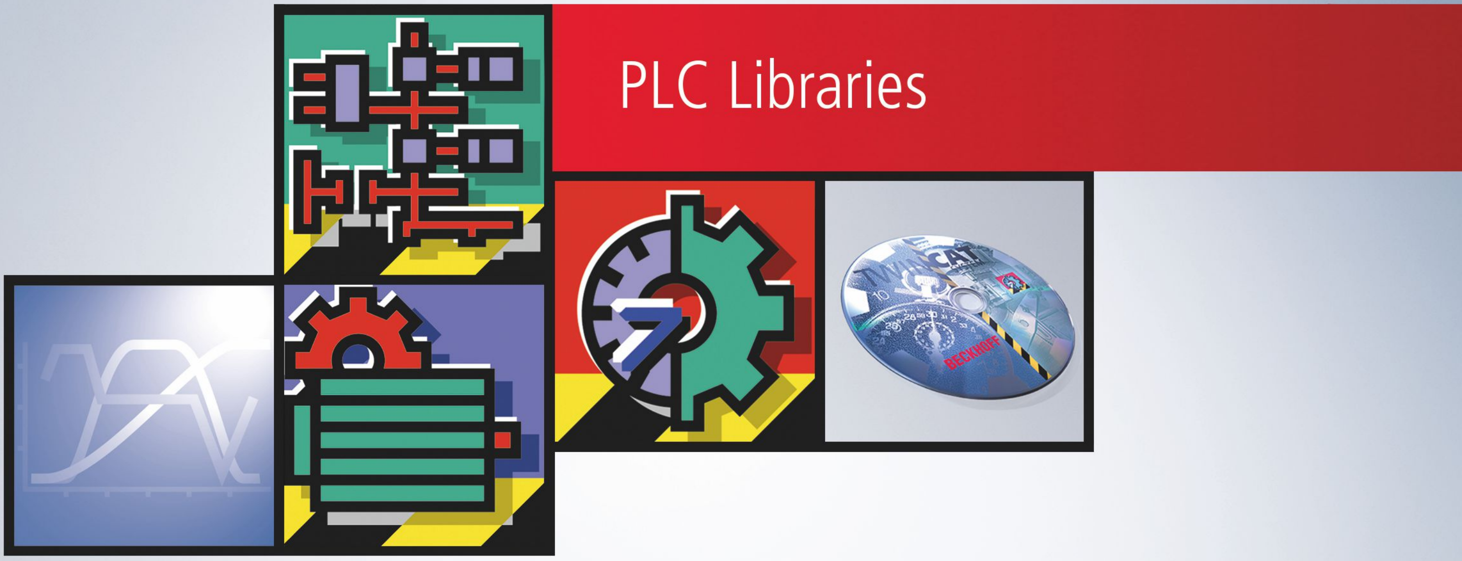
BECKHOFF New Automation Technology

Manual | EN

TX1200

TwinCAT 2 | PLC Library: TcMPBus



2023-09-04 | Version: 1.4

Table of contents

1 **Foreword** ... **5**
- 1.1 Notes on the documentation ... 5
- 1.2 For your safety ... 6
- 1.3 Notes on information security ... 7

2 **Introduction** ... **8**

3 **Target groups** ... **9**

4 **MP-Bus** ... **10**
- 4.1 Topology ... 10
- 4.2 Actuator solutions ... 10

5 **Integration into TwinCAT** ... **14**
- 5.1 KL6771 - Linking to the TwinCAT System Manager ... 14
- 5.2 Integration in TwinCAT (CX9020) ... 16
- 5.3 Integration into TwinCAT (BC9191) ... 18

6 **Programming** ... **21**
- 6.1 General Information ... 24
- 6.2 Function blocks ... 24
  - 6.2.1 KL6771 ... 26
  - 6.2.2 MP_Addressing ... 26
  - 6.2.3 MP_CMV ... 28
  - 6.2.4 MP_DamperLinearActuator ... 30
  - 6.2.5 MP_EnergyValveV4_Configuration ... 31
  - 6.2.6 MP_EnergyValveV4_Process ... 34
  - 6.2.7 MP_EPIV ... 36
  - 6.2.8 MP_EPIV_R6 ... 38
  - 6.2.9 MP_EPIV_R6_Parameter ... 39
  - 6.2.10 MP_EV ... 41
  - 6.2.11 MP_EV_Parameter ... 43
  - 6.2.12 MP_MPX ... 44
  - 6.2.13 MP_OperatingUnit ... 45
  - 6.2.14 MP_OperatingUnit_ConfigurationCO2 ... 47
  - 6.2.15 MP_OperatingUnit_ConfigurationDisplay ... 48
  - 6.2.16 MP_OperatingUnit_ConfigurationOffsetValues ... 49
  - 6.2.17 MP_OperatingUnit_ConfigurationStatusIcons ... 50
  - 6.2.18 MP_OperatingUnit_ConfigurationTemp ... 51
  - 6.2.19 MP_OperatingUnit_ConfigurationVentilation ... 52
  - 6.2.20 MP_PTH ... 53
  - 6.2.21 MP_RoomSensor ... 54
  - 6.2.22 MP_RoomSensor_Parameter ... 55
  - 6.2.23 MP_Smoker ... 57
  - 6.2.24 MP_TEM_Configuration ... 58
  - 6.2.25 MP_TEM_Process ... 59
  - 6.2.26 MP_UST_3 ... 61
  - 6.2.27 MP_VAV ... 64

6.2.28 MP_VRU_Configuration 66
6.2.29 MP_VRU_Process 67
6.2.30 MP_Window 69
6.2.31 MPL_DamperLinearActuator 70
6.3 Functions 71
6.3.1 NI1000_LuS_TO_INT : INT 71
6.3.2 NI1000_TO_INT : INT 71
6.3.3 NTC_TO_INT : INT 72
6.3.4 PT1000_TO_INT : INT 72
6.4 Data types 73
6.4.1 Enums 73
6.4.2 Structures 84
6.5 Error codes 87

7 Appendix 89
7.1 Examples 89
7.2 Support and Service 89

1 Foreword

1.1 Notes on the documentation

This description is only intended for the use of trained specialists in control and automation engineering who are familiar with applicable national standards.
It is essential that the documentation and the following notes and explanations are followed when installing and commissioning the components.
It is the duty of the technical personnel to use the documentation published at the respective time of each installation and commissioning.

The responsible staff must ensure that the application or use of the products described satisfy all the requirements for safety, including all the relevant laws, regulations, guidelines and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without prior announcement.
No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.
Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
with corresponding applications or registrations in various other countries.

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

1.2 For your safety

Safety regulations

Read the following explanations for your safety.
Always observe and follow product-specific safety instructions, which you may find at the appropriate places in this document.

Exclusion of liability

All the components are supplied in particular hardware and software configurations which are appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation, and drive technology who are familiar with the applicable national standards.

Signal words

The signal words used in the documentation are classified below. In order to prevent injury and damage to persons and property, read and follow the safety and warning notices.

Personal injury warnings

⚠ DANGER
Hazard with high risk of death or serious injury.

⚠ WARNING
Hazard with medium risk of death or serious injury.

⚠ CAUTION
There is a low-risk hazard that could result in medium or minor injury.

Warning of damage to property or environment

NOTICE
The environment, equipment, or data may be damaged.

Information on handling the product

This information includes, for example:
recommendations for action, assistance or further information on the product.

1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.

2 Introduction

The MP-Bus library is a TwinCAT PLC library for data exchange with MP-Bus devices.

All function blocks from the library must be called in the same task.

This library is to be used only in conjunction with a MP-Bus master terminal KL6771.

3 Target groups

The user of this library requires basic knowledge of the following:

- TwinCAT PLC-Control
- TwinCAT System Manager
- PCs and networks
- Structure and properties of the Beckhoff Embedded PC and its Bus Terminal system
- Technology of MP-Bus devices
- Relevant safety regulations for building technical equipment

This software library is intended for building automation system partners of Beckhoff Automation GmbH & Co. KG. The system partners operate in the field of building automation and are concerned with the installation, commissioning, expansion, maintenance and service of measurement, control and regulating systems for the technical equipment of buildings.

4 MP-Bus

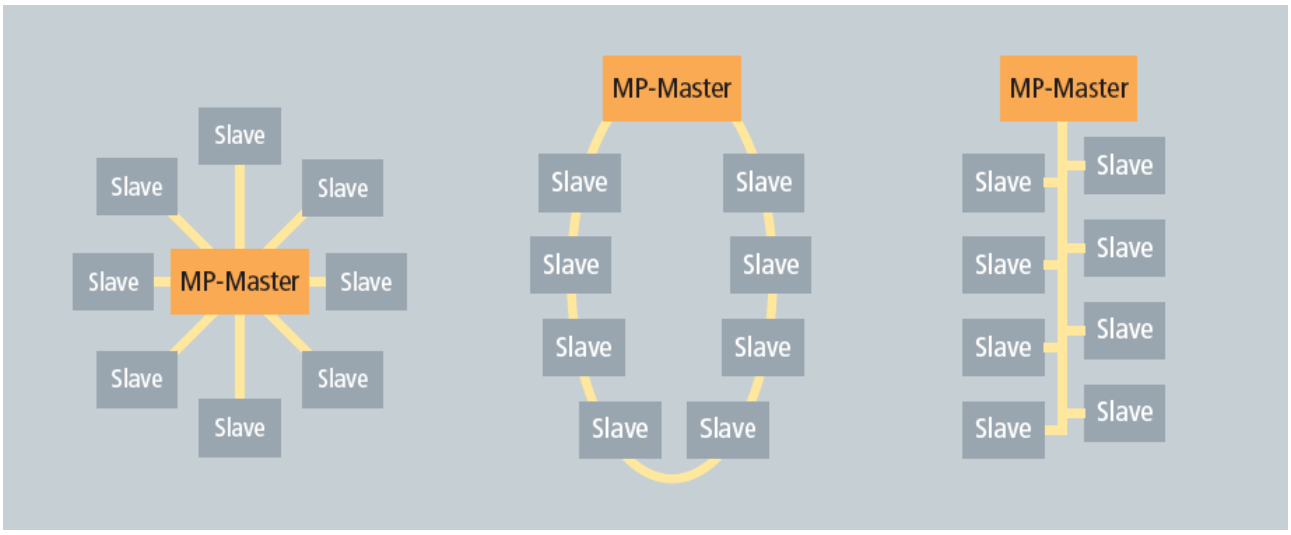
MP-bus = multi-point bus

The MP-Bus (Multi-Point) is a simple sensor/actuator bus, which is used for certain technical systems of the building automation system. The MP-Bus serves to control HVAC actuators for dampers, control valves and volume flow controllers from the Belimo product range. Up to eight different devices from HVAC systems can be connected to an MP-Bus master using 3-wire technology. Additionally, a sensor can be connected to each of these eight devices; the sensor is addressed by the MP-Bus. An additional range of products with an MP-Bus connection is the FLS window ventilation system by Belimo (see Belimo documentation for the connection of the Belimo actuators).

The MP-Bus was developed by Belimo for connecting valves, throttle valves, air valves, fire dampers, and for window ventilation systems.

4.1 Topology

There are no restrictions whatsoever with regard to the topological structure of MP-Bus strands: star, ring, tree or mixed topologies are possible. The length of the entire bus strand depends on the selected cable cross-section and the type as well as the number of connected drives! Documentation from the Belimo Automation AG contains further information.



4.2 Actuator solutions

Types ¹	Unit	ready for extended MP-Bus® ²	Beckhoff function block
General air solution			
Rotary actuators			
CM24-MPL-L	2 Nm	•	MPL_DamperLinearActuator
CM24-MPL-R	2 Nm	•	
LM24A-MP	5 Nm		MP_DamperLinearActuator
NM24A-MP	10 Nm		
SM24A-MP	20 Nm		
GM24A-MP	40 Nm		
Rotary actuators with emergency control function			
TF24-MFT	2 Nm		MP_DamperLinearActuator
LF24-MFT2	4 Nm		
NF24A-MP	10 Nm		

Types [1]	Unit	ready for extended MP-Bus® [2]	Beckhoff function block
SF24A-MP	20 Nm		
EF24A-MP	30 Nm		
GK24A-MP	40 Nm		
Linear actuators			
LH24A-MP... 60/100/200/300	150 Nm		MP_DamperLinearActuator
SH24A-MP... 100/200/300	450 Nm		
Linear actuators with emergency control function			
LHK24A-MP100	150 Nm		MP_DamperLinearActuator
SHK24A-MP100	450 Nm		
Rotary actuators			
LU24A-MP	3 Nm		MP_DamperLinearActuator
Fast running rotary actuators			
NMC24A-MP	10 Nm		MP_DamperLinearActuator
SMC24A-MP	20 Nm		
Rotary actuators for special applications			
NM24P-MP	10 Nm		MP_DamperLinearActuator
SM24P-MP	20 Nm		
GM24G-MP-T	40 Nm		
Rotary actuators with emergency control function for special applications			
NF24G-MP-L	10 Nm		MP_DamperLinearActuator
SF24G-MP-L	20 Nm		
GK24G-MP	40 Nm		
Water solutions			
Actuators for control ball valves / open-close control ball valves			
CQ24A-MPL	1 Nm	•	MPL_DamperLinearActuator
LR24A-MP	5 Nm		MP_DamperLinearActuator
NR24A-MP	10 Nm		
SR24A-MP	20 Nm		
Actuators with emergency control function for control ball valves / open-close control ball valves			
TRF24-MFT	2 Nm		MP_DamperLinearActuator
LRF24-MP	4 Nm		
NRF24A-MP	10 Nm		
Actuators for globe valves			
LV24A-MP-TPC	500 N		MP_DamperLinearActuator
NV24A-MP-TPC	1000 N		
SV24A-MP-TPC	1500 N		
EV24A-MP-TPC	2500 N		
Actuators with emergency control function for globe valves			
NVK24A-MP-TPC	1000 N		MP_DamperLinearActuator
AVK24A-MP-TPC	2000 N		
Fast running actuators for globe valves			
LVC24A-MP-TPC	500 N		MP_DamperLinearActuator
NVC24A-MP-TPC	1000 N		
SVC24A-MP-TPC	1500 N		
Fast running actuators with emergency control function for globe valves			

Types ¹	Unit	ready for extended MP-Bus® ²	Beckhoff function block
NVKC24A-MP-TPC	1000 N		MP_DamperLinearActuator
Actuators for dampers			
SR24A-MP-5	20 Nm		MP_DamperLinearActuator
GR24A-MP-5/7	40 Nm		
DR24A-MP-...5/7	90 Nm		
PRCA-BAC-S2-T	160 Nm		
PRKCA-BAC-S2-T	160 Nm		
SY2-24-MP-T	90 Nm		
SY2-230-MP-T	90 Nm		
SY3-24-MP-T	150 Nm		
SY3-230-MP-T	150 Nm		
SY4-24-MP-T	400 Nm		
SY4-230-MP-T	400 Nm		
SY5-24-MP-T	500 Nm		
Safety solutions			
"BF-TopLine" actuators for fire dampers			
BKN230-24MP for connection of BF(G)24TL-T-ST	11 / 18 Nm		MP_Smoker
Standard actuator for fire dampers			
BKN230-24-C-MP for connection of BF..24-..ST	4 / 9 / 11 / 18 Nm		MP_Smoker
Room and system solutions			
VAV compact rotary actuators			
LMV-D3-MP	5 Nm		MP_VAV
NMV-D3-MP	10 Nm		
SMV-D3-MP	20 Nm		
VAV compact linear actuators			
LHV-D3-MP	150 N		MP_VAV
VAV Universal			
VRP-M			MP_VAV
VRU-D3-BAC			MP_VRU_Process MP_VRU_Configuration
VRU-M1-BAC			
VRU-M1R-BAC			
VAV indoor climate control system			
CMV-100-MP	DN 100	•	MP_CMV
CMV-125-MP	DN 125	•	
CMV-150-MP	DN 150	•	
CMV-160-MP	DN 160	•	
HVAC performance devices			
EPIV			
EP..R-R6+BAC	DN 15-20	•	MP_EPIV_R6 MP_EPIV_R6_Parameter
EP..R+MP	DN 15-20		MP_EPIV
P..W..E-MP	DN 65-150		
Energy Valve™			
EV..R+BAC	DN 15-50	•	MP_EV

Types [1]	**Unit**	**ready for extended MP-Bus®** [2]	**Beckhoff function block**
P..W..EV-BAC	DN 65-150	•	MP_EV_Parameter
Energy Valve™ V4			
EV..R2+..	DN 15-50	•	MP_EnergyValveV4_Process MP_EnergyValveV4_Configuration
Thermal Energy Meter			
22PE- ..		•	MP_TEM_Process MP_TEM_Configuration
22PEM-..		•	
Sensors			
Room sensors in the comfort zone			
22RT-19-1 (temp.)		•	MP_RoomSensor MP_RoomSensor_Parameter
22RTH-19-1 (temp., rh)		•	
22RTM-19-1 (temp., CO_2, rh)		•	
MS24A-R02-MPX (Temp., CO_2)		•	MP_MPX
MS24A-R08-MPX (Temp., VOC, CO_2, rH)		•	
Room control units			
P-22Rxx-1900x-1		•	MP_OperatingUnit MP_OperatingUnit_ConfigurationCO2 MP_OperatingUnit_ConfigurationTemp MP_OperatingUnit_ConfigurationDisplay MP_OperatingUnit_ConfigurationVentilation MP_OperatingUnit_ConfigurationStatusIcons MP_OperatingUnit_ConfigurationOffsetValues

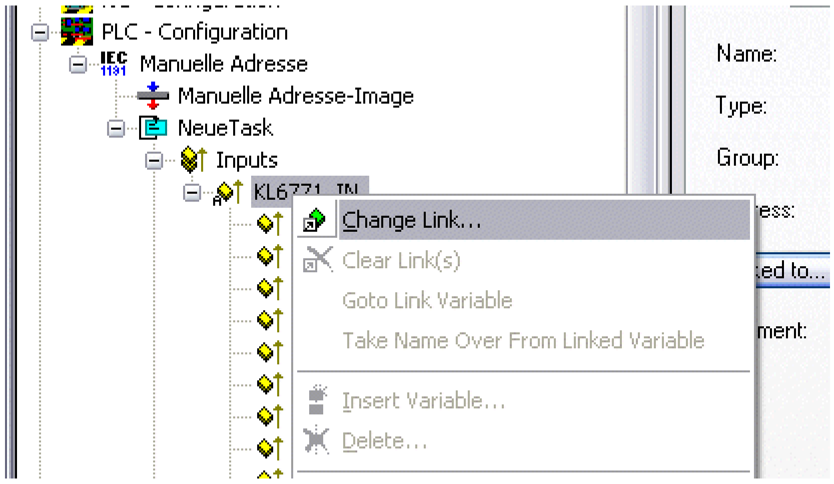
[1] The currently available product range can be found online at www.belimo.eu.

[2] The marked devices support the extended address range. Up to 16 MP devices (addressed MP1 ... MP16) can be connected to a data line. If unmarked devices are connected to the same data line, the common address range should be limited to 8 MP devices.

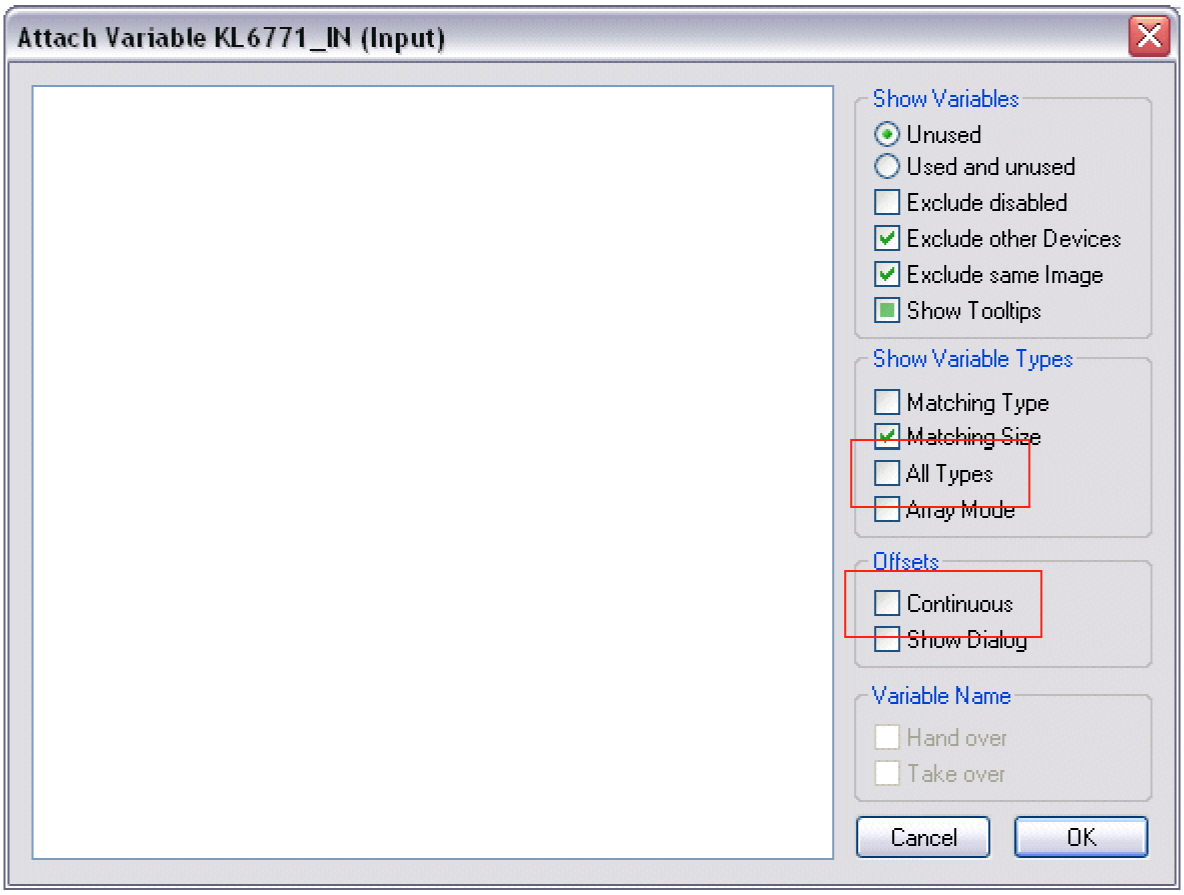
5 Integration into TwinCAT

5.1 KL6771 - Linking to the TwinCAT System Manager

How do I link the KL6771 to the System Manager?



1. Select **All Types** and **Continuous**


Attach Variable KL6771_IN (Input)

Show Variables
- [✓ Unused]
- Used and unused
- ☐ Exclude disabled
- ✓ Exclude other Devices
- ✓ Exclude same Image
- 🟩 Show Tooltips

Show Variable Types
- ☐ Matching Type
- ✓ Matching Size
- ☐ All Types
- ☐ Array Mode

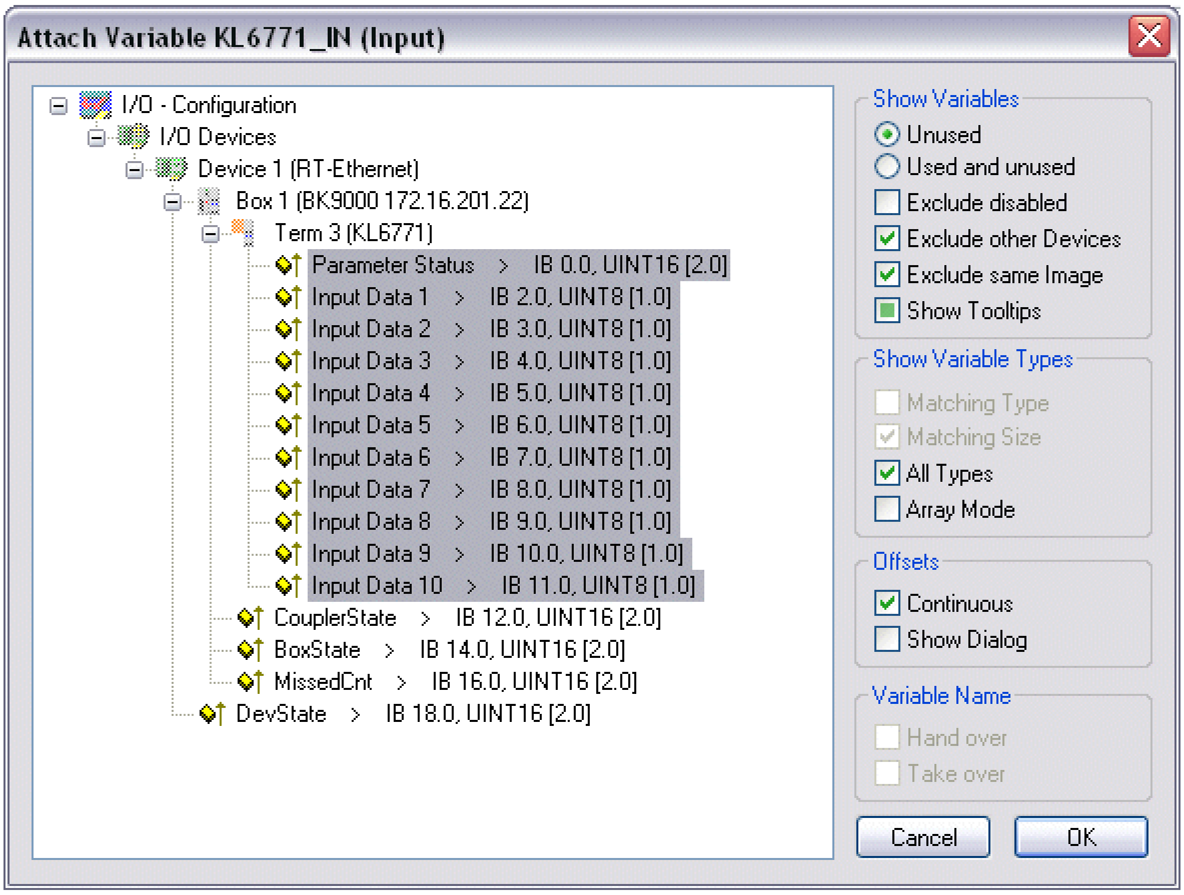
Offsets
- ☐ Continuous
- ☐ Show Dialog

Variable Name
- ☐ Hand over
- ☐ Take over

2. Click the mouse on the first variable in the KL6771 **Parameter Status**. Then press the [SHIFT] key, and hold it down. Take the mouse pointer to the last variable in the KL6771 **Input Data 10**, and again click the left mouse button. Now release the [SHIFT] key again.

   ⇨ All the terminal's data should now be highlighted (see Figure).

3. Then press the **OK** button.


Attach Variable KL6771_IN (Input)

I/O - Configuration
I/O Devices
Device 1 (RT-Ethernet)
Box 1 (BK9000 172.16.201.22)
Term 3 (KL6771)
Parameter Status > IB 0.0, UINT16 [2.0]
Input Data 1 > IB 2.0, UINT8 [1.0]
Input Data 2 > IB 3.0, UINT8 [1.0]
Input Data 3 > IB 4.0, UINT8 [1.0]
Input Data 4 > IB 5.0, UINT8 [1.0]
Input Data 5 > IB 6.0, UINT8 [1.0]
Input Data 6 > IB 7.0, UINT8 [1.0]
Input Data 7 > IB 8.0, UINT8 [1.0]
Input Data 8 > IB 9.0, UINT8 [1.0]
Input Data 9 > IB 10.0, UINT8 [1.0]
Input Data 10 > IB 11.0, UINT8 [1.0]
CouplerState > IB 12.0, UINT16 [2.0]
BoxState > IB 14.0, UINT16 [2.0]
MissedCnt > IB 16.0, UINT16 [2.0]
DevState > IB 18.0, UINT16 [2.0]

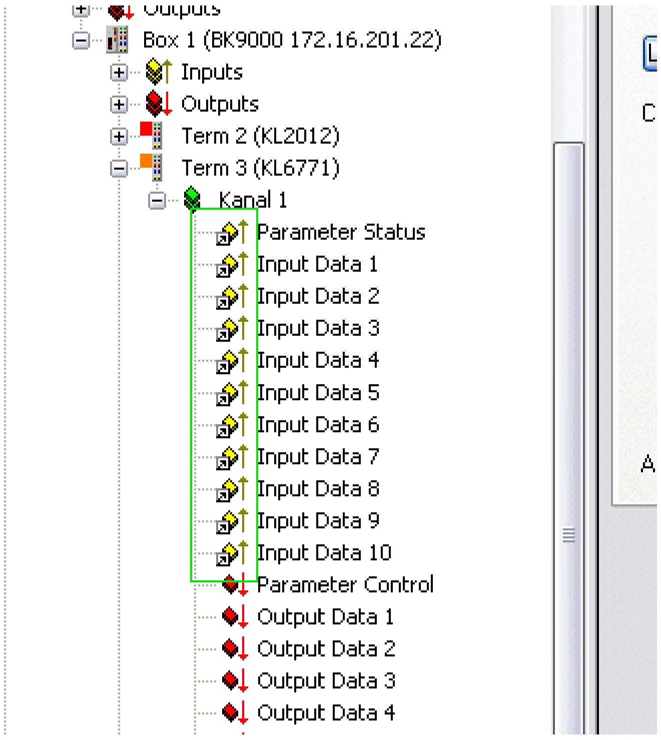
Show Variables
[● Unused] ○ Used and unused
☐ Exclude disabled
☑ Exclude other Devices
☑ Exclude same Image
☑ Show Tooltips

Show Variable Types
☐ Matching Type
☑ Matching Size
☑ All Types
☐ Array Mode

Offsets
☑ Continuous
☐ Show Dialog

Variable Name
☐ Hand over
☐ Take over

4. You can now check the link. To do this, go to the KL6771 and open it. All the terminal's data should now be marked by a small arrow (see Figure).


Outputs
Box 1 (BK9000 172.16.201.22)
Inputs
Outputs
Term 2 (KL2012)
Term 3 (KL6771)
Kanal 1
Parameter Status
Input Data 1
Input Data 2
Input Data 3
Input Data 4
Input Data 5
Input Data 6
Input Data 7
Input Data 8
Input Data 9
Input Data 10
Parameter Control
Output Data 1
Output Data 2
Output Data 3
Output Data 4

⇨ If that is the case, then proceed in exactly the same way with the outputs.

5.2 Integration in TwinCAT (CX9020)

This example describes how a simple PLC program for MP-Bus can be written in TwinCAT and how it is linked with the hardware. The task is to control an individual flap drive and change it with a button.

https://infosys.beckhoff.com/content/1033/tcplclibmpbus/Resources/12063685643/.zip https://infosys.beckhoff.com/content/1033/tcplclibmpbus/Resources/12063685643/.zip

Hardware

Setting up the components

The following hardware is required:

- 1x Bus Terminal Controller CX9020
- 1x digital 2-channel input terminal KL1002 (for the Open and Close functions)
- 1x MP-Bus master terminal KL6771
- 1x end terminal KL9010

Set up the hardware and the MP-Bus components as described in the associated documentation.

This example assumes that the address of the flap drive is known.

Software

Creation of the PLC program

Create a new PLC project for PC-based systems (ARM) and add the *TcMPBus.lib* library.

Next, generate the following global variables:

```
VAR_GLOBAL
    bOpen        AT %I*      : BOOL;
    bClose       AT %I*      : BOOL;
    arrKL6771_IN     AT %I* : ARRAY[0..11] OF BYTE;
    arrKL6771_OUT    AT %Q* : ARRAY[0..11] OF BYTE;
    stDataKL6771            : DataKL6771;
END_VAR
```

bOpen: input variable for the Open button.

bClose: input variable for the Close button.

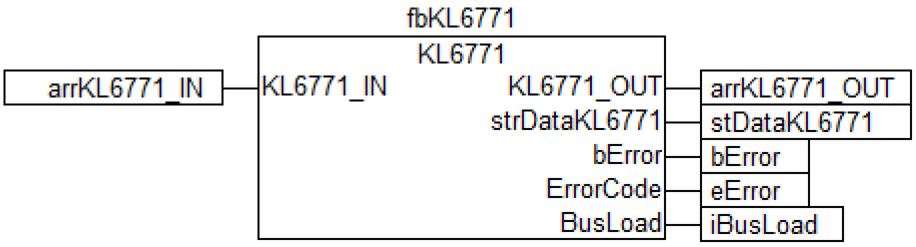
arrKL6771_IN: input variable for the MP-Bus terminal.

arrKL6771_OUT: output variable for the MP-Bus terminal.

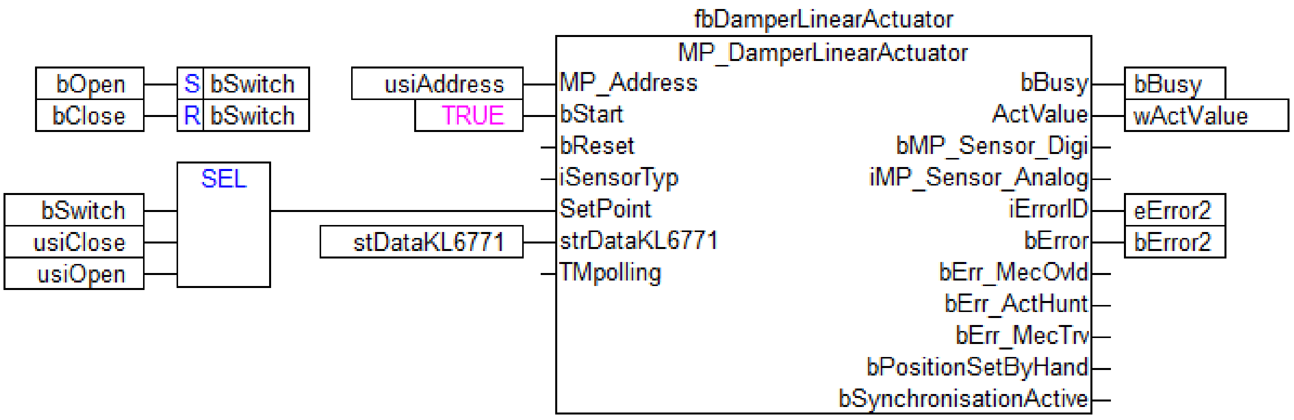
stDataKL6771: required for communication with MP-Bus (see DataKL6771 [▸ 84]).

All function blocks with MP-Bus must be called in the same task.

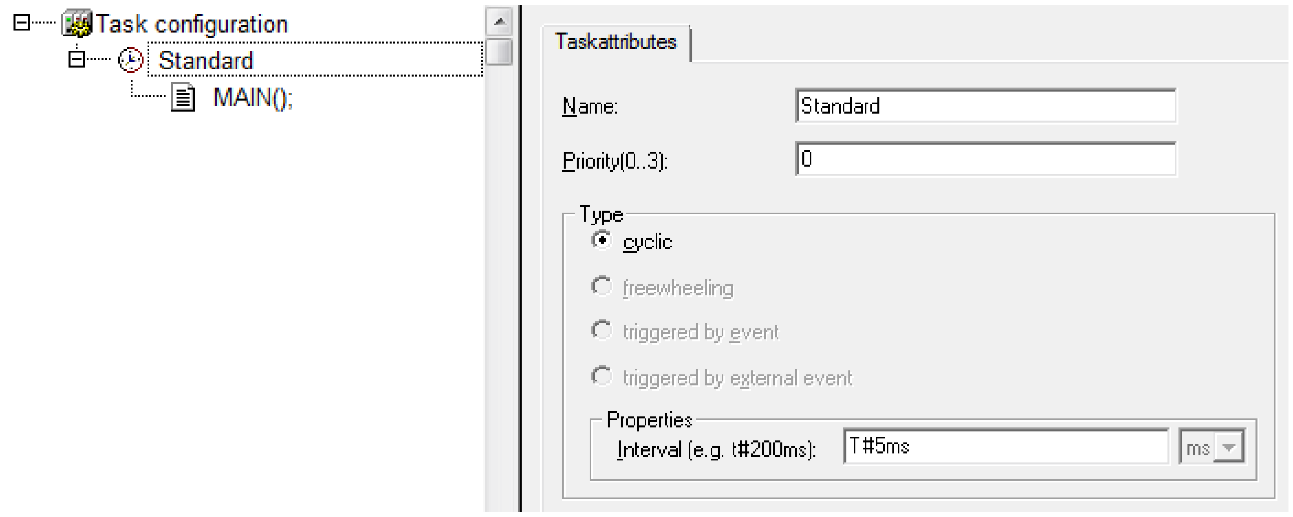
Therefore, create a MAIN program (CFC) in which the KL6771() [▸ 26] and MP_DamperLinearActuator() [▸ 30] function blocks are called. Make sure to link the communication block with *arrKL6771_IN*, *arrKL6771_OUT* and *stDataKL6771*.



The input *SetPoint* is set depending on the selected function. Link the global variables *bOpen* and *bClose* with an auxiliary variable.



Go to the task configuration and give the task a lower interval time. More detailed information can be found in the KL6771() [▸ 26] function block description.



Name: Standard

Priority(0..3): 0

Type
[✓ cyclic]
freewheeling
triggered by event
triggered by external event

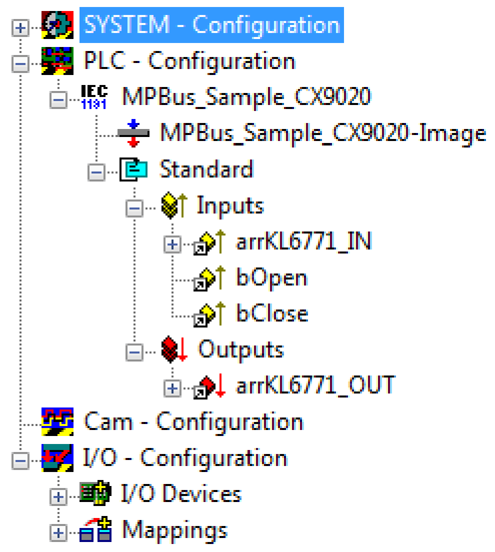
Properties
Interval (e.g. t#200ms): T#5ms ms

Load the project to the CX as the boot project and save it.

Configuration in the System Manager

Create a new TwinCAT System Manager project, select the CX as the target system, and search for the associated hardware.

Add the PLC program created above under PLC configuration.



Now link the global variables of the PLC program with the Bus Terminal inputs and outputs, create the allocations, and activate the configuration. Then start the device in run mode.

Your CX is now ready for use.

After pressing the button the flap drive can be moved. The current position is stored in variable *wActValue*.

5.3 Integration into TwinCAT (BC9191)

This example describes how a simple PLC program for MP-Bus can be written in TwinCAT and how it is linked with the hardware. The task is to control an individual flap drive and change it with a button.

https://infosys.beckhoff.com/content/1033/tcplclibmpbus/Resources/12063687051/.zip https://infosys.beckhoff.com/content/1033/tcplclibmpbus/Resources/12063687051/.zip

Hardware

Setting up the components

The following hardware is required:

- 1x Bus Terminal Controller BC9191
- 1x potential feed terminal 24V DC
- 1x digital 2-channel input terminal KL1002 (for the Open and Close functions)
- 1x MP-Bus master terminal KL6771
- 1x end terminal KL9010

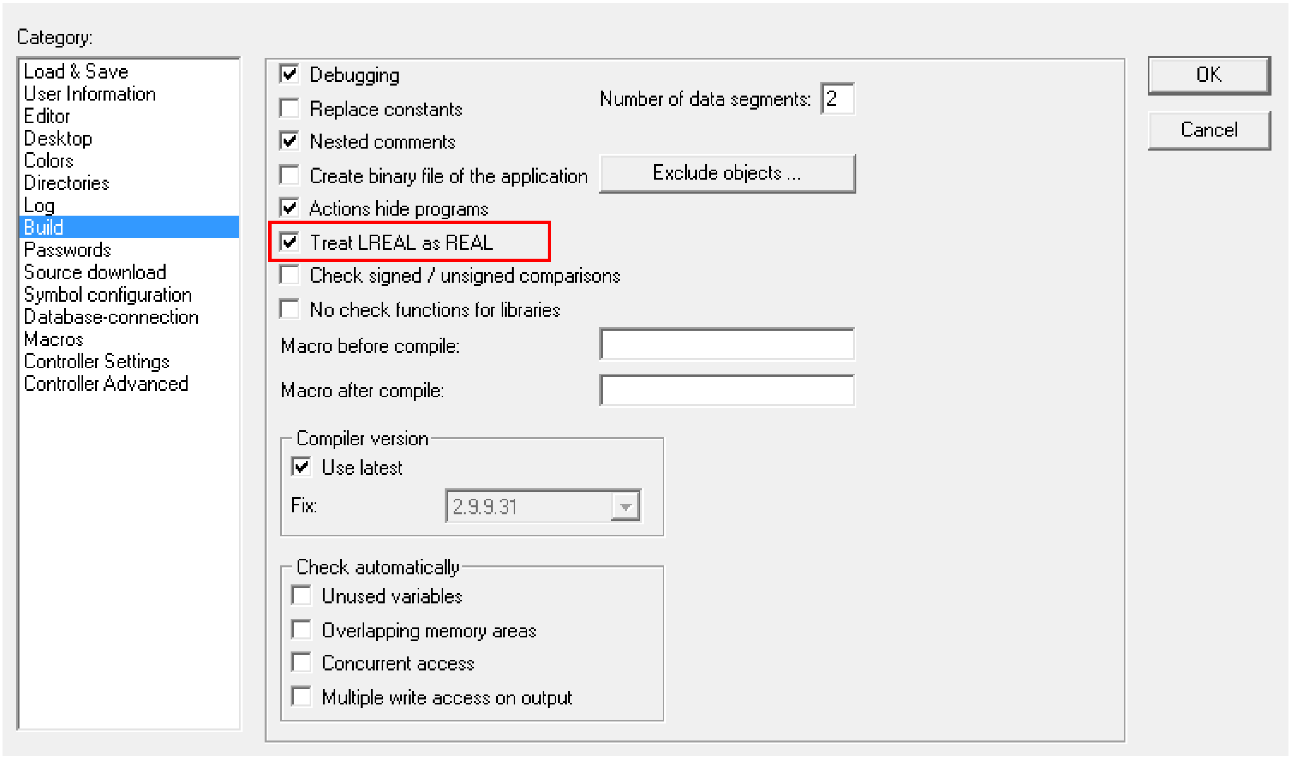
Set up the hardware and the MP-Bus components as described in the associated documentation.

This example assumes that the address of the flap drive is known.

Software

Creation of the PLC program

Create a new PLC project for BC-based systems (BCxx50 via AMS) and add the libraries *TcMPBus.lbx* and *TcSystemBCxx50.lbx*. Then navigate to *Project* → *Options...* → *Build* and select *Treat LREAL as REAL*.


Category: Build

[✓] Debugging
[ ] Replace constants
[✓] Nested comments
[ ] Create binary file of the application
[✓] Actions hide programs
[✓] Treat LREAL as REAL
[ ] Check signed / unsigned comparisons
[ ] No check functions for libraries

Number of data segments: 2

Macro before compile:
Macro after compile:

Compiler version
[✓] Use latest
Fix: 2.9.9.31

Check automatically
[ ] Unused variables
[ ] Overlapping memory areas
[ ] Concurrent access
[ ] Multiple write access on output

Next, generate the following global variables:

```
VAR_GLOBAL
    bOpen       AT %I*      : BOOL;
    bClose      AT %I*      : BOOL;
    arrKL6771_IN    AT %I* : ARRAY[0..11] OF BYTE;
    arrKL6771_OUT   AT %Q* : ARRAY[0..11] OF BYTE;
    stDataKL6771           : DataKL6771;
END_VAR
```

bOpen: input variable for the Open button.

bClose: input variable for the Close button.

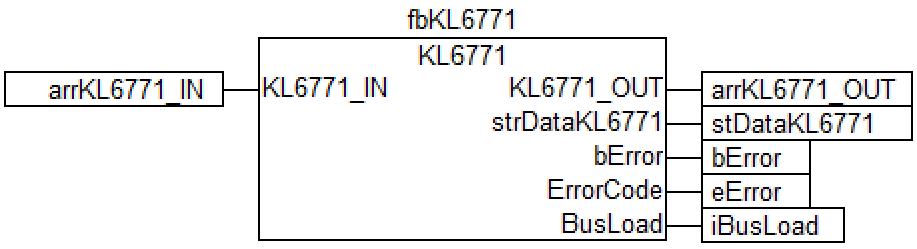
arrKL6771_IN: input variable for the MP-Bus terminal.

arrKL6771_OUT: output variable for the MP-Bus terminal.

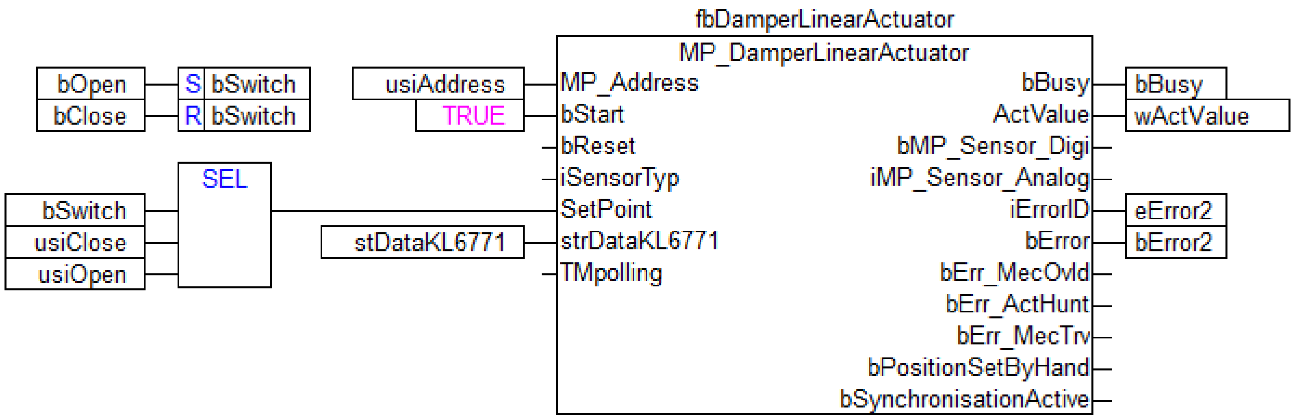
stDataKL6771: required for communication with MP-Bus (see DataKL6771 [▸ 84]).

All function blocks with MP-Bus must be executed in one task.

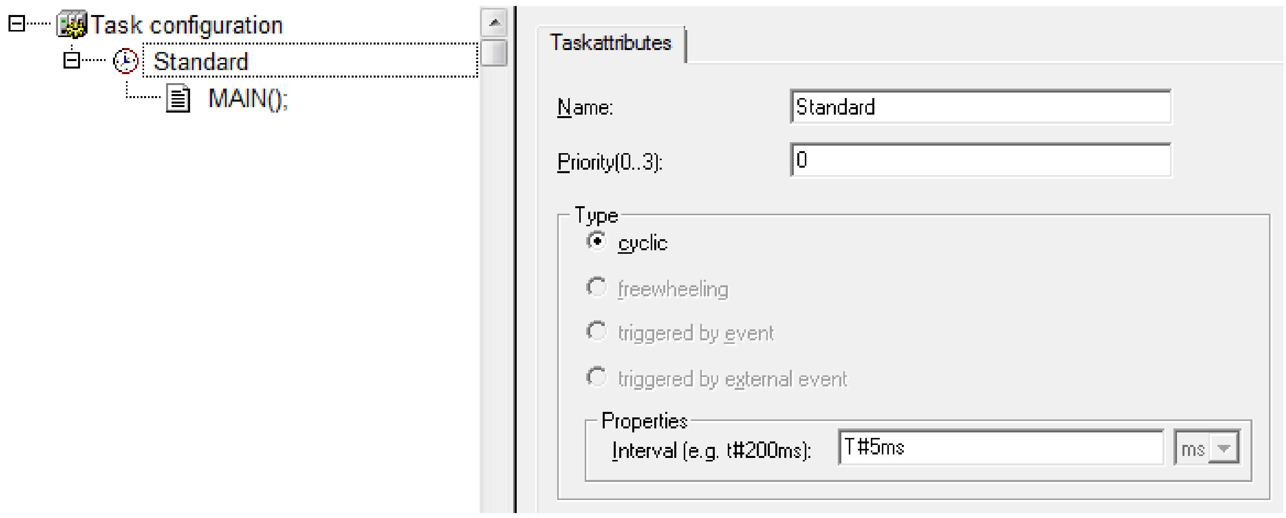
Therefore, create a MAIN program (CFC) in which the KL6771() [▸ 26] and MP_DamperLinearActuator() [▸ 30] function blocks are called. Make sure to link the communication block with *arrKL6771_IN*, *arrKL6771_OUT* and *stDataKL6771*.



The input *SetPoint* is set depending on the selected function. Link the global variables *bOpen* and *bClose* with an auxiliary variable.



Go to the task configuration and give the task a lower interval time. More detailed information can be found in the KL6771() [▸ 26] function block description.


Task configuration
Standard
MAIN();

Taskattributes

Name: Standard

Priority(0..3): 0

Type
[✓ cyclic]
freewheeling
triggered by event
triggered by external event

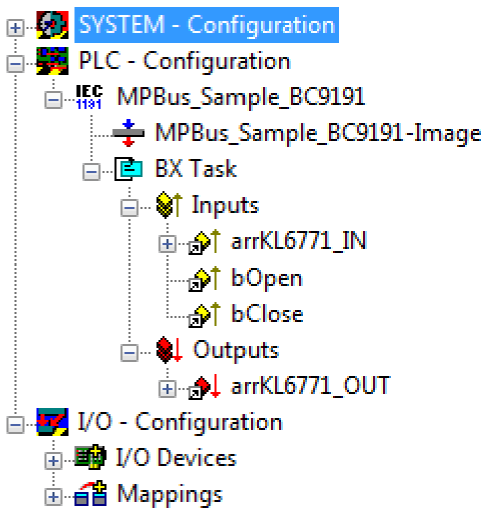
Properties
Interval (e.g. t#200ms): T#5ms ms

Now load the project as a boot project to the BC and save it.

Configuration in the System Manager

Create a new TwinCAT System Manager project, select the BC as the target system, and search for the associated hardware.

Add the PLC program created above under PLC configuration.


```
SYSTEM - Configuration
PLC - Configuration
    MPBus_Sample_BC9191
        MPBus_Sample_BC9191-Image
        BX Task
            Inputs
                arrKL6771_IN
                bOpen
                bClose
            Outputs
                arrKL6771_OUT
I/O - Configuration
    I/O Devices
    Mappings
```

Now link the global variables of the PLC program with the Bus Terminal inputs and outputs, create the allocations, and activate the configuration. Then start the device in run mode.

Your BC is now ready for use.

After pressing the button the flap drive can be moved. The current position is stored in variable *wActValue*.

6 Programming

Content
General Information [▶ 24]
Linking to the TwinCAT System Manager [▶ 14]

POUs

POUs	Description
KL6771 [▶ 26]	Communication with the MP-Bus master terminal KL6771

POUs/Function blocks

Function blocks	Description
KL6771 [▶ 26]	Communication with a KL6771 MP-Bus master terminal.
MP_Addressing [▶ 26]	This function block allows an MP-Bus slave to be addressed.
MP_CMV [▶ 28]	This function block is used to control and monitor a volume flow controller.
MP_DamperLinearActuator [▶ 30]	This function block is used to control and monitor an actuator of a damper and of a globe valve.
MP_EnergyValveV4_Configuration [▶ 31]	This function block is used to configure the Energy Valve actuators EV..R2+... (V4).
MP_EnergyValveV4_Process [▶ 34]	This function block is suitable for Energy Valve actuators EV..R2+... (V4).
MP_EPIV [▶ 36]	This function block is used to control a pressure-independent control ball valve.
MP_EPIV_R6 [▶ 38]	This function block is used to control a control ball valve of series EP..R-R6+BAC.
MP_EPIV_R6_Parameter [▶ 39]	This function block is used to parameterize actuators of series EP..R-R6+BAC.
MP_EV [▶ 41]	This function block is used to control a control ball valve of series P6...W..EV-BAC.
MP_EV_Parameter [▶ 43]	This function block is used to parameterize actuators.
MP_MPX [▶ 44]	For BELIMO room sensor MS24A-R..-MPX.
MP_OperatingUnit [▶ 45]	This function block is used to read room sensors of types P-22Rxx-1900x-1.
MP_OperatingUnit_ConfigurationCO2 [▶ 47]	This function block is used to configure room sensors of types P-22Rxx-1900x-1.
MP_OperatingUnit_ConfigurationDisplay [▶ 48]	This function block is used to configure room sensors of types P-22Rxx-1900x-1.
MP_OperatingUnit_ConfigurationOffsetValues [▶ 49]	This function block is used to configure room sensors of types P-22Rxx-1900x-1.
MP_OperatingUnit_ConfigurationStatusIcons [▶ 50]	This function block is used to configure room sensors of types P-22Rxx-1900x-1.
MP_OperatingUnit_ConfigurationTemp [▶ 51]	This function block is used to configure room sensors of types P-22Rxx-1900x-1.
MP_OperatingUnit_ConfigurationVentilation [▶ 52]	This function block is used to configure room sensors of types P-22Rxx-1900x-1.
MP_PTH [▶ 53]	This function block is used to control and monitor a PTH sensor.

Function blocks	Description
MP_RoomSensor [▸ 54]	This function block is used to read room sensors.
MP_RoomSensor_Parameter [▸ 55]	This function block is used for the parameterization of room sensors.
MP_Smoker [▸ 57]	This function block is used to control and monitor a fire damper.
MP_TEM_Configuration [▸ 58]	This function block is used to configure the Thermal Energy Meter type 22PE-.. and 22PEM-.. .
MP_TEM_Process [▸ 59]	This function block is suitable for the thermal energy meter type 22PE-.. and 22PEM-... .
MP_UST_3 [▸ 61]	This function block is used to control and monitor a multi-I/O module UST3.
MP_VAV [▸ 64]	This function block is used to control and monitor a volume flow controller.
MP_VRU_Configuration [▸ 66]	This function block is used to configure the VAV actuators VRU-D3-BAC, VRU-M1-BAC and VRU-M1R-BAC.
MP_VRU_Process [▸ 67]	This function block is suitable for VAV actuators VRU-D3-BAC, VRU-M1-BAC and VRU-M1R-BAC.
MP_Window [▸ 69]	This function block is used to control and monitor a window ventilation system (FLS).
MPL_DamperLinearActuator [▸ 70]	This function block is used to control and monitor an actuator of a damper and of a globe valve.

Functions

POUs	Description
NI1000_LuS_TO_INT [▸ 71]	This function calculates a temperature from the value of an NI1000 L&S resistor.
NI1000_TO_INT [▸ 71]	This function calculates a temperature from the value of an NI1000 resistor.
NTC_TO_INT [▸ 72]	This function calculates a temperature from the value of an NTC resistor.
PT1000_TO_INT [▸ 72]	This function calculates a temperature from the value of an PT1000 resistor.

POUs/Slave addressing

POUs/Slave addressing

POUs	Description
MP_Addressing [▸ 26]	This function block allows an MP-Bus slave to be addressed.

Enums

Data types	Description
Data_Window [▸ 73]	Ventilation type.
E_MP_AirQualityStatus [▸ 73]	Status of the measured air quality.
E_MP_DisplayBackground [▸ 74]	Background color of the display.
E_MP_DisplayModeButton [▸ 74]	Display mode of the buttons.
E_MP_DisplayModeHeatingCooling [▸ 74]	Display mode of heating or cooling icons.
E_MP_DisplayModeIconWarning [▸ 74]	Display mode of the warning icon.
E_MP_DisplayModeIconWindow [▸ 75]	Display mode of the window icon.

Data types	Description
E_MP_DisplayModeTemp [▸ 75]	Display mode temperature.
E_MP_DisplayModeTempUnit [▸ 75]	Display mode of the unit for the temperature.
E_MP_DisplayModeVentilationStage [▸ 75]	Number of adjustable ventilation stages shown on the display.
E_MP_DisplayVisibility [▸ 76]	Enabled status.
E_MP_EnabledStatus [▸ 76]	Scaling.
E_MP_EP_R_R6_UnitSel [▸ 76]	Scaling.
E_MP_EV_V4_Command [▸ 76]	Command for service and test functions of the actuator.
E_MP_EV_V4_ControlMode [▸ 77]	Control mode.
E_MP_EV_V4_DeltaTLimitation [▸ 77]	Response to a low delta T.
E_MP_EV_V4_DeltaTManagerStatus [▸ 77]	Status from Delta T Manager.
E_MP_EV_V4_OverrideControl [▸ 77]	Setpoint override.
E_MP_EV_V4_Sensor1Type [▸ 78]	External sensor at input S1.
E_MP_SystemOperationMode [▸ 78]	Operation mode of the system.
E_MP_VRU_Application [▸ 78]	Visualization of the application selected by the manufacturer.
E_MP_VRU_Command [▸ 79]	Commands for service and test functions of the actuator.
E_MP_VRU_OverrideControl [▸ 79]	Setpoint override.
E_MP_VRU_RoomPressureCascade [▸ 79]	Room pressure cascade control.
E_MP_VRU_Sensor1Type [▸ 80]	External sensor at input S1.
E_MPBus_ControlMode [▸ 80]	Control mode.
E_MPBus_DeltaTLimitation [▸ 80]	Delta T (dT) limitation.
E_MPBus_Override [▸ 81]	Override mode.
E_MPBus_Override_6wayMPIV [▸ 81]	Override control mode.
MP_Error [▸ 81]	Error messages.
UST3_Ex [▸ 83]	Voltage scaling.
UST3_R_set [▸ 83]	Resistance scaling.

Data types/Structures

Data types	Description
DataKL6771 [▸ 84]	Links the send and receive function blocks with the function block *KL6771*.
MP_BUS_MPX_ERROR [▸ 84]	Error messages of the "MPX" sensors.
MP_Serial_Number [▸ 84]	Serial number of the device.
St_MP_EV_V4_MalfunctionServiceInfo [▸ 85]	Fault and service information.
St_MP_VRU_ServiceInfo [▸ 85]	Fault and service information.
St_StateEV [▸ 86]	Information on the state of the EV.
UST3_SET [▸ 86]	Data structure for setting the scaling and the resistance measurement.

6.1 General Information

Installation

Beginning with TwinCAT 2.11 Build 2229 (R3 and x64 Engineering), the libraries "TcMPBus.lib/.lb6/.lbx" will be installed automatically.

Name of the library

This library replaces the "TcKL6771.lib/.lb6./.lbx". Only the name of the libraries has changed. The modules are still compatible.

Hardware documentation in Beckhoff Information System: KL6771 - MP-Bus master terminal

Further libraries are required

For PC systems (x86) and Embedded-PCs (CXxxxx):

- Standard.lib
- TcBase.lib
- TcSystem.lib

For Bus Terminal Controller of BCxx00 series:

- Standard.lb6
- PlcHelperBC.lb6

For Bus Terminal Controller of BCxx50, BCxx20 and BC9191 series:

- Standard.lbx
- TcBaseBCxx50.lbx
- TcSystemBCxx50.lbx

For Bus Terminal Controller of BXxx00 series:

- Standard.lbx
- TcBaseBX.lbx
- TcSystemBX.lbx

Memory usage

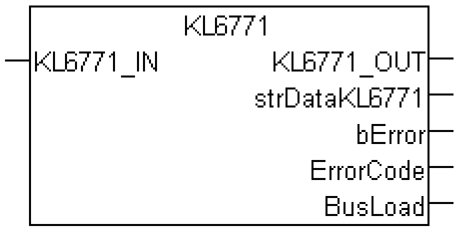
By linking the library PLC program memory is already consumed. Depending on the application program the remaining memory can not be sufficient.

6.2 Function blocks

Function blocks	Description
KL6771 [▸ 26]	Communication with a KL6771 MP-Bus master terminal.
MP_Addressing [▸ 26]	This function block allows an MP-Bus slave to be addressed.
MP_CMV [▸ 28]	This function block is used to control and monitor a volume flow controller.
MP_DamperLinearActuator [▸ 30]	This function block is used to control and monitor an actuator of a damper and of a globe valve.
MP_EnergyValveV4_Configuration [▸ 31]	This function block is used to configure the Energy Valve actuators EV..R2+... (V4).
MP_EnergyValveV4_Process [▸ 34]	This function block is suitable for Energy Valve actuators EV..R2+... (V4).
MP_EPIV [▸ 36]	This function block is used to control a pressure-independent control ball valve.

Function blocks	Description
MP_EPIV_R6 [▸ 38]	This function block is used to control a control ball valve of series EP..R-R6+BAC.
MP_EPIV_R6_Parameter [▸ 39]	This function block is used to parameterize actuators of series EP..R-R6+BAC.
MP_EV [▸ 41]	This function block is used to control a control ball valve of series P6...W..EV-BAC.
MP_EV_Parameter [▸ 43]	This function block is used to parameterize actuators.
MP_MPX [▸ 44]	For BELIMO room sensor MS24A-R..-MPX.
MP_OperatingUnit [▸ 45]	This function block is used to read room sensors of types P-22Rxx-1900x-1.
MP_OperatingUnit_ConfigurationCO2 [▸ 47]	This function block is used to configure room sensors of types P-22Rxx-1900x-1.
MP_OperatingUnit_ConfigurationDisplay [▸ 48]	This function block is used to configure room sensors of types P-22Rxx-1900x-1.
MP_OperatingUnit_ConfigurationOffsetValues [▸ 49]	This function block is used to configure room sensors of types P-22Rxx-1900x-1.
MP_OperatingUnit_ConfigurationStatusIcons [▸ 50]	This function block is used to configure room sensors of types P-22Rxx-1900x-1.
MP_OperatingUnit_ConfigurationTemp [▸ 51]	This function block is used to configure room sensors of types P-22Rxx-1900x-1.
MP_OperatingUnit_ConfigurationVentilation [▸ 52]	This function block is used to configure room sensors of types P-22Rxx-1900x-1.
MP_PTH [▸ 53]	This function block is used to control and monitor a PTH sensor.
MP_RoomSensor [▸ 54]	This function block is used to read room sensors.
MP_RoomSensor_Parameter [▸ 55]	This function block is used for the parameterization of room sensors.
MP_Smoker [▸ 57]	This function block is used to control and monitor a fire damper.
MP_TEM_Configuration [▸ 58]	This function block is used to configure the Thermal Energy Meter type 22PE-.. and 22PEM-.. .
MP_TEM_Process [▸ 59]	This function block is suitable for the thermal energy meter type 22PE-.. and 22PEM-... .
MP_UST_3 [▸ 61]	This function block is used to control and monitor a multi-I/O module UST3.
MP_VAV [▸ 64]	This function block is used to control and monitor a volume flow controller.
MP_VRU_Configuration [▸ 66]	This function block is used to configure the VAV actuators VRU-D3-BAC, VRU-M1-BAC and VRU-M1R-BAC.
MP_VRU_Process [▸ 67]	This function block is suitable for VAV actuators VRU-D3-BAC, VRU-M1-BAC and VRU-M1R-BAC.
MP_Window [▸ 69]	This function block is used to control and monitor a window ventilation system (FLS).
MPL_DamperLinearActuator [▸ 70]	This function block is used to control and monitor an actuator of a damper and of a globe valve.

6.2.1 KL6771



This function block takes care of communication with the KL6771 MP-Bus master terminal. The KL6771 is configured with this function block, and data exchange with the MP network is started.

Restrictions:

- This function block must only be called once for each KL6771!
- It must be called in the same task as the actuator function blocks!
- In the PLC project this function block may only be called once per cycle!

VAR_INPUT

```
KL6771_IN       : ARRAY [0..11] OF BYTE;
```

KL6771_IN: Is linked with the input process image of the MP-Bus master terminal.

VAR_OUTPUT

```
KL6771_OUT      : ARRAY [0..11] OF BYTE;
strDataKL6771   : DataKL6771;
bError          : BOOL;
ErrorCode       : MP_Error;
BusLoad         : INT := -1;
```

KL6771_OUT: is linked with the MP-Bus master terminal's output process image.

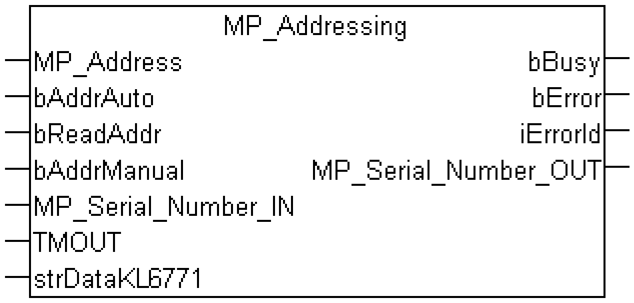
strDataKL6771: data structure, which is linked with the different MP-Bus function blocks and includes the communication with the KL6771 function block (see DataKL6771 [▸ 84]).

bError: the output becomes TRUE as soon as an error occurs. This error is described via the *ErrorCode* variable.

ErrorCode: this output outputs an error code in the event of an error (see MP_ERROR [▸ 81]). *bError* goes TRUE at the same time.

BusLoad: MP-Bus load in percent.

6.2.2 MP_Addressing



This function block allows an MP-Bus slave to be addressed. It is also possible to use this function block to read the serial number of the slave.
The slave address that the slave is to receive when addressed, or of the bus device whose serial number is to be read, is passed in **MP_Address**. The serial number is read out on a rising edge of **bReadAddr**. It is output through **MP_Serial_Numer_Out**. A rising edge of **bAddrAuto** addresses a slave whose address is

MP_Address. The block waits for the time set in **TMOUT** for the transmission of the slave with its serial number. Transmission of the serial number is initiated differently from one slave to another. Please see the documentation for the MP-Bus device for how the serial number can be sent (in most cases there is a switch on the drive which will trigger this action when pressed). No telegrams are sent to the slaves during the time specified by **TMOUT**.

A rising edge at **bAddrManual** initiates manual addressing. This requires the serial number of the drive to be stored in **MP_Serial_Number_In**. The serial number of the slave can be found on an adhesive label on the drive.

Example: 00234-00016-002-031 The following must be entered in the **MP_Serial_Numer_IN** variable:

YearAndWeek = 234

DayAndNumber = 16

DeviceFamily = 2

TestStation = 31

FamilySuffix = is not evaluated, and should therefore be ignored.

bBusy is set for as long as the block is active. An error is indicated through **bError**, while the error number can be read with **iErrorId**.

VAR_INPUT

```
MP_Address              : USINT := 1;
bAddrAuto               : BOOL;
bReadAddr               : BOOL;
bAddrManual             : BOOL;
MP_Serial_Number_IN     : MP_Serial_Number;
TMOUT                   : TIME := t#30s;
strDataKL6771           : DataKL6771;
```

MP_Address: the MP address that is to be used for the addressing or for reading the serial number. Valid values (1...8).

bAddrAuto: a positive edge starts the function block. The function block halts other MP-Bus communication, and waits until the time set through *TMOUT* has elapsed for an MP-Bus slave has transmitted its serial number, for instance in response to pressing a switch. The MP-Bus address that has been configured in the *MP_Address* variable is then transmitted to the slave.

bReadAddr: a positive edge starts the function block. The function block reads the serial number of the MP-Bus slave whose address is *MP_Address*.

bAddrManual: a positive edge starts the function block. The function block addresses the slave that has serial number *MP_Serial_Number_IN*. The slave's addresses configured with *MP_Address*.

MP_Serial_Number_IN: serial number of the device (see MP_Serial_Number [▶ 84]).

TMOUT: time after which automatic addressing is canceled.

strDataKL6771: the data structure with which the KL6771 [▶ 26] function block must be linked (see DataKL6771 [▶ 84]).

VAR_OUTPUT

```
bBusy                   : BOOL;
bError                  : BOOL;
iErrorId                : MP_Error;
MP_Serial_Number_OUT    : MP_Serial_Number;
```

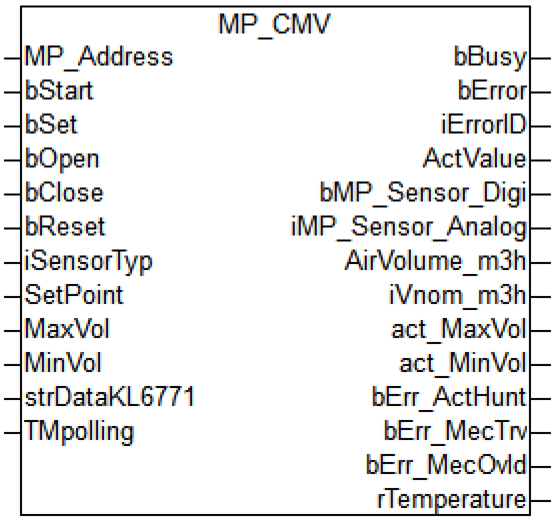
bBusy: This bit is set for as long as the block is active.

bError: The output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorId*.

iErrorId: The output issues an error code when an error occurs (see MP_ERROR [▶ 81]). Simultaneously *bError* is TRUE.

MP_Serial_Number_OUT: The serial number of the slave that is addressed or interrogated (see MP_Serial_Number [▶ 84]).

6.2.3 MP_CMV



This function block is used to control and monitor a volume flow controller.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. If **bStart** remains TRUE, the device is addressed cyclically with a period specified by the time in **TMPolling**. The time should be set longer than 1 s. **bError** is used to indicate an error in communication with the actuator. The type of the error can be read with **iErrorId**.

VAR_INPUT

```
MP_Address      : USINT := 1;
bStart          : BOOL;
bSet            : BOOL;
bOpen           : BOOL;
bClose          : BOOL;
bReset          : BOOL;
iSensorTyp      : INT;
SetPoint        : USINT;
MaxVol          : WORD;
MinVol          : WORD;
strDataKL6771   : DataKL6771;
TMpolling       : TIME:= t#10S;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block. If this remains continuously TRUE, the function block will be activated cyclically with a period specified by the time in *TMPolling*.

bSet: a positive edge writes the *MaxVol* and *MinVol* data to the actuator.

bOpen: a positive edge opens the dampers of the actuator, while a negative edge cancels the forced ventilation.

bClose: a positive edge closes the dampers of the actuator, while a negative edge cancels the forced closure.

bReset: a positive edge resets the actuator's error messages.

iSensorTyp: 0: no sensor connected, 1: digital sensor connected, 2: analog sensor connected (0...35 V), 3...6: output of a resistance in ohms (3..5 applies to PT1000, NI1000 and NI1000LuS; 6 applies to NTC). To convert to a temperature, use the corresponding conversion functions.

SetPoint: set volume flow rate (0...100 %).

MaxVol: maximum volume flow (30...100 %).

MinVol: minimum volume flow (0...100 %).

strDataKL6771: the data structure with which the KL6771 [▸ 26] function block must be linked (see DataKL6771 [▸ 84]).

TMpolling: the time for which the function block should address the actuator. Default 10 s, minimum time 1 s.

VAR_OUTPUT

```
bBusy                : BOOL;
bError               : BOOL;
iErrorId             : MP_Error;
ActValue             : WORD;
bMP_Sensor_Digi      : BOOL;
iMP_Sensor_Analog    : INT;
AirVolume_m3h        : WORD;
iVnom_m3h            : INT;
act_MaxVol           : INT;
act_MinVol           : INT;
bErr_ActHunt         : BOOL;
bErr_MecTrv          : BOOL;
bErr_MecOvld         : BOOL;
rTemperature         : LREAL;
```

bBusy: this bit is set for as long as the function block is active.

bError: the output becomes TRUE as soon as an error occurs. This error is described via the *iErrorId* variable.

iErrorId: this output outputs an error code in the event of an error (see MP_Error [▸ 81]). *bError* goes TRUE at the same time.

ActValue: contains the current position of the actuator (0...100 %).

bMP_Sensor_Digi: if an analog sensor is connected, its value is indicated through this variable. *iSensorTyp* must be "1".

iMP_Sensor_Analog: if an analog sensor is connected, its value is indicated through this variable. *iSensorTyp* must be in the range "2...6".

AirVolume_m3h: output of the volume flow rate in m^3/h.

iVnom_m3h: nominal air volume flow in m^3/h. This output is available from version 1.12.0. VAV is read and must be > 0. If 0, then the calculation of *AirVolume_m3h* is not correct.

act_MaxVol: maximum set volume flow rate in %.

act_MinVol: minimum set volume flow rate in %.

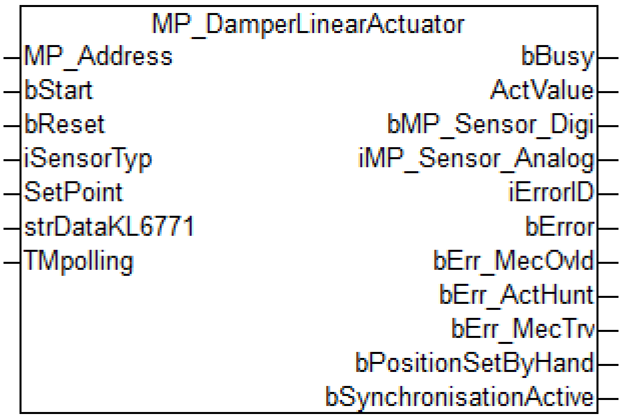
bErr_ActHunt: actuator error, "Regulating oscillation"; the actuator is swinging backwards and forwards.

bErr_MecTrv: actuator error, "Positioning angle exceeded"; the actuator has passed more than 10° beyond the adaptation position.

bErr_MecOvld: actuator error, "Overload"; the set position could not be reached.

rTemperature: temperature in the channel in °C.

6.2.4 MP_DamperLinearActuator



This function block is used to control and monitor an actuator of a damper and of a globe valve.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. If **bStart** remains TRUE, the device is addressed cyclically with a period specified by the time in **TMPolling**. The time should be set longer than 1 s. **bError** is used to indicate an error in communication with the actuator. The type of the error can be read with **iErrorID**.

A positive edge at **bReset** clears any pending error messages from the actuator. This resets errors that affect the output variables **bErr_MecOcld**, **bErr_ActHunt** and **bErrMecTrv**. If the error itself is still present, the actuator will set these error bits again.

SetPoint is used to adjust the position of the damper from 0...100 %. The current position of the actuator can be read through **ActValue**.

If a sensor is connected to the actuator, its type is to be indicated in **iSensorTyp**. If no sensor is connected, the value "0" should be entered, or the variable left open. A digital sensor should be parameterized with "1". The state of the sensor can be queried through **bMP_Sensor_Digi**. Analog sensors "2...6" are output in variable **iMP_Sensor_Analog**.

VAR_INPUT

```
MP_Address      : USINT := 1;
bStart          : BOOL;
bReset          : BOOL;
iSensorTyp      : INT;
SetPoint        : USINT;
strDataKL6771   : DataKL6771;
TMpolling       : TIME := t#10s;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block. If this remains continuously TRUE, the function block will be activated cyclically with a period specified by the time in *TMPolling*.

bReset: a positive edge resets the actuator's error messages.

iSensorTyp: 0: no sensor connected, 1: digital sensor connected, 2: analog sensor connected (0...35 V), 3...6: output of a resistance in ohms (3...5 applies to PT1000, NI1000 and NI1000LuS; 6 applies to NTC). To convert to a temperature, use the corresponding conversion functions.

SetPoint: 0...100 % the set damper position specified for the actuator.

strDataKL6771: the data structure with which the KL6771 [▶ 26] function block must be linked (see DataKL6771 [▶ 84]).

TMpolling: the time for which the function block should address the actuator. Default 10 s, minimum time 1 s.

VAR_OUTPUT

```
bBusy                   : BOOL;
ActValue                : WORD;
bMP_Sensor_Digi         : BOOL;
iMP_Sensor_Analog       : INT;
iErrorID                : MP_Error;
bError                  : BOOL;
bErr_MecOvld            : BOOL;
bErr_ActHunt            : BOOL;
bErr_MecTrv             : BOOL;
bPositionSetByHand      : BOOL;
bSynchronisationActive : BOOL;
```

bBusy: this bit is set for as long as the function block is active.

ActValue: contains the current position of the actuator (0...100 %).

bMP_Sensor_Digi: if an analog sensor is connected, its value is indicated through this variable. *iSensorTyp* must be "1".

iMP_Sensor_Analog: if an analog sensor is connected, its value is indicated through this variable. *iSensorTyp* must be in the range "2...6".

iErrorID: this output outputs an error code in the event of an error (see MP_Error [▸ 81]). *bError* goes TRUE at the same time.

bError: the output becomes TRUE as soon as an error occurs. This error is described via the *iErrorID* variable.

bErr_MecOvld: actuator error, "Overload"; the set position could not be reached.

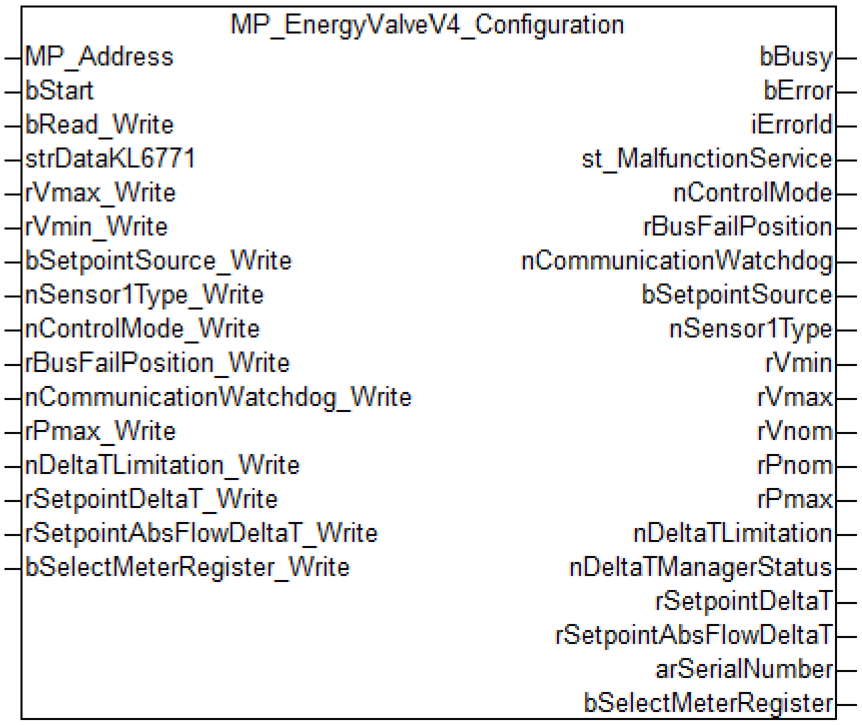
bErr_ActHunt: actuator error, "Regulating oscillation"; the actuator is swinging backwards and forwards.

bErr_MecTrv: actuator error, "Positioning angle exceeded"; the actuator has passed more than 10° beyond the adaptation position.

bPositionSetByHand: the position of the actuator was changed by hand.

bSynchronizationActive: the actuator synchronizes.

6.2.5 MP_EnergyValveV4_Configuration



This function block is used to configure the Energy Valve actuators EV..R2+... (V4). For more information please visit www.belimo.ch.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. **bError** is used to indicate an error in communication with the actuator. The type of the error can be read with **iErrorId**.

VAR_INPUT

```
MP_Address                    : USINT := 1;
bStart                        : BOOL;
bRead_Write                   : BOOL;
strDataKL6771                 : DataKL6771;
rVmax_Write                   : LREAL := 100;
rVmin_Write                   : LREAL;
bSetpointSource_Write         : BOOL := TRUE;
nSensor1Type_Write            : E_MP_EV_V4_Sensor1Type;
nControlMode_Write            : E_MP_EV_V4_ControlMode := 1;
rBusFailPosition_Write        : LREAL;
nCommunicationWatchdog_Write  : UINT;
rPmax_Write                   : LREAL := 100;
nDeltaTLimitation_Write       : E_MP_EV_V4_DeltaTLimitation;
rSetpointDeltaT_Write         : LREAL := 1.0;
rSetpointAbsFlowDeltaT_Write  : LREAL;
bSelectMeterRegister_Write    : BOOL;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block.

bRead_Write: if FALSE then READ only; if TRUE then READ and WRITE.

strDataKL6771: the data structure with which the KL6771() [▸ 26] function block must be linked (see DataKL6771 [▸ 84]).

rVmax_Write: max setpoint in % (25...100). Standard 100 %. Refers to Vnom and is taken into account when control mode = flow control.

rVmin_Write: min setpoint in % (0...Vmax). Vmin must be smaller than Vmax.

bSetpointSource_Write: TRUE = bus; FALSE = analog.

nSensor1Type_Write: sensor 1 type (see E_MP_EV_V4_Sensor1Type [▸ 78]).

nControlMode_Write: control mode (see E_MP_EV_V4_ControlMode [▸ 77]).

rBusFailPosition_Write: bus fail position in % (0...100). Not functional (reserved for future extensions).

nCommunicationWatchdog_Write: communication watchdog in s (0...3600). Not functional (reserved for future extensions).

rPmax_Write: max setpoint in % (5...100). Standard 100 %. Refers to Pnom and is taken into account if control mode = power control.

nDeltaTLimitation_Write: determines whether the device responds to a low delta T (see E_MP_EV_V4_DeltaTLimitation [▸ 77]).

rSetpointDeltaT_Write: setpoint delta T in K (0...60). Standard 1.0K. Is taken into account if the delta T limitation is active (not disabled).

rSetpointAbsFlowDeltaT_Write: setpoint Abs flow delta T in l/s (0...100,000). Considered when Delta T limitation is set to "Delta T Manager scaled".

bSelectMeterRegister_Write: FALSE = certified meter; TRUE = lifetime meter.

VAR_OUTPUT

```
bBusy                         : BOOL;
bError                        : BOOL;
iErrorId                      : MP_Error;
st_MalfunctionService         : St_MP_EV_V4_MalfunctionServiceInfo;
nControlMode                  : E_MP_EV_V4_ControlMode;
```

```
rBusFailPosition              : LREAL;
nCommunicationWatchdog        : UINT;
bSetpointSource               : BOOL;
nSensor1Type                  : E_MP_EV_V4_Sensor1Type;
rVmin                         : LREAL;
rVmax                         : LREAL;
rVnom                         : LREAL;
rPnom                         : LREAL;
rPmax                         : LREAL;
nDeltaTLimitation             : E_MP_EV_V4_DeltaTLimitation;
nDeltaTManagerStatus          : E_MP_EV_V4_DeltaTManagerStatus;
rSetpointDeltaT               : LREAL;
rSetpointAbsFlowDeltaT        : LREAL;
arSerialNumber                : ARRAY[0..1] OF DWORD;
bSelectMeterRegister          : BOOL;
```

bBusy: this bit is set for as long as the function block is active.

bError: the output becomes TRUE as soon as an error occurs. This error is described via the *iErrorId* variable.

iErrorId: this output outputs an error code in the event of an error (see MP_ERROR [▶ 81]). *bError* goes TRUE at the same time.

st_MalfunctionService: malfunction and service information (see St_MP_EV_V4_MalfunctionServiceInfo [▶ 85]).

nControlMode: control mode (see E_MP_EV_V4_ControlMode [▶ 77]).

rBusFailPosition: bus failure position in % (0...100). Not functional (reserved for future extensions).

nCommunicationWatchdog: communication watchdog in sec (0...3600). Not functional (reserved for future extensions).

bSetpointSource: TRUE = bus; FALSE = analog.

nSensor1Type: sensor 1 type (see E_MP_EV_V4_Sensor1Type [▶ 78]).

rVmin: min setpoint in % (0...Vmax). Refers to Vnom and is taken into account when control mode = flow control.

rVmax: max setpoint in % (25...100). Refers to Vnom and is taken into account when control mode = flow control.

rVnom: nominal volume flow rate in l/s (0...100).

rPnom: nominal output in kW (0...21.5).

rPmax: max setpoint in % (5...100). Refers to Pnom and is taken into account if control mode = power control.

nDeltaTLimitation: specifies whether the device responds to a low delta T (see E_MP_EV_V4_DeltaTLimitation [▶ 77]).

nDeltaTManagerStatus: indicates the status of the DeltaT Manager (see E_MP_EV_V4_DeltaTManagerStatus [▶ 77]).

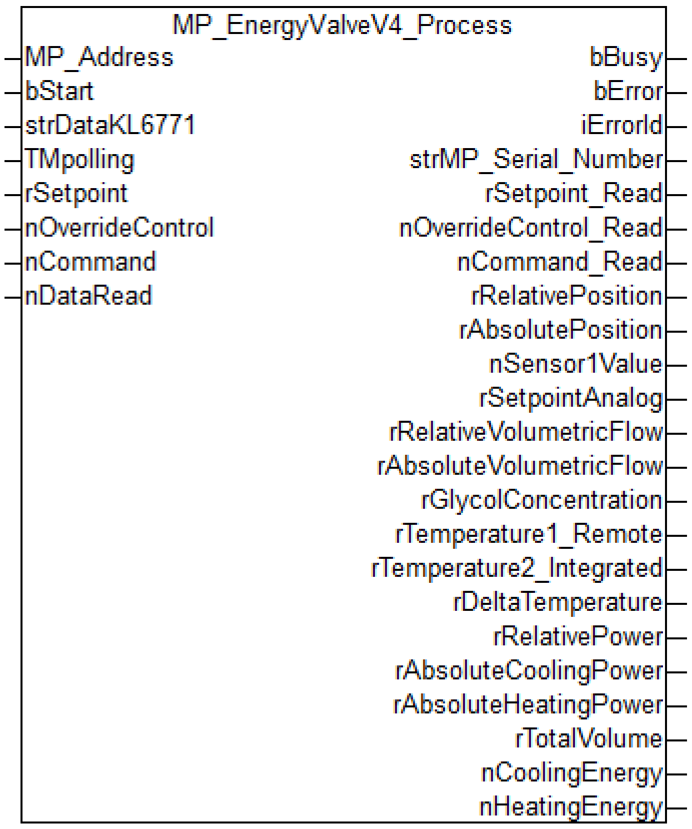
rSetpointDeltaT: setpoint DeltaT in K (0...60). Is taken into account if DeltaT limitation is active (not disabled).

rSetpointAbsFlowDeltaT: setpoint Abs flow DeltaT in l/s (0...100,000). Considered when DeltaT limitation is set to DeltaT Manager scaled.

arSerialNumber: serial number of the device.

bSelectMeterRegister: FALSE = certified meter; TRUE = lifetime meter.

6.2.6 MP_EnergyValveV4_Process



This function block is suitable for Energy Valve actuators EV..R2+... (V4). For more information please visit www.belimo.ch.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. If **bStart** remains TRUE, the device is addressed cyclically with a period specified by the time in **TMPolling**. The time should be set longer than 1 s. **bError** is used to indicate an error in communication with the actuator. The type of the error can be read with **iErrorId**.

VAR_INPUT

```
MP_Address                   : USINT := 1;
bStart                       : BOOL;
strDataKL6771                : DataKL6771;
TMpolling                    : TIME:= t#10s;
rSetpoint                    : LREAL;
nOverrideControl             : E_MP_EV_V4_OverrideControl;
nCommand                     : E_MP_EV_V4_Command;
nDataRead                    : WORD := 16#FFFF;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block. If this remains continuously TRUE, the function block will be activated cyclically with a period specified by the time in *TMPolling*.

strDataKL6771: the data structure with which the KL6771 [▸ 26] function block must be linked (see DataKL6771 [▸ 84]).

TMpolling: the time for which the function block should address the actuator. Default 10 s, minimum time 1 s.

rSetpoint: setpoint in % (0...100).

nOverrideControl: override the setpoint with defined values (see E_MP_EV_V4_OverrideControl [▸ 77]).

nCommand: command for service and test functions of the actuator (see E_MP_EV_V4_Command [▸ 76]).

nDataRead: 0xFFFF - read all data; bit 0 - read relative position; bit 1 - read absolute position; bit 2 - read value sensor 1; bit 3 - read analog setpoint; bit 4 - read relative volume flow rate; bit 5 - read absolute volume flow rate; bit 6 - read glycol concentration; bit 7 - read temperature1 remote; bit 8 - read temperature2 integrated; bit 9 - read temperature delta; bit 10 - read relative power; bit 11 - read absolute cooling power; bit 12 - read absolute heating power; bit 13 - total volume; bit 14 - cooling energy; bit 15 - heating energy

VAR_OUTPUT

```
bBusy                      : BOOL;
bError                     : BOOL;
iErrorId                   : MP_Error;
strMP_Serial_Number        : MP_Serial_Number;
rSetpoint_Read             : LREAL;
nOverrideControl_Read      : E_MP_EV_V4_OverrideControl;
nCommand_Read              : E_MP_EV_V4_Command;
rRelativePosition          : LREAL;
rAbsolutePosition          : LREAL;
nSensor1Value              : DINT;
rSetpointAnalog            : LREAL;
rRelativeVolumetricFlow    : LREAL;
rAbsoluteVolumetricFlow    : LREAL;
rGlycolConcentration       : LREAL;
rTemperature1_Remote       : LREAL;
rTemperature2_Integrated   : LREAL;
rDeltaTemperature          : LREAL;
rRelativePower             : LREAL;
rAbsoluteCoolingPower      : LREAL;
rAbsoluteHeatingPower      : LREAL;
rTotalVolume               : LREAL;
nCoolingEnergy             : DINT;
nHeatingEnergy             : DINT;
```

bBusy: this bit is set for as long as the function block is active.

bError: the output becomes TRUE as soon as an error occurs. This error is described via the *iErrorId* variable.

iErrorId: this output outputs an error code in the event of an error (see MP_ERROR [▶ 81]). *bError* goes TRUE at the same time.

strMP_Serial_Number: structure for the serial number. (see MP_Serial_Number [▶ 84]).

rSetpoint_Read: setpoint in % (0...100).

nOverrideControl_Read: overridden setpoint (see E_MP_EV_V4_OverrideControl [▶ 77]).

nCommand_Read: command (see E_MP_EV_V4_Command [▶ 76]).

rRelativePosition: relative position in % (0...100). A value of -1 means that the data is disabled (see VAR_INPUT nDataRead.0 = TRUE).

rAbsolutePosition: absolute position in ° (0...96). A value of -1 means that the data is disabled (see VAR_INPUT nDataRead.1 = TRUE).

nSensor1Value: sensor 1 value in mV/Ohm (0...65535). A value of -1 means that the data is disabled (see VAR_INPUT nDataRead.2 = TRUE).

rSetpointAnalog: analog setpoint in % (0...100). A value of -1 means that the data is disabled (see VAR_INPUT nDataRead.3 = TRUE).

rRelativeVolumetricFlow: relative volume flow rate in % (0...100). A value of -1 means that the data is disabled (see VAR_INPUT nDataRead.4 = TRUE).

rAbsoluteVolumetricFlow: absolute volume flow rate in l/s (0...100). A value of -1 means that the data is disabled (see VAR_INPUT nDataRead.5 = TRUE).

rGlycolConcentration: glycol concentration in % (0...100). A value of -1 means that the data is disabled (see VAR_INPUT nDataRead.6 = TRUE).

rTemperature1_Remote: temperature1 remote in °C (-20...12). A value of -1 means that the data is disabled (see VAR_INPUT nDataRead.7 = TRUE).

rTemperature2_Integrated: temperature2 integrated in °C (-20...12). A value of -1 means that the data is disabled (see VAR_INPUT nDataRead.8 = TRUE).

rDeltaTemperature: temperature delta in K (0...14). A value of -1 means that the data is disabled (see VAR_INPUT nDataRead.9 = TRUE).

rRelativePower: relative power in % (0...100). A value of -1 means that the data is disabled (see VAR_INPUT nDataRead.10 = TRUE).

rAbsoluteCoolingPower: absolute cooling power in kW (0...21.5). A value of -1 means that the data is disabled (see VAR_INPUT nDataRead.11 = TRUE).

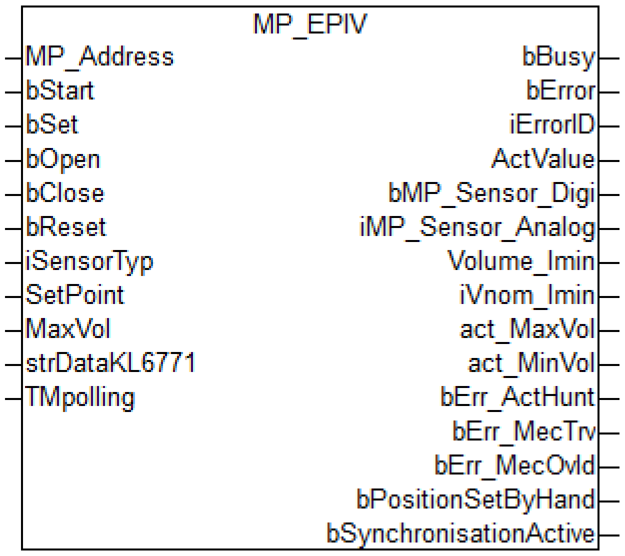
rAbsoluteHeatingPower: absolute heating power in kW (0...21.5). A value of -1 means that the data is disabled (see VAR_INPUT nDataRead.12 = TRUE).

rTotalVolume: total volume in m³ (0...214748.36). A value of -1 means that the data is disabled (see VAR_INPUT nDataRead.13 = TRUE).

nCoolingEnergy: cooling energy in kWh (0...21474836). A value of -1 means that the data is disabled (see VAR_INPUT nDataRead.14 = TRUE).

nHeatingEnergy: heating energy in kWh (0...21474836). A value of -1 means that the data is disabled (see VAR_INPUT nDataRead.15 = TRUE).

6.2.7 MP_EPIV



This function block is used to control a pressure-independent control ball valve. For more information please visit www.belimo.ch.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. If **bStart** remains TRUE, the device is addressed cyclically with a period specified by the time in **TMPolling**. The time should be set longer than 1 s. **bError** is used to indicate an error in communication with the actuator. The type of the error can be read with **iErrorId**.

VAR_INPUT

```
MP_Address      : USINT := 1;
bStart          : BOOL;
bSet            : BOOL;
bOpen           : BOOL;
bClose          : BOOL;
bReset          : BOOL;
iSensorTyp      : INT;
```

```
SetPoint        : USINT;
MaxVol          : WORD;
strDataKL6771   : DataKL6771;
TMpolling       : TIME := t#10s;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block. If this remains continuously TRUE, the function block will be activated cyclically with a period specified by the time in *TMPolling*.

bSet: a positive edge writes the *MaxVol* data to the actuator.

bOpen: a positive edge opens the dampers of the actuator, while a negative edge cancels the forced ventilation.

bClose: a positive edge closes the dampers of the actuator, while a negative edge cancels the forced closure.

bReset: a positive edge resets the actuator's error messages.

iSensorTyp: 0: no sensor connected, 1: digital sensor connected, 2: analog sensor connected (0...35 V), 3...6: output of a resistance in ohms (3...5 applies to PT1000, NI1000 and NI1000LuS; 6 applies to NTC). To convert to a temperature, use the corresponding conversion functions.

SetPoint: set volume flow rate (0...100 %).

MaxVol: maximum volume flow (30...100 %).

strDataKL6771: the data structure with which the KL6771 [▸ 26] function block must be linked (see DataKL6771 [▸ 84]).

TMpolling: the time for which the function block should address the actuator. Default 10 s, minimum time 1 s.

VAR_OUTPUT

```
bBusy                   : BOOL;
bError                  : BOOL;
iErrorId                : MP_Error;
ActValue                : WORD;
bMP_Sensor_Digi         : BOOL;
iMP_Sensor_Analog       : INT;
Volume_lmin             : WORD;
iVnom_lmin              : INT;
act_MaxVol              : INT;
act_MinVol              : INT;
bErr_ActHunt            : BOOL;
bErr_MecTrv             : BOOL;
bErr_MecOvld            : BOOL;
bPositionSetByHand      : BOOL;
bSynchronisationActive  : BOOL;
```

bBusy: this bit is set for as long as the function block is active.

bError: the output becomes TRUE as soon as an error occurs. This error is described via the *iErrorId* variable.

iErrorId: this output outputs an error code in the event of an error (see MP_ERROR [▸ 81]). *bError* goes TRUE at the same time.

ActValue: contains the current position of the actuator (0...100 %).

bMP_Sensor_Digi: if an analog sensor is connected, its value is indicated through this variable. *iSensorTyp* must be "1".

iMP_Sensor_Analog: if an analog sensor is connected, its value is indicated through this variable. *iSensorTyp* must be in the range "2...6".

Volume_lmin: output from volume flow rate in l/min.

iVnom_lmin: nominal air volume flow in l/min. This output is available from version 1.12.0.

act_MaxVol: maximum set volume flow rate in %.

act_MinVol: minimum set volume flow rate in %.

bErr_ActHunt: actuator error, "Regulating oscillation": the actuator is swinging backwards and forwards.

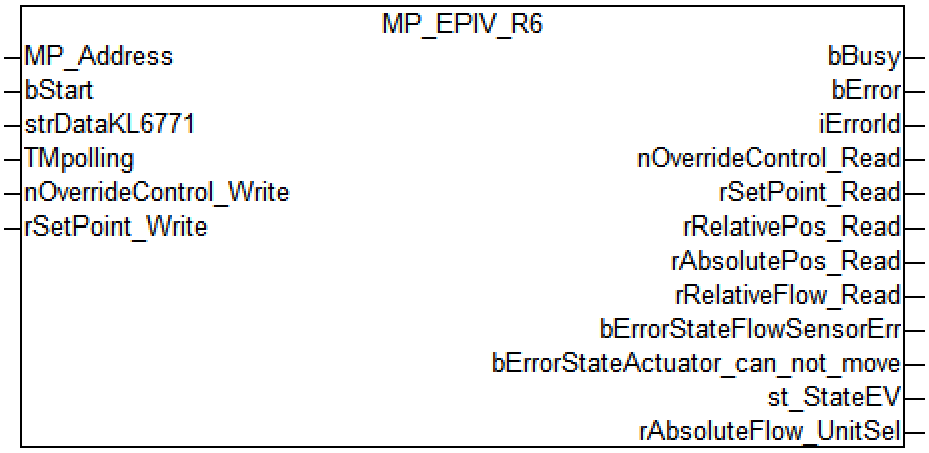
bErr_MecTrv: actuator error, "Positioning angle exceeded": the actuator has passed more than 10° beyond the adaptation position.

bErr_MecOvld: actuator error, "Overload": the set position could not be reached.

bPositionSetByHand: the position of the actuator was changed by hand.

bSynchronizationActive: the actuator synchronizes.

6.2.8 MP_EPIV_R6



This function block is used to control a control ball valve of series EP..R-R6+BAC.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. If **bStart** remains TRUE, the device is addressed cyclically with a period specified by the time in **TMPolling**. The time should be set longer than 1 s. **bError** is used to indicate an error in communication with the actuator. The type of the error can be read with **iErrorId**.

VAR_INPUT

```
MP_Address              : USINT := 1;
bStart                  : BOOL;
strDataKL6771           : DataKL6771;
TMpolling               : TIME := t#10s;
nOverrideControl_Write  : E_MPBus_Override_6wayMPIV := MPBus_6wayMPIV_None;
rSetPoint_Write         : LREAL;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block. If this remains continuously TRUE, the function block will be activated cyclically with a period specified by the time in *TMPolling*.

strDataKL6771: the data structure with which the KL6771 [▶ 26] function block must be linked (see DataKL6771 [▶ 84]).

TMpolling: the time for which the function block should address the actuator. Default 10 s, minimum time 1 s.

nOverrideControl_Write: the relative setpoint is ignored in override control mode (see E_MPBus_Override_6wayMPIV [▶ 81]). Overriding is disabled if the command is not repeated within 120 minutes.

rSetPoint_Write: 0...100 %.

VAR_OUTPUT

```
bBusy                             : BOOL;
bError                            : BOOL;
iErrorId                          : MP_Error;
nOverrideControl_Read             : E_MPBus_Override_6wayMPIV;
rSetPoint_Read                    : LREAL;
rRelativePos_Read                 : LREAL;
rAbsolutePos_Read                 : LREAL;
rRelativeFlow_Read                : LREAL;
bErrorStateFlowSensorErr          : BOOL;
bErrorStateActuator_can_not_move  : BOOL;
st_StateEV                        : st_StateEV;
rAbsoluteFlow_UnitSel             : LREAL;
```

bBusy: this bit is set for as long as the function block is active.

bError: the output becomes TRUE as soon as an error occurs. This error is described via the *iErrorId* variable.

iErrorId: this output outputs an error code in the event of an error (see MP_Error [▶ 81]). *bError* goes TRUE at the same time.

nOverrideControl_Read: current override control mode (see E_MPBus_Override_6wayMPIV [▶ 81]).

rSetPoint_Read: setpoint in % (0...100 %).

rRelativePos_Read: relative position in % (0...100 %).

rAbsolutePos_Read: absolute position in °.

rRelativeFlow_Read: relative flow rate in % (0...100 %).

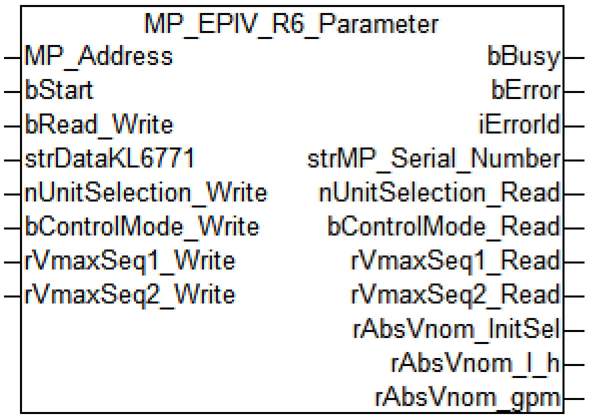
bErrorStateFlowSensorErr: flow sensor is faulty.

bErrorStateActuator_can_not_move: actuator cannot move.

st_StateEV: only devices from 27 March 2014 (see St_StateEV [▶ 86]).

rAbsoluteFlow_UnitSel: absolute flow rate in UnitSel (0...4294967295).

6.2.9 MP_EPIV_R6_Parameter



This function block is used for parameterization of valves of EP..R-R6+BAC series.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the block is active. **bError** is used to indicate an error in communication with the drive. The type of the error can be read with **iErrorId**.

VAR_INPUT

```
MP_Address              : USINT := 1;
bStart                  : BOOL;
bRead_Write             : BOOL;
```

```
strDataKL6771          : DataKL6771;
nUnitSelection_Write   : E_MP_EP_R_R6_UnitSel;
bControlMode_Write     : BOOL;
rVmaxSeq1_Write        : LREAL;
rVmaxSeq2_Write        : LREAL;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block. If this remains continuously TRUE, the function block will be activated cyclically with a period specified by the time in *TMPolling*.

bRead_Write: if FALSE then READ only; if TRUE then READ and WRITE.

strDataKL6771: the data structure with which the KL6771 [▶ 26] function block must be linked (see DataKL6771 [▶ 84]).

nUnitSelection_Write: scaling for *rAbsoluteFlow_UnitSel* (see E_MP_EP_R_R6_UnitSel [▶ 76]).

bControlMode_Write: FALSE: position-controlled, TRUE: flow-controlled.

rVmaxSeq1_Write: 0...100 %.

rVmaxSeq2_Write: 0...100 %.

VAR_OUTPUT

```
bBusy                 : BOOL;
bError                : BOOL;
iErrorId              : MP_Error;
strMP_Serial_Number   : MP_Serial_Number;
nUnitSelection_Read   : E_MP_EP_R_R6_UnitSel;
bControlMode_Read     : BOOL;
rVmaxSeq1_Read        : LREAL;
rVmaxSeq2_Read        : LREAL;
rAbsVnom_InitSel      : LREAL;
rAbsVnom_l_h          : LREAL;
rAbsVnom_gpm          : LREAL;
```

bBusy: this bit is set for as long as the function block is active.

bError: the output becomes TRUE as soon as an error occurs. This error is described via the *iErrorId* variable.

iErrorId: this output outputs an error code in the event of an error (see MP_Error [▶ 81]). *bError* goes TRUE at the same time.

strMP_Serial_Number: structure for the serial number. (see MP_Serial_Number [▶ 84]).

nUnitSelection_Read: set the scaling (see E_MP_EP_R_R6_UnitSel).

bControlMode_Read: FALSE: position-controlled, TRUE: flow-controlled.

rVmaxSeq1_Read: maximum sequence speed 1 in % (0...100 %).

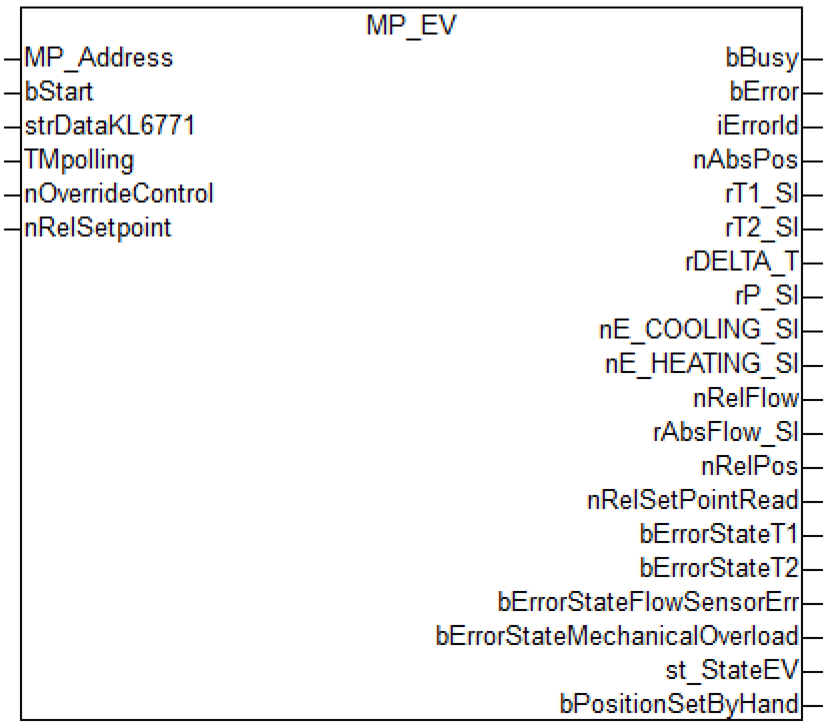
rVmaxSeq2_Read: maximum sequence speed 2 in % (0...100 %).

rAbsVnom_InitSel: nominal volume (see *rAbsoluteFlow_UnitSel*).

rAbsVnom_l_h: nominal volume in l/h (0...4294967295).

rAbsVnom_gpm: nominal volume in gpm (0...4294967295).

6.2.10 MP_EV



This function block is used to control a control ball valve of series P6...W..EV-BAC. For more information please visit www.belimo.ch.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. If **bStart** remains TRUE, the device is addressed cyclically with a period specified by the time in **TMPolling**. The time should be set longer than 1 s. **bError** is used to indicate an error in communication with the actuator. The type of the error can be read with **iErrorId**.

VAR_INPUT

```
MP_Address          : USINT := 1;
bStart              : BOOL;
strDataKL6771       : DataKL6771;
TMpolling           : TIME  := t#10s;
nOverrideControl    : E_MPBus_Override  := MPBus_Override_Auto;
nRelSetpoint        : INT;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block. If this remains continuously TRUE, the function block will be activated cyclically with a period specified by the time in *TMPolling*.

strDataKL6771: the data structure with which the KL6771 [▸ 26] function block must be linked (see DataKL6771 [▸ 84]).

TMpolling: the time for which the function block should address the actuator. Default 10 s, minimum time 1 s.

nOverrideControl: the relative setpoint *nRelSetpoint* is ignored in override mode (see E_MPBus_Override [▸ 81]). Overriding is disabled if the command is not repeated within 120 minutes.

nRelSetpoint: the set value is interpreted either as position setpoint or as advance setpoint.

VAR_OUTPUT

```
bBusy                           : BOOL;
bError                          : BOOL;
iErrorId                        : MP_Error;
nAbsPos                         : INT;
```

```
rT1_SI                          : REAL;
rT2_SI                          : REAL;
rDELTA_T                        : REAL;
rP_SI                           : REAL;
nE_COOLING_SI                   : DINT;
nE_HEATING_SI                   : DINT;
nRelFlow                        : INT;
rAbsFlow_SI                     : REAL;
nRelPos                         : INT;
nRelSetPointRead                : INT;
bErrorStateT1                   : BOOL;
bErrorStateT2                   : BOOL;
bErrorStateFlowSensorErr        : BOOL;
bErrorStateMechanicalOverload : BOOL;
st_StateEV                      : St_StateEV;
bPositionSetByHand              : BOOL;
```

bBusy: This bit is set for as long as the block is active.

bError: The output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorId*.

iErrorId: The output issues an error code when an error occurs (see MP_ERROR [▶ 81]). Simultaneously *bError* is TRUE.

nAbsPos: Absolute position in °.

rT1_SI: Temperature 1 (remote) in °C.

rT2_SI: Temperature 2 (embedded) in °C.

rDELTA_T: Delta temperature in °C.

rP_SI: Power in kW.

nE_COOLING_SI: Cooling energy in kWh.

nE_HEATING_SI: Heating energy in kWh.

nRelFlow: Relative flow in %.

rAbsFlow_SI: Absolute flow in l/min.

nRelPos: Relative position in %.

nRelSetPointRead: The setpoint is interpreted either as position setpoint or as flow setpoint (related to Vmax) in %.

bErrorStateT1: Failure temperature sensor T1.

bErrorStateT2: Failure temperature sensor T2.

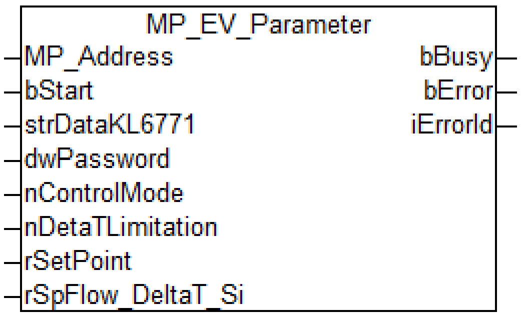
bErrorStateFlowSensorErr: Failure flow sensor.

bErrorStateMechanicalOverload: Mechanical overload detected.

st_StateEV: Only devices from 27.03.2014 (see St_StateEV [▶ 86]).

bPositionSetByHand: The position of the drive has been changed manually.

6.2.11 MP_EV_Parameter



This function block is used for parameterization of valves.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the block is active. **bError** is used to indicate an error in communication with the drive. The type of the error can be read with **iErrorId**.

VAR_INPUT

```
MP_Address         : USINT := 1;
bStart             : BOOL;
strDataKL6771      : DataKL6771;
dwPassword         : DWORD;
nControlMode       : E_MPBus_ControlMode := MPBus_ControlMode_Disable;
nDetaTLimitation   : E_MPBus_DeltaTLimitation := MPBus_DeltaTLimitation_Disable;
rSetPoint          : REAL := 0.0;
rSpFlow_DeltaT_Si  : REAL := 0.0;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block.

strDataKL6771: the data structure with which the KL6771 [▸ 26] function block must be linked (see DataKL6771 [▸ 84]).

dwPassword: the actuator password is usually 0x0000.

nControlMode: specifies the control mode (see E_MPBus_ControlMode [▸ 80]).

nDetaTLimitation: dT limitation (see E_MPBus_DeltaTLimitation [▸ 80]).

rSetPoint: dT limit.

rSpFlow_DeltaT_Si: flow at saturation.

VAR_OUTPUT

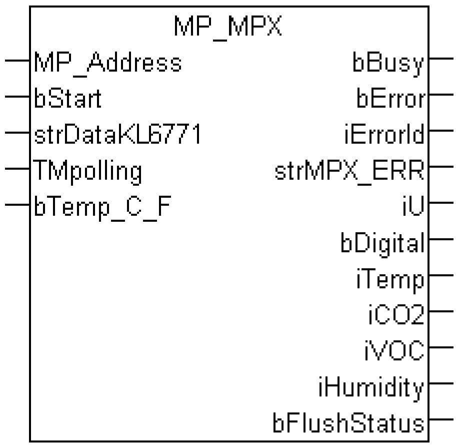
```
bBusy              : BOOL;
bError             : BOOL;
iErrorId           : MP_Error;
```

bBusy: This bit is set for as long as the block is active.

bError: The output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorId*.

iErrorId: The output issues an error code when an error occurs (see MP_ERROR [▸ 81]). Simultaneously *bError* is TRUE.

6.2.12 MP_MPX



For BELIMO room sensor MS24A-R..-MPX.

MS24A-R01-MPX temperature

MS24A-R02-MPX temperature, CO2

MS24A-R03-MPX temperature, VOC

MS24A-R04-MPX temperature, CO2, VOC

MS24A-R05-MPX temperature, humidity

MS24A-R06-MPX temperature, humidity, CO2

MS24A-R07-MPX temperature, humidity, VOC

MS24A-R08-MPX temperature, humidity, CO2, VOC

VAR_INPUT

```
MP_Address      : USINT := 1;
bStart          : BOOL;
strDataKL6771       : DataKL6771;
TMpolling       : TIME := t#10s;
bTemp_C_F       : BOOL;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block. If this remains continuously TRUE, the function block will be activated cyclically with a period specified by the time in *TMPolling*.

strDataKL6771: the data structure with which the KL6771 [▶ 26] function block must be linked (see DataKL6771 [▶ 84]).

TMpolling: the time for which the function block should address the sensor. Default 10 s, minimum time 1 s.

bTemp_C_F: FALSE = °C / TRUE = °F.

VAR_OUTPUT

```
bBusy           : BOOL;
bError          : BOOL;
iErrorId        : MP_Error;
strMPX_ERR      : MP_BUS_MPX_ERROR;
iU              : INT;
bDigital        : BOOL;
iTemp           : INT;
iCO2            : INT;
```

```
iVOC           : INT;
iHumidity      : INT;
bFlushStatus   : BOOL;
```

bBusy: this bit is set for as long as the function block is active.

bError: the output becomes TRUE as soon as an error occurs. This error is described via the *iErrorId* variable.

iErrorId: this output outputs an error code in the event of an error (see MP_Error [▸ 81]). *bError* goes TRUE at the same time.

strMPX_ERR: sensor error messages (see MP_BUS_MPX_ERROR [▸ 84]).

iU: 0...10 V UNIT 1 mV.

bDigital: DI 24 V.

iTemp: 0...50 °C Unit: 0.01 °C.

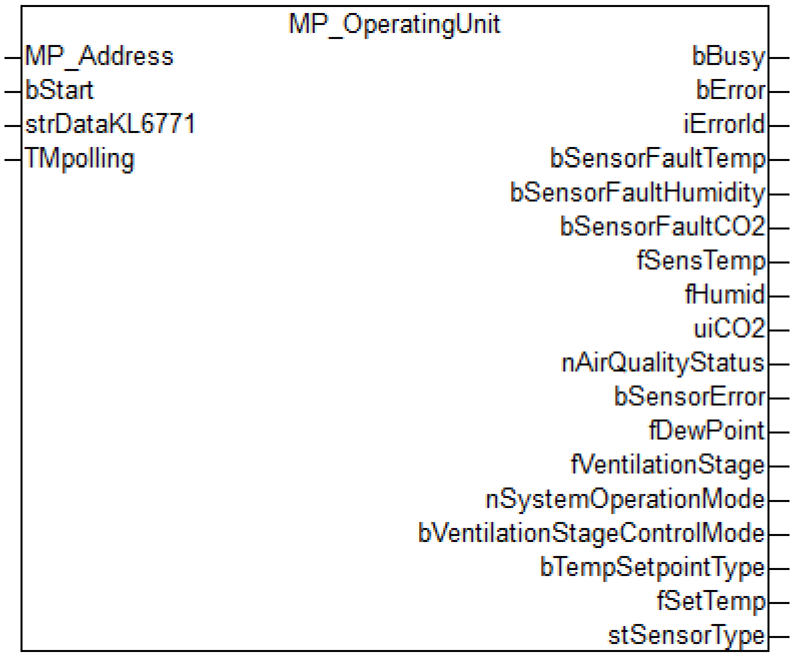
iCO2: 0...2000 ppm Unit: 1 ppm.

iVOC: 0...2000 ppm Unit: 1 ppm (pseudo).

iHumidity: 10...90 % Unit: 0.01 %.

bFlushStatus: VOC gradient threshold exceeded, FALSE = air quality OK, TRUE = air quality not OK, flush.

6.2.13 MP_OperatingUnit



This function block is used to read room sensors of types P-22Rxx-1900x-1. It is compatible with 22Rxx-19-1 room sensors produced as of May 2022 (before May 2022, see MP_RoomSensor [▸ 54]).
For more information, see www.belimo.ch.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. If **bStart** remains TRUE, the device is addressed cyclically with a period specified by the time in **TMPolling**. The time should be set longer than 1 s. **bError** is used to indicate an error in communication with the actuator. The type of the error can be read with **iErrorId**.

VAR_INPUT

```
MP_Address     : USINT := 1;
bStart         : BOOL;
strDataKL6771  : DataKL6771;
TMpolling      : TIME := t#10s;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block. If this remains continuously TRUE, the function block will be activated cyclically with a period specified by the time in *TMPolling*.

strDataKL6771: the data structure with which the KL6771() [▶ 26] function block must be linked (see DataKL6771 [▶ 84]).

TMpolling: the time for which the function block should address the actuator. Default 10 s. Minimum time 1 s.

VAR_OUTPUT

```
bBusy                        : BOOL;
bError                       : BOOL;
iErrorId                     : MP_Error;
bSensorFaultTemp             : BOOL;
bSensorFaultHumidity         : BOOL;
bSensorFaultCO2              : BOOL;
fSensTemp                    : LREAL;
fHumid                       : LREAL;
uiCO2                        : UINT;
nAirQualityStatus            : E_MP_AirQualityStatus;
bSensorError                 : BOOL;
fDewPoint                    : LREAL;
fVentilationStage            : LREAL;
nSystemOperationMode         : E_MP_SystemOperationMode;
bVentilationStageControlMode : BOOL;
bTempSetpointType            : BOOL;
fSetTemp                     : LREAL;
stSensorType                 : STRING;
```

bBusy: this bit is set for as long as the function block is active.

bError: the output becomes TRUE as soon as an error occurs. This error is described via the *iErrorId* variable.

iErrorId: this output outputs an error code in the event of an error (see MP_ERROR [▶ 81]). *bError* goes TRUE at the same time.

bSensorFaultTemp: temperature sensor has a fault.

bSensorFaultHumidity: humidity sensor has a fault.

bSensorFaultCO2: CO2 sensor has a fault.

fSensTemp: room temperature in °C or °F (0..50 or 32..122).

fHumid: relative humidity in % (0..100).

uiCO2: CO2 value in ppm (0..2000).

nAirQualityStatus: air quality status (see E_MP_AirQualityStatus [▶ 73]).

bSensorError: one of the sensors has a fault.

fDewPoint: dew point temperature in °C (-50..50).

fVentilationStage: ventilation stage in % (0..100).

nSystemOperationMode: operation mode of the system (see E_MP_SystemOperationMode [▶ 78]).

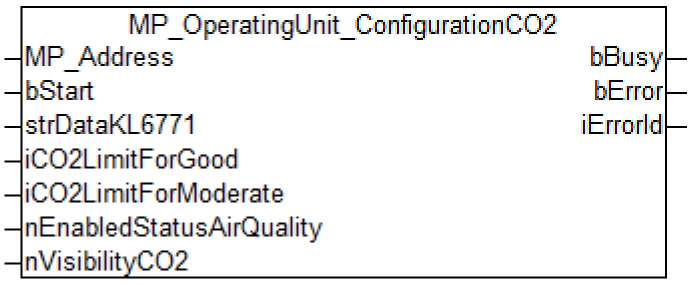
bVentilationStageControlMode: FALSE = manual; TRUE = automatic.

bTempSetpointType: FALSE = absolute; TRUE = relative.

fSetTemp: bTempSetpointType FALSE = setpoint room temperature in °C; bTempSetpointType TRUE = setpoint relative room temperature in °C.

stSensorType: DSensortyp. '?' = not read; 'unknown' = number unknown.

6.2.14 MP_OperatingUnit_ConfigurationCO2



This function block is used to configure room sensors of types P-22Rxx-1900x-1. It is compatible with 22Rxx-19-1 room sensors produced as of May 2022 (before May 2022, see MP_RoomSensor_Parameter [▸ 55]).
For more information, see www.belimo.ch.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. **bError** is used to indicate an error in communication with the actuator. The type of the error can be read with **iErrorId**.

VAR_INPUT

```
MP_Address                : USINT := 1;
bStart                    : BOOL;
strDataKL6771             : DataKL6771;
iCO2LimitForGood          : UINT := 1000;
iCO2LimitForModerate      : UINT := 1500;
nEnabledStatusAirQuality : E_MP_EnabledStatus;
nVisibilityCO2            : E_MP_DisplayVisibility;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block.

strDataKL6771: the data structure with which the KL6771() [▸ 26] function block must be linked (see DataKL6771 [▸ 84]).

iCO2LimitForGood: CO2 limit for "Good" air quality in ppm (600..1249).

iCO2LimitForModerate: CO2 limit for "Moderate" air quality in ppm (1250..2000).

nEnabledStatusAirQuality: status of the air quality display (see E_MP_EnabledStatus [▸ 76]).

nVisibilityCO2: visibility of the CO2 value (see E_MP_DisplayVisibility [▸ 76]).

VAR_OUTPUT

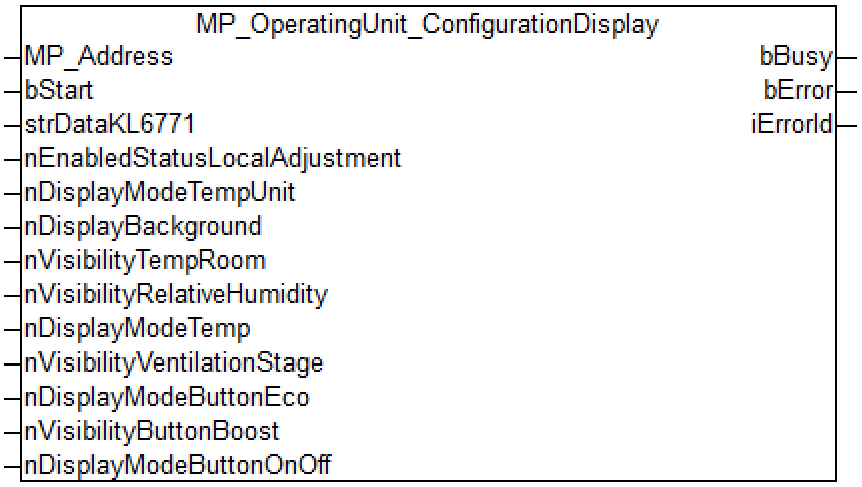
```
bBusy    : BOOL;
bError   : BOOL;
iErrorId : MP_Error;
```

bBusy: this bit is set for as long as the function block is active.

bError: the output becomes TRUE as soon as an error occurs. This error is described via the *iErrorId* variable.

iErrorId: this output outputs an error code in the event of an error (see MP_ERROR [▸ 81]). *bError* goes TRUE at the same time.

6.2.15 MP_OperatingUnit_ConfigurationDisplay



This function block is used to configure room sensors of types P-22Rxx-1900x-1. It is compatible with 22Rxx-19-1 room sensors produced as of May 2022 (before May 2022, see MP_RoomSensor_Parameter [▶ 55]).
For more information, see www.belimo.ch.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. **bError** is used to indicate an error in communication with the actuator. The type of the error can be read with **iErrorId**.

VAR_INPUT

```
MP_Address                     : USINT := 1;
bStart                         : BOOL;
strDataKL6771                  : DataKL6771;
nEnabledStatusLocalAdjustment  : E_MP_EnabledStatus;
nDisplayModeTempUnit           : E_MP_DisplayModeTempUnit;
nDisplayBackground             : E_MP_DisplayBackground;
nVisibilityTempRoom            : E_MP_DisplayVisibility;
nVisibilityRelativeHumidity    : E_MP_DisplayVisibility;
nDisplayModeTemp               : E_MP_DisplayModeTemp;
nVisibilityVentilationStage    : E_MP_DisplayVisibility;
nDisplayModeButtonEco          : E_MP_DisplayModeButton;
nVisibilityButtonBoost         : E_MP_DisplayVisibility;
nDisplayModeButtonOnOff        : E_MP_DisplayModeButton;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block.

strDataKL6771: the data structure with which the KL6771() [▶ 26] function block must be linked (see DataKL6771 [▶ 84]).

nEnabledStatusLocalAdjustment: status of access permission for local adjustments (see E_MP_EnabledStatus [▶ 76]).

nDisplayModeTempUnit: display mode of the unit for the temperature in the display (see E_MP_DisplayModeTempUnit [▶ 75]).

nDisplayBackground: background color of the display (see E_MP_DisplayBackground [▶ 74]).

nVisibilityTempRoom: visibility of the room temperature (see E_MP_DisplayVisibility [▶ 76]).

nVisibilityRelativeHumidity: visibility of the relative humidity (see E_MP_DisplayVisibility [▶ 76]).

nDisplayModeTemp: display mode of the temperature (see E_MP_DisplayModeTemp [▶ 75]).

nVisibilityVentilationStage: visibility of the ventilation stage (see E_MP_DisplayVisibility [▶ 76]).

nDisplayModeButtonEco: display mode of the Eco button (see E_MP_DisplayModeButton [▶ 74]).

nVisibilityButtonBoost: display mode of the Boost button (see E_MP_DisplayVisibility [▸ 76]).

nDisplayModeButtonOnOff: display mode of the OnOff button (see E_MP_DisplayModeButton [▸ 74]).

VAR_OUTPUT

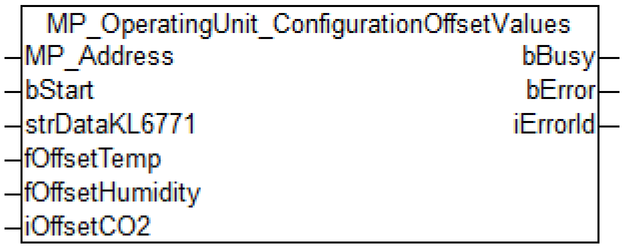
```
bBusy    : BOOL;
bError   : BOOL;
iErrorId : MP_Error;
```

bBusy: this bit is set for as long as the function block is active.

bError: the output becomes TRUE as soon as an error occurs. This error is described via the *iErrorId* variable.

iErrorId: this output outputs an error code in the event of an error (see MP_ERROR [▸ 81]). *bError* goes TRUE at the same time.

6.2.16 MP_OperatingUnit_ConfigurationOffsetValues



This function block is used to configure room sensors of types P-22Rxx-1900x-1. It is compatible with 22Rxx-19-1 room sensors produced as of May 2022 (before May 2022, see MP_RoomSensor_Parameter [▸ 55]).
For more information, see www.belimo.ch.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. **bError** is used to indicate an error in communication with the actuator. The type of the error can be read with **iErrorId**.

VAR_INPUT

```
MP_Address      : USINT := 1;
bStart          : BOOL;
strDataKL6771   : DataKL6771;
fOffsetTemp     : LREAL := 0;
fOffsetHumidity : LREAL := 0;
iOffsetCO2      : INT := 0;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block.

strDataKL6771: the data structure with which the KL6771() [▸ 26] function block must be linked (see DataKL6771 [▸ 84]).

fOffsetTemp: offset applied to the measured temperature in °C or °F (-15..15 or -27..27).

fOffsetHumidity: offset applied to the measured relative humidity in % (-20..20).

iOffsetCO2: offset applied to the measured CO2 content in ppm (-500..500).

VAR_OUTPUT

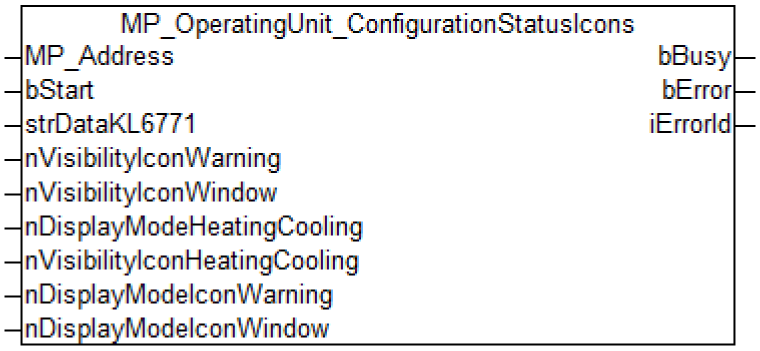
```
bBusy    : BOOL;
bError   : BOOL;
iErrorId : MP_Error;
```

bBusy: this bit is set for as long as the function block is active.

bError: the output becomes TRUE as soon as an error occurs. This error is described via the *iErrorId* variable.

iErrorId: this output outputs an error code in the event of an error (see MP_ERROR [▶ 81]). *bError* goes TRUE at the same time.

6.2.17 MP_OperatingUnit_ConfigurationStatusIcons



This function block is used to configure room sensors of types P-22Rxx-1900x-1. It is compatible with 22Rxx-19-1 room sensors produced as of May 2022 (before May 2022, see MP_RoomSensor_Parameter [▶ 55]).
For more information, see www.belimo.ch.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. **bError** is used to indicate an error in communication with the actuator. The type of the error can be read with **iErrorId**.

VAR_INPUT

```
MP_Address                    : USINT := 1;
bStart                        : BOOL;
strDataKL6771                 : DataKL6771;
nVisibilityIconWarning        : E_MP_DisplayVisibility;
nVisibilityIconWindow         : E_MP_DisplayVisibility;
nDisplayModeHeatingCooling    : E_MP_DisplayModeHeatingCooling;
nVisibilityIconHeatingCooling : E_MP_DisplayVisibility;
nDisplayModeIconWarning       : E_MP_DisplayModeIconWarning;
nDisplayModeIconWindow        : E_MP_DisplayModeIconWindow;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block.

strDataKL6771: the data structure with which the KL6771() [▶ 26] function block must be linked (see DataKL6771 [▶ 84]).

nVisibilityIconWarning: visibility of the warning icon. Applies if nDisplayModeIconWarning is set to 1 (see E_MP_DisplayVisibility [▶ 76]).

nVisibilityIconWindow: visibility of the window icon. Applies if nDisplayModeIconWindow is set to 1 (see E_MP_DisplayVisibility [▶ 76]).

nDisplayModeHeatingCooling: display mode of heating or cooling icons (see E_MP_DisplayModeHeatingCooling [▶ 74]).

nVisibilityIconHeatingCooling: visibility of heating or cooling icons (see E_MP_DisplayVisibility [▶ 76]).

nDisplayModeIconWarning: display mode of the warning icon (see E_MP_DisplayModeIconWarning [▶ 74]).

nDisplayModeIconWindow: display mode of the window icon (see E_MP_DisplayModeIconWindow [▶ 75]).

VAR_OUTPUT

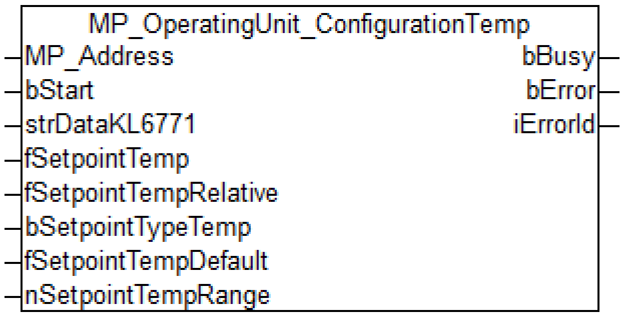
```
bBusy    : BOOL;
bError   : BOOL;
iErrorId : MP_Error;
```

bBusy: this bit is set for as long as the function block is active.

bError: the output becomes TRUE as soon as an error occurs. This error is described via the *iErrorId* variable.

iErrorId: this output outputs an error code in the event of an error (see MP_ERROR [▸ 81]). *bError* goes TRUE at the same time.

6.2.18 MP_OperatingUnit_ConfigurationTemp



This function block is used to configure room sensors of types P-22Rxx-1900x-1. It is compatible with 22Rxx-19-1 room sensors produced as of May 2022 (before May 2022, see MP_RoomSensor_Parameter [▸ 55]).
For more information, see www.belimo.ch.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. **bError** is used to indicate an error in communication with the actuator. The type of the error can be read with **iErrorId**.

VAR_INPUT

```
MP_Address            : USINT := 1;
bStart                : BOOL;
strDataKL6771         : DataKL6771;
fSetpointTemp         : LREAL := 20;
fSetpointTempRelative : LREAL := 0;
bSetpointTypeTemp     : BOOL;
fSetpointTempDefault  : LREAL := 20;
nSetpointTempRange    : USINT := 10;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block.

strDataKL6771: the data structure with which the KL6771() [▸ 26] function block must be linked (see DataKL6771 [▸ 84]).

fSetpointTemp: setpoint for the room temperature in °C (5..45).

fSetpointTempRelative: setpoint for the relative room temperature in °C (-5.6..5.6).

bSetpointTypeTemp: FALSE = absolute; TRUE = relative.

fSetpointTempDefault: default setpoint for the room temperature in °C (15..35).

nSetpointTempRange: setpoint range for the temperature in °C (0..10).

VAR_OUTPUT

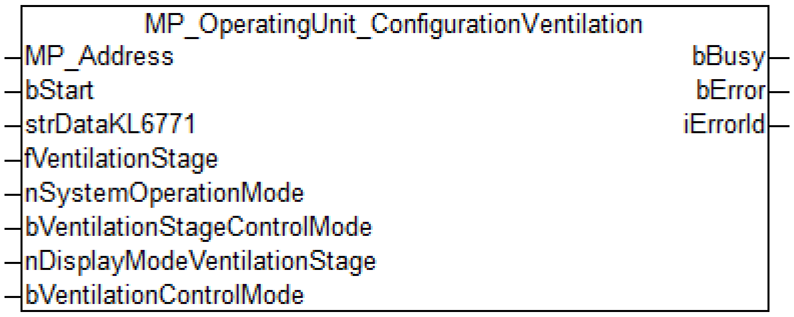
```
bBusy    : BOOL;
bError   : BOOL;
iErrorId : MP_Error;
```

bBusy: this bit is set for as long as the function block is active.

bError: the output becomes TRUE as soon as an error occurs. This error is described via the *iErrorId* variable.

iErrorId: this output outputs an error code in the event of an error (see MP_ERROR [▶ 81]). *bError* goes TRUE at the same time.

6.2.19 MP_OperatingUnit_ConfigurationVentilation



This function block is used to configure room sensors of types P-22Rxx-1900x-1. It is compatible with 22Rxx-19-1 room sensors produced as of May 2022 (before May 2022, see MP_RoomSensor_Parameter [▶ 55]).
For more information, see www.belimo.ch.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. **bError** is used to indicate an error in communication with the actuator. The type of the error can be read with **iErrorId**.

VAR_INPUT

```
MP_Address                   : USINT := 1;
bStart                       : BOOL;
strDataKL6771                : DataKL6771;
fVentilationStage            : LREAL;
nSystemOperationMode         : E_MP_SystemOperationMode;
bVentilationStageControlMode : BOOL;
nDisplayModeVentilationStage : E_MP_DisplayModeVentilationStage := MPBus_DisplayModeVentilationStage
_7;
bVentilationControlMode      : BOOL;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block.

strDataKL6771: the data structure with which the KL6771() [▶ 26] function block must be linked (see DataKL6771 [▶ 84]).

fVentilationStage: ventilation stage in the room/zone in % (0..100).

nSystemOperationMode: operation mode of the system (see E_MP_SystemOperationMode [▶ 78]).

bVentilationStageControlMode: mode of ventilation stage control. Applies if bVentilationControlMode is set to TRUE.

nDisplayModeVentilationStage: display mode of the ventilation stage (see E_MP_DisplayModeVentilationStage [▶ 75]).

bVentilationControlMode: FALSE = manual mode only; TRUE = hybrid control mode (setpoint invisible in auto mode).

VAR_OUTPUT

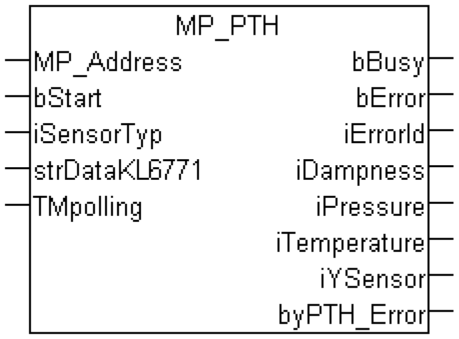
```
bBusy    : BOOL;
bError   : BOOL;
iErrorId : MP_Error;
```

bBusy: this bit is set for as long as the function block is active.

bError: the output becomes TRUE as soon as an error occurs. This error is described via the *iErrorId* variable.

iErrorId: this output outputs an error code in the event of an error (see MP_ERROR [▸ 81]). *bError* goes TRUE at the same time.

6.2.20 MP_PTH



This function block is used to control and monitor a PTH sensor.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. If **bStart** remains TRUE, the device is addressed cyclically with a period specified by the time in **TMPolling**. The time should be set longer than 1 s. **bError** is used to indicate an error in communication with the sensor. The type of the error can be read with **iErrorId**.

If an external sensor is connected to the sensor, its type is to be indicated in **iSensorTyp**. If no sensor is connected, the value "0" should be entered, or the variable left open. A digital sensor should be parameterized with "3". The state of the sensor is output through the variable **iYSensor**.

VAR_INPUT

```
MP_Address      : USINT := 1;
bStart          : BOOL;
iSensorTyp      : INT;
strDataKL6771   : DataKL6771;
TMpolling       : TIME  := t#10s;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block. If this remains continuously TRUE, the function block will be activated cyclically with a period specified by the time in *TMPolling*.

iSensorTyp: "0" or blank - no sensor is connected; "1" an analog sensor is connected with voltage output in mV; "2" output of a resistance in ohms - 1.0 ohm; "3" output of a resistance in ohms - 0.1 ohm; "4" digital sensor.

strDataKL6771: the data structure with which the KL6771 [▸ 26] function block must be linked (see DataKL6771 [▸ 84]).

TMpolling: the time for which the function block should address the sensor. Default 10 s, minimum time 1 s.

VAR_OUTPUT

```
bBusy           : BOOL;
bError          : BOOL;
iErrorId        : MP_Error;
iDampness       : INT;
iPressure       : INT;
iTemperature    : INT;
iYSensor        : INT;
byPTH_Error     : BYTE;
```

bBusy: this bit is set for as long as the function block is active.

bError: the output becomes TRUE as soon as an error occurs. This error is described via the *iErrorId* variable.

iErrorId: this output outputs an error code in the event of an error (see MP_Error [▸ 81]). *bError* goes TRUE at the same time.

iDampness: relative humidity in 0.01 %.

iPressure: differential pressure, output in 0.1 Pa.

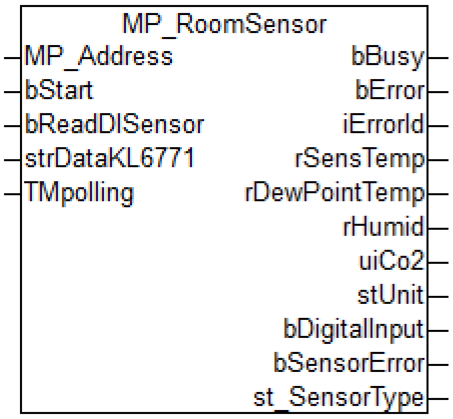
iTemperature: temperature in 0.01 °C.

iYSensor: Y-input, iSensorTyp = "1" - voltage 0...10 V output in [mV]; iSensorTyp = "2" resistance output in 1.0 ohm; iSensorTyp = "3" resistance output in 0.1 ohm; iSensorTyp = "4" digital switch 0 or 1.

byPTH_Error: sensor error - 0 - no error.

byPTH_Error	Description
Bit 0	Servicing error
Bit 1	Error message, sensor faulty
Bit 2	-
Bit 3	-
Bit 4	Sensor (temperature/humidity) faulty
Bit 5	A/D converter (pressure) faulty
Bit 6	A/D converter (Y-input) faulty
Bit 7	-

6.2.21 MP_RoomSensor



This function block is used to read out room sensors of types 22Rxx-19-1 that were produced before May 2022 (as of May 2022, see MP_OperatingUnit [▸ 45]).
For more information, see www.belimo.ch.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. If **bStart** remains TRUE, the device is addressed cyclically with a period specified by the time in **TMPolling**. The time should be set longer than 1 s. **bError** is used to indicate an error in communication with the actuator. The type of the error can be read with **iErrorId**.

VAR_INPUT

```
MP_Address    : USINT := 1;
bStart        : BOOL;
bReadDISensor : BOOL;
strDataKL6771 : DataKL6771;
TMpolling     : TIME := t#10s;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block. If this remains continuously TRUE, the function block will be activated cyclically with a period specified by the time in *TMPolling*.

bReadDISensor: if TRUE, the DI sensor is read and the result is available in *bDigitalInput*.

strDataKL6771: the data structure with which the KL6771 [▸ 26] function block must be linked (see DataKL6771 [▸ 84]).

TMpolling: the time for which the function block should address the actuator. Default 10 s, minimum time 1 s.

VAR_OUTPUT

```
bBusy          : BOOL;
bError         : BOOL;
iErrorId       : MP_Error;
rSensTemp      : LREAL;
rDewPointTemp  : LREAL;
rHumid         : LREAL;
uiCo2          : UINT;
stUnit         : STRING;
bDigitalInput  : BOOL;
bSensorError   : BOOL;
st_SensorType  : STRING;
```

bBusy: This bit is set for as long as the block is active.

bError: The output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorId*.

iErrorId: The output issues an error code when an error occurs (see MP_ERROR [▸ 81]). Simultaneously *bError* is TRUE.

rSensTemp: Sensor temperature in °C or °F.

rDewPointTemp: Temperature of the calculated dew point in °C or °F.

rHumid: Humidity in percent (% 0.01).

uiCo2: CO2 content in ppm.

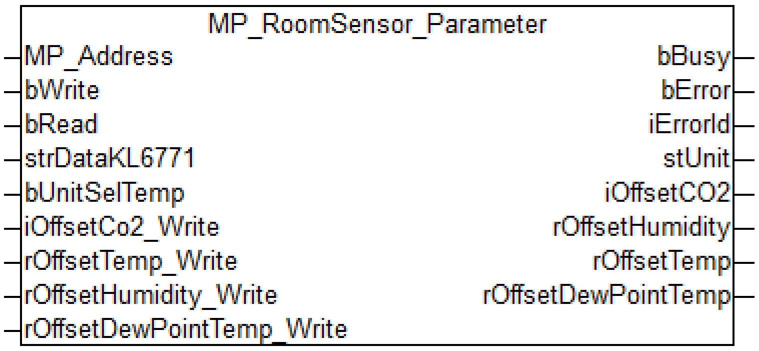
stUnit: C = °C or F = °F, ? = not read.

bDigitalInput: DI sensor read if *bReadDISensor* is TRUE.

bSensorError: One of the sensors has an error.

st_SensorType: Sensor type '?' = not read / type / 'unknown' = number unknown.

6.2.22 MP_RoomSensor_Parameter



This function block is used to parameterize room sensors of types 22Rxx-19-1 that were produced before May 2022 (as of May 2022, see MP_OperatingUnit_Configuration).
For more information, see www.belimo.ch.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. *bRead* reads the parameters, *bWrite* writes them to the room sensor. **bBusy** indicates that the function block is active. **bError** is used to indicate an error in communication with the actuator. The type of the error can be read with **iErrorId**.

VAR_INPUT

```
MP_Address                 : USINT := 1;
bWrite                     : BOOL;
bRead                      : BOOL;
strDataKL6771              : DataKL6771;
bUnitSelTemp               : BOOL;
iOffsetCo2_Write           : INT;
rOffsetTemp_Write          : LREAL;
rOffsetHumidity_Write      : LREAL;
rOffsetDewPointTemp_Write  : LREAL;
```

MP_Address: MP-Bus address of the slave.

bWrite: a positive edge starts the function block and writes the parameter.

bRead: a positive edge starts the function block and reads the parameter.

strDataKL6771: the data structure with which the KL6771 [▸ 26] function block must be linked (see DataKL6771 [▸ 84]).

bUnitSelTemp: FALSE = °C, TRUE = °F.

iOffsetCo2_Write: OffsetCO2 [ppm] -500...500.

rOffsetTemp_Write: OffsetTemp [UnitSel] -15...15 °C (-27...27 °F)

rOffsetHumidity_Write: OffsetHumidity [%] -20...+20.

rOffsetDewPointTemp_Write: OffsetDewPointTemp [UnitSel] -15...15 °C (-27...27 °F).

VAR_OUTPUT

```
bBusy                : BOOL;
bError               : BOOL;
iErrorId             : MP_Error;
stUnit               : STRING;
iOffsetCO2           : INT;
rOffsetHumidity      : LREAL;
rOffsetTemp          : LREAL;
rOffsetDewPointTemp  : LREAL;
```

bBusy: This bit is set for as long as the block is active.

bError: The output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorId*.

iErrorId: The output issues an error code when an error occurs (see MP_ERROR [▸ 81]). Simultaneously *bError* is TRUE.

stUnit: C = °C or F = °F, ? = not read.

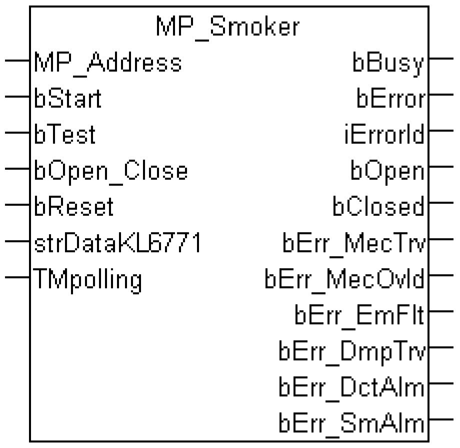
iOffsetCO2: OffsetCO2 [ppm].

rOffsetHumidity: OffsetHumidity [%] 0.01.

rOffsetTemp: OffsetTemp [°C or °F] 0.01.

rOffsetDewPointTemp: DewOffsetTemp [°C or °F] 0.01.

6.2.23 MP_Smoker



This function block is used to control and monitor a fire damper.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. If **bStart** remains TRUE, the device is addressed cyclically with a period specified by the time in **TMPolling**. The time should be set longer than 1 s, maximum 30 s. **bError** is used to indicate an error in communication with the actuator. The type of the error can be read with **iErrorID**.

A positive edge at **bReset** clears any pending error messages from the actuator. A telegram is thus sent to the actuator that acknowledges the errors in the actuator. If they persist, they remain set. This applies to all bErr_* error bits.

bOpen_Close is used to open or close the fire damper. TRUE causes the fire damper to open, while FALSE closes it. **bOpen** indicates that the damper is open, and **bClosed** indicates that it is closed. If both bits are FALSE, the actuator is currently opening or closing.

A positive edge at **bTest** initiates a test run on the fire damper. Errors that have been set can be cleared with this if they are no longer present.

VAR_INPUT

```
MP_Address      : USINT := 1;
bStart          : BOOL := TRUE;
bTest           : BOOL;
bOpen_Close     : BOOL;
bReset          : BOOL;
strDataKL6771   : DataKL6771;
TMpolling       : TIME := t#10s;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block. If this remains continuously TRUE, the function block will be activated cyclically with a period specified by the time in *TMPolling*.

bTest: a positive edge starts the test run at a fire damper.

bOpen_Close: TRUE opens a damper, while FALSE closes a damper.

bReset: a positive edge resets the actuator's error messages.

strDataKL6771: the data structure with which the KL6771 [▸ 26] function block must be linked (see DataKL6771 [▸ 84]).

TMpolling: the time for which the function block should address the actuator. Default 10 s, minimum time 1 s.

VAR_OUTPUT

```
bBusy           : BOOL;
bError          : BOOL;
iErrorId        : MP_Error;
bOpen           : BOOL;
bClosed         : BOOL;
bErr_MecTrv     : BOOL;
bErr_MecOvld    : BOOL;
bErr_EmFlt      : BOOL;
bErr_DmpTrv     : BOOL;
bErr_DctAlm     : BOOL;
bErr_SmAlm      : BOOL;
```

bBusy: This bit is set for as long as the block is active.

bError: The output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorId*.

iErrorId: The output issues an error code when an error occurs (see MP_ERROR [▶ 81]). Simultaneously *bError* is TRUE.

bOpen: Fire protection flap is open.

bClosed: Fire protection flap is closed.

bErr_MecTrv: Drive error, "Positioning angle exceeded"; the drive has passed more than 10° beyond the adaptation position.

bErr_MecOvld: Drive error, "Overload"; the desired position could not be reached.

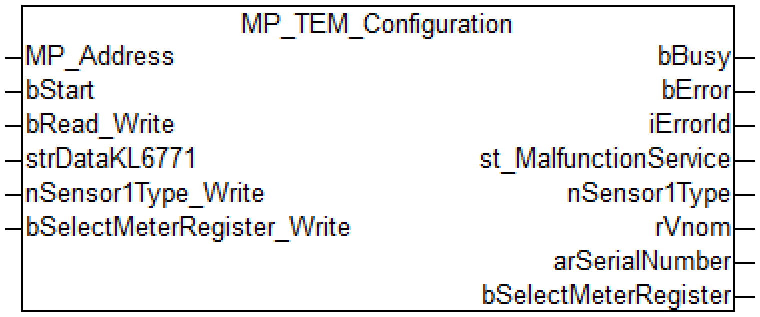
bErr_EmFlt: Drive error, "Safety-relevant error"; ambient temperature above 72°C or motor temperature above 85°C. Error can only be reset at the factory.

bErr_DmpTrv: Drive error, "Free flap movement error"; is cleared if the test run is okay.

bErr_DctAlm: Drive error, "Channel temperature too high"; the drive is swinging backwards and forwards.

bErr_SmAlm: Drive error, "Smoke alarm".

6.2.24 MP_TEM_Configuration



This function block is used to configure the Thermal Energy Meter type 22PE-.. and 22PEM-... . For more information please visit www.belimo.ch.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. **bError** is used to indicate an error in communication with the actuator. The type of the error can be read with **iErrorId**.

VAR_INPUT

```
MP_Address                 : USINT := 1;
bStart                     : BOOL;
bRead_Write                : BOOL;
strDataKL6771              : DataKL6771;
nSensor1Type_Write         : E_MP_EV_V4_Sensor1Type;
bSelectMeterRegister_Write : BOOL;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block.

bRead_Write: if FALSE then READ only; if TRUE then READ and WRITE.

strDataKL6771: the data structure with which the KL6771() [▸ 26] function block must be linked (see DataKL6771 [▸ 84]).

nSensor1Type_Write: sensor 1 type (see E_MP_EV_V4_Sensor1Type [▸ 78]).

bSelectMeterRegister_Write: FALSE = certified meter; TRUE = lifetime meter.

VAR_OUTPUT

```
bBusy                       : BOOL
bError                      : BOOL
iErrorId                    : MP_Error
st_MalfunctionService       : St_MP_EV_V4_MalfunctionServiceInfo
nSensor1Type                : E_MP_EV_V4_Sensor1Type
rVnom                       : LREAL
arSerialNumber              : ARRAY[0..1] OF DWORD
bSelectMeterRegister        : BOOL
```

bBusy: this bit is set for as long as the function block is active.

bError: the output becomes TRUE as soon as an error occurs. This error is described via the *iErrorId* variable.

iErrorId: this output outputs an error code in the event of an error (see MP_ERROR [▸ 81]). *bError* goes TRUE at the same time.

st_MalfunctionService: malfunction and service information (see St_MP_EV_V4_MalfunctionServiceInfo [▸ 85]).

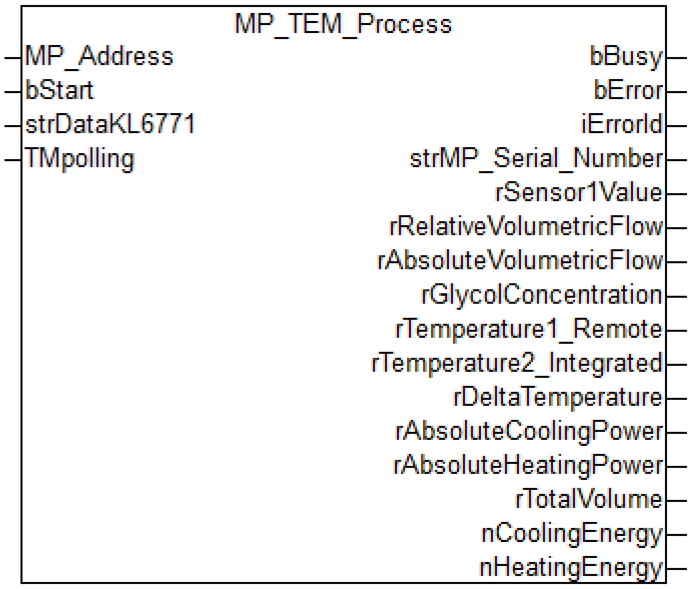
nSensor1Type: sensor 1 type (see E_MP_EV_V4_Sensor1Type [▸ 78]).

rVnom: nominal volume flow rate in l/s (0...100).

arSerialNumber: serial number of the device.

bSelectMeterRegister: FALSE = certified meter; TRUE = lifetime meter.

6.2.25 MP_TEM_Process



This function block is is suitable for the Thermal Energy Meter type 22PE-.. and 22PEM-... . For more information please visit www.belimo.ch.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. If **bStart** remains TRUE, the device is addressed cyclically with a period specified by the time in **TMPolling**. The time should be set longer than 1 s. **bError** is used to indicate an error in communication with the actuator. The type of the error can be read with **iErrorId**.

VAR_INPUT

```
MP_Address                 : USINT := 1;
bStart                     : BOOL;
strDataKL6771              : DataKL6771;
TMpolling                  : TIME:= t#10s;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block. If this remains continuously TRUE, the function block will be activated cyclically with a period specified by the time in *TMPolling*.

strDataKL6771: the data structure with which the KL6771 [▸ 26] function block must be linked (see DataKL6771 [▸ 84]).

TMpolling: the time for which the function block should address the actuator. Default 10 s, minimum time 1 s.

VAR_OUTPUT

```
bBusy                      : BOOL;
bError                     : BOOL;
iErrorId                   : MP_Error;
strMP_Serial_Number        : MP_Serial_Number;
rSensor1Value              : LREAL;
rRelativeVolumetricFlow    : LREAL;
rAbsoluteVolumetricFlow    : LREAL;
rGlycolConcentration       : LREAL;
rTemperature1_Remote       : LREAL;
rTemperature2_Integrated   : LREAL;
rDeltaTemperature          : LREAL;
rAbsoluteCoolingPower      : LREAL;
rAbsoluteHeatingPower      : LREAL;
rTotalVolume               : LREAL;
nCoolingEnergy             : DINT;
nHeatingEnergy             : DINT;
```

bBusy: this bit is set for as long as the function block is active.

bError: the output becomes TRUE as soon as an error occurs. This error is described via the *iErrorId* variable.

iErrorId: this output outputs an error code in the event of an error (see MP_ERROR [▸ 81]). *bError* goes TRUE at the same time.

strMP_Serial_Number: structure for the serial number. (see MP_Serial_Number [▸ 84]).

rSensor1Value: sensor 1 value in mV/Ohm (0...65535).

rRelativeVolumetricFlow: relative volume flow rate in % (0...100).

rAbsoluteVolumetricFlow: absolute volume flow rate in l/s (0...100).

rGlycolConcentration: glycol concentration in % (0...100).

rTemperature1_Remote: temperature1 remote in °C (-20...12).

rTemperature2_Integrated: temperature2 integrated in °C (-20...12).

rDeltaTemperature: temperature delta in K (0...14).

rAbsoluteCoolingPower: absolute cooling power in kW (0...21.5).

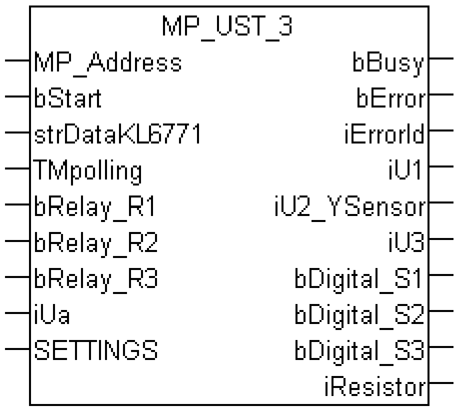
rAbsoluteHeatingPower: absolute heating power in kW (0...21.5).

rTotalVolume: total volume in m³ (0...214748.36).

nCoolingEnergy: cooling energy in kWh (0...21474836).

nHeatingEnergy: heating energy in kWh (0...21474836).

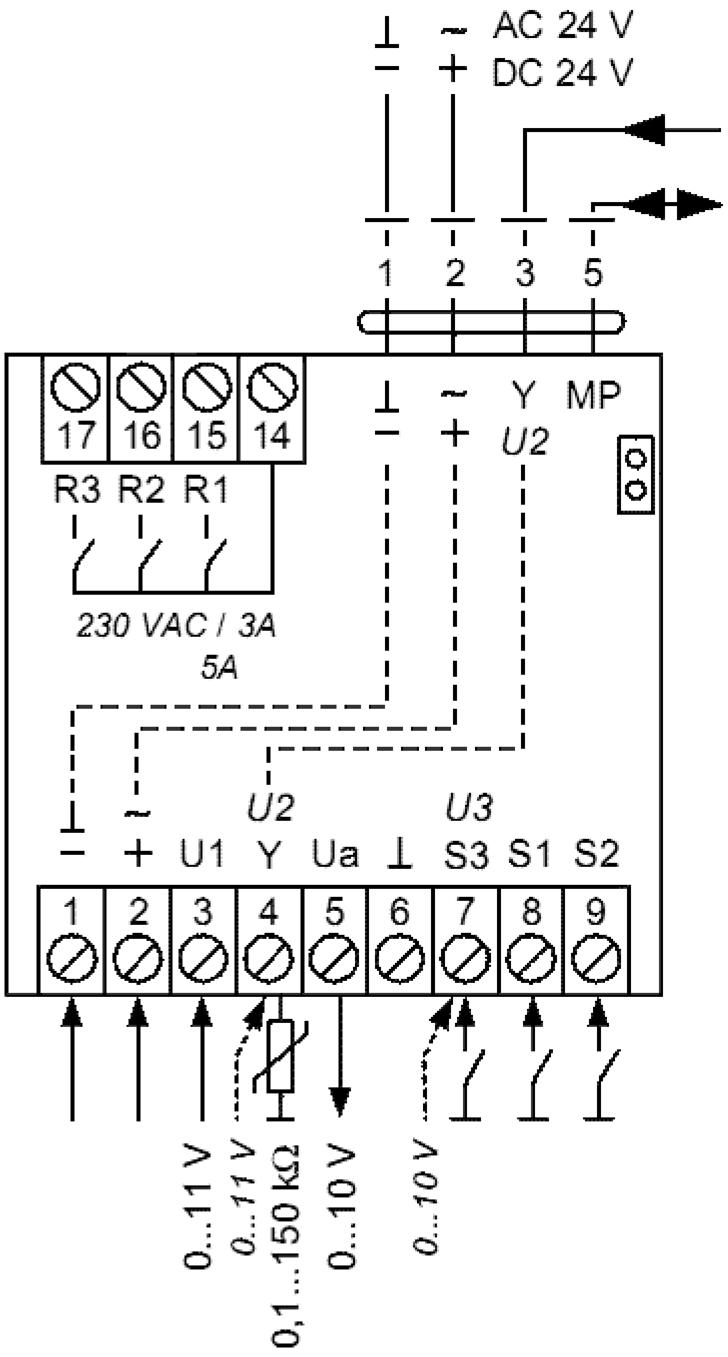
6.2.26 MP_UST_3



This function block is used to control and monitor a multi-I/O module UST3.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. If **bStart** remains TRUE, the device is addressed cyclically with a period specified by the time in **TMPolling**. The time must be set greater than one second. **bError** indicates an error in communication with the actuator. The type of the error can be read with **iErrorID**.

https://infosys.beckhoff.com/content/1033/tcplclibmpbus/Resources/12063688459/.gif



The input data **bRelay_R1** to **bRelay_R3** switch the relays R1 to R3 (PIN 15 to PIN 17). The variable **iUa** switches the analog output 0...10 V to PIN 5. One digit corresponds to 1 mV.

The data structure **SETTINGS** is used for parameterization of the UST3. The scaling of the analog input data can be set, and the resistance measurement can be enabled on PIN4. For resistance measurement, the scaling of the resistance reading can be changed. This can also be done during operation. **iU1** is the analog input on PIN 3. One digit corresponds to one mV or, if the scaling in the data structure **SETTINGS** has been changed, 250 µV. The same applies to the analog inputs **iU2_YSensor** (PIN 4) and **iU3** (PIN 7).**iU2_YSensor** can also be used for resistance measurement. This must be set via the data structure **SETTINGS** **bDigital_S1** to **bDigital_S3** correspond to the digital inputs of the UST3, PIN 7 to PIN 9.

All data is automatically polled by the KL6771 MP-Bus master terminal. The polling speed depends on the number of connected MP-Bus devices and the set polling time. The digital inputs are **unsuitable** for connecting push buttons or sensors, which only issue short pulses. In order to be able to register a change in signal level reliably, it must be present for at least one second.

VAR_INPUT

```
MP_Address      : USINT := 1;
bStart          : BOOL;
strDataKL6771   : DataKL6771;
TMpolling       : TIME := t#10s;
bRelay_R1       : BOOL;
bRelay_R2       : BOOL;
bRelay_R3       : BOOL;
iUa             : UINT;
SETTINGS        : UST3_SET;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block. If this remains continuously TRUE, the function block will be activated cyclically with a period specified by the time in *TMPolling*.

strDataKL6771: the data structure with which the KL6771 [▸ 26] function block must be linked (see DataKL6771 [▸ 84]).

TMpolling: the time for which the function block should address the actuator. Default 10 s, minimum time 1 s.

bRelay_R1: relay PIN 15.

bRelay_R2: relay PIN 16.

bRelay_R3: relay PIN 17.

iUa: analog output PIN 5 (1 mV = 1 digit).

SETTINGS: data structure for setting the scaling and the resistance measurement (see UST3_SET [▸ 86]).

VAR_OUTPUT

```
bBusy           : BOOL;
bError          : BOOL;
iErrorId        : MP_Error;
iU1             : INT;
iU2_YSensor     : INT;
iU3             : INT;
bDigital_S1     : BOOL;
bDigital_S2     : BOOL;
bDigital_S3     : BOOL;
iResistor       : INT;
```

bBusy: This bit is set for as long as the block is active.

bError: The output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorId*.

iErrorId: The output issues an error code when an error occurs (see MP_ERROR [▸ 81]). Simultaneously *bError* is TRUE.

iU1: Analog input PIN 3 (1 digit = 1 mV or 1 digit = 250 µV).

iU2_YSensor: Analog input PIN 4 (1 digit = 1 mV or 1 digit = 250 µV).

iU3: Analog input PIN 7 (1 digit = 1 mV or 1 digit = 250 µV).

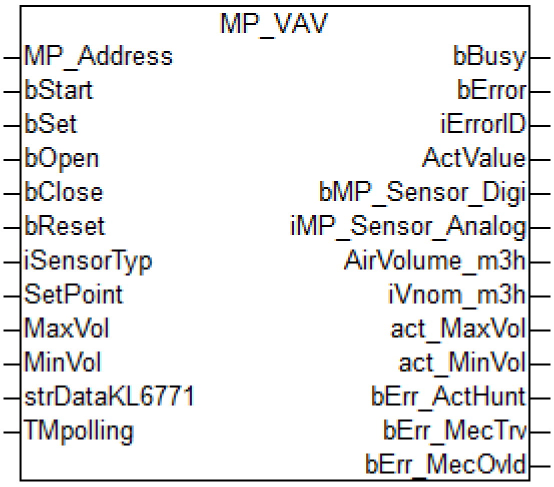
bDigital_S1: Digital input PIN 8.

bDigital_S2: Digital input PIN 9.

bDigital_S3: Digital input PIN 7.

iResistor: Resistor value PIN 4.

6.2.27 MP_VAV



This function block is used to control and monitor a volume flow controller.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. If **bStart** remains TRUE, the device is addressed cyclically with a period specified by the time in **TMPolling**. The time should be set longer than 1 s. **bError** is used to indicate an error in communication with the actuator. The type of the error can be read with **iErrorID**.

A positive edge at **bReset** clears any pending error messages from the actuator. This resets errors that affect the output variables **bErr_MecOcld**, **bErr_ActHunt** and **bErrMecTrv**. If the error itself is still present, the actuator will set these error bits again.

SetPoint is used to adjust the volume flow rate from 0...100 %. The current position of the damper can be read through **ActValue**.

If a sensor is connected to the actuator, its type is to be indicated in **iSensorTyp**. If no sensor is connected, the value "0" should be entered, or the variable left open. A digital sensor should be parameterized with "1". The state of the sensor can be queried through **bMP_Sensor_Digi**. Analog sensors "2...6" are output in variable **iMP_Sensor_Analog**.

A positive edge at the **bOpen** or **bClose** inputs opens or closes the damper of the actuator. A negative edge at these two inputs clears the command again.

MaxVol and **MinVol** can be used to store a maximum and minimum volume flow rate in the actuator. A positive edge at **bSet** writes the data to the actuator. You can obtain the current value from the output data **act_MaxVol** and **act_MinVol**. The current volume flow rate is output in the **AirVolume_m3h** variable.

VAR_INPUT

```
MP_Address      : USINT := 1;
bStart          : BOOL;
bSet            : BOOL;
bOpen           : BOOL;
bClose          : BOOL;
bReset          : BOOL;
iSensorTyp      : INT;
SetPoint        : USINT;
MaxVol          : WORD;
MinVol          : WORD;
strDataKL6771   : DataKL6771;
TMpolling       : TIME := t#10s;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block. If this remains continuously TRUE, the function block will be activated cyclically with a period specified by the time in *TMPolling*.

bSet: a positive edge writes the *MaxVol* and *MinVol* data to the actuator.

bOpen: a positive edge opens the dampers of the actuator, while a negative edge cancels the forced ventilation.

bClose: a positive edge closes the dampers of the actuator, while a negative edge cancels the forced closure.

bReset: a positive edge resets the actuator's error messages.

iSensorTyp: 0: no sensor connected, 1: digital sensor connected, 2: analog sensor connected (0...35 V), 3...6: output of a resistance in ohms (3..5 applies to PT1000, NI1000 and NI1000LuS; 6 applies to NTC). To convert to a temperature, use the corresponding conversion functions.

SetPoint: set volume flow rate (0...100 %).

MaxVol: maximum volume flow (30...100 %).

MinVol: minimum volume flow (0...100 %).

strDataKL6771: the data structure with which the KL6771 [▶ 26] function block must be linked (see DataKL6771 [▶ 84]).

TMpolling: the time for which the function block should address the actuator. Default 10 s, minimum time 1 s.

VAR_OUTPUT

```
bBusy               : BOOL;
bError              : BOOL;
iErrorID            : MP_Error;
ActValue            : WORD;
bMP_Sensor_Digi     : BOOL;
iMP_Sensor_Analog   : INT;
AirVolume_m3h       : WORD;
iVnom_m3h           : INT;
act_MaxVol          : INT;
act_MinVol          : INT;
bErr_ActHunt        : BOOL;
bErr_MecTrv         : BOOL;
bErr_MecOvld        : BOOL;
```

bBusy: this bit is set for as long as the function block is active.

bError: the output becomes TRUE as soon as an error occurs. This error is described via the *iErrorID* variable.

iErrorID: this output outputs an error code in the event of an error (see MP_Error [▶ 81]). *bError* goes TRUE at the same time.

ActValue: contains the current position of the actuator (0...100 %).

bMP_Sensor_Digi: if an analog sensor is connected, its value is indicated through this variable. *iSensorTyp* must be "1".

iMP_Sensor_Analog: if an analog sensor is connected, its value is indicated through this variable. *iSensorTyp* must be in the range "2...6".

AirVolume_m3h: output of the volume flow rate in m^3/h.

iVnom_m3h: nominal air volume flow in m3/h. This output is available from version 1.12.0. VAV is read and must be > 0. If 0, then the calculation of *AirVolume_m3h* is not correct.

act_MaxVol: maximum set volume flow rate in %.

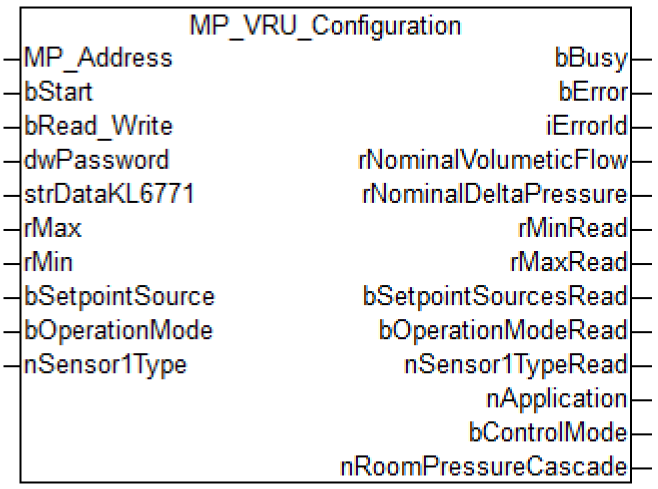
act_MinVol: minimum set volume flow rate in %.

bErr_ActHunt: actuator error, "Regulating oscillation"; the actuator is swinging backwards and forwards.

bErr_MecTrv: actuator error, "Positioning angle exceeded"; the actuator has passed more than 10° beyond the adaptation position.

bErr_MecOvld: actuator error, "Overload"; the set position could not be reached.

6.2.28 MP_VRU_Configuration



This function block is used to configure the VAV actuators VRU-D3-BAC, VRU-M1-BAC and VRU-M1R-BAC (max. 8 slaves). Further information can be found at www.belimo.ch.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the block is active. **bError** is used to indicate an error in communication with the drive. The type of the error can be read with **iErrorId**.

VAR_INPUT

```
MP_Address          : USINT := 1;
bStart              : BOOL;
bRead_Write         : BOOL;
dwPassword          : DWORD;
strDataKL6771       : DataKL6771;
rMax                : LREAL := 100;
rMin                : LREAL;
bSetpointSource     : BOOL;
bOperationMode      : BOOL  := TRUE;
nSensor1Type        : E_MP_VRU_Sensor1Type  := MPBus_VRU_Sensor_Active;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block.

bRead_Write: FALSE = READ only. TRUE = READ and WRITE.

dwPassword: the password for the actuators. Usually 0x0000.

strDataKL6771: the data structure with which the KL6771 [▸ 26] function block must be linked (see DataKL6771 [▸ 84]).

rMax: max in % (20...100 %).

rMin: min in % (0...*rMax*). *rMin* must be smaller than *rMax*.

bSetpointSource: TRUE = bus; FALSE = analog

bOperationMode: TRUE = overpressure; FALSE = negative pressure

nSensor1Type: sensor 1 type [▸ 80].

VAR_OUTPUT

```
bBusy                 : BOOL;
bError                : BOOL;
iErrorId              : MP_Error;
rNominalVolumeticFlow : LREAL;
rNominalDeltaPressure : LREAL;
rMinRead              : LREAL;
rMaxRead              : LREAL;
```

```
bSetpointSourcesRead  : BOOL;
bOperationModeRead    : BOOL;
nSensor1TypeRead      : E_MP_VRU_Sensor1Type;
nApplication          : E_MP_VRU_Application;
bControlMode          : BOOL;
nRoomPressureCascade  : E_MP_VRU_RoomPressureCascade;
```

bBusy: this bit is set for as long as the function block is active.

bError: the output becomes TRUE as soon as an error occurs. This error is described via the *iErrorId* variable.

iErrorId: this output outputs an error code in the event of an error (see MP_ERROR [▸ 81]). *bError* goes TRUE at the same time.

rNominalVolumeticFlow: nominal volume flow in m³/h (0...60.000).

rNominalDeltaPressure: nominal differential pressure in Pa (0...10.000).

rMinRead: min in % (0...*rMax*). *rMin* must be smaller than *rMax*.

rMaxRead: max in % (20...100 %).

bSetpointSourcesRead: TRUE = bus; FALSE = analog

bOperationModeRead: TRUE = overpressure; FALSE = negative pressure

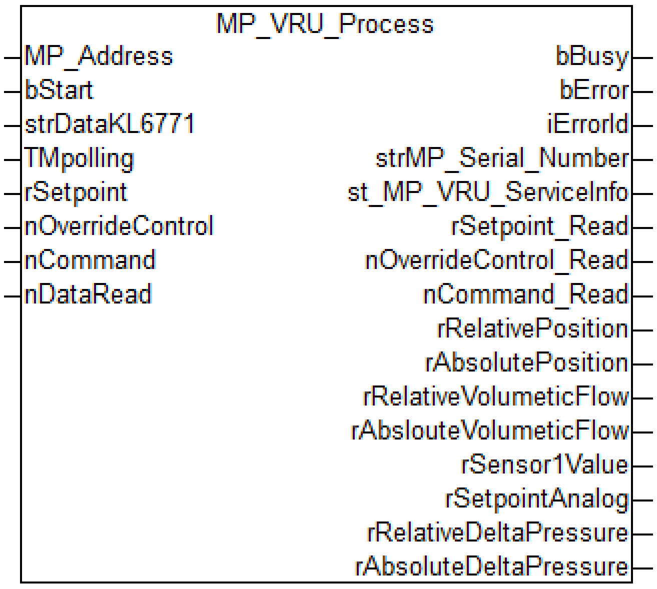
nSensor1TypeRead: sensor 1 type [▸ 80].

nApplication: visualization of the application [▸ 78] selected by the manufacturer.

bControlMode: visualization of the control function selected by the manufacturer. TRUE = volume flow rate control; FALSE = position control

nRoomPressureCascade: room pressure Cascade control [▸ 79].

6.2.29 MP_VRU_Process



This function block is suitable for VAV actuators VRU-D3-BAC, VRU-M1-BAC and VRU-M1R-BAC (max. 8 slaves). For more information please visit www.belimo.ch.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. If **bStart** remains TRUE, the device is addressed cyclically with a period specified by the time in **TMPolling**. The time should be set longer than 1 s. **bError** is used to indicate an error in communication with the actuator. The type of the error can be read with **iErrorId**.

VAR_INPUT

```
MP_Address          : USINT := 1;
bStart              : BOOL;
strDataKL6771       : DataKL6771;
TMpolling           : TIME := t#10s;
rSetpoint           : LREAL;
nOverrideControl    : E_MP_VRU_OverrideControl;
nCommand            : E_MP_VRU_Command;
nDataRead           : BYTE;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block. If this remains continuously TRUE, the function block will be activated cyclically with a period specified by the time in *TMPolling*.

strDataKL6771: the data structure with which the KL6771 [▶ 26] function block must be linked (see DataKL6771 [▶ 84]).

TMpolling: the time for which the function block should address the actuator. Default 10 s, minimum time 1 s.

rSetpoint: value in % (0...100 %).

nOverrideControl: override the setpoint [▶ 79].

nCommand: command [▶ 79] for service and test functions of the actuator.

nDataRead: 0xFF - read all data; bit 0 - read relative position; bit 1 - read absolute position; bit 2 - read relative volume flow rate; bit 3 - read absolute volume flow rate; bit 4 - read value sensor 1; bit 5 - read analog setpoint; bit 6 - read relative differential pressure; bit 7 - read absolute differential pressure

VAR_OUTPUT

```
bBusy                    : BOOL;
bError                   : BOOL;
iErrorId                 : MP_Error;
strMP_Serial_Number      : MP_Serial_Number;
st_MP_VRU_ServiceInfo    : St_MP_VRU_ServiceInfo;
rSetpoint_Read           : LREAL;
nOverrideControl_Read    : E_MP_VRU_OverrideControl;
nCommand_Read            : E_MP_VRU_Command;
rRelativePosition        : LREAL;
rAbsolutePosition        : LREAL;
rRelativeVolumeticFlow : LREAL;
rAbslouteVolumeticFlow : LREAL;
rSensor1Value            : LREAL;
rSetpointAnalog          : LREAL;
rRelativeDeltaPressure : LREAL;
rAbsoluteDeltaPressure : LREAL;
```

bBusy: This bit is set for as long as the block is active.

bError: The output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorId*.

iErrorId: The output issues an error code when an error occurs (see MP_ERROR [▶ 81]). Simultaneously *bError* is TRUE.

strMP_Serial_Number: Structure [▶ 84] for serial number.

st_MP_VRU_ServiceInfo: Malfunction and Service Information [▶ 85].

rSetpoint_Read: Setpoint.

nOverrideControl_Read: Override Control [▶ 79].

nCommand_Read: Command [▶ 79].

rRelativePosition: Relative Position in %. Value of -1 means data are disabled (see VAR_INPUT *nDataRead*).

rAbsolutePosition: Absolute Position in °. Value of -1 means data are disabled (see VAR_INPUT *nDataRead*).

rRelativeVolumeticFlow: Relative Volumetic Flow in %. Value of -1 means data are disabled (see VAR_INPUT *nDataRead*).

rAbslouteVolumeticFlow: Absloute Volumetic Flow in m³/h. Value of -1 means data are disabled (see VAR_INPUT *nDataRead*).

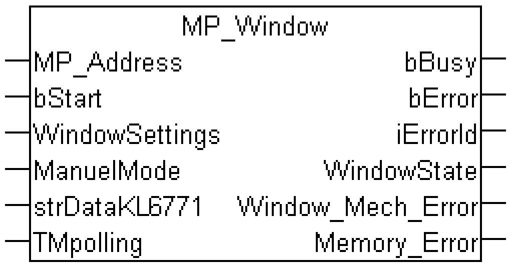
rSensor1Value: Sensor 1 Value in mV/Ohm. Value of -1 means data are disabled (see VAR_INPUT *nDataRead*).

rSetpointAnalog: Setpoint Analog in %. Value of -1 means data are disabled (see VAR_INPUT *nDataRead*).

rRelativeDeltaPressure: Relative Delta Pressure in %. Value of -1 means data are disabled (see VAR_INPUT *nDataRead*).

rAbsoluteDeltaPressure: Absolute Delta Pressure in Pa. Value of -1 means data are disabled (see VAR_INPUT *nDataRead*).

6.2.30 MP_Window



This function block is used to control and monitor a window ventilation system (FLS).

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the block is active. If **bStart** remains TRUE, the device is addressed cyclically with a period specified by the time in **TMPolling**. The time should be set longer than 1 second. **bError** is used to indicate an error in communication with the FLS. The type of the error can be read with **iErrorId**.

ManuelMode can be used to activate or deactivate manual operation. The type of ventilation can be specified through **WindowSettings**.

VAR_INPUT

```
MP_Address          : USINT := 1;
bStart              : BOOL;
WindowSettings      : Data_Window;
ManuelMode          : BOOL;
strDataKL6771       : DataKL6771;
TMpolling           : TIME := t#10s;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block. If this remains continuously TRUE, the function block will be activated cyclically with a period specified by the time in *TMPolling*.

WindowSettings: nominal ventilation settings (see Data_Window [▸ 73]).

ManuelMode: FALSE: manual operation allowed. TRUE: manual operation disabled.

strDataKL6771: the data structure with which the KL6771 [▸ 26] function block must be linked (see DataKL6771 [▸ 84]).

TMpolling: the time for which the function block should address the actuator. Default 10 s, minimum time 1 s.

VAR_OUTPUT

```
bBusy             : BOOL;
bError            : BOOL;
iErrorId          : MP_Error;
WindowState       : Data_Window;
Window_Mech_Error : BOOL;
Memory_Error      : BOOL;
```

bBusy: This bit is set for as long as the block is active.

bError: The output becomes TRUE as soon as an error occurs. The error is described via the variable *iErrorId*.

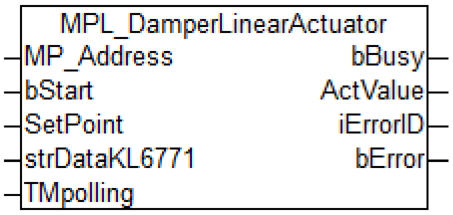
iErrorId: The output issues an error code when an error occurs (see MP_ERROR [▸ 81]). Simultaneously *bError* is TRUE.

WindowState: Current value of the parameters (see Data_Window [▸ 73]).

Window_Mech_Error: The window or drive is blocked.

Memory_Error: The drive has a memory error. Reprogram or replace.

6.2.31 MPL_DamperLinearActuator



This function block is used to control and monitor an actuator of a damper and of a globe valve.

MP_Address is used to specify the MP-Bus device with which the function block is to communicate. **bStart** activates communication with the MP-Bus device. **bBusy** indicates that the function block is active. If **bStart** remains TRUE, the device is addressed cyclically with a period specified by the time in **TMPolling**. The time should be set longer than 1 s. **bError** is used to indicate an error in communication with the actuator. The type of the error can be read with **iErrorID**.

SetPoint is used to adjust the position of the damper from 0...100 %. The current position of the actuator can be read through **ActValue**.

VAR_INPUT

```
MP_Address        : USINT := 1;
bStart            : BOOL;
SetPoint          : USINT;
strDataKL6771     : DataKL6771;
TMpolling         : TIME  := t#10s;
```

MP_Address: MP-Bus address of the slave.

bStart: a positive edge starts the function block. If this remains continuously TRUE, the function block will be activated cyclically with a period specified by the time in *TMPolling*.

SetPoint: 0...100 % the set damper position specified for the actuator.

strDataKL6771: the data structure with which the KL6771 [▸ 26] function block must be linked (see DataKL6771 [▸ 84]).

TMpolling: the time for which the function block should address the actuator. Default 10 s, minimum time 1 s.

VAR_OUTPUT

```
bBusy          : BOOL;
ActValue       : WORD;
iErrorID       : MP_Error;
bError         : BOOL;
```

bBusy: this bit is set for as long as the function block is active.

ActValue: contains the current position of the actuator (0...100 %).

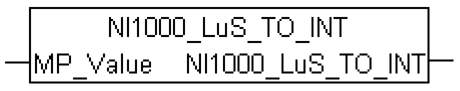
iErrorID: this output outputs an error code in the event of an error (see MP_Error [▶ 81]). *bError* goes TRUE at the same time.

bError: the output becomes TRUE as soon as an error occurs. This error is described via the *iErrorID* variable.

6.3 Functions

POUs	Description
NI1000_LuS_TO_INT [▶ 71]	This function calculates a temperature from the value of an NI1000 L&S resistor.
NI1000_TO_INT [▶ 71]	This function calculates a temperature from the value of an NI1000 resistor.
NTC_TO_INT [▶ 72]	This function calculates a temperature from the value of an NTC resistor.
PT1000_TO_INT [▶ 72]	This function calculates a temperature from the value of an PT1000 resistor.

6.3.1 NI1000_LuS_TO_INT : INT



This function calculates a temperature from the value of an NI1000 L&S resistor.

Connect this function to **iMP_Sensor_Analog**. As output, you receive an INT variable that represents the temperature with a resolution of 0.01 °C (20.5 °C, for example, is represented as 2050).

The lowest valid value of 872 ohms corresponds to -30 °C. If the value is smaller than this, 16#7FFD is output.

The largest valid value of 1586 ohms corresponds to 115 °C. If the value is greater than this, 16#7FFE is output.

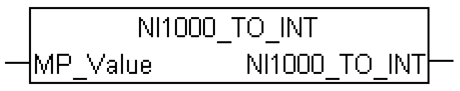
VAR_INPUT

```
MP_Value      : WORD;
```

MP_Value: input for an ohmic NI1000 L&S sensor.

NI1000_LuS_TO_INT: temperature in 0.01 °C.

6.3.2 NI1000_TO_INT : INT



This function calculates a temperature from the value of an NI1000 resistor.

Connect this function to **iMP_Sensor_Analog**. As output, you receive an INT variable that represents the temperature with a resolution of 0.01 °C (20.5 °C, for example, is represented as 2050).

The lowest valid value of 867 ohms corresponds to -25 °C. If the value is smaller than this, 16#7FFD is output.

The largest valid value of 1583 ohms corresponds to 95 °C. If the value is greater than this, 16#7FFE is output.

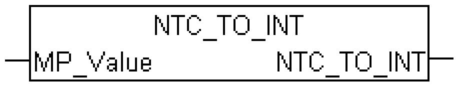
VAR_INPUT

```
MP_Value       : WORD;
```

MP_Value: input for an ohmic NI1000 sensor

NI1000_TO_INT: temperature in 0.01 °C.

6.3.3 NTC_TO_INT : INT



This function calculates a temperature from the value of an NTC resistor.

Connect this function to **iMP_Sensor_Analog**. As output, you receive an INT variable that represents the temperature with a resolution of 0.01 °C (20.5 °C, for example, is represented as 2050).

The lowest valid value of 104 ohms corresponds to 145 °C. If the value is smaller than this, 16#7FFD is output.

The largest valid value of 48555 ohms corresponds to -20 °C. If the value is greater than this, 16#7FFE is output.

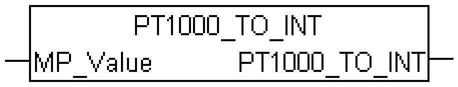
VAR_INPUT

```
MP_Value       : WORD;
```

MP_Value: input for an ohmic NTC sensor.

NTC_TO_INT: temperature in 0.01 °C.

6.3.4 PT1000_TO_INT : INT



This function calculates a temperature from the value of a PT1000 resistor.

Connect this function to **iMP_Sensor_Analog**. As output, you receive an INT variable that represents the temperature with a resolution of 0.01 °C (20.5 °C, for example, is represented as 2050).

The lowest valid value of 862 ohms corresponds to -35 °C. If the value is smaller than this, 16#7FFD is output.

The largest valid value of 1592 ohms corresponds to 155 °C. If the value is greater than this, 16#7FFE is output.

VAR_INPUT

```
MP_Value       : WORD;
```

MP_Value: input for an ohmic PT1000 sensor.

PT1000_TO_INT: temperature in 0.01 °C.

6.4 Data types

6.4.1 Enums

6.4.1.1 Data_Window

This ENUM can be used to specify the ventilation method.

```
TYPE Data_Window :
(
  Window_Close   := 8,
  Window_Unlock  := 9,
  Window_Open    := 16#0A,
  Window_20      := 16#0B,
  Window_40      := 16#0C,
  Window_60      := 16#0D,
  Window_80      := 16#0E,
  Window_100     := 16#0F,
  Auto_Close     := 1,
  Auto_5_15min   := 2,
  Auto_8_30min   := 3,
  Auto_10_50min  := 4,
  Auto_open      := 5,
  Auto           := 0,
)
END_TYPE
```

Window_Close: closes the window.

Window_Unlock: if you use "Unlock", use the switch in the window to restart the MP-Bus communication.

Window_Open: opens the window.

Window_20: opens the window by 20 %.

Window_40: opens the window by 40 %.

Window_60: opens the window by 60 %.

Window_80: opens the window by 80 %.

Window_100: opens the window by 100 %.

Auto_Close: automatic closing.

Auto_5_15min: automatic ventilation every 5...15 minutes.

Auto_8_30min: automatic ventilation every 8...30 minutes.

Auto_10_50min: automatic ventilation every 10...50 minutes.

Auto_open: automatic ventilation open.

Auto: automatic mode.

6.4.1.2 E_MP_AirQualityStatus

Status of the measured air quality.

```
TYPE E_MP_AirQualityStatus :
(
  MPBus_AirQualityStatus_Deactivated := 0,
  MPBus_AirQualityStatus_OK          := 1,
  MPBus_AirQualityStatus_Warning     := 2,
  MPBus_AirQualityStatus_Alarm       := 3
);
END_TYPE
```

MPBus_AirQualityStatus_Deactivated: deactivated.

MPBus_AirQualityStatus_OK: OK.

MPBus_AirQualityStatus_Warning: warning.

MPBus_AirQualityStatus_Alarm: alarm.

6.4.1.3 E_MP_DisplayBackground

Background color of the display.

```
TYPE E_MP_DisplayBackground :
(
  MPBus_DisplayBackground_WhiteOnBlack := 0,
  MPBus_DisplayBackground_BlackOnWhite := 1
);
END_TYPE
```

MPBus_DisplayBackground_WhiteOnBlack: white on black.

MPBus_DisplayBackground_BlackOnWhite: black on white.

6.4.1.4 E_MP_DisplayModeButton

Display mode of the buttons.

```
TYPE E_MP_DisplayModeButton :
(
  MPBus_DisplayModeButton_Invisible := 0,
  MPBus_DisplayModeButton_Status    := 1,
  MPBus_DisplayModeButton_Setpoint  := 2
);
END_TYPE
```

MPBus_DisplayModeButton_Invisible: invisible.

MPBus_DisplayModeButton_Status: status.

MPBus_DisplayModeButton_Setpoint: setpoint.

6.4.1.5 E_MP_DisplayModeHeatingCooling

Display mode of heating or cooling icons.

```
TYPE E_MP_DisplayModeHeatingCooling :
(
  MPBus_DisplayModeHeatingCooling_None    := 0,
  MPBus_DisplayModeHeatingCooling_Heating := 1,
  MPBus_DisplayModeHeatingCooling_Cooling := 2
);
END_TYPE
```

MPBus_DisplayModeHeatingCooling_None: none.

MPBus_DisplayModeHeatingCooling_Heating: heating.

MPBus_DisplayModeHeatingCooling_Cooling: cooling.

6.4.1.6 E_MP_DisplayModeIconWarning

Display mode of the warning icon.

```
TYPE E_MP_DisplayModeIconWarning :
(
  MPBus_DisplayModeIconWarning_Invisible        := 0,
  MPBus_DisplayModeIconWarning_Icon             := 1,
  MPBus_DisplayModeIconWarning_DeviceErrorState := 2
);
END_TYPE
```

MPBus_DisplayModeIconWarning_Invisible: invisible.

MPBus_DisplayModeIconWarning_Icon: icon.

MPBus_DisplayModeIconWarning_DeviceErrorState: error state of the device.

6.4.1.7 E_MP_DisplayModeIconWindow

Display mode of the window icon.

```
TYPE E_MP_DisplayModeIconWindow :
(
  MPBus_DisplayModeIconWindow_Invisible := 0,
  MPBus_DisplayModeIconWindow_Icon      := 1,
  MPBus_DisplayModeIconWindow_Reserve   := 2
);
END_TYPE
```

MPBus_DisplayModeIconWindow_Invisible: invisible.

MPBus_DisplayModeIconWindow_Icon: icon.

MPBus_DisplayModeIconWindow_Reserve: reserve.

6.4.1.8 E_MP_DisplayModeTemp

Display mode temperature.

```
TYPE E_MP_DisplayModeTemp :
(
  MPBus_DisplayModeTemp_Invisible        := 0,
  MPBus_DisplayModeTemp_ActualRoomTemp   := 1,
  MPBus_DisplayModeTemp_RoomTempSetpoint := 2
);
END_TYPE
```

MPBus_DisplayModeTemp_Invisible: invisible.

MPBus_DisplayModeTemp_ActualRoomTemp: actual room temperature.

MPBus_DisplayModeTemp_RoomTempSetpoint: setpoint of the room temperature.

6.4.1.9 E_MP_DisplayModeTempUnit

Display mode of the unit for the temperature.

```
TYPE E_MP_DisplayModeTempUnit :
(
  MPBus_DisplayModeTempUnit_C       := 0,
  MPBus_DisplayModeTempUnit_Reserve := 1,
  MPBus_DisplayModeTempUnit_F       := 2
);
END_TYPE
```

MPBus_DisplayModeTempUnit_C: °C.

MPBus_DisplayModeTempUnit_Reserve: reserve.

MPBus_DisplayModeTempUnit_F: °F.

6.4.1.10 E_MP_DisplayModeVentilationStage

Number of adjustable ventilation stages shown on the display.

```
TYPE E_MP_DisplayModeVentilationStage :
(
  MPBus_DisplayModeVentilationStage_Reserve1 := 0,
  MPBus_DisplayModeVentilationStage_Reserve2 := 1,
  MPBus_DisplayModeVentilationStage_3        := 2,
  MPBus_DisplayModeVentilationStage_4        := 3,
  MPBus_DisplayModeVentilationStage_7        := 4

);
END_TYPE
```

MPBus_DisplayModeVentilationStage_Reserve1: reserve 1.

MPBus_DisplayModeVentilationStage_Reserve2: reserve 2.

MPBus_DisplayModeVentilationStage_3: 3 ventilation stages.

MPBus_DisplayModeVentilationStage_4: 4 ventilation stages.

MPBus_DisplayModeVentilationStage_7: 7 ventilation stages.

6.4.1.11 E_MP_DisplayVisibility

Visibility on the display.

```
TYPE E_MP_DisplayVisibility :
(
  MPBus_DisplayVisibility_Invisible := 0,
  MPBus_DisplayVisibility_Visible   := 1

);
END_TYPE
```

MPBus_DisplayVisibility_Invisible: invisible.

MPBus_DisplayVisibility_Visible: visible.

6.4.1.12 E_MP_EnabledStatus

Enabled status.

```
TYPE E_MP_EnabledStatus :
(
  MPBus_EnabledStatus_Disabled := 0,
  MPBus_EnabledStatus_Enabled  := 1
);
END_TYPE
```

MPBus_EnabledStatus_Disabled: disabled.

MPBus_EnabledStatus_Enabled: enabled.

6.4.1.13 E_MP_EP_R_R6_UnitSel

Scaling.

```
TYPE E_MP_EP_R_R6_UnitSel :
(
  E_MP_m3_s  := 0,
  E_MP_m3_h  := 1,
  E_MP_l_s   := 2,
  E_MP_l_min := 3,
  E_MP_l_h   := 4,
  E_MP_gpm   := 5,
  E_MP_cfm   := 6
);
END_TYPE
```

E_MP_m3_s: Set scaling to m3/s.

E_MP_m3_h: Set scaling to m3/h.

E_MP_l_s: Set scaling to l/s.

E_MP_l_min: Set scaling to l/min.

E_MP_l_h: Set scaling to l/h.

E_MP_gpm: Set scaling to gpm.

E_MP_cfm: Set scaling to cfm.

6.4.1.14 E_MP_EV_V4_Command

Command for service and test functions of the actuator.

```
TYPE E_MP_EV_V4_Command :
(
  MPBus_EV_Command_None := 0,
  MPBus_EV_Command_Sync := 2
);
END_TYPE
```

MPBus_EV_Command_None: none.

MPBus_EV_Command_Sync: sync.

6.4.1.15 E_MP_EV_V4_ControlMode

Control mode.

```
TYPE E_MP_EV_V4_ControlMode :
(
  MPBus_EV_PositionControl := 0,
  MPBus_EV_FlowControl     := 1,
  MPBus_EV_PowerControl    := 2
);
END_TYPE
```

MPBus_EV_PositionControl: position control.

MPBus_EV_FlowControl: flow control.

MPBus_EV_PowerControl: power control.

6.4.1.16 E_MP_EV_V4_DeltaTLimitation

Response to a low delta T.

```
TYPE E_MP_EV_V4_DeltaTLimitation :
(
  MPBus_EV_Disabled               := 0,
  MPBus_EV_DeltaT_Manager         := 1,
  MPBus_EV_DeltaT_Manager_Scaled := 2
);
END_TYPE
```

MPBus_EV_Disabled: disabled.

MPBus_EV_DeltaT_Manager: Delta T Manager.

MPBus_EV_DeltaT_Manager_Scaled: Delta T Manager scaled.

6.4.1.17 E_MP_EV_V4_DeltaTManagerStatus

Status from Delta T Manager.

```
TYPE E_MP_EV_V4_DeltaTManagerStatus :
(
  MPBus_EV_NotSelect := 0,
  MPBus_EV_Standby   := 1,
  MPBus_EV_Active    := 2
);
END_TYPE
```

MPBus_EV_NotSelect: not selected.

MPBus_EV_Standby: standby.

MPBus_EV_Active: active.

6.4.1.18 E_MP_EV_V4_OverrideControl

Setpoint override.

```
TYPE E_MP_EV_V4_OverrideControl :
(
  MPBus_EV_Override_None       := 0,
  MPBus_EV_Override_Open       := 1,
  MPBus_EV_Override_Close      := 2,
  MPBus_EV_Override_MinFlow    := 3,
  MPBus_EV_Override_MaxFlow    := 5,
  MPBus_EV_Override_NomFlow    := 6,
  MPBus_EV_Override_Motor_Stop := 10
);
END_TYPE
```

MPBus_EV_Override_None: none.

MPBus_EV_Override_Open: open.

MPBus_EV_Override_Close: close.

MPBus_EV_Override_MinFlow: minimum flow.

MPBus_EV_Override_MaxFlow: maximum flow.

MPBus_EV_Override_NomFlow: nominal flow.

MPBus_EV_Override_Motor_Stop: motor stop.

6.4.1.19 E_MP_EV_V4_Sensor1Type

External sensor at input S1.

```
TYPE E_MP_EV_V4_Sensor1Type :
(
  MPBus_EV_Sensor_None    := 0,
  MPBus_EV_Sensor_Active  := 1,
  MPBus_EV_Sensor_Passive := 3,
  MPBus_EV_Sensor_Switch  := 4
);
END_TYPE
```

MPBus_EV_Sensor_None: none.

MPBus_EV_Sensor_Active: active.

MPBus_EV_Sensor_Passive: passive.

MPBus_EV_Sensor_Switch: switch.

6.4.1.20 E_MP_SystemOperationMode

Operation mode of the system.

```
TYPE E_MP_SystemOperationMode :
(
  MPBus_SystemOperationMode_OffProtection := 0,
  MPBus_SystemOperationMode_OnComfort     := 1,
  MPBus_SystemOperationMode_Eco           := 2,
  MPBus_SystemOperationMode_Boost         := 3
);
END_TYPE
```

MPBus_SystemOperationMode_OffProtection: off/protection.

MPBus_SystemOperationMode_OnComfort: on/comfort.

MPBus_SystemOperationMode_Eco: eco mode.

MPBus_SystemOperationMode_Boost: boost mode.

6.4.1.21 E_MP_VRU_Application

Visualization of the application selected by the damper manufacturer.

```
TYPE E_MP_VRU_Application :
(
  MPBus_VRU_Application_FlowControl         := 0,
  MPBus_VRU_Application_PressureControl     := 1,
  MPBus_VRU_Application_RoomPressureControl := 2,
  MPBus_VRU_Application_FlowMeasurement     := 4
);
END_TYPE
```

MPBus_VRU_Application_FlowControl: Flow control.

MPBus_VRU_Application_PressureControl: Pressure control.

MPBus_VRU_Application_RoomPressureControl: Room pressure control.

MPBus_VRU_Application_FlowMeasurement: Flow measurement.

6.4.1.22 E_MP_VRU_Command

Commands for service and test functions of the actuator.

```
TYPE E_MP_VRU_Command :
(
  MPBus_VRU_Command_None     := 0,
  MPBus_VRU_Command_Adaption := 1,
  MPBus_VRU_Command_Test     := 2,
  MPBus_VRU_Command_Sync     := 3
);
END_TYPE
```

MPBus_VRU_Command_None: None.

MPBus_VRU_Command_Adaption: Adaption.

MPBus_VRU_Command_Test: Test.

MPBus_VRU_Command_Sync: Sync.

6.4.1.23 E_MP_VRU_OverrideControl

Override the setpoint.

```
TYPE E_MP_VRU_OverrideControl :
(
  MPBus_VRU_Override_None       := 0,
  MPBus_VRU_Override_Open       := 1,
  MPBus_VRU_Override_Close      := 2,
  MPBus_VRU_Override_Max        := 3,
  MPBus_VRU_Override_Min        := 4,
  MPBus_VRU_Override_Reserve    := 5,
  MPBus_VRU_Override_Motor_Stop := 6,
  MPBus_VRU_Override_Vnom_Pnom  := 7
);
END_TYPE
```

MPBus_VRU_Override_None: None.

MPBus_VRU_Override_Open: Open.

MPBus_VRU_Override_Close: Close.

MPBus_VRU_Override_Max: Maximum.

MPBus_VRU_Override_Min: Minimum.

MPBus_VRU_Override_Reserve: Reserve.

MPBus_VRU_Override_Motor_Stop: Motor stop.

MPBus_VRU_Override_Vnom_Pnom: Vnom / Pnom.

6.4.1.24 E_MP_VRU_RoomPressureCascade

Room pressure cascade.

```
TYPE E_MP_VRU_RoomPressureCascade :
(
  MPBus_VRU_RoomPressureCascade_Disabled    := 0,
  MPBus_VRU_RoomPressureCascade_Enabled     := 1,
  MPBus_VRU_RoomPressureCascade_EnabledFast := 2
);
END_TYPE
```

MPBus_VRU_RoomPressureCascade_Disabled: Room pressure cascade disabled.

MPBus_VRU_RoomPressureCascade_Enabled: Room pressure cascade enabled.

MPBus_VRU_RoomPressureCascade_EnabledFast: Room pressure cascade enabled (fast).

6.4.1.25 E_MP_VRU_Sensor1Type

External sensor on input S1.

```
TYPE E_MP_VRU_Sensor1Type :
(
  MPBus_VRU_Sensor_None    := 0,
  MPBus_VRU_Sensor_Active  := 1,
  MPBus_VRU_Sensor_Passive := 2,
  MPBus_VRU_Sensor_Switch  := 4
);
END_TYPE
```

MPBus_VRU_Sensor_None: None.

MPBus_VRU_Sensor_Active: Active.

MPBus_VRU_Sensor_Passive: Passive.

MPBus_VRU_Sensor_Switch: Switch.

6.4.1.26 E_MPBus_ControlMode

Control mode.

```
TYPE E_MPBus_ControlMode :
(
  MPBus_ControlMode_PosCtrl   := 0,
  MPBus_ControlMode_FlowCtrl  := 1,
  MPBus_ControlMode_PowerCtrl := 2,
  MPBus_ControlMode_Disable   := 16#FF
);
END_TYPE
```

MPBus_ControlMode_PosCtrl: Regulated by position.

MPBus_ControlMode_FlowCtrl: Regulated by flow volume.

MPBus_ControlMode_PowerCtrl: Regulated by energy.

MPBus_ControlMode_Disable: Disabled.

6.4.1.27 E_MPBus_DeltaTLimitation

Delta T (dT) limitation. Details are given in the documentation of Belimo Energy Valves.

```
TYPE E_MPBus_DeltaTLimitation :
(
  MPBus_DeltaTLimitation_Disable        := 0,
  MPBus_DeltaTLimitation_dT_Manager     := 1,
  MPBus_DeltaTLimitation_dT_ManagerScal := 2
);
END_TYPE
```

MPBus_DeltaTLimitation_Disable: dT disabled.

MPBus_DeltaTLimitation_dT_Manager: Simple dT limitation.

MPBus_DeltaTLimitation_dT_ManagerScal: Advanced dT limitation.

6.4.1.28 E_MPBus_Override

Override mode.

```
TYPE E_MPBus_Override :
(
  MPBus_Override_None  := 0,
  MPBus_Override_Auto  := 1,
  MPBus_Override_Close := 2,
  MPBus_Override_Open  := 3,
  MPBus_Override_Vnom  := 4,
  MPBus_Override_Vmax  := 5,
  MPBus_Override_Stop  := 6,
  MPBus_Override_Pnom  := 7,
  MPBus_Override_Pmax  := 8
);
END_TYPE
```

6.4.1.29 E_MPBus_Override_6wayMPIV

Override control mode.

```
TYPE E_MPBus_Override_6wayMPIV :
(
  MPBus_6wayMPIV_None     := 0,
  MPBus_6wayMPIV_Seq1Open := 1,
  MPBus_6wayMPIV_Seq2Open := 2,
  MPBus_6wayMPIV_Close    := 3,
  MPBus_6wayMPIV_Seq1Vmax := 4,
  MPBus_6wayMPIV_Seq2Vmax := 5
);
END_TYPE
```

6.4.1.30 MP_ERROR

Library error messages.

```
TYPE MP_ERROR :
(
  NO_MP_ERROR                                := 0,
  WRONG_TERMINAL                             := 1,
  NO_ANSWER_FROM_KL6771                      := 2,
  NO_LINK_TO_STRUCTURE_strDataKL6771         := 3,
  WRONG_MP_ADDRESS_IS_0                      := 10,
  WRONG_MP_ADDRESS                           := 11,
  WRONG_SET_POINT                            := 21,
  MP_BUS_TIMEOUT_NO_ANSWER_FROM_SLAVE        := 25,
  MP_BUS_SETPOINT_DIFF_TOO_HIGH              := 26,
  KL6771_TIME_OUT                            := 31,
  MP_ADDRESS_IS_IN_USE                       := 32,
  MP_DISABLED                                := 33,
  MP_BUS_ERROR                               := 87,
  MP_NO_ANSWER_ON_EVENT                      := 88,
  MP_NO_ANSWER                               := 89,
  MP_COM_BREAK                               := 90,
  MP_LENGTH_PARITY_ERROR                     := 98,
  MP_CROSS_PARITY_ERROR                      := 99,
  MP_MASTER_CONFLICT_ERROR                   := 101,
  MP_GAP_TIMEOUT_ERROR                       := 102,
  MP_NO_ANSWER_SLAVE                         := 103,
  MP_ANSWER_ERROR_FLAG                       := 110,
  MP_ANSWER_WRONG_LEN                        := 111,
  MP_ANSWER_WRONG_TELEG                      := 112,
  MP_ANSWER_WITH_ERROR                       := 115,
  MP_ERROR_WrongDeviceFamily                 := 200,
  MP_CONF_ERROR_CO2LimitGood                 := 300,
  MP_CONF_ERROR_CO2LimitModerate             := 301,
  MP_CONF_ERROR_CO2Limit                     := 302
  MP_CONF_ERROR_WRONG_TEMP_OFFSET            := 303,
  MP_CONF_ERROR_WRONG_HUMIDITY_OFFSET        := 304,
  MP_CONF_ERROR_WRONG_CO2_OFFSET             := 305,
  MP_CONF_ERROR_WRONG_TempSetpoint           := 306,
  MP_CONF_ERROR_WRONG_RelativeTempSetpoint   := 307,
  MP_CONF_ERROR_WRONG_DefaultTempSetpoint    := 308,
  MP_CONF_ERROR_WRONG_TempSetpointRange      := 309,
  MP_ANSWER_Reserve                          := 16#800A,
  MP_ANSWER_UnknowCommand                    := 16#800B,
  MP_ANSWER_WrongOrNoPassword                   := 16#800C,
  MP_ANSWER_CommandExecution                    := 16#800D,
  MP_ANSWER_ParameterError                      := 16#800E,
  MP_ANSWER_UnknowId                            := 16#800F,
  MP_ANSWER_SizeMismatch                        := 16#8010,
  MP_ANSWER_IllegalBlockNr                      := 16#8011,
  MP_ANSWER_InternalBusBusy                     := 16#8012,
  MP_ANSWER_ReservedForFuture                   := 16#80FF
)
END_TYPE
```

NO_MP_ERROR: No error.

WRONG_TERMINAL: Wrong terminal connected.

NO_ANSWER_FROM_KL6771: No answer from KL6771. This message usually means that there is no connection to the terminal. Are the I/O variables of the terminal linked? Terminal plugged in incorrectly? Everything revised, compiled and read again?

NO_LINK_TO_STRUCTURE_strDataKL6771: Check link to structure DataKL6771.

WRONG_MP_ADDRESS_IS_0: MP-Bus address is 0. Only addresses between 1 and 8 are allowed.

WRONG_MP_ADDRESS: MP-Bus address is >8. Only addresses between 1 and 8 are allowed.

WRONG_SET_POINT: Wrong setpoint.

MP_BUS_TIMEOUT_NO_ANSWER_FROM_SLAVE: MP-Bus timeout, no answer from slave.

MP_BUS_SETPOINT_DIFF_TOO_HIGH: Difference from setpoint too high.

KL6771_TIME_OUT: KL6771 timeout.

MP_ADDRESS_IS_IN_USE: MP-Bus address is in use.

MP_DISABLED: MP-Bus disabled.

MP_BUS_ERROR: MP-Bus error.

MP_NO_ANSWER_ON_EVENT: MP-Bus no answer on event.

MP_NO_ANSWER: MP-Bus no answer.

MP_COM_BREAK: MP-Bus communication break.

MP_LENGTH_PARITY_ERROR: MP-Bus length parity error.

MP_CROSS_PARITY_ERROR: MP-Bus cross parity error.

MP_MASTER_CONFLICT_ERROR: MP-Bus MASTER_CONFLICT_ERROR.

MP_GAP_TIMEOUT_ERROR: MP-Bus GAP timeout.

MP_NO_ANSWER_SLAVE: MP-Bus no answer from slave.

MP_ANSWER_ERROR_FLAG: MP-Bus error bit in the answer telegram is set.

MP_ANSWER_WRONG_LEN: MP-Bus wrong telegram length.

MP_ANSWER_WRONG_TELEG: MP-Bus wrong telegram received.

MP_ANSWER_WITH_ERROR: MP-Bus answer contains an error.

MP_ERROR_WrongDeviceFamily: Wrong device family.

MP_CONF_ERROR_CO2LimitGood: Configuration error for CO2 limit "Good".

MP_CONF_ERROR_CO2LimitModerate: Configuration error for CO2 limit "Moderate".

MP_CONF_ERROR_CO2Limit: Configuration error for CO2 limit.

MP_CONF_ERROR_WRONG_TEMP_OFFSET: Configuration error for temperature offset.

MP_CONF_ERROR_WRONG_HUMIDITY_OFFSET: Configuration error for humidity offset.

MP_CONF_ERROR_WRONG_CO2_OFFSET: Configuration error for CO2 offset.

MP_CONF_ERROR_WRONG_TempSetpoint: Configuration error for temperature setpoint.

MP_CONF_ERROR_WRONG_RelativeTempSetpoint: Configuration error for relative temperature setpoint.

MP_CONF_ERROR_WRONG_DefaultTempSetpoint: Configuration error for default temperature setpoint.

MP_CONF_ERROR_WRONG_TempSetpointRange: Configuration error for temperature setpoint range.

MP_ANSWER_Reserve: Reserve.

MP_ANSWER_UnknowCommand: Unknown command.

MP_ANSWER_WrongOrNoPassword: Wrong or no password.

MP_ANSWER_CommandExecution: Execution of the command.

MP_ANSWER_ParameterError: Parameter error.

MP_ANSWER_UnknowId: Unknown Id.

MP_ANSWER_SizeMismatch: Size does not match.

MP_ANSWER_IllegalBlockNr: Invalid block number.

MP_ANSWER_InternalBusBusy: Internal bus is busy.

MP_ANSWER_ReservedForFuture: Reserve.

6.4.1.31 UST3_Ex

Voltage scaling.

```
TYPE UST3_Ex :
(
  Ex_1mV    := 0,
  Ex_250uV  := 1,
)
END_TYPE
```

Ex_1mV**:** scaling 0...11 V.

Ex_250uV**:** scaling 0...3 V.

6.4.1.32 UST3_R_set

Resistance scaling.

```
TYPE UST3_R_set :
(
  R_1Ohm      := 0,
  R_250mOhm   := 1,
  R_4Ohm      := 2,
)
END_TYPE
```

R_1Ohm**:** scaling 0...20 kOhm.

R_250mOhm**:** scaling 0...5 kOhm.

R_4Ohm**:** scaling 0...262 kOhm.

6.4.2 Structures

6.4.2.1 DataKL6771

Links the send and receive function blocks with the function block KL6771() [▶ 26].

```
TYPE DataKL6771 :
STRUCT
  OrderNumber   : BYTE;
  ReciveData    : BOOL;
  SendData      : BOOL;
  Error         : BOOL;
  ErrorID       : MP_Error;
  pNumber       : DWORD;
END_STRUCT
END_TYPE
```

OrderNumber: internal byte.

ReciveData: data are received.

SendData: data are sent.

Error: the output becomes TRUE as soon as an error occurs. This error is described via the *ErrorID* variable.

ErrorID: this output outputs an error code in the event of an error (see MP_ERROR [▶ 81]). *Error* goes TRUE at the same time.

pNumber: internal pointer.

6.4.2.2 MP_BUS_MPX_ERROR

Error messages of the "MPX" sensors (function block MP_MPX [▶ 44]).

```
TYPE MP_BUS_MPX_ERROR :
STRUCT
  MP_BUS_MPX_TempSensorErr       : BOOL;
  MP_BUS_MPX_HumiditySensorErr   : BOOL;
  MP_BUS_MPX_CO2SensorErr        : BOOL;
  MP_BUS_MPX_VocSensorErr        : BOOL;
END_STRUCT
END_TYPE
```

MP_BUS_MPX_TempSensorErr: The temperature sensor is faulty.

MP_BUS_MPX_HumiditySensorErr: The humidity sensor is faulty.

MP_BUS_MPX_CO2SensorErr: The CO2 sensor is faulty.

MP_BUS_MPX_VocSensorErr: The VOC sensor is faulty.

6.4.2.3 MP_Serial_Number

Serial number of the device.

```
TYPE MP_Serial_Number :
STRUCT
  YearAndWeek    : WORD;
  DayAndNumber   : WORD;
  DeviceFamily   : BYTE;
  TestStation    : BYTE;
  FamilySuffix   : BYTE;
END_STRUCT
END_TYPE
```

YearAndWeek: Year and week.

DayAndNumber: Day and number.

DeviceFamily: Device family.

TestStation: Test station.

FamilySuffix: Device family suffix.

6.4.2.4 St_MP_EV_V4_MalfunctionServiceInfo

Fault and service information.

```
TYPE St_MP_EV_V4_MalfunctionServiceInfo :
STRUCT
  bNoCommunicationToActuator : BOOL;
  bGearDisengaged            : BOOL;
  bActuatorCannotMove        : BOOL;
  bReverseFlow               : BOOL;
  bFlowSetpointNotReached    : BOOL;
  bFlowWithClosedValve       : BOOL;
  bActualFlowVnom            : BOOL;
  bFlowMeasurementError      : BOOL;
  bRemoteTempError           : BOOL;
  bIntegratedTempError       : BOOL;
  bCommToSensorInterrupted   : BOOL;
  bFreezeWarning             : BOOL;
  bGlycolDetected            : BOOL;
  bPowerSetpointNotReached   : BOOL;
END_STRUCT
END_TYPE
```

bNoCommunicationToActuator: No communication to the actuator.

bGearDisengaged: Gear disengaged.

bActuatorCannotMove: Actuator cannot move.

bReverseFlow: Reverse flow.

bFlowSetpointNotReached: Setpoint for flow not reached.

bFlowWithClosedValve: Flow with closed valve.

bActualFlowVnom: Actual flow > nominal flow.

bFlowMeasurementError: Error during flow measurement.

bRemoteTempError: Remote temperature error.

bIntegratedTempError: Integrated temperature error.

bCommToSensorInterrupted: Communication to the sensor is interrupted.

bFreezeWarning: Freeze warning.

bGlycolDetected: Glycol detected.

bPowerSetpointNotReached: Power setpoint not reached.

6.4.2.5 St_MP_VRU_ServiceInfo

Malfunction and service information.

```
TYPE St_MP_VRU_ServiceInfo :
STRUCT
  bError_dP_Sensor                  : BOOL;
  bReverseAirflowDetected           : BOOL;
  bAirflowNotReached                : BOOL;
  bFlowInClosedPosition             : BOOL;
  bInternalActivity                 : BOOL;
  bGearDisengaged                   : BOOL;
  bBusWatchdogTriggered             : BOOL;
  bActuatorDoseNotFitToApplication : BOOL;
  bPressSensorWrongConnected        : BOOL;
  bPressureSensorNotReached         : BOOL;
  bError_dP_SensorOutOfRange        : BOOL;
END_STRUCT
END_TYPE
```

bError_dP_Sensor: Error dp sensor.

bReverseAirflowDetected: Reverse air flow detected.

bAirflowNotReached: Air flow not reached.

bFlowInClosedPosition: Flow in closed position.

bInternalActivity: Internal activity.

bGearDisengaged: Gear disengaged.

bBusWatchdogTriggered: Bus watchdog triggered.

bActuatorDoseNotFitToApplication: Actuator does not fit to application.

bPressSensorWrongConnected: Pressure sensor connected wrong.

bPressureSensorNotReached: Pressure sensor not reached.

bError_dP_SensorOutOfRange: dP sensor out of range.

6.4.2.6 St_StateEV

Information on the status of the EV.

```
TYPE St_StateEV :
STRUCT
  bFlow_with_closed_valve : BOOL;
  bAir_bubbles            : BOOL;
  bFlow_not_reached       : BOOL;
  bPower_not_realized     : BOOL;
  bGear_disengaged        : BOOL;
END_STRUCT
END_TYPE
```

bFlow_with_closed_valve: Flow with the valve closed. check valve.

bAir_bubbles: Too many air bubbles (system inadequately vented). Flow measurement is no longer accurate. EV will change from flow control to position control.

bFlow_not_reached: Flow is not reached, although the control valve is fully open. Check hydraulic, turn on pump or increase pump pressure.

bPower_not_realized: Performance is not achieved. The EV can next to the position control and flow control directly conduct a performance. Primary side gives too little flow or dT.

bGear_disengaged: Gear disengagement active. Manual adjustment on site possible. In other MP drives, this information can be read with MP_Get_State.

6.4.2.7 UST3_SET

Data structure for setting and adjusting the scaling of the resistor measurement.

```
TYPE UST3_SET :
STRUCT
  E1        : UST3_Ex;
  E2        : UST3_Ex;
  E3        : UST3_Ex;
  R_SET     : UST3_R_set;
  R_ON_OFF  : BOOL;
END_STRUCT
END_TYPE
```

E1: Parameter U1 (see UST3_Ex [▶ 83]).

E2: Parameter U2 (see UST3_Ex [▶ 83]).

E3: Parameter U3 (see UST3_Ex [▶ 83]).

R_SET: Parameter Y (see UST3_R_set [▶ 83]).

R_ON_OFF: Measurement R or U.

6.5 Error codes

Value (hex)	Value (dec)	Value (enum)	Description
0x0000	0	NO_MP_ERROR	No error.
0x0001	1	WRONG_TERMINAL	Incorrect terminal connected.
0x0002	2	NO_ANSWER_FROM_KL6771	No answer from the KL6771 MP-Bus master terminal.
0x0003	3	NO_LINK_TO_STRUCTURE_strDataKL6771	Check the link to the structure DataKL6771 [▸ 84].
0x000A	10	WRONG_MP_ADDRESS_IS_0	MP-Bus address is 0. Only addresses between 1 and 8 are allowed.
0x000B	11	WRONG_MP_ADDRESS	MP-Bus address is >8. Only addresses between 1 and 8 are allowed.
0x0015	21	WRONG_SET_POINT	Wrong setpoint.
0x0019	25	MP_BUS_TIMEOUT_NO_ANSWER_FROM_SLAVE	MP-Bus timeout, no answer from slave.
0x0020	26	MP_BUS_SETPOINT_DIFF_TOO_HIGH	Difference from setpoint too high.
0x001F	31	KL6771_TIME_OUT	KL6771 timeout.
0x0020	32	MP_ADDRESS_IS_IN_USE	MP-Bus address is in use.
0x0021	33	MP_DISABLED	MP-Bus disabled.
0x0057	87	MP_BUS_ERROR	MP-Bus error.
0x0058	88	MP_NO_ANSWER_ON_EVENT	MP-Bus no answer to an event.
0x0059	89	MP_NO_ANSWER	MP-Bus no answer.
0x005A	90	MP_COM_BREAK	MP-Bus communication break.
0x0062	98	MP_LENGTH_PARITY_ERROR	MP-Bus length parity error.
0x0063	99	MP_CROSS_PARITY_ERROR	MP-Bus cross parity error.
0x0065	101	MP_MASTER_CONFLICT_ERROR	MP-Bus MASTER_CONFLICT_ERROR.
0x0066	102	MP_GAP_TIMEOUT_ERROR	MP-Bus GAP Timeout.
0x0067	103	MP_NO_ANSWER_SLAVE	MP-Bus no answer from slave.
0x006E	110	MP_ANSWER_ERROR_FLAG	MP-Bus error bit in answer telegram is set.
0x006F	111	MP_ANSWER_WRONG_LEN	MP-Bus wrong telegram length.
0x0070	112	MP_ANSWER_WRONG_TELEG	MP-Bus wrong telegram received.
0x0073	115	MP_ANSWER_WITH_ERROR	Answer contains an error.
0x00C8	200	MP_ERROR_WrongDeviceFamily	Wrong device family.
0x012C	300	MP_CONF_ERROR_CO2LimitGood	Configuration error for CO2 limitation "Good".
0x012D	301	MP_CONF_ERROR_CO2LimitModerate	Configuration error for CO2 limitation "Moderate".
0x012E	302	MP_CONF_ERROR_CO2Limit	Configuration error for CO2 limitation.
0x012F	303	MP_CONF_ERROR_WRONG_TEMP_OFFSET	Configuration error for temperature offset.
0x0130	304	MP_CONF_ERROR_WRONG_HUMIDITY_OFFSET	Configuration error for humidity offset.
0x0131	305	MP_CONF_ERROR_WRONG_CO2_OFFSET	Configuration error for CO2 offset.

Value (hex)	Value (dec)	Value (enum)	Description
0x0132	306	MP_CONF_ERROR_WRONG_TempSetpoint	Configuration error for temperature setpoint.
0x0133	307	MP_CONF_ERROR_WRONG_RelativeTempSetpoint	Configuration error for relative temperature setpoint.
0x0134	308	MP_CONF_ERROR_WRONG_DefaultTempSetpoint	Configuration error for preset temperature setpoint.
0x0135	309	MP_CONF_ERROR_WRONG_TempSetpointRange	Configuration error for temperature setpoint range.
0x800A	32778	MP_ANSWER_Reserve	Reserve.
0x800B	32779	MP_ANSWER_UnknowCommand	Unknown command.
0x800C	32780	MP_ANSWER_WrongOrNoPassword	Wrong or no password.
0x800D	32781	MP_ANSWER_CommandExecution	Command execution.
0x800E	32782	MP_ANSWER_ParameterError	Parameter error.
0x800F	32783	MP_ANSWER_UnknowId	Unknown ID.
0x8010	32784	MP_ANSWER_SizeMismatch	Size does not match.
0x8011	32785	MP_ANSWER_IllegalBlockNr	Invalid block number.
0x8012	32786	MP_ANSWER_InternalBusBusy	Internal bus is busy.
0x80FF	33023	MP_ANSWER_ReservedForFuture	Reserve.

7 Appendix

7.1 Examples

Requirements

Example	Description
https://infosys.beckhoff.com/content/1033/tcplclibmpbus/Resources/12063689867/.zip	TwinCAT PLC project for the KL6771.
https://infosys.beckhoff.com/content/1033/tcplclibmpbus/Resources/12063691275/.zip	TwinCAT PLC project for the KL6771 (TwinCAT PLC Control).
https://infosys.beckhoff.com/content/1033/tcplclibmpbus/Resources/12063692683/.zip	TwinCAT PLC project for the KL6771 (TwinCAT System Manager).
https://infosys.beckhoff.com/content/1033/tcplclibmpbus/Resources/12063694091/.zip	Manual assignment of the MP-Bus address (TwinCAT PLC Control).
https://infosys.beckhoff.com/content/1033/tcplclibmpbus/Resources/12063695499/.zip	Manual assignment of the MP-Bus address (TwinCAT System Manager).

7.2 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Download finder

Our download finder contains all the files that we offer you for downloading. You will find application reports, technical documentation, technical drawings, configuration files and much more.

The downloads are available in various formats.

Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on our internet page: www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline:	+49 5246 963-157
e-mail:	support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline:	+49 5246 963-460
e-mail:	service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone:	+49 5246 963-0
e-mail:	info@beckhoff.com
web:	www.beckhoff.com

More Information:
www.beckhoff.com/tx1200

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com