting device transmits the TSC1 message.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_TSC1TR_100_MS</td>
<td>100 ms transfer rate</td>
</tr>
<tr>
<td>mpJ1939_TSC1TR_250_MS</td>
<td>250 ms transfer rate</td>
</tr>
<tr>
<td>mpJ1939_TSC1TR_500_MS</td>
<td>500 ms transfer rate</td>
</tr>
<tr>
<td>mpJ1939_TSC1TR_750_MS</td>
<td>750 ms transfer rate</td>
</tr>
<tr>
<td>mpJ1939_TSC1TR_1000_MS</td>
<td>1000 ms transfer rate</td>
</tr>
<tr>
<td>mpJ1939_TSC1TR_STANDARD_10_MS</td>
<td>10 ms transfer rate</td>
</tr>
<tr>
<td>mpJ1939_TSC1TR_20_MS</td>
<td>20 ms transfer rate</td>
</tr>
<tr>
<td>mpJ1939_TSC1TR_50_MS</td>
<td>50 ms transfer rate</td>
</tr>
</tbody>
</table>
5.4.1.2.114 MpJ1939VARLSEnum

This enumerated data type specifies the state of the system that limits the maximum vehicle forward acceleration.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_VARLS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_VARLS_RESERVED</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_VARLS_LIMIT_ACTIVE</td>
<td>Limiting active.</td>
</tr>
<tr>
<td>mpJ1939_VARLS_LIMIT_NOT_ACTIVE</td>
<td>Limiting not active.</td>
</tr>
</tbody>
</table>

5.4.1.2.115 MpJ1939VLSGDSEnum

This enumerated data type specifies whether the vehicle limiting speed governor (VLSG) is decremented.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_VLSGDS_ERROR</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_VLSGDS_ON_STATE</td>
<td>VLSG is decremented.</td>
</tr>
<tr>
<td>mpJ1939_VLSGDS_OFF_STATE</td>
<td>VLSG is not decremented.</td>
</tr>
<tr>
<td>mpJ1939_VLSGDS_NOT_AVAILABLE</td>
<td>VLSG is not available.</td>
</tr>
</tbody>
</table>

5.4.1.2.116 MpJ1939VLSGESEnum

This enumerated data type specifies the state of the vehicle limiting speed governor (VLSG).

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_VLSGES_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_VLSGES_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_VLSGES_SWITCHED_ENABLED</td>
<td>VLSG is enabled.</td>
</tr>
<tr>
<td>mpJ1939_VLSGES_SWITCH_DISABLED</td>
<td>VLSG is disabled.</td>
</tr>
</tbody>
</table>

5.4.1.2.117 MpJ1939VLSGISEnum

This enumerated data type specifies whether the vehicle limiting speed governor (VLSG) is incremented.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_VLSGIS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_VLSGIS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_VLSGIS_ON_STATE</td>
<td>VLSG is incremented.</td>
</tr>
<tr>
<td>mpJ1939_VLSGIS_OFF_STATE</td>
<td>VLSG is not incremented.</td>
</tr>
</tbody>
</table>

5.4.1.2.118 MpJ1939WRPO21RSEnum

This enumerated data type specifies the state of the oxygen measured value.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_WRPO21RS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_WRPO21RS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_WRPO21RS_STABLE</td>
<td>The oxygen measured value is stable.</td>
</tr>
<tr>
<td>mpJ1939_WRPO21RS_NOT_STABLE</td>
<td>The oxygen measured value is not stable.</td>
</tr>
</tbody>
</table>

5.4.1.2.119 MpJ1939ACHPFSEnum

This enumerated data type indicates whether the pressure in the coolant circuit of an air conditioner is too high and the fan must be switched on.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_ACHPFS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_ACHPFS_ERROR</td>
<td>Error state.</td>
</tr>
<tr>
<td>mpJ1939_ACHPFS_PRESSURE_HIGH</td>
<td>Pressure is too high.</td>
</tr>
<tr>
<td>mpJ1939_ACHPFS_PRESSURE_NORMAL</td>
<td>Pressure is normal.</td>
</tr>
</tbody>
</table>

5.4.1.2.120 MpJ1939DEFTLLIEnum

This enumeration data type specifies the state of the operator prompting system for anomalies with the SCR system.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_DEFSTLL_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_DEFSTLL_RESERVED6</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DEFSTLL_RESERVED5</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DEFSTLL_ON_FAST_BLINK</td>
<td>The operator prompting system is blinking.</td>
</tr>
<tr>
<td>mpJ1939_DEFSTLL_RESERVED3</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DEFSTLL_RESERVED2</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_DEFSTLL_ON_SOLID</td>
<td>The operator prompting system is lit.</td>
</tr>
<tr>
<td>mpJ1939_DEFSTLL_OFF</td>
<td>The operator prompting system is off.</td>
</tr>
</tbody>
</table>
5.4.1.2.121 MpJ1939SCROISEnum

This enumerated data type specifies the desired brightness of the driver's diesel exhaust gas warning indicator.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpJ1939_SCROIS_NOT_AVAILABLE</td>
<td>State not available.</td>
</tr>
<tr>
<td>mpJ1939_SCROIS_TEMP_OVERRIDE</td>
<td>The lamp is lit.</td>
</tr>
<tr>
<td>mpJ1939_SCROIS_INDUCEMENT_LEVEL5</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_SCROIS_INDUCEMENT_LEVEL4</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_SCROIS_INDUCEMENT_LEVEL3</td>
<td>The lamp is blinking.</td>
</tr>
<tr>
<td>mpJ1939_SCROIS_INDUCEMENT_LEVEL2</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_SCROIS_INDUCEMENT_LEVEL1</td>
<td>Reserved for SAE assignment.</td>
</tr>
<tr>
<td>mpJ1939_SCROIS_DRIVER_WARNING</td>
<td>The lamp is in an error state.</td>
</tr>
</tbody>
</table>

5.4.2 Function blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MpJ1939Generic</td>
<td>This function block can be used to receive and send PGNs.</td>
</tr>
<tr>
<td>MpJ1939Receive</td>
<td>This function block can be used to receive a specified PGN cyclically.</td>
</tr>
<tr>
<td>MpJ1939SpecificRequest</td>
<td>This function block can be used to request a PGN that cannot be read cyclically.</td>
</tr>
<tr>
<td>MpJ1939Transmit</td>
<td>This function block can be used to transmit a PGN cyclically.</td>
</tr>
</tbody>
</table>

5.4.2.1 MpJ1939Generic

This function block can be used to receive and send PGNs.

Function block

```
MpJ1939Generic

&MpComIdentType MpLink

BOOL Enable

BOOL ErrorReset

BOOL Active

BOOL Error

DINT StatusID

MpJ1939GenericInfoType Info
```

Interface

<table>
<thead>
<tr>
<th>I/O</th>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
</table>
| IN    | MpLink          | Pointer to Mp-
                             | ComIdentType Connection to mapp (MpLink of an MpJ1939Generic configuration) |
| IN    | Enable          | BOOL               | The function block is active as long as this input is set.                 |
| IN    | ErrorReset      | BOOL               | Resets function block errors.                                              |
| OUT   | Active          | BOOL               | Function block active.                                                     |
| OUT   | Error           | BOOL               | Error occurred during execution.                                           |
| OUT   | StatusID        | DINT               | Status information.                                                        |
| OUT   | Info            | MpJ1939Gener-
                             | icInfoType Additional information about the component.                     |

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 be carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

5.4.2.1.1 Description

This function block uses the MpLink from an MpJ1939Generic configuration.

The parameter group numbers (PGNs) defined in theMpJ1939Generic configuration are transmitted and received on the specified CAN interface using MpJ1939Generic. The CAN interface is defined in the MpJ1939Generic configuration under "CAN interface". How data is transmitted and received is explained in section Transmitting and receiving information.

Some PGNs must be sent in certain cycle times. For more information, see Timing.

The informational structure displays diagnostic information of the function module.
If signals in PGNs have scaling or an offset, they are automatically adjusted when transmitting and receiving the PGN. For example, signal EngRequestedSpeed_SpeedLimit in PGN TSC1 has a factor of 0.125. This factor is automatically taken into account when transmitting and receiving the PGN.

A PGN can also have minimum and maximum values. These are also taken into account automatically. If a PGN is transmitted in which a value would overshoot the maximum value of the PGN, then error Value outside limits is triggered and the value cannot be sent.

If and which scaling or offset a PGN has can be taken from a PGN database.

5.4.2.2 MpJ1939Receive

This function block can be used to receive a specified PGN cyclically.

**Function block**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MpLink</td>
<td>BOOL</td>
<td>Pointer to MpComIdentType. Connection to mapp (MpLink of an MpJ1939Receive configuration).</td>
</tr>
<tr>
<td>Enable</td>
<td>BOOL</td>
<td>The function block is active as long as this input is set.</td>
</tr>
<tr>
<td>ErrorReset</td>
<td>BOOL</td>
<td>Resets function block errors.</td>
</tr>
<tr>
<td>PGN</td>
<td>UDINT</td>
<td>Value for the numerical value of the PGN.</td>
</tr>
<tr>
<td>Priority</td>
<td>USINT</td>
<td>Priority of the PGN.</td>
</tr>
<tr>
<td>Data</td>
<td>UDINT</td>
<td>Value for the address of the PGN's data buffer. For more information, see Memory management.</td>
</tr>
<tr>
<td>DataLength</td>
<td>UINT</td>
<td>Defines the size of data buffer &quot;Data&quot;. Specified in bytes.</td>
</tr>
<tr>
<td>Receive</td>
<td>BOOL</td>
<td>Command to receive a PGN.</td>
</tr>
<tr>
<td>Active</td>
<td>BOOL</td>
<td>Function block active.</td>
</tr>
<tr>
<td>Error</td>
<td>BOOL</td>
<td>Error occurred during execution.</td>
</tr>
<tr>
<td>StatusID</td>
<td>DINT</td>
<td>Status information.</td>
</tr>
<tr>
<td>CommandDone</td>
<td>BOOL</td>
<td>Execution successful. Function block is finished.</td>
</tr>
<tr>
<td>CommandBusy</td>
<td>BOOL</td>
<td>Function block currently executing command.</td>
</tr>
<tr>
<td>Info</td>
<td>MpJ1939ReceiveInfoType</td>
<td>Additional information about the component.</td>
</tr>
</tbody>
</table>

**Interface**

**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 be carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

5.4.2.2.1 Description

This function block uses the MpLink from an MpJ1939Receive configuration.

This function block can be used to receive information from a cyclically readable PGN. For more information, see Transmitting and receiving information.
The PGN specified on input parameter "PGN" is queried by the function block. The numerical value of the PGN must be specified. The buffer for the PGN must be provided on "Data". How to create this buffer is explained in section Memory management.

The priority of the PGN must be specified on input "Priority". This can be different for each PGN. The priority of a PGN can be taken from a J1939 database. For PGN DTBI1, for example, the priority is 5.

The requested information is indicated by command "Receive" = TRUE on parameter "Data". Output "Command-Done" set to TRUE indicates that the command was successful.

It is important to note that PGNs must be queried within a certain cycle time. For more information, see Timing.

Input parameter "DataLength" specifies the size of data buffer "Data".

If the information of a PGN has been received and a new PGN should be requested, the function block must be disabled via "Enable" = FALSE and re-enabled via "Enable" = TRUE. After re-enabling of the function block, a new PGN can be specified.

A PGN can have limit values and an offset. These must be taken into account when further processing the PGN. The limit values of a PGN can be taken from a J1939 database that can be downloaded here.

If command "Receive" was executed successfully, the number of bytes received can be seen on parameter "ReceivedBytes" of the info structure.

5.4.2.3 MpJ1939SpecificRequest

This function block can be used to request a PGN that cannot be read cyclically.

**Function block**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;MpComIdentType</td>
<td></td>
<td>Pointer to MpComIdentType</td>
</tr>
<tr>
<td>MpLink</td>
<td>BOOL</td>
<td>Connection to mapp (MpLink of an MpJ1939SpecificRequest configuration).</td>
</tr>
<tr>
<td>Enable</td>
<td>BOOL</td>
<td>The function block is active as long as this input is set.</td>
</tr>
<tr>
<td>ErrorReset</td>
<td>BOOL</td>
<td>Resets function block errors.</td>
</tr>
<tr>
<td>PGN</td>
<td>UDINT</td>
<td>Value for the numerical value of the PGN.</td>
</tr>
<tr>
<td>Priority</td>
<td>USINT</td>
<td>Priority of the PGN.</td>
</tr>
<tr>
<td>Data</td>
<td>UDINT</td>
<td>Value for the address of the PGN's data buffer. For more information, see Memory management.</td>
</tr>
<tr>
<td>DataLength</td>
<td>UINT</td>
<td>Defines the size of data buffer &quot;Data&quot;. Specified in bytes.</td>
</tr>
<tr>
<td>Request</td>
<td>BOOL</td>
<td>Command to start request.</td>
</tr>
<tr>
<td>CommandBusy</td>
<td>BOOL</td>
<td>Function block currently executing command.</td>
</tr>
<tr>
<td>CommandDone</td>
<td>BOOL</td>
<td>Execution successful. Function block is finished.</td>
</tr>
<tr>
<td>StatusID</td>
<td>DINT</td>
<td>Status information.</td>
</tr>
<tr>
<td>Info</td>
<td>MpJ1939SpecificRequestInfoType</td>
<td>Additional information about the component.</td>
</tr>
</tbody>
</table>

**Interface**

**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 be carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

5.4.2.3.1 Description

This function block uses the MpLink from an MpJ1939SpecificRequest configuration.

This function block can be used to query a PGN that cannot be read cyclically. For more information, see Transmitting and receiving information.

The PGN specified on input parameter "PGN" is queried by the function block. The numerical value of the PGN must be specified. The numerical value of a PGN can be taken from a J1939 database. This can be downloaded here. The buffer for the PGN must be provided on "Data". How to create this buffer is explained in section Memory management.

The priority of the PGN must be specified on input "Priority". This can be different for each PGN. The priority of a PGN can be taken from a J1939 database. For PGN DTBI1, for example, the priority is 5.

The requested information is indicated by command "Request" = TRUE on parameter "Data". Output "CommandDone" set to TRUE indicates that the command was successful.

Input parameter "DataLength" specifies the size of data buffer "Data".

If the information of a PGN has been received and a new PGN should be requested, the function block must be disabled via "Enable" = FALSE and re-enabled via "Enable" = TRUE. After re-enabling of the function block, a new PGN can be specified.

A PGN can have limit values and an offset. These must be taken into account when further processing the PGN. The limit values of a PGN can be taken from a J1939 database that can be downloaded here.

If command "Request" was executed successfully, the number of bytes received can be seen on parameter "ReceivedBytes" of the info structure.

5.4.2.4 MpJ1939Transmit

This function block can be used to transmit a PGN cyclically.

Function block

![Function block diagram]

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 MpJ1939Transmit 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>PGN</td>
<td>UDINT</td>
<td>Value for the numerical value of the PGN.</td>
</tr>
<tr>
<td>IN</td>
<td>Priority</td>
<td>USINT</td>
<td>Priority of the PGN.</td>
</tr>
<tr>
<td>IN</td>
<td>Data</td>
<td>UDINT</td>
<td>Value for the address of the PGN's data buffer. For more information, see Memory management.</td>
</tr>
<tr>
<td>IN</td>
<td>DataLength</td>
<td>UINT</td>
<td>Defines the size of data buffer &quot;Data&quot;. Specified in bytes.</td>
</tr>
<tr>
<td>IN</td>
<td>Transmit</td>
<td>BOOL</td>
<td>Command to transmit PGN.</td>
</tr>
<tr>
<td>OUT</td>
<td>Active</td>
<td>BOOL</td>
<td>Function block active.</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 be carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

**5.4.2.4.1 Description**

This function block uses the MpLink from an MpJ1939Transmit configuration. This function block can be used to transmit a cyclically readable PGN. For more information, see Transmitting and receiving information.

The PGN specified on input parameter "PGN" is queried by the function block. The numerical value of the PGN must be specified. The numerical value of a PGN can be taken from a J1939 database. This can be downloaded [here](#). The buffer for the PGN must be provided on "Data". How to create this buffer is explained in section Memory management.

The priority of the PGN must be specified on input "Priority". This can be different for each PGN. The priority of a PGN can be taken from a J1939 database. For PGN DTBI1, for example, the priority is 5.

The information specified on parameter "Data" is transmitted by command "Transmit" = TRUE. The successful transmission is confirmed by "CommandDone" = TRUE.

It is important to note that PGNs must be transmitted within a certain cycle time. For more information, see Timing. Input parameter "DataLength" specifies the size of data buffer "Data".

If the PGN has been transmitted and a new PGN should be transmitted, the function block must be disabled via "Enable" = FALSE and re-enabled via "Enable" = TRUE. After re-enabling of the function block, a new PGN can be transmitted.

A PGN can have limit values and an offset. These must be taken into account when further processing the PGN. The limit values of a PGN can be taken from a J1939 database that can be downloaded [here](#).

**5.4.3 Status numbers**

**5.4.3.1 -1064171516: Invalid data buffer**

**Description:**

Input parameter "Data" is not connected or a 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:**

- Connect parameter to "Data".

**Constant**

mpJ1939_ERR_INVALID_DATA_BUFFER

These function blocks / functions can report this error:

- MpJ1939Receive
- MpJ1939SpecificRequest
- MpJ1939Transmit
5.4.3.2 -1064171517: J1939 communication failed

Description:
An error occurred during communication or cyclic execution of the PGNs. 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 CAN interface.
- Check provided data buffer "Data". For more information, see Memory management.
- See the Logger for additional information.

Constant
mpJ1939_ERR_COMMUNICATION

These function blocks / functions can report this error:
- MpJ1939Generic
- MpJ1939Receive
- MpJ1939SpecificRequest
- MpJ1939Transmit

5.4.3.3 -1064171518: Simulation not permitted

Description:
The component cannot be used in a simulated environment. 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:
- Use real hardware.

Constant
mpJ1939_ERR_ARSIM_NOT_ALLOWED

These function blocks / functions can report this error:
- MpJ1939Generic
- MpJ1939Receive
- MpJ1939SpecificRequest
- MpJ1939Transmit

5.4.3.4 -1064171519: Error opening CAN interface

Description:
An error occurred when trying to open the CAN interface. 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.
mapp Services

Cause/Solution:

- Check the CAN path specified in the MpJ1939Generic configuration.

Constant

mpJ1939_ERR_INVALID_CAN_IF

These function blocks / functions can report this error:

- MpJ1939Generic
- MpJ1939Receive
- MpJ1939SpecificRequest
- MpJ1939Transmit

5.4.3.5 -1064171520: Value outside limits

Description:
The value of a PGN signal is outside the defined limits.
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 maximum and minimum values of a PGN signal. For more information, see Memory management.

Constant

mpJ1939_ERR_LIMITS

These function blocks / functions can report this error:

- MpJ1939Generic

5.4.3.6 -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.
- The function is not enabled in the configuration.

These function blocks / functions can report this error:

- MpJ1939Generic
- MpJ1939Receive
- MpJ1939SpecificRequest
- MpJ1939Transmit
5.4.3.7 -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:
- MpJ1939Generic
- MpJ1939Receive
- MpJ1939SpecificRequest
- MpJ1939Transmit

5.4.3.8 -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:
- MpJ1939Generic
- MpJ1939Receive
- MpJ1939SpecificRequest
- MpJ1939Transmit

5.4.3.9 -1064239099: Invalid MpLink contents

Description:
The value of variable "MpLink" 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.
mapp Services

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:

- MpJ1939Generic
- MpJ1939Receive
- MpJ1939SpecificRequest
- MpJ1939Transmit

5.4.3.10 -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:

- MpJ1939Generic
- MpJ1939Receive
- MpJ1939SpecificRequest
- MpJ1939Transmit

5.4.3.11 -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:

- MpJ1939Generic
- MpJ1939Receive
- MpJ1939SpecificRequest
- MpJ1939Transmit

5.4.3.12 -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:
- MpJ1939Generic
- MpJ1939Receive
- MpJ1939SpecificRequest
- MpJ1939Transmit

5.4.3.13 -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:
- MpJ1939Generic
- MpJ1939Receive
- MpJ1939SpecificRequest
- MpJ1939Transmit

5.4.3.14 -1064290619: Invalid PGN configuration

Description:
Specified configuration {2:PGN} is invalid. 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 PGN.
- The PGN does not exist.

Additional information:
- {2:PGN}: Number of the affected parameter group.

Constant
mpJ1939_ERR_INVALID_PGN_CFG
These function blocks / functions can report this error:

- MpJ1939Generic

5.4.3.15 -1064290621: Undefined CAN interface

Description:
No path to the CAN interface was defined. 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:
- Define the path to the CAN interface in the used configurations.

Constant
mpJ1939_ERR_EMPTY_CAN_IF_PATH

These function blocks / functions can report this error:

- MpJ1939Generic
- MpJ1939Receive
- MpJ1939SpecificRequest
- MpJ1939Transmit

5.4.3.16 -1064290622: PGN request failed

Description:
PGN {2:PGN} is not provided by the device. 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 desired PGN was requested in the correct cycle time. For more information, see Timing.
- Check the PGN number.

Additional information:
- {2:PGN}: Number of the affected parameter group.

Constant
mpJ1939_ERR_REQUEST_NACK

These function blocks / functions can report this error:

- MpJ1939Generic
- MpJ1939SpecificRequest

5.4.3.17 -1064290623: Receiving data aborted

Description:
Receiving data for {2:PGN} was aborted. 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:
• Receiving data was aborted by the transmitter.
• Request the PGN again.

Additional information:
• \{2:PGN\}: Number of the affected parameter group.

Constant
\texttt{mpJ1939\_ERR\_RECEIVE\_ABORT}

These function blocks / functions can report this error:
• \texttt{MpJ1939Generic}
• \texttt{MpJ1939Receive}

5.4.3.18 -1064290624: Invalid target address

Description:
The target address for \{2:PGN\} is invalid. 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:
• Use \texttt{MpComDump} to create a file that contains all information about the components and services being used. Save the Logger information as well. Contact B&R Support with this information.

Additional information:
• \{2:PGN\}: Number of the affected parameter group.

Constant
\texttt{mpJ1939\_ERR\_DEST\_ADR\_OUT\_OF\_RANGE}

These function blocks / functions can report this error:
• \texttt{MpJ1939SpecificRequest}

5.4.3.19 -1064290625: Receive timeout

Description:
A timeout occurred while receiving data for \{2:PGN\}. 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:
• Receive the desired PGN again.
• Use \texttt{MpComDump} to create a file that contains all information about the components and services being used. Save the Logger information as well. Contact B&R Support with this information.
Additional information:

- {2:PGN}: Number of the affected parameter group.

Constant

mpJ1939_ERR_RECEIVE_TIMEOUT

These function blocks / functions can report this error:

- MpJ1939Generic
- MpJ1939Receive

5.4.3.20 -1064290626: Transfer timeout

Description:
A timeout occurred while transferring data for {2:PGN}. 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:

- Transmit the desired PGN again.
- Use MpComDump to create a file that contains all information about the components and services being used. Save the Logger information as well. Contact B&R Support with this information.

Additional information:

- {2:PGN}: Number of the affected parameter group

Constant

mpJ1939_ERR_TRANSMIT_TIMEOUT

These function blocks / functions can report this error:

- MpJ1939Generic
- MpJ1939Transmit

5.4.3.21 -1064290627: Receive failed

Description:
An error occurred while receiving data for {2:PGN}. 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:

- Use MpComDump to create a file that contains all information about the components and services being used. Save the Logger information as well. Contact B&R Support with this information.

Additional information:

- {2:PGN}: Number of the affected parameter group.
These function blocks / functions can report this error:

- MpJ1939Generic
- MpJ1939Receive

5.4.3.22 -1064290630: Invalid pointer for data information

Description:
Invalid pointer for file information. PGN cannot be sent or received. 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:

- Use MpComDump to create a file that contains all information about the components and services being used. Save the Logger information as well. Contact B&R Support with this information.

Constant
mpJ1939_ERR_NO_DATA_INFO_POINTER

5.4.3.23 -1064290632: Transfer failed

Description:
Data transfer for {2:PGN} 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:

- Use MpComDump to create a file that contains all information about the components and services being used. Save the Logger information as well. Contact B&R Support with this information.

Additional information:

- {2:PGN}: Number of the affected parameter group.

Constant
mpJ1939_ERR_TRANSMIT_FAILED

5.4.3.24 -1064290634: Invalid data pointer

Description:
The attached data pointer is invalid for {2:PGN}. 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:

- Use MpComDump to create a file that contains all information about the components and services being used. Save the Logger information as well. Contact B&R Support with this information.

Additional information:

- {2:PGN}: Number of the affected parameter group.
Constant

mpJ1939_ERR_NO_DATA_POINTER

5.4.3.25 -1064290635: Invalid PGN

Description:
Specified PGN {2:PGN} is outside the valid range. 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 specified PGN number.

Additional information:
- {2:PGN}: Number of the affected parameter group.

Constant

mpJ1939_ERR_PGN_OUT_OF_RANGE

5.4.3.26 -1064290636: Invalid data length

Description:
The specified data length fro {2:PGN} is outside the permissible range. 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:
- Use MpComDump to create a file that contains all information about the components and services being used. Save the Logger information as well. Contact B&R Support with this information.

Additional information:
- {2:PGN}: Number of the affected parameter group.

Constant

mpJ1939_ERR_DATA_LENGTH_OUT_OF_RANGE

5.4.3.27 -1064290637: Invalid priority

Description:
The specified priority is outside the valid range for {2:PGN}. 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 PGN priority must be ≤7.
- The priority of a PGN can be taken from the J1939 standard.
Additional information:

- {2:PGN}: Number of the affected parameter group.

**Constant**

mpJ1939_ERR_PRIORITY_OUT_OF_RANGE

5.4.3.28 -1064290639: No CAN transmitter specified

**Description:**

No CAN transmitter was specified. 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:**

- Use MpComDump to create a file that contains all information about the components and services being used. Save the Logger information as well. Contact B&R Support with this information.

**Constant**

mpJ1939_ERR_NO_CAN_SENDER_HANDLE

5.4.3.29 -1064290620: Invalid PV mapping

**Description:**

The specified PV mapping for {2: PGN} is invalid. 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 PV in the MpJ1939Generic configuration.
- The PV that should be used for the desired PGN has an incorrect data type.

**Additional information:**

- {2:PGN}: Number of the affected parameter group.

**Constant**

mpJ1939_ERR_INVALID_PV_MAPPING

6 mapp File: File management system

mapp File provides a file management system as well as a connection to Visual Components 4 to display files.
6.1 Concept

User stories

Description

Machine operator

• As a machine operator, I would like to be able to browse through my file structure and do things like create/delete folders, etc.
• As a machine operator, I would like to copy files from one data storage device to another.

Application engineer

• As an application engineer, I would like to invest as little time as possible implementing a file explorer as a base machine function.

6.2 Guides

6.2.1 Getting started

6.2.1.1 File explorer with VC4 mapp template

This section provides a step-by-step explanation of how to create a file explorer and use it with the VC4 mapp template with the help of MpFileManagerUI.

6.2.1.1.1 Creating a project

You must first create a new project in Automation Studio.

6.2.1.1.2 Add mapp component

Adding a configuration for MpFile

The next step is to select the desired mapp component in the Toolbox and then drag-and-drop the configuration into the "mapp Services" folder.
Name the MpLink

When a configuration is added, the MpLink is automatically added along with it. When the configuration is opened, the MpLink is shown at the very top.

6.2.1.1.3 Adding a program

The next step is to add a program. In this case, we will add a Ladder Diagram program. We could use any of the other programming languages, however.
6.2.1.1.4 Adding MpFileManagerUI to the program

Searching for MpFile in the Ladder Diagram Catalog

We can select the "mapp" option in the Ladder Diagram Catalog. This will display all of the available mapp components that are in the project. In our case, this includes the file explorer.

Selecting a function block

Function block MpFileManagerUI is added to the program using drag-and-drop.

Adding function block

A function block instance is generated after we have pulled the function block into the program.
6.2.1.1.5 Configuring MpFileManagerUI

Connecting MpLink

First, connect input "MpLink" to the MpLink previously created in the Configuration View. Its address is passed on to the function block.

Enabling the component

The component is enabled by setting input "Enable". This is indicated by output "Active".
Defining UISetup
Now a UISetup variable must be created. The important thing here is that the "Value" field is empty.

Defining UIConnect
A UIConnect variable must be created in the next step. It is specified as an address.
Determining the data storage device

The files created in the file explorer are shown on the data storage device.

The data storage device for the file explorer is defined in the initialization subprogram using "MpFileMngConnect.DeviceList.DeviceNames[x]". A total of 10 data storage devices can be used.

6.2.1.1.6 Creating the data storage device

The next step is to create the data storage device ("CF"). 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.
6.2.1.1.7 Creating an HMI application

Adding a visualization object

To display the file explorer, we will create a VC4-based HMI application.

Selecting a template

In order to show the mapp HMI application, the "mapp Technology VGA 640x480 landscape" template must be selected.
Configuring VNC server for the HMI application

The HMI application is then connected to a VNC server.

6.2.1.8 Integrating the template into the project

Creating the connection between the task and visualization object

After opening the visualization object, select and update the data source. After the update, you will see which variables are connected in the HMI application but do not yet exist in the project.
Select the variable from the list that should be replaced by your own. Right-click on "MpFileManagerUI" and select the "Refactor" option.

Now select your own "MpFileMngConnect" UIConnect structure in the dialog box that opens.
You must then confirm the replacement. All connections are now created automatically.

After "refactoring", the data source is updated again. The previous "MpFileManagerUI" connection should now no longer be available.

6.2.1.1.9 Generating the file structure

Now generate a file structure for the memory card for ARsim.
6.2.1.1.10 Testing the HMI application

Opening the VNC viewer

The HMI application can be opened with VNC Viewer.

Home screen of the HMI application

The start page is displayed after successfully connecting to the HMI application using the VNC viewer. The file explorer appears by clicking on the “MpFile” button.
File explorer

The file explorer is now operational. All necessary data points are already connected. Folders can be created, deleted, etc.

6.3 Configuration

6.3.1 MpFileManagerUI configuration

Parameter "Customized file type" allows the user to define custom file extensions and display an icon for them in the HMI applications. The configuration of the extension here corresponds to MpFileManagerUIItemTypeEnum. If a new file extension is entered, it receives the value mpFILE_ITEM_TYPE_USERX in the enumeration.

6.4 Use cases

This section outlines several possible use cases for MpFile components.

6.4.1 Use case 1: File management system for various data storage devices

Requirement

A file management system should be implemented on the controller. It should be possible to manage various data sources such as CF, USB and FTP. In addition, basic file system functions (copy, rename, etc.) as well as the VC4 visualization connection to the file management system should be implemented.

Solution

Component list

- MpFileManagerUI (own MpLink): File management system with VC4 connection
Using the mapp components

**MpFileManagerUI** provides basic file system functions such as refresh, rename, copy, cut, paste, etc. It is possible to set up filters and sorting orders.

The contents of the selected data source (files/folders) are grouped together in structure **MpFileManagerUIFileListType**. The list with the data storage devices (FileDevices) are located under **MpFileManagerUIDeviceListType**. The different data storage devices must be entered here manually.

**UIConnect.Status** allows the respective status of **MpFileManagerUI** to be viewed. When idle, this is mpFILE_UI_STATUS_IDLE. During a command, the status is represented accordingly. It is then reset again to mpFILE_UI_STATUS_IDLE after the command has finished processing.

### 6.5 Libraries

#### 6.5.1 Function blocks

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MpFileManagerUI</td>
<td>This function block provides a file management system as well as a connection to Visual Components 4.</td>
</tr>
<tr>
<td>MpFileManagerConfig</td>
<td>This function block configures the file browser.</td>
</tr>
</tbody>
</table>

#### 6.5.1.1 MpFileManagerUI

This function block provides a file management system as well as a connection to Visual Components 4 to display files.

**Function block**

**Interface**

<table>
<thead>
<tr>
<th>ID</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 MpFileManagerUI 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>USetup</td>
<td>MpFileManagerUISetupType</td>
<td>Used to configure the elements connected to the HMI application.</td>
</tr>
<tr>
<td>IN</td>
<td>UIConnect</td>
<td>Pointer to MpFileManagerUIConnectType</td>
<td>This structure contains the parameters needed for the connection to the HMI application.</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>Info</td>
<td>MpFileInfoType</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 be carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

6.5.1.1 Description

The MpLink of an MpFileManagerUI configuration is used for this function block.

Initializing the file management system

The two structures MpFileManagerUISetupType and MpFileManagerUIConnectType must be configured before the function block is enabled.

MpFileManagerUIConnectType

Structure "UIConnect" is divided into the following areas:

- **Status**: Parameter "Status" provides information about the current activity of recipe management.
- **DeviceList**: Structure MpFileManagerUIDeviceListType is used to select the data storage medium (File Device) being accessed by the file manager. This list must be managed by the user. This means that the entries in the "DeviceNames" list are not created by the file manager; instead, valid file devices must be entered there.
- **File**: Structure MpFileManagerUIFileType contains the list of files and folders in the current path. This list can be sorted and filtered to make it easier to find a file or folder. This structure can also be used to navigate through folders and create new folders. In addition, standard functions such as copy, cut, paste, delete and rename (file/folder) are possible with this structure. File extensions and filter entries are not case sensitive. For example, a file can have extension .txt, .TxT or .TXT. All variants of the file will be stored as mpFILE_ITEM_TYPE_TXT.
- **MessageBox**: The various dialog boxes for MpFile can be configured using structure MpFileManagerUIMessageBoxType.
- **DefaultLayerStatus**: The background layer can be locked for the duration that a dialog box is open using this data point.

Search function

Parameter "Filter" in data type MpFileManagerUIFileType can be used to search for certain files. It is possible to use the "*" wildcard in the search function. If "*Test*.xml" is used for filtering, for example, the files "Test01.xml", "MyTest.xml" and "MyText01.xml" will be displayed. If the "*" wildcard is not used, then it will only search for "Text.xml".

If no filter is specified or only the "*" symbol, then all existing files and folders will be displayed.

MpFileManagerUISetupType

"FileListSize" defines in this structure how many entries from the file list should be displayed on one page of the HMI application. "FileListScrollWindow" determines how many entries from the list are displayed in advance when scrolling up and down.

6.5.1.2 MpFileManagerConfig

This function block configures the file browser.

Function block

## Optional parameters
mapp Services

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 MpFileManager 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>Configuration</td>
<td>Pointer to MpFileManagerConfigType</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>MpFileInfoType</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 be carried out in the initialization subroutine or in an acyclic task. However, an appropriately high stack must be configured in acyclic tasks (~6 kB).

6.5.1.2.1 Description

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

This function block reads and writes the configuration. The complete configuration 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 respective configuration data types for the function block explains when parameters in the configuration are applied.
Configuring structure **MpFileManagerConfigType**

User-defined file types can be added at runtime using structure **MpFileManagerConfigType**.

### 6.5.2 Data types and enumerators

#### 6.5.2.1 Data types

##### 6.5.2.1.1 MpFileDiagType

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>MpFileStatusIDType</td>
<td>StatusID diagnostic structure</td>
</tr>
</tbody>
</table>

##### 6.5.2.1.2 MpFileInfoType

This data type provides additional information for the **MpFile** component.

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

##### 6.5.2.1.3 MpFileManagerUIConnectType

This data type contains all of the information necessary to establish a connection to the HMI application.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>MpFileManagerUIStatusEnum</td>
<td>Current operation</td>
</tr>
<tr>
<td>DeviceList</td>
<td>MpFileManagerUIDeviceListType</td>
<td>List of data storage devices (file devices) accessible by the file manager</td>
</tr>
<tr>
<td>File</td>
<td>MpFileManagerUIFileType</td>
<td>Contains the file list as well as additional information about files, folders and possible commands</td>
</tr>
<tr>
<td>Message box</td>
<td>MpFileManagerUIMessageBoxType</td>
<td>Controls dialog boxes</td>
</tr>
<tr>
<td>DefaultLayerStatus</td>
<td>UINT</td>
<td>Status data point for the default layer of the visualization page where the file manager is being displayed</td>
</tr>
<tr>
<td>VC4 connection:</td>
<td>StatusDatapoint from Layer</td>
<td></td>
</tr>
</tbody>
</table>

##### 6.5.2.1.4 MpFileManagerConfigType

The file browser is configured using this data type.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustomizedFileType</td>
<td>ARRAY[0..9] of STRING[5]</td>
<td>User-defined file type</td>
</tr>
</tbody>
</table>

##### 6.5.2.1.5 MpFileManagerUIDeviceListType

List of possible data storage devices (file devices) and associated navigation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeviceNames</td>
<td>ARRAY[0..9] of STRING[50]</td>
<td>List of all available data storage devices</td>
</tr>
<tr>
<td>VC4 connection:</td>
<td>TextDatapoint from Dropdown</td>
<td></td>
</tr>
<tr>
<td>SelectedIndex</td>
<td>UINT</td>
<td>Index of the entry currently selected in the list</td>
</tr>
<tr>
<td>VC4 connection:</td>
<td>IndexDatapoint from Dropdown</td>
<td></td>
</tr>
</tbody>
</table>
### 6.5.2.1.6 MpFileManagerUIFileListType

**List of possible files/folders and associated navigation**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items</td>
<td>ARRAY[0..49] of MpFileManagerUIItemType</td>
<td>List of all files and folders under the current path</td>
</tr>
<tr>
<td>PageUp</td>
<td>BOOL</td>
<td>Jumps to the start of the current page and then scrolls up one page at a time The size of the page is defined using parameter &quot;FileListSize&quot; in structure MpFileManagerUISetupType. <strong>VC4 connection:</strong> Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0</td>
</tr>
<tr>
<td>PageDown</td>
<td>BOOL</td>
<td>Jumps to the end of the current page and then scrolls down one page at a time The size of the page is defined using parameter &quot;FileListSize&quot; in structure MpFileManagerUISetupType. <strong>VC4 connection:</strong> Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0</td>
</tr>
</tbody>
</table>
### 6.5.2.1.7 MpFileManagerUIFileType

This data type contains all parameters relevant to managing files.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>List</td>
<td>MpFileManagerUIFileListType</td>
<td>List of all folders and files under the current path, as well as the navigation of the list</td>
</tr>
<tr>
<td>Filter</td>
<td>STRING[255]</td>
<td>Filter used to sort the files. It is possible to use wildcard &quot;*&quot; in the filter function. For more information about the filter function, see section &quot;Search function&quot; in the description of MpFileManagerUI.Description</td>
</tr>
<tr>
<td>SortOrder</td>
<td>MpFileManagerUISortOrderEnum</td>
<td>Responsible for the order of files in the list</td>
</tr>
<tr>
<td>Refresh</td>
<td>BOOL</td>
<td>Updates the file list in the current folder</td>
</tr>
<tr>
<td>PathInfo</td>
<td>MpFileManagerUIPathInfoType</td>
<td>Information about the current path</td>
</tr>
<tr>
<td>EnterFolder</td>
<td>BOOL</td>
<td>Navigates one folder down</td>
</tr>
<tr>
<td>FolderUp</td>
<td>BOOL</td>
<td>Navigates one folder up</td>
</tr>
<tr>
<td>Parameter</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MultiSelect</td>
<td>BOOL</td>
<td>Allows the selection of multiple elements at the same time. VC4 connection: Datapoint from Button / Type: ToggleDatapoint / SetValue: 1 / ResetValue: 0</td>
</tr>
<tr>
<td>Delete</td>
<td>BOOL</td>
<td>Deletes a file or folder. VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0</td>
</tr>
<tr>
<td>Copy</td>
<td>BOOL</td>
<td>Copies a file/folder. When copying to the same folder where the source is located, (1), (2) will be appended to the name of the copied file. VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0</td>
</tr>
<tr>
<td>Cut</td>
<td>BOOL</td>
<td>Cuts a file or folder. VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0</td>
</tr>
<tr>
<td>Paste</td>
<td>BOOL</td>
<td>Pastes a file or folder. VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0</td>
</tr>
<tr>
<td>NewName</td>
<td>STRING[255]</td>
<td>The new name of a file or folder when using the &quot;Rename&quot; or &quot;CreateFolder&quot; command. The file can also be stored in a folder on the specified data storage device. In this case, the folder name must also be specified (e.g. &quot;FolderName/Filename&quot;). VC4 connection: Datapoint from String / Type: SetDatapoint / SetValue: 1 / ResetValue: 0</td>
</tr>
<tr>
<td>Rename</td>
<td>BOOL</td>
<td>Renames a file or folder. VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0</td>
</tr>
<tr>
<td>CreateFolder</td>
<td>BOOL</td>
<td>Creates a folder. VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1 / ResetValue: 0</td>
</tr>
</tbody>
</table>
6.5.2.1.8 MpFileManagerUIItemType

Provides a more exact description of a file or folder.

<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>Filename / Folder name</td>
</tr>
<tr>
<td>Size</td>
<td>UDINT</td>
<td>File size [bytes]</td>
</tr>
<tr>
<td>LastModified</td>
<td>DATE_AND_TIME</td>
<td>Date when this file/folder was last modified</td>
</tr>
<tr>
<td>Type</td>
<td>MpFileManagerUIItemTypeEnum</td>
<td>File type (.csv, jpg, etc.)</td>
</tr>
<tr>
<td>IsSelected</td>
<td>BOOL</td>
<td>Indicates that a file/folder is selected</td>
</tr>
<tr>
<td>IsFolder</td>
<td>BOOL</td>
<td>Indicates that the element is a folder</td>
</tr>
</tbody>
</table>

Info:

Buttons with the ToggleDatapoint action are used for file selection. The name a file has is defined by a text snippet in a text group. The size of the text group results from the number of files that should be displayed at one time. A text snippet is assigned to each element in the text group with a different index. (TextSnippet[0] connected with FileManUIConnect.File.List.Items[0].Name)
none

Displaying the file format:

The Type parameter is used to determine whether a file is a folder or an actual file. This enumerator can also be used to determine which file format is being used (.csv, .xml, .jpg, etc.).

A bitmap group is created for this that contains an image for each file format. Which index this image has in the bitmap ground is determined by the \texttt{MpFileManagerUIItemTypeEnum} enumerator.

Lastly, this bitmap group is specified as a property of the button (BitmapSource) and connected as a bitmap index offset to the Type parameter.

Figure 16: BitmapGroup file formats

6.5.2.1.9 \texttt{MpFileManagerUIMessageBoxType}

Data type used to control the display of a dialog box.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LayerStatus</td>
<td>UINT</td>
<td>Visibility of the dialog box (status data point for the dialog box layer)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VC4 connection: StatusDatapoint from Layer</td>
</tr>
<tr>
<td>Type</td>
<td>MpFileManagerUIMessageEnum</td>
<td>Type of dialog box</td>
</tr>
<tr>
<td>ErrorNumber</td>
<td>UINT</td>
<td>Current error number to be displayed (corresponds to the contents of &quot;StatusID&quot;) (reserved for future use)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VC4 connection: Datapoint from Numeric</td>
</tr>
<tr>
<td>Confirm</td>
<td>BOOL</td>
<td>Confirms the operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1</td>
</tr>
<tr>
<td>Cancel</td>
<td>BOOL</td>
<td>Cancels the operation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VC4 connection: Datapoint from Button / Type: SetDatapoint / SetValue: 1</td>
</tr>
</tbody>
</table>

### 6.5.2.1.10 MpFileManagerUIPathInfoType

Additional information that can be displayed in the HMI application.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentDir</td>
<td>STRING[255]</td>
<td>Current folder name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VC4 connection: TextDatapoint from Text</td>
</tr>
<tr>
<td>FileCount</td>
<td>UDINT</td>
<td>Number of files in the current folder</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VC4 connection: Datapoint from Numeric</td>
</tr>
<tr>
<td>FolderCount</td>
<td>UDINT</td>
<td>Number of subfolders in the current folder</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VC4 connection: Datapoint from Numeric</td>
</tr>
</tbody>
</table>
6.5.2.1.11 MpFileManagerUISetupType

Additional configuration options for the HMI application.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Data type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileListSize</td>
<td>UINT</td>
<td>10</td>
<td>Number of files to be displayed on one page of the list in the HMI application. Maximum 50 files/folders can be displayed.</td>
</tr>
<tr>
<td>FileListScrollWindow</td>
<td>USINT</td>
<td>1</td>
<td>Determines how many entries from the list are initially displayed when scrolling up and down.</td>
</tr>
</tbody>
</table>

**FileListSize**

FileListSize specifies how many files are displayed on one side of a VC4-based HMI application. The number of buttons in the HMI application that are used to show the files must be equal to FileListSize. (See image above.)
FileListScrollWindow specifies how many entries should be "previewed" when scrolling. In this example, FileListScrollWindow = 1 is defined. If the current selection is fourth in the list, then pressing the StepDown button will select the next entry (Any File 2) and preview the next entry (Any File 3).

6.5.2.1.12 MpFileStatusIDType

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>MpFileErrorEnum</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>

6.5.2.2 Enumerators

6.5.2.2.1 MpFileManagerUIItemTypeEnum

This enumerated data type describes which type a file has.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpFILE_ITEM_TYPE_NONE</td>
<td>No type defined</td>
</tr>
<tr>
<td>mpFILE_ITEM_TYPE_FOLDER</td>
<td>Indicates that the element is a folder</td>
</tr>
<tr>
<td>mpFILE_ITEM_TYPE_FILE</td>
<td>Indicates that the element is a file</td>
</tr>
<tr>
<td>mpFILE_ITEM_TYPE_TXT</td>
<td>Indicates that the element is a text file</td>
</tr>
<tr>
<td>mpFILE_ITEM_TYPE_HTML</td>
<td>Indicates that the element is an HTML file</td>
</tr>
<tr>
<td>mpFILE_ITEM_TYPE_DOC</td>
<td>Indicates that the element is a Word file (.doc)</td>
</tr>
<tr>
<td>mpFILE_ITEM_TYPE_XLS</td>
<td>Indicates that the element is an Excel file (.xls)</td>
</tr>
<tr>
<td>mpFILE_ITEM_TYPE_XML</td>
<td>Indicates that the element is an XML file (.xml)</td>
</tr>
<tr>
<td>mpFILE_ITEM_TYPE_CSV</td>
<td>Indicates that the element is a CSV file (.csv)</td>
</tr>
<tr>
<td>mpFILE_ITEM_TYPE_JPG</td>
<td>Indicates that the element is an image file (.jpg)</td>
</tr>
<tr>
<td>mpFILE_ITEM_TYPE_BMP</td>
<td>Indicates that the element is an image file (.bmp)</td>
</tr>
<tr>
<td>mpFILE_ITEM_TYPE_USER1</td>
<td>The element is a user-defined file (can be specified in the configuration).</td>
</tr>
<tr>
<td>mpFILE_ITEM_TYPE_USER2</td>
<td>The element is a user-defined file (can be specified in the configuration).</td>
</tr>
<tr>
<td>mpFILE_ITEM_TYPE_USER3</td>
<td>The element is a user-defined file (can be specified in the configuration).</td>
</tr>
<tr>
<td>mpFILE_ITEM_TYPE_USER4</td>
<td>The element is a user-defined file (can be specified in the configuration).</td>
</tr>
<tr>
<td>mpFILE_ITEM_TYPE_USER5</td>
<td>The element is a user-defined file (can be specified in the configuration).</td>
</tr>
<tr>
<td>mpFILE_ITEM_TYPE_USER6</td>
<td>The element is a user-defined file (can be specified in the configuration).</td>
</tr>
<tr>
<td>mpFILE_ITEM_TYPE_USER7</td>
<td>The element is a user-defined file (can be specified in the configuration).</td>
</tr>
<tr>
<td>mpFILE_ITEM_TYPE_USER8</td>
<td>The element is a user-defined file (can be specified in the configuration).</td>
</tr>
<tr>
<td>mpFILE_ITEM_TYPE_USER9</td>
<td>The element is a user-defined file (can be specified in the configuration).</td>
</tr>
<tr>
<td>mpFILE_ITEM_TYPE_USER10</td>
<td>The element is a user-defined file (can be specified in the configuration).</td>
</tr>
<tr>
<td>mpFILE_ITEM_TYPE_UNKNOWN</td>
<td>Indicates that the file type is unknown</td>
</tr>
</tbody>
</table>

6.5.2.2.2 MpFileManagerUIMessageEnum

This enumerated data type determines the type of dialog box.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpFILE_MSG_OK</td>
<td>Dialog box for requesting confirmation of input</td>
</tr>
<tr>
<td>mpFILE_MSG_CONFIRM_DELETE</td>
<td>Dialog box for requesting confirmation of deletion</td>
</tr>
<tr>
<td>mpFILE_MSG_CONFIRM_OVERWRITE</td>
<td>Dialog box for requesting confirmation of overwriting</td>
</tr>
<tr>
<td>mpFILE_MSG_BUSY</td>
<td>Dialog box that shows that another actions is currently being executed</td>
</tr>
</tbody>
</table>

6.5.2.2.3 MpFileManagerUISortOrderEnum

This enumerated data type is a template for all other mapp component enumerated data types.
6.5.2.2.4 MpFileManagerUIStatusEnum

This enumerated data type provides information about the current activity of a component.

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpFILE_UI_STATUS_IDLE</td>
<td>No process is currently active.</td>
</tr>
<tr>
<td>mpFILE_UI_STATUS_REFRESH</td>
<td>Updating elements</td>
</tr>
<tr>
<td>mpFILE_UI_STATUS_CHANGE_PAGE</td>
<td>Page change active</td>
</tr>
<tr>
<td>mpFILE_UI_STATUS_CHANGE_DIR</td>
<td>Folder change active</td>
</tr>
<tr>
<td>mpFILE_UI_STATUS_CHANGE_DEVICE</td>
<td>Data storage device (FileDevice) being changed</td>
</tr>
<tr>
<td>mpFILE_UI_STATUS_CREATE</td>
<td>Creating a file/folder active</td>
</tr>
<tr>
<td>mpFILE_UI_STATUS_RENAME</td>
<td>Renaming a file/folder active</td>
</tr>
<tr>
<td>mpFILE_UI_STATUS_DELETE</td>
<td>Deleting a file/folder active</td>
</tr>
<tr>
<td>mpFILE_UI_STATUS_COPY</td>
<td>Copying a file/folder active</td>
</tr>
<tr>
<td>mpFILE_UI_STATUS_CUT</td>
<td>Cutting a file/folder active</td>
</tr>
<tr>
<td>mpFILE_UI_STATUS_PASTE</td>
<td>Pasting a file/folder active</td>
</tr>
<tr>
<td>mpFILE_UI_STATUS_ERROR</td>
<td>Error active</td>
</tr>
</tbody>
</table>

6.5.3 Error numbers

6.5.3.1 -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:
- MpFileManagerUI

Constant:
mpFILE_ERR_ACTIVATION

6.5.3.2 -1064329102: 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:
- **MpFileManagerUI**

**Constant:**

mpFILE_ERR_MPLINK_NULL

**6.5.3.3 -1064329101: 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:**
- MpFileManagerUI

**Constant:**

mpFILE_ERR_MPLINK_INVALID

**6.5.3.4 -1064329100: 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:**
- MpFileManagerUI

**Constant:**

mpFILE_ERR_MPLINK_CHANGED

**6.5.3.5 -1064239099: Invalid MpLink contents**

**Description:**
The value of variable "MpLink" 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:

- MpFileManagerUI

**Constant:**

mpFILE_ERR_MPLINK_CORRUPT

**6.5.3.6 -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:

- MpFileManagerUI

**Constant:**

mpFILE_ERR_MPLINK_IN_USE

**6.5.3.7 -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.
- The function is not enabled in the configuration.

These function blocks / functions can report this error:

- MpFileManagerUI

**Constant:**

mpFILE_ERR_CONFIG_INVALID

**6.5.3.8 -1064165376: Missing value on UIConnect**

**Description:**

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

Cause/Solution:
- Input "UIConnect" forgotten

Constant:
mpFILE_ERR_MISSING_UICONNECT

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

6.5.3.9 -1064165375: Command not permitted

Description:
This command is not allowed in multi-select mode.

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

Cause/Solution:
- Turn off multi-select.
- Change the command.

Constant:
mpFILE_ERR_CMD_NOT_ALLOWED

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

6.5.3.10 -1064165374: No element to paste

Description:
No element was selected that can be pasted.

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

Cause/Solution:
- Copy or cut the element before issuing the "Paste" command.

Constant:
mpFILE_ERR_NOTHING_TO_PASTE

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

6.5.3.11 -1064165373: Nothing selected

Description:
Nothing was selected for an operation.
**Reaction:**
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

**Cause/Solution:**
- First select a file or folder.

**Constant:**
`mpFILE_ERR NOTHING SELECTED`

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

**6.5.3.12 -1064165372: Folder already exists**

**Description:**
An error occurred while creating or renaming a folder since the new name 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:**
- The new name ("NewName" in `MpFileManagerUIFileType`) already exists in this path.

**Constant:**
`mpFILE_ERR_DIR_ALREADY_EXISTS`

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

**6.5.3.13 -1064165371: Invalid file device**

**Description:**
The selected data storage device (file device) 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:**
- Check the connection to the data storage device (FileDevice) ("DeviceList" and "DeviceIndex" in structure `MpFileManagerUIConnectType`).

**Constant:**
`mpFILE_ERR INVALID_FILE_DEV`

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

**6.5.3.14 -1064165370: New name not entered**

**Description:**
A new name was not found when creating or renaming a file/folder.
Reaction:
The function block indicates an active error on outputs "StatusID" and "Error". No other functions are available during this time.

Cause/Solution:
- The new name ("NewName" in MpFileManagerUIFileType) is blank.

Constant:
mpFILE_ERR_NAME_EMPTY

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

6.5.3.15 -1064165369: Invalid name

Description:
The name of the file or folder 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:
- The following special characters are not permitted in names: \\ /*?"<>|
- A file with the same name already exists.

Constant:
mpFILE_ERR_INVALID_NAME

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

6.5.3.16 -1064165368: Paste not allowed

Description:
It is not possible to paste in this path.

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

Cause/Solution:
- Pasting in a path where it is not permitted (locked, already exists, etc.)
- Check the path.

Constant:
mpFILE_ERR_PASTE_NOT_ALLOWED

These function blocks / functions can report this error:
- MpFileManagerUI
6.5.3.17 -1064165367: General error

Description:
General error in file system. See Logger for more information.

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

Cause/Solution:
• See Logger for more information.

Constant:
mpFILE_ERR_FILE_SYSTEM

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

6.5.4 Alarms

6.5.4.1 mpFILE_ALM_DEVICE_NOT_FOUND: FileDevice not found

Description:
Action failed due to a missing data storage device (file device).

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

Cause/Solution:
• Check the parameter on input "DeviceList" in data type .
• The value on input "DeviceIndex" may not be valid.

Behavior
Edge alarm

6.5.4.2 mpFILE_ALM_ALREADY_EXISTING: File/Folder already exists

Description:
The action failed since the file/folder already exists.

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

Cause/Solution:
• Delete or rename the file/folder.

Behavior
Edge alarm

6.5.4.3 mpFILE_ALM_ERROR_FILE_SYSTEM: General error in file system

Description:
General error in file system. Details can be found in the additional information on the function block’s output.
Reaction:
A mapp alarm is triggered if a MpAlarmX component is active.

Cause/Solution:
- The exact cause can be found in the additional information (structure "Info" - parameter "Code").

Behavior
Edge alarm