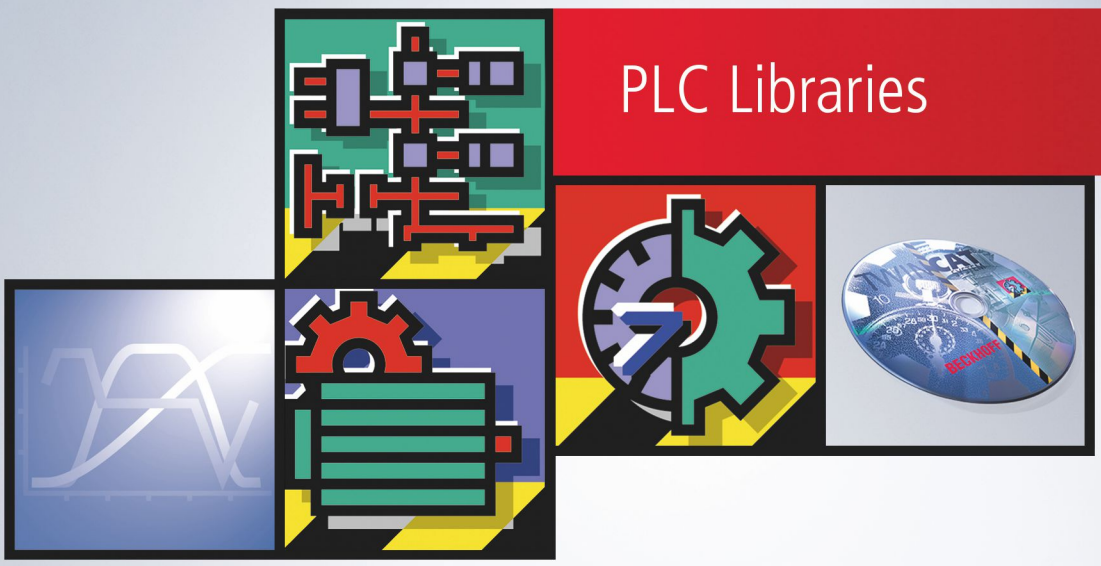
BECKHOFF New Automation Technology

Manual | EN

TX1200

TwinCAT 2 | PLC Library: TcSMI



2023-11-15 | Version: 1.1

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.
For installation and commissioning of the components, it is absolutely necessary to observe the documentation and the following notes and explanations.
The qualified personnel is obliged to always use the currently valid documentation.

The responsible staff must ensure that the application or use of the products described satisfies all 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 notice.
No claims to modify 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 and licensed trademarks of Beckhoff Automation GmbH.
If third parties make use of designations or trademarks used in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.

Patents

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

EtherCAT®

EtherCAT® is 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 Target groups

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

- TwinCAT PLC Control
- TwinCAT System Manager
- PC and network knowledge
- Structure and properties of the Beckhoff Embedded PC and its Bus Terminal system
- Technology of SMI 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.

3 SMI

The automation of roller shutters and sun blinds in building automation is simplified with the SMI-Bus system (Standard Motor Interface). With SMI drives roller shutters can drive to precise positions and blind drives can drive to angular positions with an accuracy of a degree. The SMI drives can transmit actual positions, error messages and service information back to the SMI master terminal.

Major European manufacturers have joined forces in the SMI working group and have developed the digital interface. Drives are controlled by means of telegrams via this uniform interface. Using standard commands, functions can be implemented that are not so easy to realize with conventional drives. Examples are the precise movement to positions, the feedback of the current position and diagnosis. For the adjustment of louvers in shading systems, for example, angular resolutions of 2° can be achieved. Adjustment of the louvers in relation to the position of the sun is thus possible for constant light control. Powerful PLC function blocks are available in the TwinCAT HVAC library for room automation according to VDI 3813.

Distances of up to 350 meters between the controller and the drive are possible. A normal 5-core power cable can be used for the cabling (with PE, N, L and the SMI-specific I+ and I-); I+ and I- are protected against pole reversal. Up to 16 drives can be connected in parallel and addressed individually. SMI drives are available for mains voltage (230 V AC) and for low voltage (24 V DC).

To ensure the compatibility of the SMI products with one another, all products that are to be marked with the SMI logo must be certified. A positive certification can be read on the SMI Group's homepage (https://standard-motor-interface.com). Further information on the SMI-Bus and SMI drives can also be found there.

3.1 Device addressing

SMI defines various modes of device addressing. In principle distinction can be made between individual addressing, group addressing and the collective call (broadcast). Most PLC function blocks have the *dwAddr* input and *eAddrType* for this. While the *dwAddr* input contains the necessary address details, *eAddrType* defines the mode of addressing. The individual modes of addressing are described below. Note that not every command supports all addressing modes. Details can be found in the description of the respective PLC function block.

by the address of a device (eAddrType: = eSMIAddrTypeAddress)

Each SMI device can be assigned an address from 0 to 15. The address is stored in the SMI device and must be correctly set again when exchanging the drive. Since each address should only be assigned once, each SMI device can be addressed individually. With this mode of addressing the *dwAddr* input contains the address in the range 0 to 15. If a value outside the valid range is specified, then the respective function block outputs an error [▶ 57].

This address is also occasionally called the slave address. The slave address must not be confused with the slave ID (see below).

per slave ID (eAddrType := eSMIAddrTypeSlaveId)

The individual device manufacturers store a unique 32-bit number in each SMI device. This slave ID, also called the key ID, can also be used for the addressing of a device. With this mode of addressing the *dwAddr* input contains the 32-bit slave ID and the *dwAddrOption* input contains the manufacturer code (see below). This ensures that SMI devices can be addressed worldwide without them requiring an address.

With some SMI devices the slave ID is printed on the name plate or is made visible by a label on the cable.

Most read commands do not support the addressing by slave ID.

by the manufacturer code (eAddrType := eSMIAddrTypeManufacturer)

In addition to the slave ID, SMI defines a further unique ID, the so-called manufacturer code [▶ 65]. The manufacturer code is permanently stored in the SMI device and cannot be changed. The possible range of values is from 0 to 15, wherein the values 0 and 14 have a special meaning. The manufacturer code 0 addresses all devices, irrespective of the manufacturer. This addressing mode can therefore also be used to dispatch broadcast commands. The value 15 is reserved for future expansions and may not be used. The

English designation *manufacturer code* or the abbreviation *M-ID* is frequently found here. All devices from a manufacturer are always addressed by this addressing. With this mode of addressing the *dwAddr* input contains the manufacturer code in the range from 0 to 14. If a value outside the valid range is specified, then the respective function block outputs an error [▶ 57].

With some SMI devices the manufacturer code is printed on the name plate or is made visible by a label on the cable.

by group addressing (eAddrType: = eSMIAddrTypeGroup)

Each device that is to be controlled via group addressing must have an address from 0 to 15. Each bit of the *dwAddr* input corresponds to an address in the case of group addressing. If bit 0 of *dwAddr* is set, then the device with the address 0 is addressed. If bit 1 is set, then device 1 is addressed and so on. The group addressing thus occupies the bits 0 to 15, which corresponds to a range of values from 0 to 65535. If a value outside the valid range is specified, then the respective function block outputs an error [▶ 57].

Example: The drives with the addresses 1, 4, 7 and 12 are to be addressed. The value 2#00010000_10010010 or 16#1092 or 4242 must therefore be supplied to *dwAddr*.

per Broadcast (eAddrType := eSMIAddrTypeBroadcast)

When addressing by broadcast all devices are always addressed, irrespective of the address set on the device. The *dwAddr* input is not required with this method of addressing and is not evaluated either. Internally the PLC function blocks use addressing by manufacturer code, wherein the manufacturer code 0 is used.

4 Integration into TwinCAT

4.1 Integration into TwinCAT (CX9020)

This example explains how to write a simple PLC program for SMI in TwinCAT and how to link it with the hardware. The task is to control an SMI motor and to change the direction of rotation with a button.

https://infosys.beckhoff.com/content/1033/tcplclibsmi/Resources/12074354059/.zip https://infosys.beckhoff.com/content/1033/tcplclibsmi/Resources/12074354059/.zip

Hardware

Setting up the components

The following hardware is required:

- 1x Embedded PC CX9020
- 1x digital 2-channel input terminal KL1002 (for the step up/down function)
- 1x SMI terminal KL6831
- 1x end terminal KL9010

Set up the hardware and the SMI components as described in the associated documentation.

This example assumes that an Up button was connected to the first KL1002 input and a Down button to the second, and that an SMI motor is connected to Address 1.

Software

Creation of the PLC program

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

Next, generate the following global variables:

```
VAR_GLOBAL
    bUp           AT %I* : BOOL;
    bDown         AT %I* : BOOL;
    stInData      AT %I* : ST_KL6831KL6841InData;
    stOutData     AT %Q* : ST_KL6831KL6841OutData;
    stCommandBuffer      : ST_SMICommandBuffer;
END_VAR
```

bUp : Input variable for the Up button.

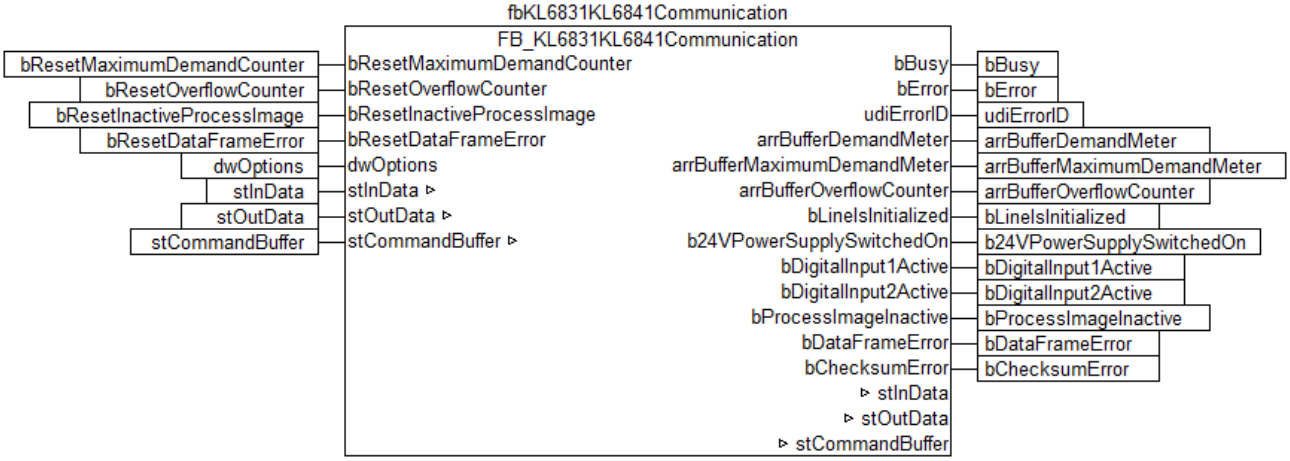
bDown : Input variable for the Down button.

stInData : Input variable [▶ 55] for the SMI terminal.

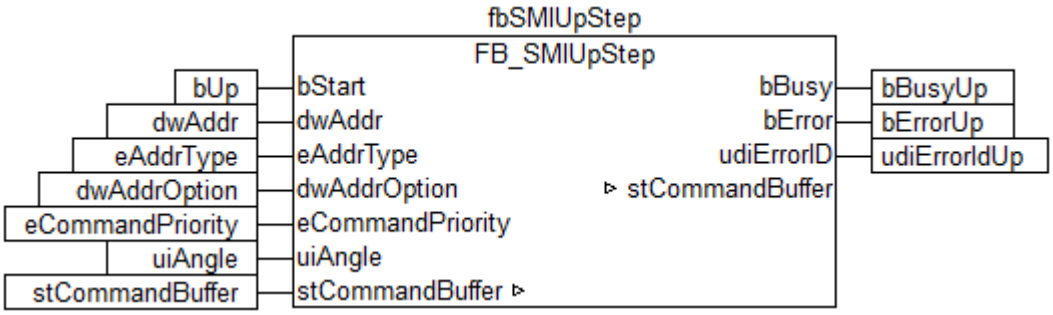
stOutData : Output variable [▶ 56] for the SMI terminal.

stCommandBuffer : required for communication with SMI.

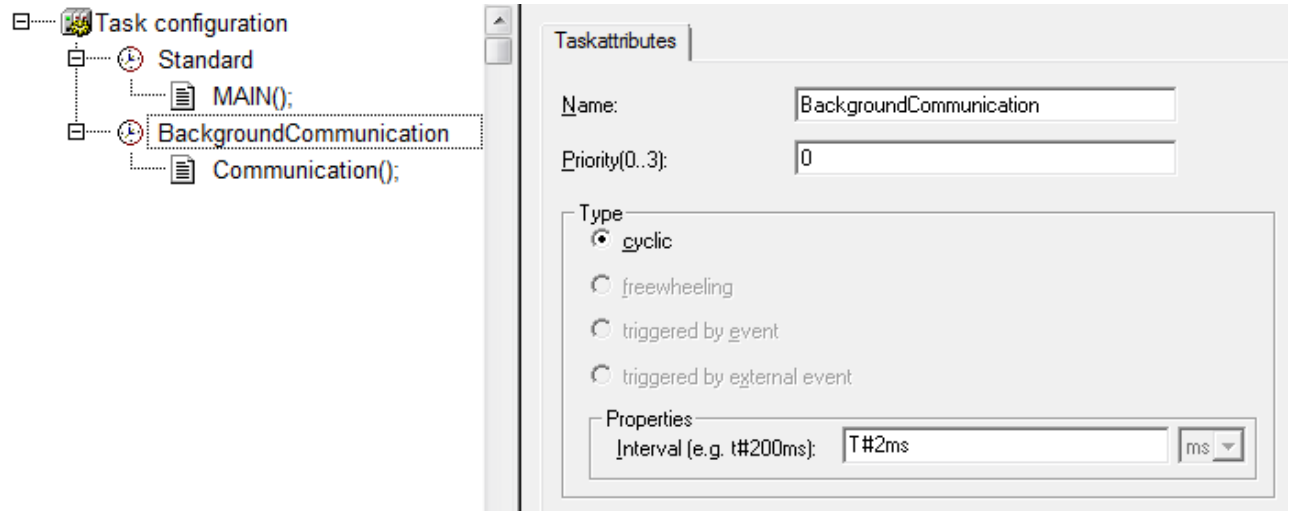
Then create a program (CFC) for background communication with SMI. The FB_KL6831KL6841Communication [▶ 19] block is called in this program. Make sure to link the communication block with **stInData**, **stOutData** and **stCommandBuffer**.



Create a MAIN program (CFC) in which the blocks FB_SMIUpStep [▶ 41] and FB_SMIDownStep [▶ 28] can be called up. Connect the input **bStart** of the StepUp block with the global variable **bUp** and **stCommandBuffer** with the global variable **stCommandBuffer**. Proceed accordingly for the StepDown block.



Go to the task configuration and give the task a lower interval time. More detailed information can be found in the FB_KL6831KL6841Communication [▶ 19] block description.

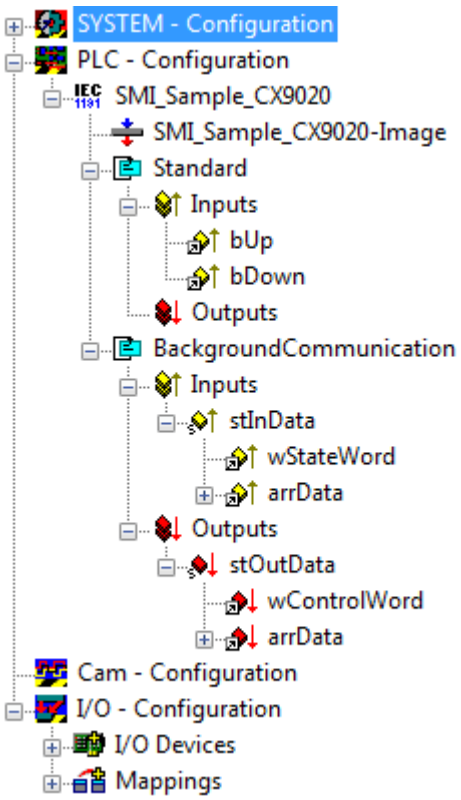


Load the project to the CX as the boot project and save it.

Configuration in the System Manager

Create a new 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. The two tasks are listed when the PLC project is expanded in the tree view. Expand the tasks – all global input and output variables should be allocated to the standard task. However, since the variables **stInData** and **stOutData** are to be processed faster, move them to the background communication task via drag & drop.



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.

By pressing one of the direction buttons the SMI motor selects the direction for the angle specified under *uiAngle*.

Also see about this

- ST_SMICommandBuffer [▶ 56]

4.2 Integration into TwinCAT (BC9191)

This example explains how to write a simple PLC program for SMI in TwinCAT and how to link it with the hardware. The task is to control an SMI motor and to change the direction of rotation with a button.

https://infosys.beckhoff.com/content/1033/tcplclibsmi/Resources/12074355467/.zip https://infosys.beckhoff.com/content/1033/tcplclibsmi/Resources/12074355467/.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 step up/down function)
- 1x SMI terminal KL6831
- 1x end terminal KL9010

Set up the hardware and the SMI components as described in the associated documentation.

This example assumes that an Up button was connected to the first KL1002 input and a Down button to the second, and that an SMI motor is connected to Address 1.

Software

Creation of the PLC program

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

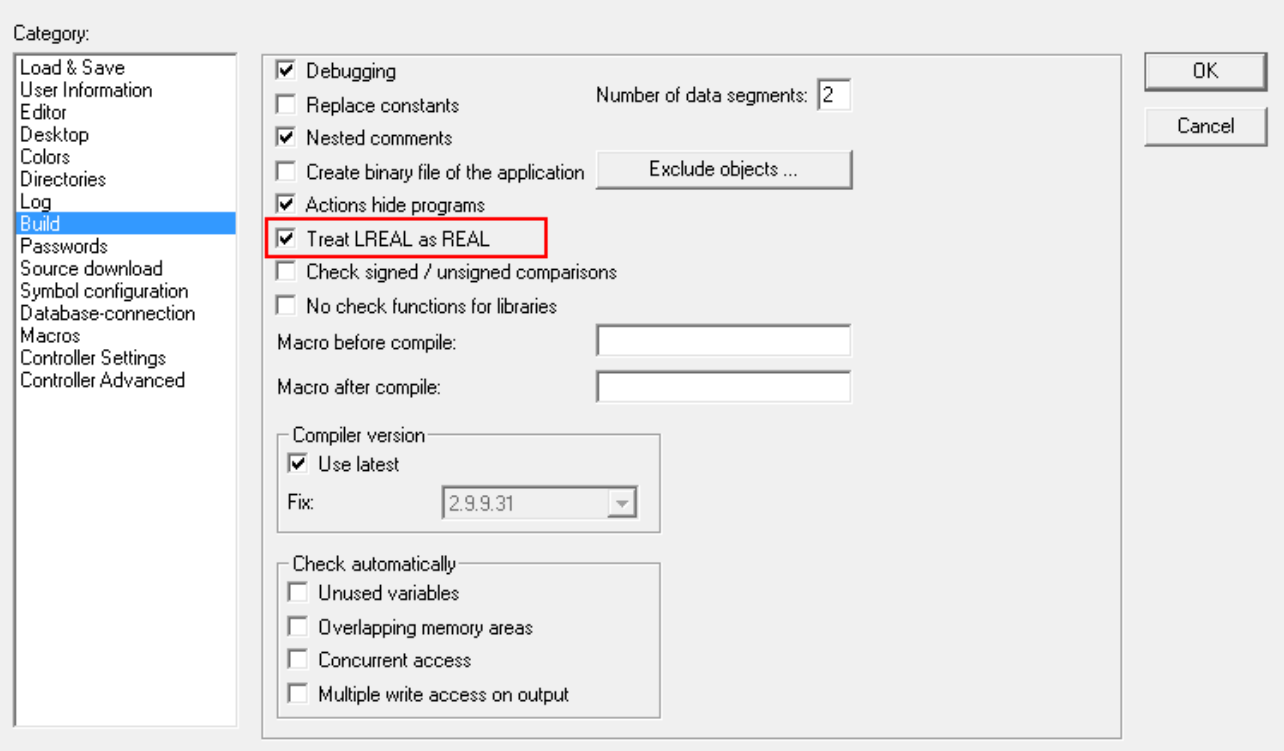


Fig. 1: Sample-BC-Options

Next, generate the following global variables:

```
VAR_GLOBAL
    bUp           AT %I* : BOOL;
    bDown         AT %I* : BOOL;
    stInData      AT %I* : ST_KL6831KL6841InData;
    stOutData     AT %Q* : ST_KL6831KL6841OutData;
    stCommandBuffer      : ST_SMICommandBuffer;
END_VAR
```

bUp : Input variable for the Up button.

bDown : Input variable for the Down button.

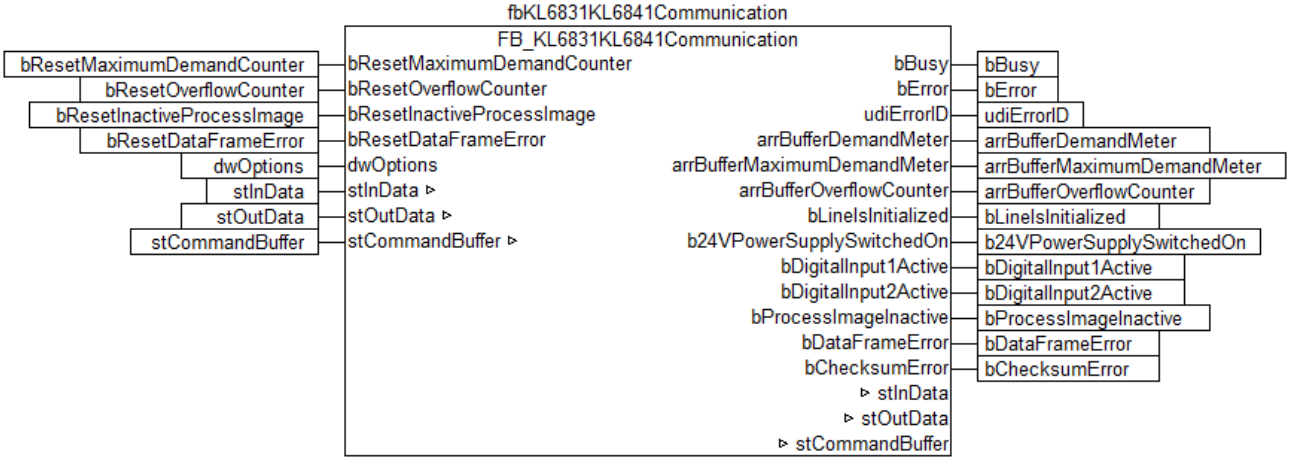
stInData : Input variable [▶ 55] for the SMI terminal.

stOutData : Output variable [▶ 56] for the SMI terminal.

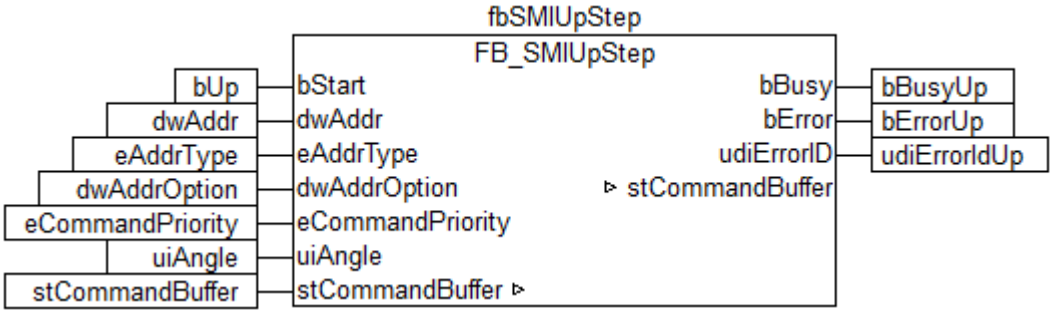
stCommandBuffer : required for communication [▶ 56] with SMI.

Since BC devices can only process one task, communication with SMI cannot run separately.

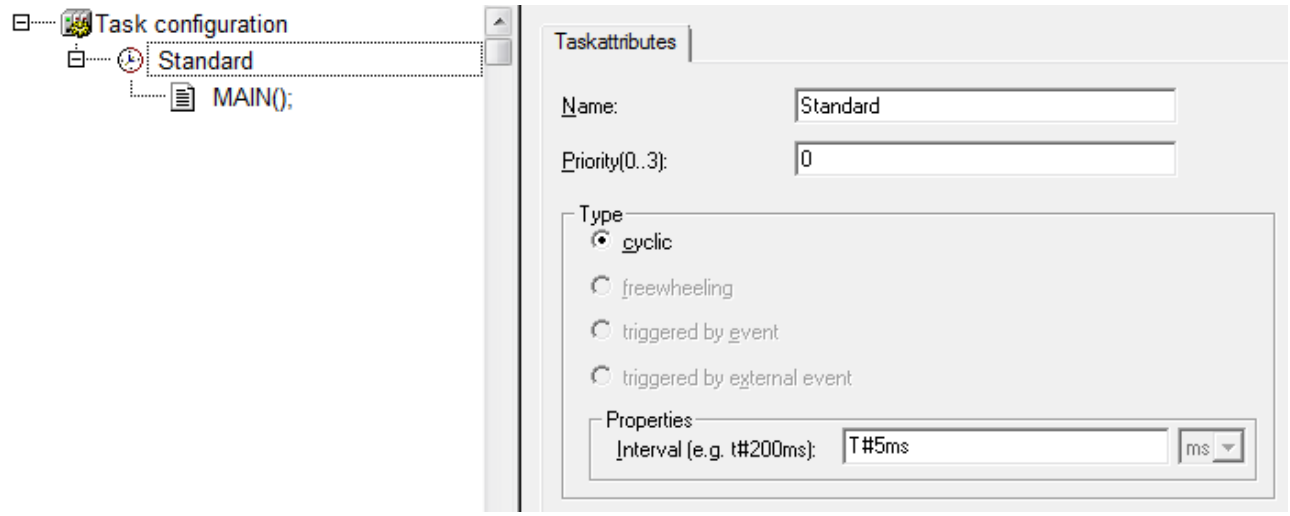
Therefore create a MAIN program (CFC) in which the blocks FB_KL6831KL6841Communication [▶ 19], FB_SMIUpStep [▶ 41] and FB_SMIDownStep [▶ 28] can be called up. Make sure to link the communication block with **stInData**, **stOutData** and **stCommandBuffer**.



Connect the input **bStart** of the StepUp block with the global variable **bUp** and **stCommandBuffer** with the global variable **stCommandBuffer**. Proceed accordingly for the StepDown block.



Go to the task configuration and give the task a lower interval time. More detailed information can be found in the FB_KL6831KL6841Communication [▶ 19] block description.

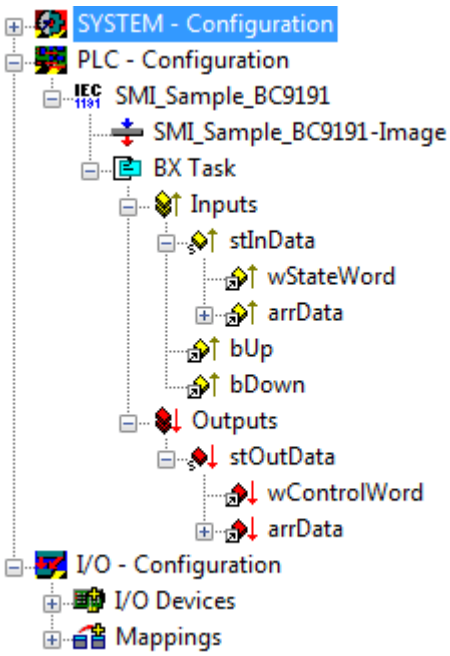


Now load the project as a boot project to the BC and save it.

Configuration in the System Manager

Create a new 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.



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.

By pressing one of the directions buttons the SMI motor selects the direction for the angle specified under *uiAngle*.

5 Programming

Installation

The 'TcSMI.lib/.lb6/.lbx' libraries are installed as standard from TwinCAT 2.11 Build 2240 (R3 and x64 Engineering).

Hardware documentation in the Beckhoff information system: KL6831/KL6841 - SMI master terminal

Further libraries required

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

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

For Bus Terminal Controllers from the BCxx00 series:

- Standard.lb6
- TcPlcUtilitiesBC.lb6
- PlcHelperBC.lb6
- PlcSystemBC.lb6

For Bus Terminal controllers from the BCxx50, BCxx20 and BC9191 series:

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

For Bus Terminal Controllers from the BXxx00 series:

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

Memory usage

Some of the PLC program memory is already used up by integrating the library. Depending on the application program it is therefore possible that the remaining memory space may be insufficient.

5.1 Function blocks

Actuators

Name	Beschreibung
FB_SMIVenetianBlind [▶ 63]	Venetian blind control.

Base commands

Name	Description
FB_KL6831KL6841Communication [▶ 19]	Reads the SMI commands sequentially from the internal buffer of the PLC library and forwards them to the KL6831/KL6841
FB_KL6831KL6841Config [▶ 21]	This function-block can be used to configure the KL6831/KL6841.
FB_SMISendSMICommand [▶ 24]	This function block is for the general sending of an SMI command.

Basic commands

Name	Description
FB_SMIDiagAll [▶ 25]	Diagnostic telegram is sent.
FB_SMIDown [▶ 27]	Motor operation to the lower final position.
FB_SMIDownStep [▶ 28]	Motor runs downwards by a specified angle.
FB_SMIPos1 [▶ 29]	Drives to the fixed position *Pos1* configured on the motor side.
FB_SMIPos1Read [▶ 30]	Reads the fixed position *Pos1* configured on the motor side.
FB_SMIPos1Write [▶ 32]	Writes the fixed position *Pos1* which is configured on the motor side.
FB_SMIPos2 [▶ 33]	Drives to the fixed position *Pos2* configured on the motor side.
FB_SMIPos2Read [▶ 34]	Reads the fixed position *Pos2* configured on the motor side.
FB_SMIPos2Write [▶ 35]	Writes the fixed position *Pos2* which is configured on the motor side.
FB_SMIPosRead [▶ 36]	Reads the current position.
FB_SMIPosWrite [▶ 37]	Drives to a position.
FB_SMIStop [▶ 38]	Stops the motor operation.
FB_SMISyn [▶ 39]	Queries the manufacturer code and drive type.
FB_SMIUp [▶ 40]	Motor operation to the upper final position.
FB_SMIUpStep [▶ 41]	Motor operation upwards by a specified angle.

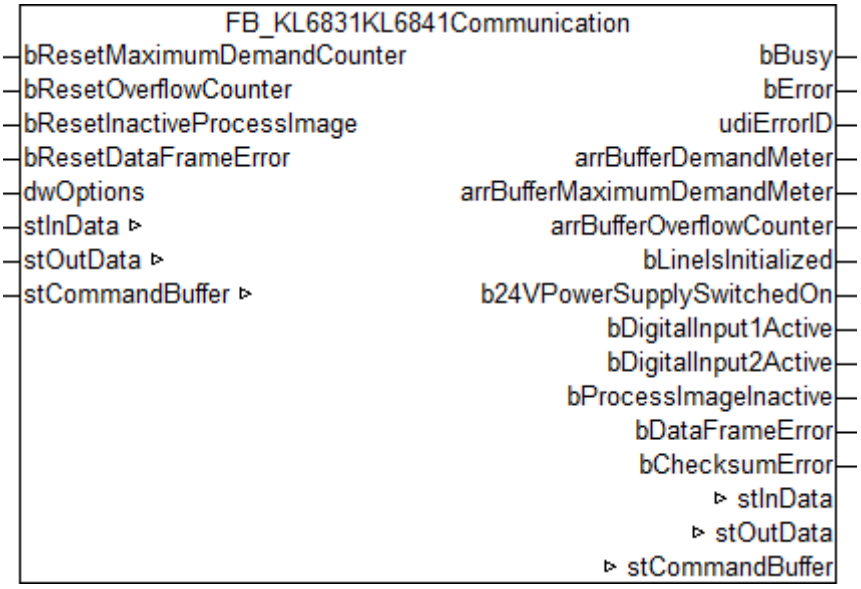
Addressing commands

Name	Description
FB_SMIAddressing [▶ 42]	Addresses SMI devices.
FB_SMIDiscoverySlaveId [▶ 43]	Searches for SMI devices by manufacturer code.
FB_SMISlaveAddrRead [▶ 45]	Reads the address (0-15) of a drive.
FB_SMISlaveAddrWrite [▶ 46]	Writes an address (0-15) to one or more drives.
FB_SMISlaveIdCompare [▶ 47]	Compares a specified slave ID (32-bit key ID) with the slave ID (32-bit key ID) of one or more drives, which is defined on the motor side.
FB_SMISlaveIdRead [▶ 49]	Reads the slave ID (32-bit Key ID).

System commands

Name	Description
FB_SMIParValueReadByte [▶ 50]	Reads a byte parameter (1 byte) stored on the motor side.
FB_SMIParValueReadWord [▶ 51]	Reads a Word parameter (2 bytes) stored on the motor side.
FB_SMIParValueReadDWord [▶ 52]	Reads a DWord parameter (4 bytes) stored on the motor side.

5.1.1 FB_KL6831KL6841Communication



The function blocks for the SMI commands do not directly access the process image of the KL6831/KL6841; instead, they place the individual SMI commands into three different buffers. The FB_KL6831KL6841Communication() block reads the SMI commands sequentially from these three buffers and forwards them to the KL6831/KL6841. This prevents multiple blocks accessing the KL6831/KL68411 process image at the same time. Each of these three buffers is processed with a different priority (high, medium or low). The user of the PLC library can use the *eCommandPriority* parameter (available in most blocks) to determine the priority with which the FB_KL6831KL6841Communication() block processes the respective SMI command.

The buffers in which the SMI commands are placed are all contained in a variable of the type ST_SMICommandBuffer [▶ 56]. For each KL6831/KL6841 there is an instance of the FB_KL6831KL6841Communication() function block and a variable of the type ST_SMICommandBuffer. If possible, the FB_KL6831KL6841Communication() function block should be called in a separate, faster task.

The extent to which the buffers are utilized can be determined from the outputs of the function block. Three arrays are output for this in which each element (0, 1 or 2) represents one of the three buffers (high, middle, or low). If you detect regular overflow for one of the three buffers, you should consider the following:

- How heavily are the individual PLC tasks utilised? The TwinCAT System Manager offers various appropriate utilities for the analysis.

- Try reducing the cycle time of the task in which the FB_KL6831KL6841Communication() block is called. The value should not exceed 6 ms. Ideally it should be 2 ms.

- Check the cycle time of the PLC task in which the blocks for the individual SMI commands are called. This value should be between 10 ms and 60 ms.

- If possible, avoid polling (regular reading) of values. Only read values when they are required.

- Distribute the individual drives evenly over several SMI lines. Since several SMI lines are processed simultaneously per PLC cycle, the data throughput as a whole is increased as a result.

VAR_INPUT

```
bResetMaximumDemandCounter    : BOOL;
bResetOverflowCounter         : BOOL;
bResetInactiveProcessImage    : BOOL;
bResetDataFrameError          : BOOL;
dwOptions                     : DWORD := 0;
```

bResetMaximumDemandCounter: a positive edge resets the stored value of the maximum command buffer utilisation *(arrBufferMaximumDemandMeter)*.

bResetOverflowCounter: a positive edge resets the stored value of the number of command buffer overflows *(arrBufferOverflowCounter)*.

bResetInactiveProcessImage: a positive edge cancels the blocking of the process image of the terminal. The *bProcessImageInactive*, *bDigitalInput1Active* and *bDigitalInput2Active* outputs are again set to FALSE. The blocking is activated as soon as one of the two digital inputs on the terminal is actuated.

bResetDataFrameError: a positive edge resets the message for a telegram error. The *bDataFrameError* output is set again to FALSE. The blocking is activated as soon as the terminal detects a telegram error.

dwOptions: reserved for future expansions.

VAR_OUTPUT

```
bBusy                       : BOOL;
bError                      : BOOL;
udiErrorId                  : UDINT;
arrBufferDemandMeter        : ARRAY [0..2] OF BYTE;
arrBufferMaximumDemandMeter : ARRAY [0..2] OF BYTE;
arrBufferOverflowCounter    : ARRAY [0..2] OF UINT;
bLineIsInitialized          : BOOL;
b24VPowerSupplySwitchedOn   : BOOL;
bDigitalInput1Active        : BOOL;
bDigitalInput2Active        : BOOL;
bProcessImageInactive       : BOOL;
bDataFrameError             : BOOL;
bChecksumError              : BOOL;
```

bBusy: this output is set as soon as the function block processes a command and remains active until the command has been processed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

arrBufferDemandMeter: occupation of respective buffer (0 - 100 %).

arrBufferMaximumDemandMeter: previous maximum occupancy of the respective buffer (0 - 100 %).

arrBufferOverflowCounter: number of buffer overflows to date.

bLineIsInitialized: if the block is being called for the first time (e.g. when the controller is starting up) an initialisation process is executed. No SMI commands can be processed during this time. This output is set to TRUE once the initialisation has been completed.

b24VPowerSupplySwitchedOn: the KL6831/KL6841 must be supplied with 24 V DC via two connections. The output is set as soon as 24 V DC is detected. If there is no 24 V DC the output goes FALSE and no SMI commands can be processed by the controller as long as there is no 24 V DC.

bDigitalInput1Active: the digital input 1 on the terminal has been or is actuated (see also the terminal documentation). The *bProcessImageInactive* output is set and no further SMI commands can be processed by the controller.

bDigitalInput2Active: the digital input 2 on the terminal has been or is actuated (see also the terminal documentation). The *bProcessImageInactive* output is set and no further SMI commands can be processed by the controller.

bProcessImageInactive: one of the two digital inputs has been actuated on the terminal. No further SMI commands can be processed by the controller. The blockage must be released again via the *bResetInactiveProcessImage* input.

bDataFrameError: the terminal has detected a telegram error on the SMI bus. The error must be reset via the *bResetDataFrameError* input.

bChecksumError: the terminal has detected a checksum error on the SMI bus. The message is automatically reset as soon as a telegram is transmitted without error once again.

VAR_IN_OUT

```
stInData              : ST_KL6831KL6841InData;
stOutData             : ST_KL6831KL6841OutData;
stCommandBuffer       : ST_SMICommandBuffer;
```

stInData: reference to the structure [▶ 55] for communication with the KL6831/KL6841.

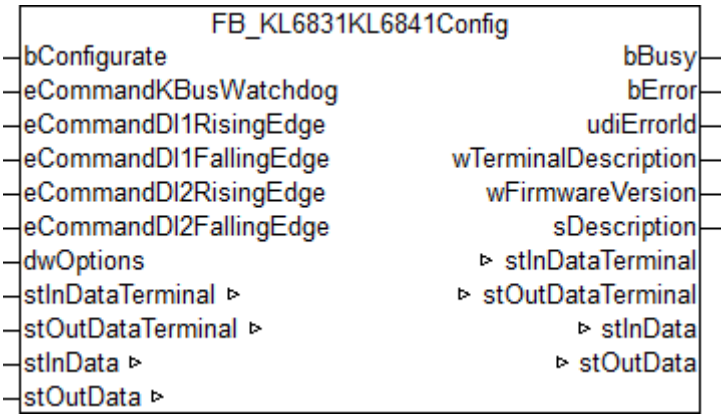
stOutData: reference to the structure [▶ 56] for communication with the KL6831/KL6841.

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the SMI function blocks.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

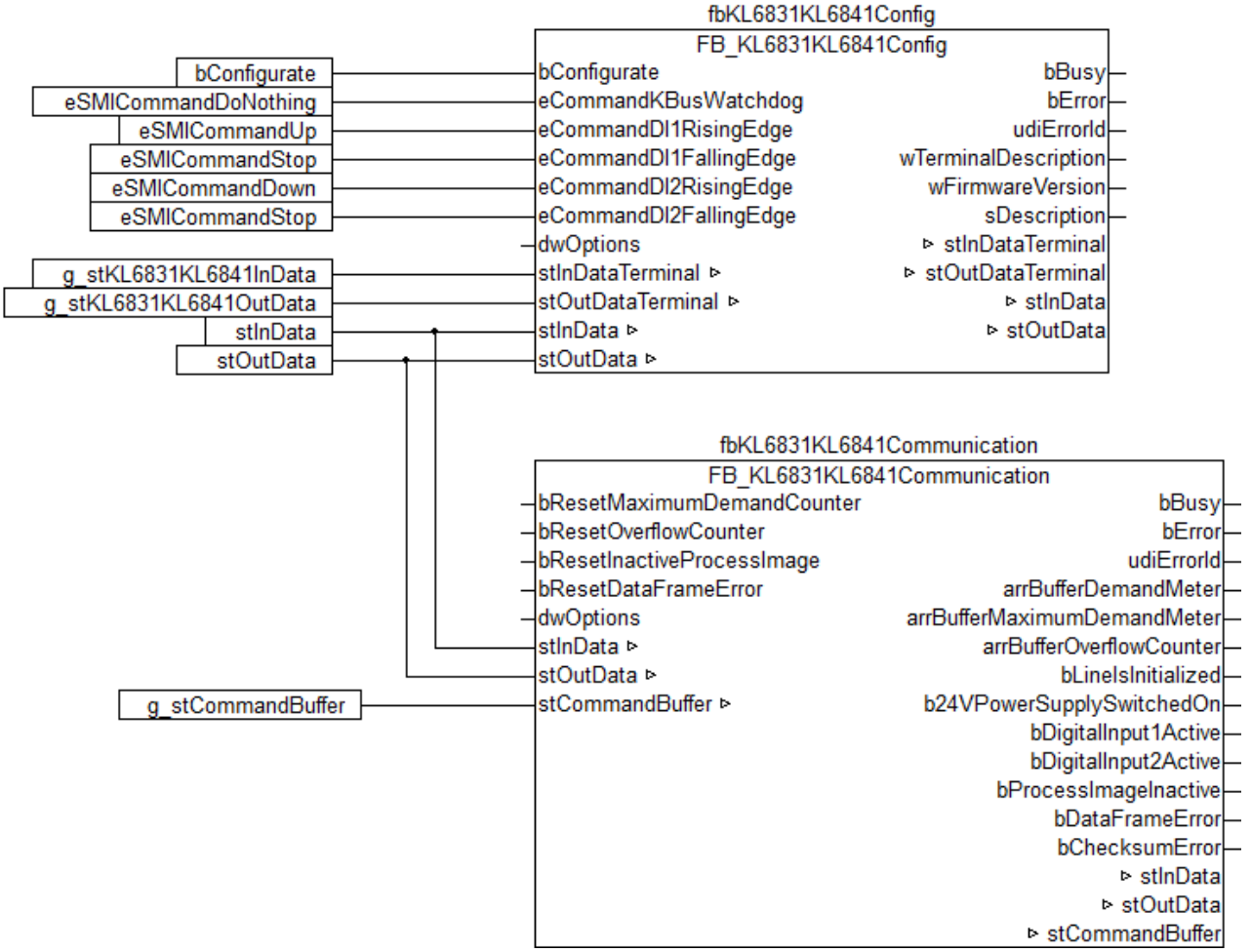
5.1.2 FB_KL6831KL6841Config



This block is used to configure the KL6831/KL6841. The configuration is executed when the PLC program starts, or it can be triggered by a positive edge at the input *bConfigurate*. The parameters are stored in the respective registers of the KL6831/KL6841 in a fail-safe manner. In addition, some general information, such as the firmware version, is read from the KL6831/KL6841.

Example

The block is called in the same task as the block FB_KL6831KL6841Communication() [▶ 19].



The block FB_KL6831KL6841Config() is linked to the process image of the KL6831/KL6841. After the configuration, the block FB_KL6831KL6841Communication() is assigned the process values of the KL6831/KL6841. During the configuration, it is not possible to send SMI commands.

Unpacking the example files https://infosys.beckhoff.com/content/1033/tcplclibsmi/Resources/

12074356875/.zip

VAR_INPUT

```
bConfigurate            : BOOL := FALSE;
eCommandKBusWatchdog    : E_SMIConfigurationCommands := eSMICommandDoNothing;
eCommandDI1RisingEdge   : E_SMIConfigurationCommands := eSMICommandUp;
eCommandDI1FallingEdge  : E_SMIConfigurationCommands := eSMICommandStop;
eCommandDI2RisingEdge   : E_SMIConfigurationCommands := eSMICommandDown;
eCommandDI2FallingEdge  : E_SMIConfigurationCommands := eSMICommandStop;
dwOptions               : DWORD := 0;
```

bConfigurate: Configuration of the Bus Terminal is started by a positive edge at this input.

eCommandKBusWatchdog: Defines the SMI command [▶ 54] that is sent as soon as the Bus Terminal is no longer addressed via the K-bus. Corresponds to register 33 to 35 of the Bus Terminal.

eCommandDI1RisingEdge: Defines the SMI command that is sent as soon as a rising edge is detected at input 1 of the Bus Terminal. Corresponds to register 36 to 38 of the Bus Terminal.

eCommandDI1FallingEdge: Defines the SMI command that is sent as soon as a falling edge is detected at input 1 of the Bus Terminal. Corresponds to register 39 to 41 of the Bus Terminal.

eCommandDI2RisingEdge: Defines the SMI command that is sent as soon as a rising edge is detected at input 2 of the Bus Terminal. Corresponds to register 42 to 44 of the Bus Terminal.

eCommandDI2FallingEdge: Defines the SMI command that is sent as soon as a falling edge is detected at input 2 of the Bus Terminal. Corresponds to register 45 to 47 of the Bus Terminal.

dwOptions: Reserved for future expansions.

VAR_OUTPUT

```
bBusy                 : BOOL;
bError                : BOOL;
udiErrorId            : UDINT;
wTerminalDescription  : WORD;
wFirmwareVersion      : WORD;
sDescription          : STRING;
```

bBusy: When the block is activated the output is set, and it remains active until execution of the command has been completed.

bError: This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is again set to FALSE by the reactivation of the function block via the *bConfigurate* input.

udiErrorId: Contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bConfigurate* input. See Error codes [▶ 57].

wTerminalDescription: Contains the terminal designation (e.g. 6831). Corresponds to register 8 of the Bus Terminal.

wFirmwareVersion: Contains the firmware version. Corresponds to register 9 of the Bus Terminal.

sDescription: Terminal designation and firmware version as string (e.g. 'Terminal KL6831 / Firmware 1D').

VAR_IN_OUT

```
stInDataTerminal          : ST_KL6831KL6841InData;
stOutDataTerminal         : ST_KL6831KL6841OutData;
stInData                  : ST_KL6831KL6841InData;
stOutData                 : ST_KL6831KL6841OutData;
```

stInDataTerminal: Reference to the structure [▶ 55] for communication with the KL6831/KL6841.

stOutDataTerminal: Reference to the structure [▶ 56] for communication with the KL6831/KL6841.

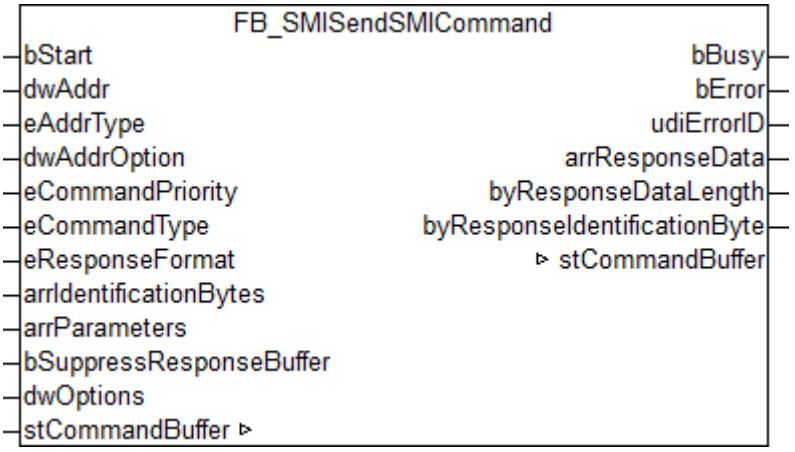
stInData: Reference to the structure for communication with the function block FB_KL6831KL6841Communication() [▶ 19].

stOutData: Reference to the structure for communication with the function block FB_KL6831KL6841Communication() [▶ 19].

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2253	PC/CX, BX or BC	TcSMI library from V1.1.0

5.1.3 FB_SMISendSMICommand



This function block is for the general sending of an SMI command. The precise structure of an SMI command and the functioning of the KL6831/KL6841 must be known for this. The use of this function block is necessary only if an SMI command is to be sent that is not covered by the other PLC function blocks.

VAR_INPUT

```
bStart                  : BOOL;
dwAddr                  : DWORD := 0;
eAddrType               : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption            : DWORD := 0;
eCommandPriority        : E_SMICommandPriority := eSMICommandPriorityMiddle;
eCommandType            : E_SMICommandType := eSMICommandTypeWrite;
eResponseFormat         : E_SMIResponseFormat := eSMIResponseFormatDiagnosis;
arrIdentificationBytes  : ARRAY [0..2] OF BYTE;
arrParameters           : ARRAY [0..2] OF DWORD;
bSuppressResponseBuffer : BOOL := FALSE;
dwOptions               : DWORD := 0;
```

bStart: the block is activated by a positive edge at this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device, for group addressing or as a slave ID.

dwAddrOption: if the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 65] must be specified via this input.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

eCommandType: type of command [▶ 54]: write/read. This parameter affects bit 5 of the start byte of the SMI telegram.

eResponseFormat: response format [▶ 55]: diagnostic special format/standard. This parameter affects bit 6 of the start byte of the SMI telegram.

arrIdentificationBytes: an SMI telegram can consist of up to 3 blocks. Each block possesses an identifier byte. The three identifier bytes are defined by this array.

arrParameters: an SMI telegram can consist of up to 3 blocks. Each block possesses up to four value bytes. The value bytes of the individual blocks are defined by this array.

bSuppressResponseBuffer: if this input is set to TRUE the internal software buffer is not filled with the responses from the FB_KL6831KL6841Communication() [▶ 19] block.

dwOptions: reserved for future expansions.

VAR_OUTPUT

```
bBusy                       : BOOL;
bError                      : BOOL;
udiErrorId                  : UDINT;
arrResponseData             : ARRAY [0..7] OF BYTE;
byResponseDataLength        : BYTE;
byResponseIdentificationByte : BYTE;
```

bBusy: when the block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is again set to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

arrResponseData: the data received from the SMI devices.

byResponseDataLength: the length of the data received in bytes.

byResponseIdentificationByte: the identifier byte received.

VAR_IN_OUT

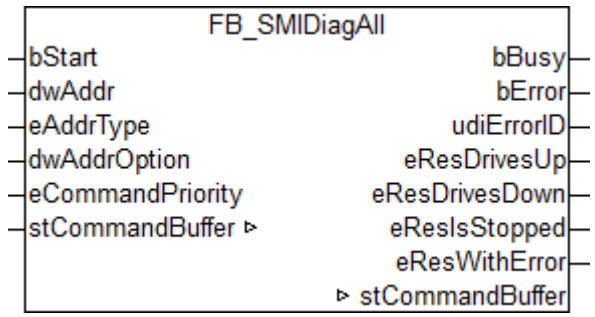
```
stCommandBuffer          : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.4 FB_SMIDiagAll



Using this command it is possible to determine the direction in which the drives are driving, whether they have stopped or whether there is a motor error. The command can also be sent also to several SMI slaves. The states of all SMI slaves can thus be queried with a single command.

The result of the query is forwarded by four outputs. Each of these outputs can assume three states:

- The condition applies to at least one drive.
- The condition does not apply to any drive.
- The condition could not be determined.

Some examples of this are explained further below.

VAR_INPUT

```
bStart            : BOOL;
dwAddr            : DWORD := 0;
eAddrType         : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption      : DWORD := 0;
eCommandPriority  : E_SMICommandPriority := eSMICommandPriorityMiddle;
```

bStart: the function block is started and the command is sent on applying a positive edge to this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device or for group addressing. Addressing by slave Id (*eAddrType* = *eSMIAddrTypeSlaveId*) is not permitted.

dwAddrOption: Reserved for future expansions.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

VAR_OUTPUT

```
bBusy           : BOOL;
bError          : BOOL;
udiErrorId      : UDINT;
eResDrivesUp    : E_SMIDiagResDrivesUp;
eResDrivesDown  : E_SMIDiagResDrivesDown;
eResIsStopped   : E_SMIDiagResIsStopped;
eResWithError   : E_SMIDiagResWithError;
```

bBusy: when the block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is again set to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

eResDrivesUp: at least one motor is driving up [▶ 55] / no motor is driving up / the value is undefined.

eResDrivesDown: at least one motor is driving down [▶ 55] / no motor is driving down / the value is undefined.

eResIsStopped: at least one motor has stopped [▶ 55] / no motor has stopped / the value is undefined.

eResWithError: at least one motor is in an error state [▶ 55] / no motor is in an error state / the value is undefined.

VAR_IN_OUT

```
stCommandBuffer   : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Examples

All drives have stopped:

Outputs	Meaning
eResDrivesUp = eSMIDiagResNoMotorDrivesUp	No drive is driving up.
eResDrivesDown = eSMIDiagResNoMotorDrivesDown	No drive is driving down.

Outputs	Meaning
eResIsStopped = eSMIDiagResAtLeastOneMotorIsStopped	At least one drive has stopped.
eResWithError = eSMIDiagResNoMotorWithError	No drive has a motor error.

All drives are driving up:

Outputs	Meaning
eResDrivesUp = eSMIDiagResAtLeastOneMotorDrivesUp	At least one drive is driving up.
eResDrivesDown = eSMIDiagResNoMotorDrivesDown	No drive is driving down.
eResIsStopped = eSMIDiagResNoMotorIsStopped	No drive has stopped.
eResWithError = eSMIDiagResNoMotorWithError	No drive has a motor error.

One drive has stopped and one drive is driving up:

Outputs	Meaning
eResDrivesUp = eSMIDiagResAtLeastOneMotorDrivesUp	At least one drive is driving up.
eResDrivesDown = eSMIDiagResNoMotorDrivesDown	No drive is driving down.
eResIsStopped = eSMIDiagResAtLeastOneMotorIsStopped	At least one drive has stopped.
eResWithError = eSMIDiagResNoMotorWithError	No drive has a motor error.

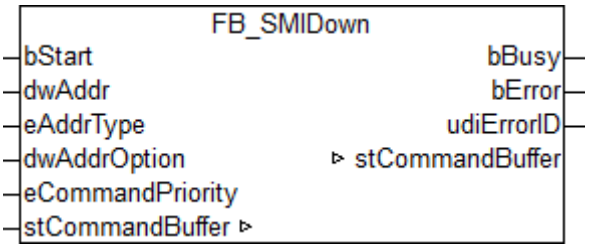
One drive has stopped, one drive is driving up and one drive is driving down:

Outputs	Meaning
eResDrivesUp = eSMIDiagResAtLeastOneMotorDrivesUp	At least one drive is driving up.
eResDrivesDown = eSMIDiagResAtLeastOneMotorDrivesDown	At least one drive is driving down.
eResIsStopped = eSMIDiagResAtLeastOneMotorIsStopped	At least one drive has stopped.
eResWithError = eSMIDiagResNoMotorWithError	No drive has a motor error.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.5 FB_SMIDown



Motor operation to the lower final position.

VAR_INPUT

```
bStart            : BOOL;
dwAddr            : DWORD := 0;
eAddrType         : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption      : DWORD := 0;
eCommandPriority  : E_SMICommandPriority := eSMICommandPriorityMiddle;
```

bStart: the function block is started and the command is sent on applying a positive edge to this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device, for group addressing or as a slave ID.

dwAddrOption: if the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 65] must be specified via this input.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

VAR_OUTPUT

```
bBusy         : BOOL;
bError        : BOOL;
udiErrorId    : UDINT;
```

bBusy: when the block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is again set to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

VAR_IN_OUT

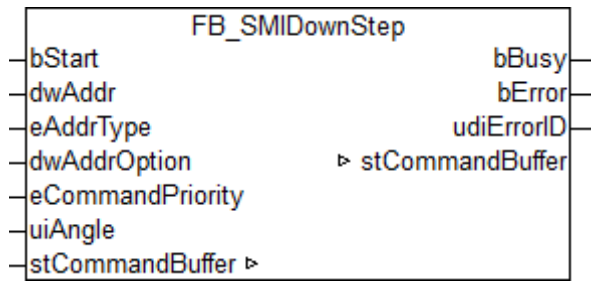
```
stCommandBuffer   : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.6 FB_SMIDownStep



Motor operation downwards by a specified angular degree (0-510 degrees).

VAR_INPUT

```
bStart            : BOOL;
dwAddr            : DWORD := 0;
eAddrType         : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption      : DWORD := 0;
eCommandPriority  : E_SMICommandPriority := eSMICommandPriorityMiddle;
uiAngle           : UINT := 0;
```

bStart: the function block is started and the command is sent on applying a positive edge to this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device, for group addressing or as a slave ID.

dwAddrOption: if the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 65] must be specified via this input.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

uiAngle: the specified angular degree. The range of values is 0 to 510 degrees. The SMI standard reduces the accuracy to a resolution of 2 degrees.

VAR_OUTPUT

```
bBusy         : BOOL;
bError        : BOOL;
udiErrorId    : UDINT;
```

bBusy: when the block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is again set to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

VAR_IN_OUT

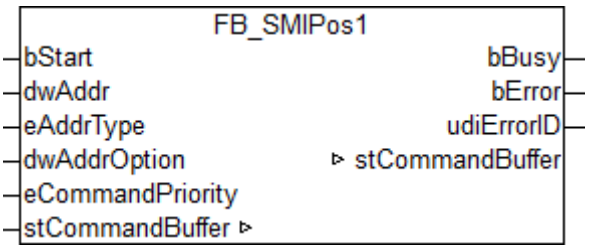
```
stCommandBuffer   : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.7 FB_SMIPos1



Drives to the fixed position *Pos1* configured on the motor side. The reading and changing of *Pos1* is possible with the FB_SMIPos1Read() [▶ 30] and FB_SMIPos1Write() [▶ 32] blocks.

VAR_INPUT

```
bStart            : BOOL;
dwAddr            : DWORD := 0;
eAddrType         : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption      : DWORD := 0;
eCommandPriority  : E_SMICommandPriority := eSMICommandPriorityMiddle;
```

bStart: the function block is started and the command is sent on applying a positive edge to this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device, for group addressing or as a slave ID.

dwAddrOption: if the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 65] must be specified via this input.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

VAR_OUTPUT

```
bBusy         : BOOL;
bError        : BOOL;
udiErrorId    : UDINT;
```

bBusy: when the block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is again set to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

VAR_IN_OUT

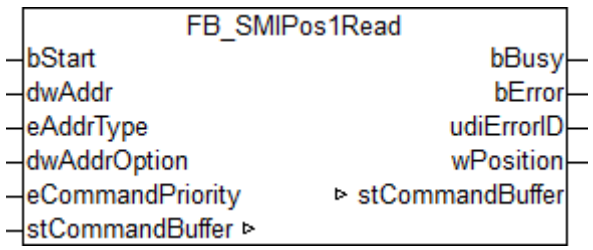
```
stCommandBuffer   : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.8 FB_SMIPos1Read



Reads the fixed position *Pos1* configured on the motor side. *Pos1* can be changed using the FB_SMIPos1Write() [▶ 32] block.

VAR_INPUT

```
bStart            : BOOL;
dwAddr            : DWORD := 0;
eAddrType         : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption      : DWORD := 0;
eCommandPriority  : E_SMICommandPriority := eSMICommandPriorityMiddle;
```

bStart: the function block is started and the command is sent on applying a positive edge to this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device or for group addressing. Addressing by slave Id (*eAddrType* = *eSMIAddrTypeSlaveId*) is not permitted.

dwAddrOption: Reserved for future expansions.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

VAR_OUTPUT

```
bBusy         : BOOL;
bError        : BOOL;
udiErrorId    : UDINT;
wPosition     : WORD;
```

bBusy: when the block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is again set to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

wPosition: The fixed position *Pos1* read out. The value 0 corresponds here to the upper final position and the value 65535 (0xFFFF) to the lower final position.

VAR_IN_OUT

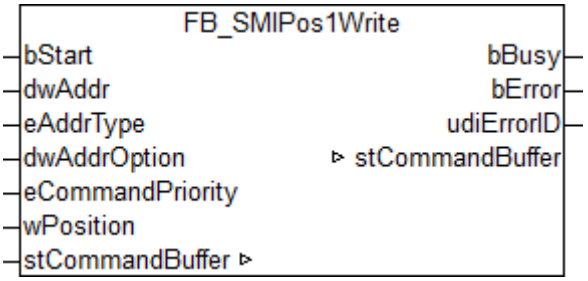
```
stCommandBuffer   : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.9 FB_SMIPos1Write



Writes the fixed position *Pos1* which is configurable on the motor side. *Pos1* can be changed using the FB_SMIPos1Read() [▶ 30] block.

VAR_INPUT

```
bStart            : BOOL;
dwAddr            : DWORD := 0;
eAddrType         : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption      : DWORD := 0;
eCommandPriority  : E_SMICommandPriority := eSMICommandPriorityMiddle;
wPosition         : WORD := 0;
```

bStart: the function block is started and the command is sent on applying a positive edge to this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device, for group addressing or as a slave ID.

dwAddrOption: if the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 65] must be specified via this input.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

wPosition: the new fixed position *Pos1*. The value 0 corresponds here to the upper final position and the value 65535 (0xFFFF) to the lower final position.

VAR_OUTPUT

```
bBusy         : BOOL;
bError        : BOOL;
udiErrorId    : UDINT;
```

bBusy: when the block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is again set to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

VAR_IN_OUT

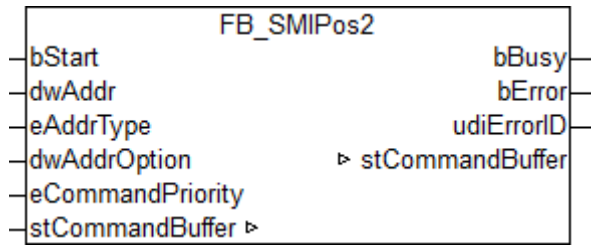
```
stCommandBuffer   : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.10 FB_SMIPos2



Drives to the fixed position *Pos2* configured on the motor side. The reading and changing of *Pos2* is possible with the FB_SMIPos2Read() [▶ 34] and FB_SMIPos2Write() [▶ 35] blocks.

VAR_INPUT

```
bStart            : BOOL;
dwAddr            : DWORD := 0;
eAddrType         : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption      : DWORD := 0;
eCommandPriority  : E_SMICommandPriority := eSMICommandPriorityMiddle;
```

bStart: the function block is started, and the command is sent on applying a positive edge to this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device, for group addressing or as a slave ID.

dwAddrOption: if the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 65] must be specified via this input.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

VAR_OUTPUT

```
bBusy         : BOOL;
bError        : BOOL;
udiErrorId    : UDINT;
```

bBusy: when the block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is again set to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

VAR_IN_OUT

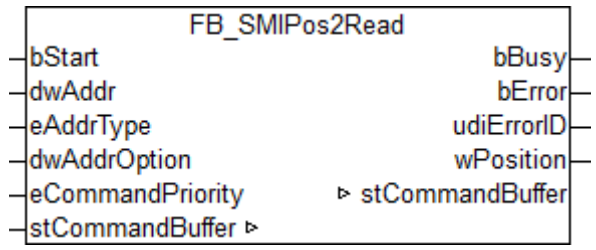
```
stCommandBuffer   : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.11 FB_SMIPos2Read



Reads the fixed position *Pos2* configured on the motor side. *Pos2* can be changed using the FB_SMIPos2Write() [▶ 35] block.

VAR_INPUT

```
bStart            : BOOL;
dwAddr            : DWORD := 0;
eAddrType         : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption      : DWORD := 0;
eCommandPriority  : E_SMICommandPriority := eSMICommandPriorityMiddle;
```

bStart: the function block is started and the command is sent on applying a positive edge to this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device or for group addressing. Addressing by slave Id (*eAddrType* = *eSMIAddrTypeSlaveId*) is not permitted.

dwAddrOption: Reserved for future expansions.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

VAR_OUTPUT

```
bBusy         : BOOL;
bError        : BOOL;
udiErrorId    : UDINT;
wPosition     : WORD;
```

bBusy: when the block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is again set to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

wPosition: The fixed position *Pos2* read. The value 0 corresponds here to the upper final position and the value 65535 (0xFFFF) to the lower final position.

VAR_IN_OUT

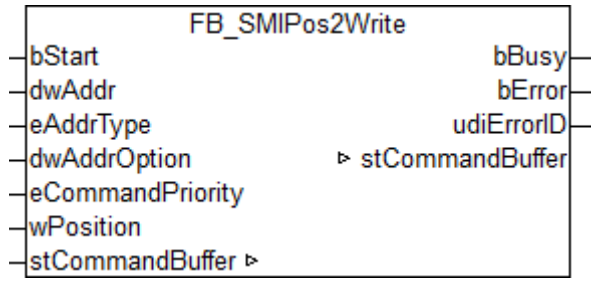
```
stCommandBuffer   : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.12 FB_SMIPos2Write



Writes the fixed position *Pos2* which is configurable on the motor side. *Pos2* can be changed using the FB_SMIPos2Read() [▶ 34] block.

VAR_INPUT

```
bStart            : BOOL;
dwAddr            : DWORD := 0;
eAddrType         : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption      : DWORD := 0;
eCommandPriority  : E_SMICommandPriority := eSMICommandPriorityMiddle;
wPosition         : WORD := 0;
```

bStart: the function block is started and the command is sent on applying a positive edge to this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device, for group addressing or as a slave ID.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

dwAddrOption: if the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 65] must be specified via this input.

wPosition: the new fixed position *Pos2*. The value 0 corresponds here to the upper final position and the value 65535 (0xFFFF) to the lower final position.

VAR_OUTPUT

```
bBusy         : BOOL;
bError        : BOOL;
udiErrorId    : UDINT;
```

bBusy: when the block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is again set to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

VAR_IN_OUT

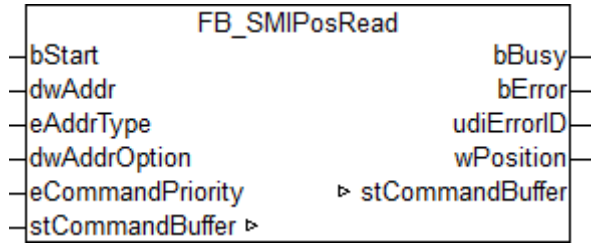
```
stCommandBuffer    : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.13 FB_SMIPosRead



The current position is read from the drive.

VAR_INPUT

```
bStart             : BOOL;
dwAddr             : DWORD := 0;
eAddrType          : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption       : DWORD := 0;
eCommandPriority   : E_SMICommandPriority := eSMICommandPriorityMiddle;
```

bStart: the function block is started, and the command is sent on applying a positive edge to this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device or for group addressing. Addressing by slave Id (*eAddrType* = *eSMIAddrTypeSlaveId*) is not permitted.

dwAddrOption: Reserved for future expansions.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

VAR_OUTPUT

```
bBusy          : BOOL;
bError         : BOOL;
udiErrorId     : UDINT;
wPosition      : WORD;
```

bBusy: when the block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is again set to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

wPosition: The position read out. The value 0 corresponds here to the upper final position and the value 65535 (0xFFFF) to the lower final position.

VAR_IN_OUT

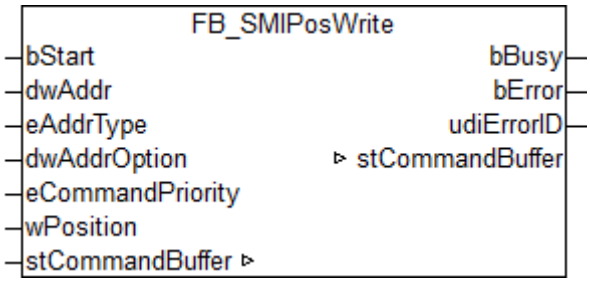
```
stCommandBuffer    : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.14 FB_SMIPosWrite



The drive is driven to the specified position.

VAR_INPUT

```
bStart            : BOOL;
dwAddr            : DWORD := 0;
eAddrType         : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption      : DWORD := 0;
eCommandPriority  : E_SMICommandPriority := eSMICommandPriorityMiddle;
wPosition         : WORD := 0;
```

bStart: the function block is started and the command is sent on applying a positive edge to this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device, for group addressing or as a slave ID.

dwAddrOption: if the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 65] must be specified via this input.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

wPosition: the new position. The value 0 corresponds here to the upper final position and the value 65535 (0xFFFF) to the lower final position.

VAR_OUTPUT

```
bBusy         : BOOL;
bError        : BOOL;
udiErrorId    : UDINT;
```

bBusy: when the block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is again set to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

VAR_IN_OUT

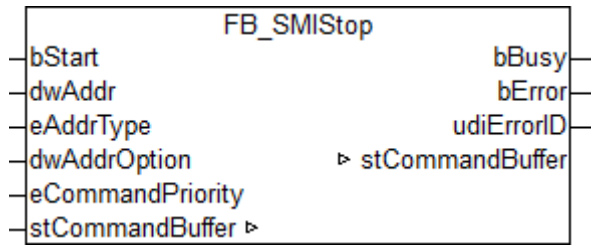
```
stCommandBuffer    : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.15 FB_SMIStop



The motor operation is stopped.

VAR_INPUT

```
bStart             : BOOL;
dwAddr             : DWORD := 0;
eAddrType          : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption       : DWORD := 0;
eCommandPriority   : E_SMICommandPriority := eSMICommandPriorityMiddle;
```

bStart: the function block is started and the command is sent on applying a positive edge to this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device, for group addressing or as a slave ID.

dwAddrOption: if the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 65] must be specified via this input.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

VAR_OUTPUT

```
bBusy          : BOOL;
bError         : BOOL;
udiErrorId     : UDINT;
```

bBusy: when the function block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is set back to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

VAR_IN_OUT

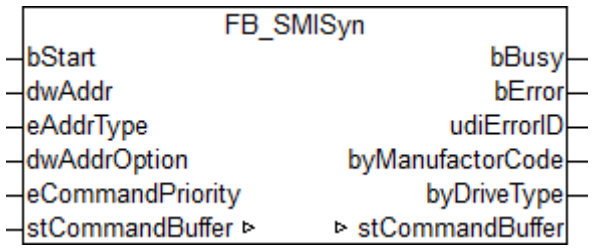
```
stCommandBuffer    : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.16 FB_SMISyn



The manufacturer code and the drive type are queried.

VAR_INPUT

```
bStart             : BOOL;
dwAddr             : DWORD := 0;
eAddrType          : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption       : DWORD := 0;
eCommandPriority   : E_SMICommandPriority := eSMICommandPriorityMiddle;
```

bStart: the function block is started and the command is sent on applying a positive edge to this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device or for group addressing. Addressing by slave Id (*eAddrType* = *eSMIAddrTypeSlaveId*) is not permitted.

dwAddrOption: Reserved for future expansions.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

VAR_OUTPUT

```
bBusy              : BOOL;
bError             : BOOL;
udiErrorId         : UDINT;
byManufactorCode   : BYTE;
byDriveType        : BYTE;
```

bBusy: when the function block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is set back to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

byManufactorCode: the manufacturer code [▶ 65] (1-14).

byDriveType: the type of drive (0-15). The meaning is manufacturer-specific.

VAR_IN_OUT

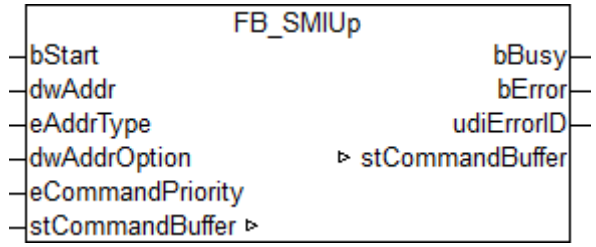
```
stCommandBuffer    : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.17 FB_SMIUp



Motor operation to the upper final position.

VAR_INPUT

```
bStart            : BOOL;
dwAddr            : DWORD := 0;
eAddrType         : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption      : DWORD := 0;
eCommandPriority  : E_SMICommandPriority := eSMICommandPriorityMiddle;
```

bStart: the function block is started and the command is sent on applying a positive edge to this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device, for group addressing or as a slave ID.

dwAddrOption: if the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 65] must be specified via this input.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

VAR_OUTPUT

```
bBusy         : BOOL;
bError        : BOOL;
udiErrorId    : UDINT;
```

bBusy: when the function block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is set back to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

VAR_IN_OUT

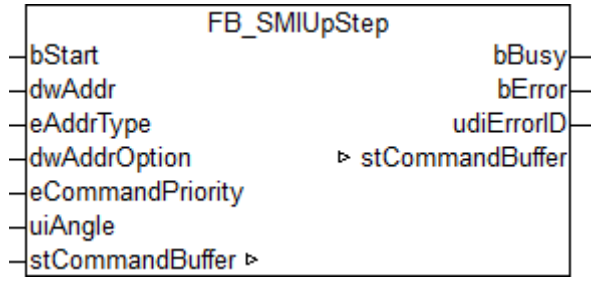
```
stCommandBuffer    : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.18 FB_SMIUpStep



Motor operation upwards by a specified angle (0-510 degrees).

VAR_INPUT

```
bStart             : BOOL;
dwAddr             : DWORD := 0;
eAddrType          : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption       : DWORD := 0;
eCommandPriority   : E_SMICommandPriority := eSMICommandPriorityMiddle;
uiAngle            : UINT := 0;
```

bStart: the function block is started and the command is sent on applying a positive edge to this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device, for group addressing or as a slave ID.

dwAddrOption: if the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 65] must be specified via this input.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

uiAngle: the specified angular degree. The range of values is 0 to 510 degrees. The SMI standard reduces the accuracy to a resolution of 2 degrees.

VAR_OUTPUT

```
bBusy          : BOOL;
bError         : BOOL;
udiErrorId     : UDINT;
```

bBusy: when the function block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is set back to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

VAR_IN_OUT

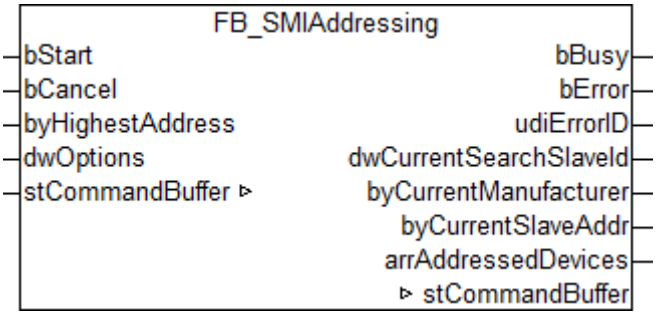
```
stCommandBuffer    : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.19 FB_SMIAddressing



This function block addresses the connected SMI devices according to the random principle. The user has no influence over which address is assigned to which SMI device. The addresses are assigned in descending order, starting with the address specified by the *byHighestAddress* parameter.

Applying a positive edge to the *bStart* input starts the block, and the *bBusy* output goes TRUE. The function block now independently addresses all SMI devices. The output variable *arrAddressedDevices* provides information which SMI devices have already received an address. Once all SMI devices have been addressed, the *bBusy* output goes FALSE again. Addressing can be aborted through a positive edge at input *bCancel*. Processing this function block can take several minutes, depending on how many SMI devices are connected.

VAR_INPUT

```
bStart           : BOOL;
bCancel          : BOOL;
byHighestAddress : BYTE := 15;
dwOptions        : DWORD := 0;
```

bStart: the function block is started and the command is sent on applying a positive edge to this input.

bCancel: the function block is deactivated and the search is aborted on applying a positive edge to this input.

byHighestAddress: address from which the SMI devices are addressed in descending order (0-15).

dwOptions: reserved for future extensions.

VAR_OUTPUT

```
bBusy                   : BOOL;
bError                  : BOOL;
udiErrorId              : UDINT;
dwCurrentSearchSlaveId  : DWORD;
byCurrentManufacturer   : BYTE;
byCurrentSlaveAddr      : BYTE;
arrAddressedDevices     : ARRAY [0..15] OF BOOL;
```

bBusy: when the block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is again set to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

dwCurrentSearchSlaveId: current slave ID used in the search algorithm.

byCurrentManufacturer: current manufacturer code [▶ 65] used in the search algorithm.

byCurrentSlaveAddr: current address used in the search algorithm. As long as the value is greater than *byHighestAddress*, still no SMI device was addressed.

arrAddressedDevices: if an address is assigned to an SMI device, then the corresponding element in the array is set to TRUE.

VAR_IN_OUT

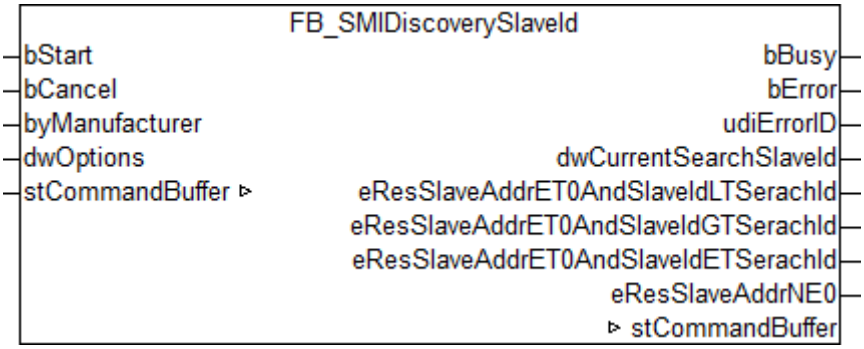
```
stCommandBuffer      : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.20 FB_SMIDiscoverySlaveId



The first drive is sought that corresponds to the specified manufacturer code and has the address 0. This function block is used in the addressing of SMI devices and is used in the FB_SMIAddressing [▶ 42] block.

VAR_INPUT

```
bStart                  : BOOL;
bCancel                 : BOOL;
byManufacturer          : BYTE := 0;
dwOptions               : DWORD := 0;
```

bStart: the function block is activated, and the search is started by applying a positive edge to this input.

bCancel: the function block is deactivated and the search is aborted on applying a positive edge to this input.

byManufacturer: the manufacturer code [▶ 65] specified for the search for the SMI device. Some SMI devices do not permit the manufacturer code 0.

dwOptions: reserved for future expansions.

VAR_OUTPUT

```
bBusy                                : BOOL;
bError                               : BOOL;
udiErrorId                           : UDINT;
dwCurrentSearchSlaveId               : DWORD;
eResSlaveAddrET0AndSlaveIdLTSerachId  : E_SMICompResSlaveAddrET0AndSlaveIdLTSearchId;
eResSlaveAddrET0AndSlaveIdGTSerachId  : E_SMICompResSlaveAddrET0AndSlaveIdGTSearchId;
eResSlaveAddrET0AndSlaveIdETSerachId  : E_SMICompResSlaveAddrET0AndSlaveIdETSearchId;
eResSlaveAddrNE0                     : E_SMICompResSlaveAddrNE0;
```

bBusy: when the block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is again set to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

dwCurrentSearchSlaveId: as soon as the execution of the function block has ended (*bBusy* changes from TRUE to FALSE) this output indicates the slave ID of the SMI device found.

eResSlaveAddrET0AndSlaveIdLTSerachId: at least one motor / no motor has the address 0 and the slave ID is smaller than the slave [▶ 54] ID sought (*dwSlave-Id*) / the value is undefined.

eResSlaveAddrET0AndSlaveIdGTSerachId: at least one motor / no motor has the address 0 and the slave ID is larger than the slave [▶ 54] ID sought (*dwSlave-Id*) / the value is undefined.

eResSlaveAddrET0AndSlaveIdETSerachId: at least one motor / no motor has the address 0 and the slave ID is also the same as the slave [▶ 54] ID sought (*dwSlave-Id*) / the value is undefined.

eResSlaveAddrNE0: at least one motor / no motor has an address unequal 0 [▶ 54] / the value is undefined.

VAR_IN_OUT

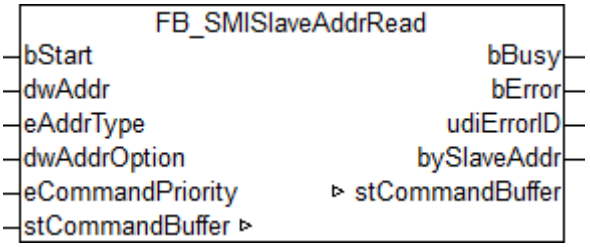
```
stCommandBuffer             : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.21 FB_SMISlaveAddrRead



The address (0-15) of a drive is read.

VAR_INPUT

```
bStart            : BOOL;
dwAddr            : DWORD := 0;
eAddrType         : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption      : DWORD := 0;
eCommandPriority  : E_SMICommandPriority := eSMICommandPriorityMiddle;
```

bStart: the function block is started and the command is sent on applying a positive edge to this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device, for group addressing or as a slave ID.

dwAddrOption: if the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 65] must be specified via this input.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

VAR_OUTPUT

```
bBusy         : BOOL;
bError        : BOOL;
udiErrorId    : UDINT;
bySlaveAddr   : BYTE;
```

bBusy: when the block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is again set to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

bySlaveAddr: The slave address (0-15) read out.

VAR_IN_OUT

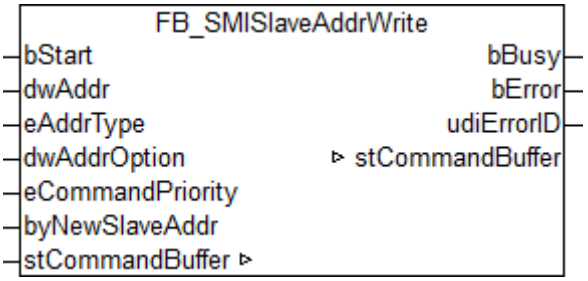
```
stCommandBuffer   : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.22 FB_SMISlaveAddrWrite



Writes the address (0-15) of one or more drives.

VAR_INPUT

```
bStart            : BOOL;
dwAddr            : DWORD := 0;
eAddrType         : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption      : DWORD := 0;
eCommandPriority  : E_SMICommandPriority := eSMICommandPriorityMiddle;
byNewSlaveAddr    : BYTE := 0;
```

bStart: the function block is started, and the command is sent on applying a positive edge to this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device, for group addressing or as a slave ID.

dwAddrOption: if the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 65] must be specified via this input.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

byNewSlaveAddr: the new slave address (0-15).

VAR_OUTPUT

```
bBusy         : BOOL;
bError        : BOOL;
udiErrorId    : UDINT;
```

bBusy: when the block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is again set to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

VAR_IN_OUT

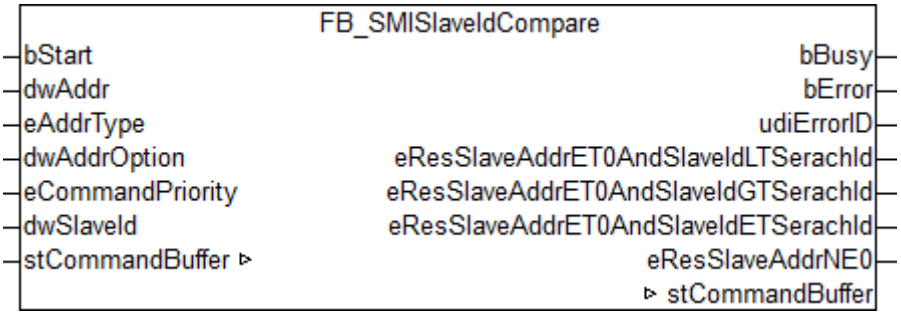
```
stCommandBuffer   : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.23 FB_SMISlaveIdCompare



A specified slave ID (32-bit key ID) is compared with the slave ID (32-bit key ID) of one or more drives which is defined on the motor side. The command can also be sent also to several SMI slaves.

The result of the query is forwarded by four outputs. Each of these outputs can assume three states:

- The condition applies to at least one drive.
- The condition does not apply to any drive.
- The condition could not be determined.

Some examples of this are explained further below.

VAR_INPUT

```
bStart                : BOOL;
dwAddr                : DWORD := 0;
eAddrType             : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption          : DWORD := 0;
eCommandPriority      : E_SMICommandPriority := eSMICommandPriorityMiddle;
dwSlaveId             : DWORD := 0;
```

bStart: the function block is started, and the command is sent on applying a positive edge to this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device, for group addressing or as a slave ID.

dwAddrOption: if the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 65] must be specified via this input.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

dwSlaveId: the Slave ID with which the Slave ID on the motor side is compared.

VAR_OUTPUT

```
bBusy                                : BOOL;
bError                               : BOOL;
udiErrorId                           : UDINT;
eResSlaveAddrET0AndSlaveIdLTSerachId  : E_SMICompResSlaveAddrET0AndSlaveIdLTSearchId;
eResSlaveAddrET0AndSlaveIdGTSerachId  : E_SMICompResSlaveAddrET0AndSlaveIdGTSearchId;
eResSlaveAddrET0AndSlaveIdETSerachId  : E_SMICompResSlaveAddrET0AndSlaveIdETSearchId;
eResSlaveAddrNE0                     : E_SMICompResSlaveAddrNE0;
```

bBusy: when the function block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is set back to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

eResSlaveAddrET0AndSlaveIdLTSerachId: at least one motor / no motor has the address 0 and the slave ID is smaller than the slave [▶ 54] ID sought (*dwSlaveId*) / the value is undefined.

eResSlaveAddrET0AndSlaveIdGTSerachId: at least one motor / no motor has the address 0 and the slave ID is larger than the slave [▶ 54] ID sought (*dwSlaveId*) / the value is undefined.

eResSlaveAddrET0AndSlaveIdETSerachId: at least one motor / no motor has the address 0 and the slave ID is also the same as the slave [▶ 54] ID sought (*dwSlaveId*) / the value is undefined.

eResSlaveAddrNE0: at least one motor / no motor has an address unequal 0 [▶ 54] / the value is undefined.

VAR_IN_OUT

```
stCommandBuffer             : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Examples

The following tables show the results of the function block with different initial situations. In all cases two SMI devices are connected to an SMI terminal and both addresses are greater than 0.

The slave ID () sought lies between the slave IDs of the two drives: *dwSlaveId*

Outputs	Meaning
eResSlaveAddrET0AndSlaveIdLTSerachId = eSMIDiagResAtLeastOneSlaveAddrET0AndSlaveIdLTSearchId	At least one motor has the slave address equal 0 and the slave ID is smaller than the slave ID sought.
eResSlaveAddrET0AndSlaveIdGTSerachId = eSMIDiagResAtLeastOneSlaveAddrET0AndSlaveIdGTSearchId	At least one motor has the slave address equal 0 and the slave ID is greater than the slave ID sought.
eResSlaveAddrET0AndSlaveIdETSerachId = eSMIDiagResNoSlaveAddrET0AndSlaveIdETSearchId	No motor has the slave address equal 0 and the slave ID is the same as the slave ID sought.
eResSlaveAddrNE0 = eSMIDiagResNoSlaveAddrNE0	No motor has the slave address unequal 0.

The slave ID () sought is greater than the slave IDs of the two drives: *dwSlaveId*

Outputs	Meaning
eResSlaveAddrET0AndSlaveIdLTSerachId = eSMIDiagResAtLeastOneSlaveAddrET0AndSlaveIdLTSearchId	At least one motor has the slave address equal 0 and the slave ID is smaller than the slave ID sought.
eResSlaveAddrET0AndSlaveIdGTSerachId = eSMIDiagResNoSlaveAddrET0AndSlaveIdGTSearchId	No motor has the slave address equal 0 and the slave ID is greater than the slave ID sought.
eResSlaveAddrET0AndSlaveIdETSerachId = eSMIDiagResNoSlaveAddrET0AndSlaveIdLTSearchId	No motor has the slave address equal 0 and the slave ID is smaller than the slave ID sought.
eResSlaveAddrNE0 = eSMIDiagResNoSlaveAddrNE0	No motor has the slave address unequal 0.

The slave ID () sought is smaller than the slave IDs of the two drives: *dwSlaveId*

Outputs	Meaning
eResSlaveAddrET0AndSlaveIdLTSerachId = eSMIDiagResNoSlaveAddrET0AndSlaveIdLTSearchId	No motor has the slave address equal 0 and the slave ID is smaller than the slave ID sought.

Outputs	Meaning
eResSlaveAddrET0AndSlaveIdGTSerachId = eSMIDiagResAtLeastOneSlaveAddrET0AndSlaveIdGTSearchId	At least one motor has the slave address equal 0 and the slave ID is greater than the slave ID sought.
eResSlaveAddrET0AndSlaveIdETSerachId = eSMIDiagResNoSlaveAddrET0AndSlaveIdETSearchId	No motor has the slave address equal 0 and the slave ID is the same as the slave ID sought.
eResSlaveAddrNE0 = eSMIDiagResNoSlaveAddrNE0	No motor has the slave address unequal 0.

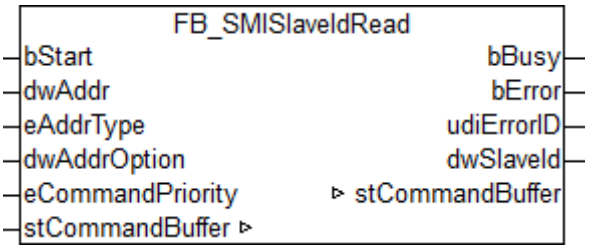
The slave ID () sought is the same as the slave ID of a drive: *dwSlaveId*

Outputs	Meaning
eResSlaveAddrET0AndSlaveIdLTSerachId = eSMIDiagResNoSlaveAddrET0AndSlaveIdLTSearchId	No motor has the slave address equal 0 and the slave ID is smaller than the slave ID sought.
eResSlaveAddrET0AndSlaveIdGTSerachId = eSMIDiagResAtLeastOneSlaveAddrET0AndSlaveIdGTSearchId	At least one motor has the slave address equal 0 and the slave ID is greater than the slave ID sought.
eResSlaveAddrET0AndSlaveIdETSerachId = eSMIDiagResAtLeastOneSlaveAddrET0AndSlaveIdETSearchId	At least one motor has the slave address equal 0 and the slave ID is the same as the slave ID sought.
eResSlaveAddrNE0 = eSMIDiagResNoSlaveAddrNE0	No motor has the slave address unequal 0.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.24 FB_SMISlaveIdRead



The slave ID (32-bit key ID) is read from a drive.

VAR_INPUT

```
bStart            : BOOL;
dwAddr            : DWORD := 0;
eAddrType         : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption      : DWORD := 0;
eCommandPriority  : E_SMICommandPriority := eSMICommandPriorityMiddle;
```

bStart: the function block is started and the command is sent on applying a positive edge to this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device, for group addressing or as a slave ID.

dwAddrOption: if the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 65] must be specified via this input.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

VAR_OUTPUT

```
bBusy          : BOOL;
bError         : BOOL;
udiErrorId     : UDINT;
dwSlaveId      : DWORD;
```

bBusy: when the function block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is set back to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

dwSlaveId: the slave ID read out.

VAR_IN_OUT

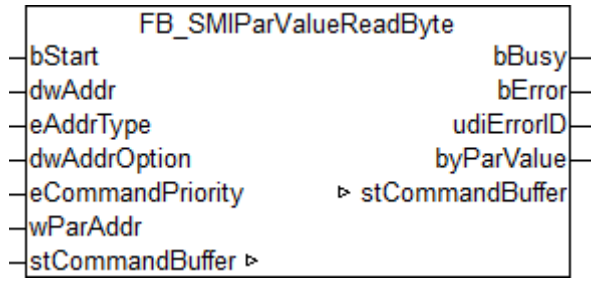
```
stCommandBuffer    : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.25 FB_SMIParValueReadByte



Reads a byte parameter (1 byte) stored on the motor side. The meaning of the individual parameters is manufacturer-specific.

VAR_INPUT

```
bStart             : BOOL;
dwAddr             : DWORD := 0;
eAddrType          : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption       : DWORD := 0;
eCommandPriority   : E_SMICommandPriority := eSMICommandPriorityMiddle;
wParAddr           : WORD := 0;
```

bStart: the function block is started, and the command is sent on applying a positive edge to this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device or for group addressing. Addressing by slave Id (*eAddrType* = *eSMIAddrTypeSlaveId*) is not permitted.

dwAddrOption: Reserved for future expansions.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

wParAddr: address of the parameter (0-4095) to be read.

VAR_OUTPUT

```
bBusy          : BOOL;
bError         : BOOL;
udiErrorId     : UDINT;
byParValue     : BYTE;
```

bBusy: when the block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is again set to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

byParValue: the byte parameter read out.

VAR_IN_OUT

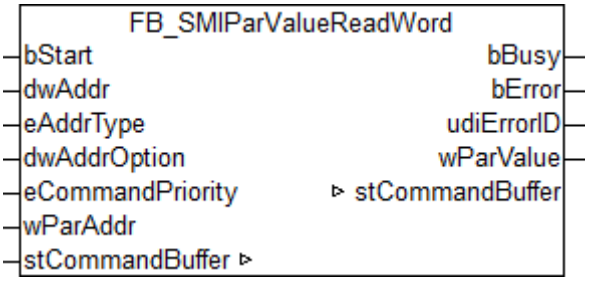
```
stCommandBuffer    : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.26 FB_SMIParValueReadWord



Reads a Word parameter (2 bytes) stored on the motor side. The meaning of the individual parameters is manufacturer-specific.

VAR_INPUT

```
bStart             : BOOL;
dwAddr             : DWORD := 0;
eAddrType          : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption       : DWORD := 0;
eCommandPriority   : E_SMICommandPriority := eSMICommandPriorityMiddle;
wParAddr           : WORD := 0;
```

bStart: the function block is started and the command is sent on applying a positive edge to this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device or for group addressing. Addressing by slave Id (*eAddrType* = *eSMIAddrTypeSlaveId*) is not permitted.

dwAddrOption: Reserved for future expansions.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

wParAddr: address of the parameter (0-4095) to be read.

VAR_OUTPUT

```
bBusy          : BOOL;
bError         : BOOL;
udiErrorId     : UDINT;
wParValue      : WORD;
```

bBusy: when the block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is again set to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

wParValue: the Word parameter read out.

VAR_IN_OUT

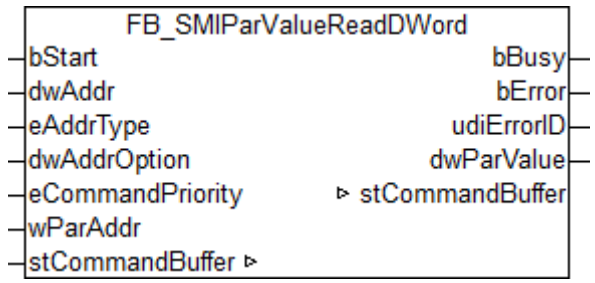
```
stCommandBuffer    : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.1.27 FB_SMIParValueReadDWord



Reads a DWord parameter (4 bytes) stored on the motor side. The meaning of the individual parameters is manufacturer-specific.

VAR_INPUT

```
bStart            : BOOL;
dwAddr            : DWORD := 0;
eAddrType         : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption      : DWORD := 0;
eCommandPriority  : E_SMICommandPriority := eSMICommandPriorityMiddle;
wParAddr          : WORD := 0;
```

bStart: the function block is started, and the command is sent on applying a positive edge to this input.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing or slave ID (32-bit key ID). This input has no meaning if a collective call (broadcast) is sent.

eAddrType: defines [▶ 53] whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device or for group addressing. Addressing by slave Id (*eAddrType* = *eSMIAddrTypeSlaveId*) is not permitted.

dwAddrOption: Reserved for future expansions.

eCommandPriority: priority [▶ 54] (high, medium or low) with which the command is processed by the PLC library.

wParAddr: address of the parameter (0-4095) to be read.

VAR_OUTPUT

```
bBusy         : BOOL;
bError        : BOOL;
udiErrorId    : UDINT;
dwParValue    : DWORD;
```

bBusy: when the block is activated the output is set, and it remains active until execution of the command has been completed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is again set to FALSE by the reactivation of the function block via the *bStart* input.

udiErrorId: contains the command-specific error code of the most recently executed command. It is set back to 0 by the reactivation of the function block via the *bStart* input. See Error codes [▶ 57].

dwParValue: the DWord parameter read out.

VAR_IN_OUT

```
stCommandBuffer   : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

5.2 Data types

5.2.1 E_SMIAddrType

```
TYPE E_SMIAddrType :
(
  eSMIAddrTypeManufacturer  := 0,
  eSMIAddrTypeAddress       := 1,
  eSMIAddrTypeGroup         := 2,
  eSMIAddrTypeSlaveId       := 3,
```

```
  eSMIAddrTypeBroadcast     := 4
);
END_TYPE
```

5.2.2 E_SMICommandPriority

```
TYPE E_SMICommandPriority :
(
  eSMICommandPriorityHigh     := 0,
  eSMICommandPriorityMiddle   := 1,
  eSMICommandPriorityLow      := 2
);
END_TYPE
```

5.2.3 E_SMICommandType

```
TYPE E_SMICommandType :
(
  eSMICommandTypeWrite    := 0,
  eSMICommandTypeRead     := 1
);
END_TYPE
```

5.2.4 E_SMICompResSlaveAddrET0AndSlaveIdETSearchId

```
TYPE E_SMICompResSlaveAddrET0AndSlaveIdETSearchId :
(
  eSMIDiagResSlaveAddrET0AndSlaveIdETSearchIdUndefined    := 0,
  eSMIDiagResNoSlaveAddrET0AndSlaveIdETSearchId           := 1,
  eSMIDiagResAtLeastOneSlaveAddrET0AndSlaveIdETSearchId   := 2
);
END_TYPE
```

5.2.5 E_SMICompResSlaveAddrET0AndSlaveIdGTSearchId

```
TYPE E_SMICompResSlaveAddrET0AndSlaveIdGTSearchId :
(
  eSMIDiagResSlaveAddrET0AndSlaveIdGTSearchIdUndefined    := 0,
  eSMIDiagResNoSlaveAddrET0AndSlaveIdGTSearchId           := 1,
  eSMIDiagResAtLeastOneSlaveAddrET0AndSlaveIdGTSearchId   := 2
);
END_TYPE
```

5.2.6 E_SMICompResSlaveAddrET0AndSlaveIdLTSearchId

```
TYPE E_SMICompResSlaveAddrET0AndSlaveIdLTSearchId :
(
  eSMIDiagResSlaveAddrET0AndSlaveIdLTSearchIdUndefined    := 0,
  eSMIDiagResNoSlaveAddrET0AndSlaveIdLTSearchId           := 1,
  eSMIDiagResAtLeastOneSlaveAddrET0AndSlaveIdLTSearchId   := 2
);
END_TYPE
```

5.2.7 E_SMICompResSlaveAddrNE0

```
TYPE E_SMICompResSlaveAddrNE0 :
(
  eSMIDiagResSlaveAddrNE0Undefined    := 0,
  eSMIDiagResNoSlaveAddrNE0           := 1,
  eSMIDiagResAtLeastOneSlaveAddrNE0   := 2
);
END_TYPE
```

5.2.8 E_SMIConfigurationCommands

```
TYPE E_SMIConfigurationCommands :
(
  eSMICommandDoNothing   := 0,
  eSMICommandUp          := 1,
  eSMICommandDown        := 2,
  eSMICommandStop        := 3,
```

```
  eSMICommandPos1        := 4,
  eSMICommandPos2        := 5
);
END_TYPE
```

5.2.9 E_SMIDiagResDrivesDown

```
TYPE E_SMIDiagResDrivesDown :
(
  eSMIDiagResDrivesDownUndefined          := 0,
  eSMIDiagResNoMotorDrivesDown            := 1,
  eSMIDiagResAtLeastOneMotorDrivesDown    := 2
);
END_TYPE
```

5.2.10 E_SMIDiagResDrivesUp

```
TYPE E_SMIDiagResDrivesUp :
(
  eSMIDiagResDrivesUpUndefined        := 0,
  eSMIDiagResNoMotorDrivesUp          := 1,
  eSMIDiagResAtLeastOneMotorDrivesUp  := 2
);
END_TYPE
```

5.2.11 E_SMIDiagResIsStopped

```
TYPE E_SMIDiagResIsStopped :
(
  eSMIDiagResIsStoppedUndefined       := 0,
  eSMIDiagResNoMotorIsStopped         := 1,
  eSMIDiagResAtLeastOneMotorIsStopped := 2
);
END_TYPE
```

5.2.12 E_SMIDiagResponse

```
TYPE E_SMIDiagResponse :
(
  eSMIDiagResponseDiscard    := 0,
  eSMIDiagResponseAck        := 1,
  eSMIDiagResponseNack       := 2
);
END_TYPE
```

5.2.13 E_SMIDiagResWithError

```
TYPE E_SMIDiagResWithError :
(
  eSMIDiagResWithErrorUndefined       := 0,
  eSMIDiagResNoMotorWithError         := 1,
  eSMIDiagResAtLeastOneMotorWithError := 2
);
END_TYPE
```

5.2.14 E_SMIResponseFormat

```
TYPE E_SMIResponseFormat :
(
  eSMIResponseFormatDiagnosis    := 0,
  eSMIResponseFormatStandard     := 1
);
END_TYPE
```

5.2.15 ST_KL6831KL6841InData

```
TYPE ST_KL6831KL6841InData :
STRUCT
  wStateWord   : WORD;
```

```
  arrData      : ARRAY [0..21] OF BYTE;
END_STRUCT
END_TYPE
```

5.2.16 ST_KL6831KL6841OutData

```
TYPE ST_KL6831KL6841OutData :
STRUCT
  wControlWord    : WORD;
  arrData         : ARRAY [0..21] OF BYTE;
END_STRUCT
END_TYPE
```

5.2.17 ST_SMICommandBuffer

```
TYPE ST_SMICommandBuffer :
STRUCT
  arrMessageQueue      : ARRAY [0..2] OF ST_SMIMessageQueue;
  stResponseTable      : ST_SMIResponseTable;
  udiMessageHandle     : UDINT;
END_STRUCT
END_TYPE
```

Also see about this

- ST_SMIMessageQueue [▶ 56]

5.2.18 ST_SMIMessageQueue

```
TYPE ST_SMIMessageQueue :
STRUCT
  arrBuffer                    : ARRAY [1..SMI_COMMAND_BUFFER_ENTRIES] OF ST_SMIMessageQueueItem;
  byBufferReadPointer          : BYTE;
  byBufferWritePointer         : BYTE;
  byBufferDemandCounter        : BYTE;
  byBufferMaximumDemandCounter : BYTE;
  uiBufferOverflowCounter      : UINT;
  bLockSemaphore               : BOOL;
END_STRUCT
END_TYPE
```

Also see about this

- ST_SMIMessageQueueItem [▶ 56]

5.2.19 ST_SMIMessageQueueItem

```
TYPE ST_SMIMessageQueueItem :
STRUCT
  dwAddr                  : DWORD;
  eAddrType               : E_SMIAddrType;
  eCommandType            : E_SMICommandType;
  eResponseFormat         : E_SMIResponseFormat;
  arrIdentificationBytes  : ARRAY [0..2] OF BYTE;
  arrParameters           : ARRAY [0..2] OF DWORD;
  udiMessageHandle        : UDINT;
  bSuppressResponseBuffer : BOOL;
END_STRUCT
END_TYPE
```

Also see about this

- E_SMIAddrType [▶ 53]
- E_SMICommandType [▶ 54]
- E_SMIResponseFormat [▶ 55]

5.2.20 ST_SMIResponseTable

```
TYPE ST_SMIResponseTable :
STRUCT
  arrResponseTable               : ARRAY [1..SMI_COMMAND_BUFFER_ENTRIES] OF ST_SMIResponseTableItem
```

```
;
  byResponseTableCounter          : BYTE;
  byResponseTableMaxCounter       : BYTE;
  uiResponseTableOverflowCounter  : UINT;
  bLockSemaphore                  : BOOL;
END_STRUCT
END_TYPE
```

Also see about this

ST_SMIResponseTableItem [▶ 57]

5.2.21 ST_SMIResponseTableItem

```
TYPE ST_SMIResponseTableItem :
STRUCT
  arrResponseData         : ARRAY [0..7] OF BYTE;
  byDataLength            : BYTE;
  byIdentificationByte    : BYTE;
  udiMessageHandle        : UDINT;
  udiErrorId              : UDINT;
END_STRUCT
END_TYPE
```

5.3 Error codes

Value (hex)	Value (dec)	Description
0x0000	0	No error.
0x8001	32769	No response from the SMI drive.
0x8002	32770	No terminal feedback for the transmit data from the SMI terminal.
0x8003	32771	The terminal has detected a telegram error (StatusWord.6 = true). This message must be acknowledged by the *bResetDataFrameError* input of FB_KL6831KL6841Communication() [▶ 19].
0x8004	32772	NACK received from the drive.
0x8005	32773	Invalid feedback received from the drive.
0x8006	32774	Communication buffer overflow.
0x8007	32775	No response from the communication block.
0x8008	32776	The *SMI_COMMAND_BUFFER_ENTRIES* constant is outside the valid range (2-250).
0x8009	32777	The ID byte received is incorrect.
0x800A	32778	The data length received is not correct.
0x800B	32779	No 24 V DC supply voltage to the KL6831/KL6841 (StatusWord.2 = false).
0x800C	32780	Process image was deactivated by the Switch1 or Switch2 input of the terminal (StatusWord.5 = true). This message must be acknowledged by the *bResetInactiveProcessImage* input of FB_KL6831KL6841Communication() [▶ 19].
0x800D	32781	The terminal has detected a checksum error (StatusWord.8 = true). This message is reset as soon as a telegram is successfully transmitted.
0x800E	32782	The SMI command does not support addressing by slave ID *(eAddrType = eSMIAddrTypeSlaveId).*
0x800F	32783	The *dwAddr* parameter (bit field for group addressing) is outside the valid range (0-65535).
0x8010	32784	The *dwAddr* parameter (address) is outside the valid range (0-15).
0x8011	32785	The *eCommandPriority* parameter is invalid.
0x8012	32786	The *eCommandType* parameter is invalid.
0x8013	32787	The *uiAngle* parameter is outside the valid range (0-510).
0x8014	32788	The *wParAddr* parameter is outside the valid range (0-4095).
0x8015	32789	The *eAddrType* parameter is invalid.
0x8016	32790	The *eResponseFormat* parameter is invalid.

Value (hex)	Value (dec)	Description
0x8017	32791	The *dwAddr* parameter (manufacturer code) is outside the valid range (0-15).
0x8018	32792	The command supports only individual addressing.
0x8019	32793	The *dwAddrOption* parameter (manufacturer code) is outside the valid range (0-15).
0x801A	32794	An internal error has occurred in the FB_SMIDiscoverySlaveId() [▶ 43] function block.
0x801B	32795	No devices were found.
0x801C	32796	All 16 addresses have already been assigned. There are possibly more than 16 devices connected to the SMI bus.
0x801D	32797	Invalid diagnostic response received (neither NACK nor ACK).
0x801E	32798	The *byHighestAddress* parameter (highest address) is outside the valid range (0-15).
0x801F	32799	FB_KL6831KL6841Config() [▶ 21]: An error occurred during configuration of the terminal.
0x8020	32800	FB_KL6831KL6841Config() [▶ 21]: Parameter *eCommandKBusWatchdog* is not in the valid range.
0x8021	32801	FB_KL6831KL6841Config() [▶ 21]: Parameter *eCommandDI1RisingEdge* is not in the valid range.
0x8022	32802	FB_KL6831KL6841Config() [▶ 21]: Parameter *eCommandDI1FallingEdge* is not in the valid range.
0x8023	32803	FB_KL6831KL6841Config() [▶ 21]: Parameter *eCommandDI2RisingEdge* is not in the valid range.
0x8024	32804	FB_KL6831KL6841Config() [▶ 21]: Parameter *eCommandDI2FallingEdge* is not in the valid range.

6 Appendix

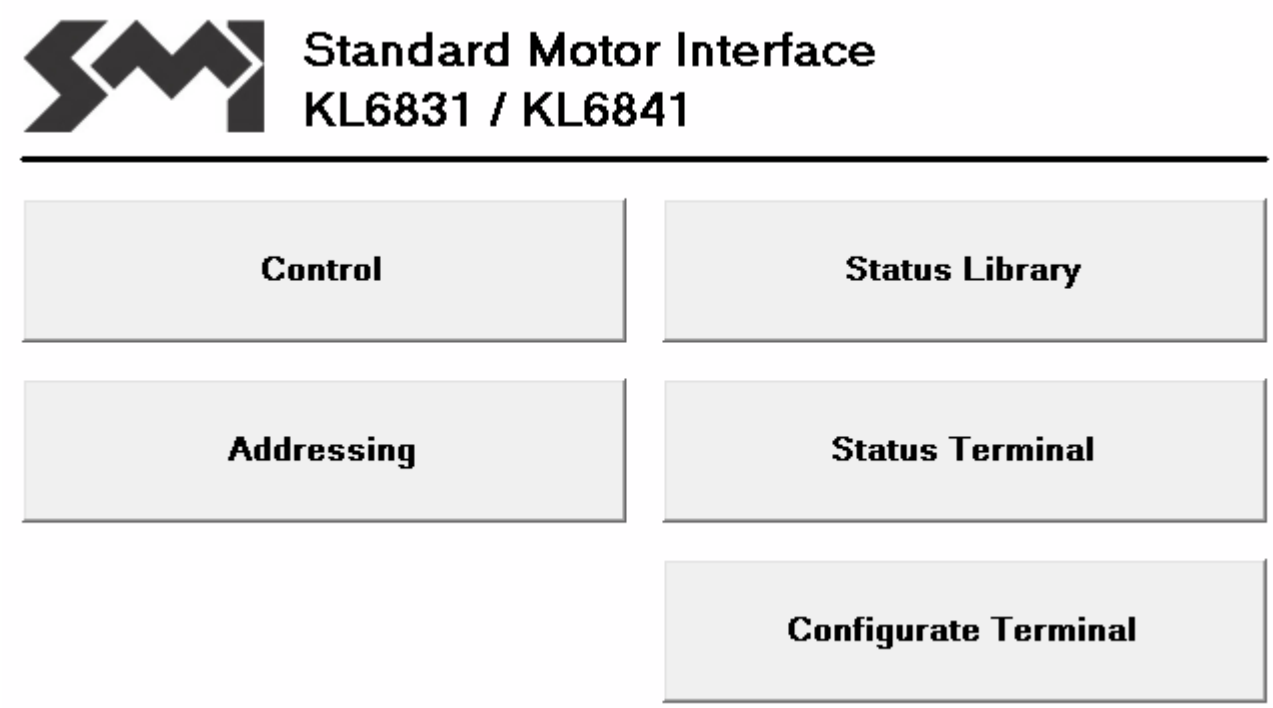
6.1 Example: Configuration of SMI devices

Requirements

Target system with TwinCAT Project Files
https://infosys.beckhoff.com/content/1033/tcplclibsmi/Resources/12074358283/.zip
https://infosys.beckhoff.com/content/1033/tcplclibsmi/Resources/12074359691/.zip
https://infosys.beckhoff.com/content/1033/tcplclibsmi/Resources/12074361099/.zip
https://infosys.beckhoff.com/content/1033/tcplclibsmi/Resources/12074362507/.zip
https://infosys.beckhoff.com/content/1033/tcplclibsmi/Resources/12074363915/.zip

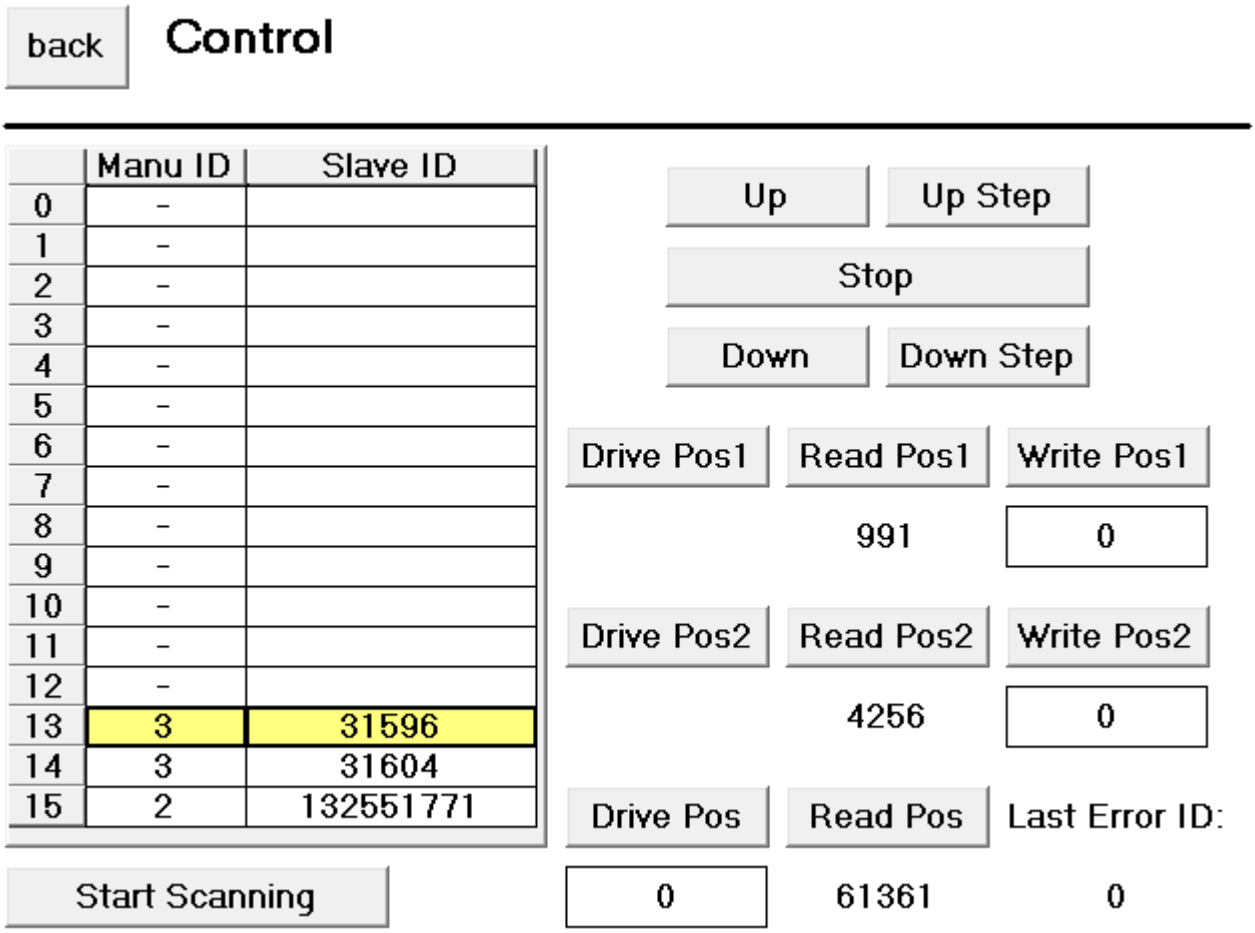
With the example it is possible to address SMI devices or to expand an existing installation. Moreover the dialogues for diagnosis and error analysis can be used. A total of 5 dialogues are available.

Start



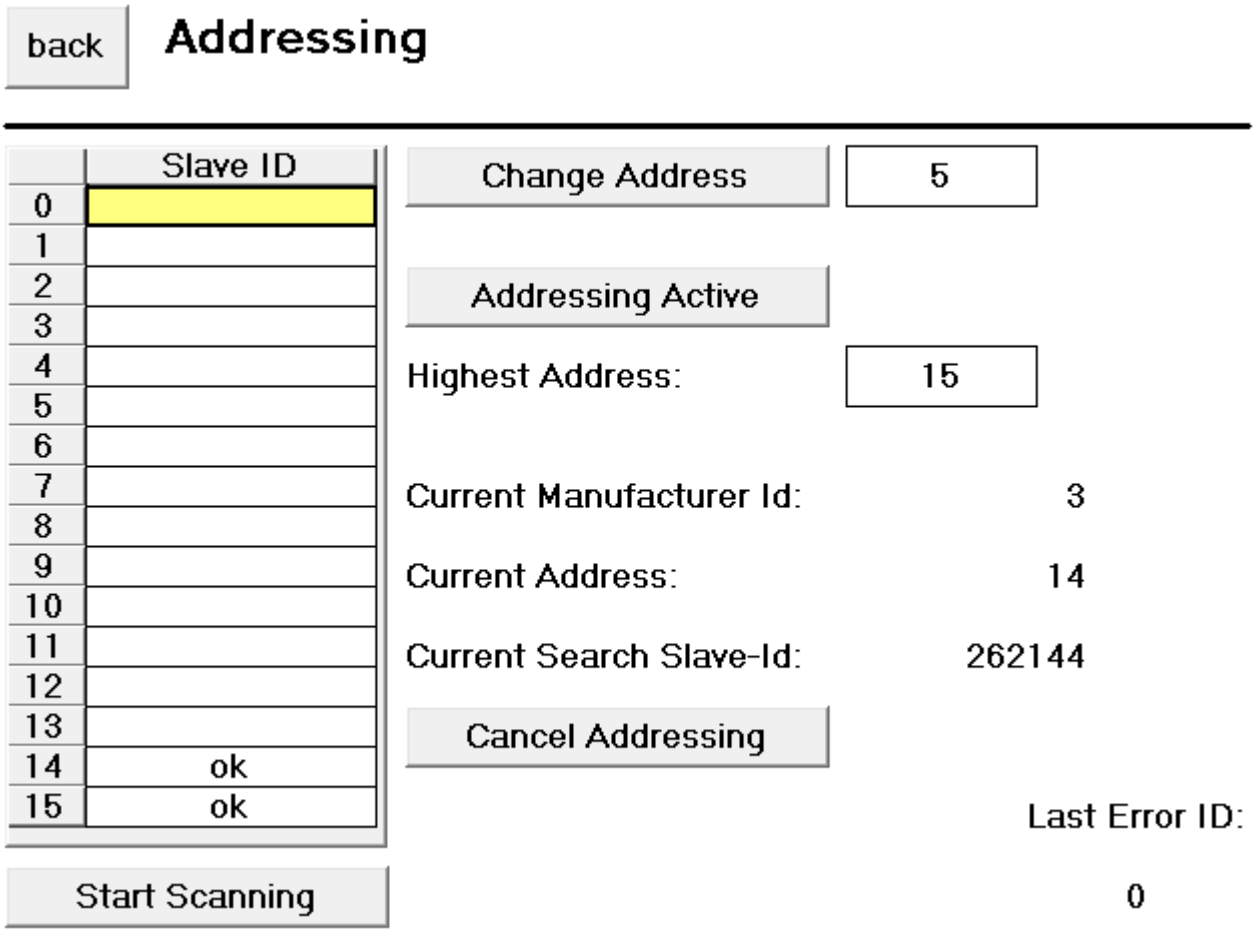
Under *SMI_Start* is the main menu, via which the five submenus can be accessed.

Control



On pressing the *StartScanning* button, a search is made for addressed SMI devices on the SMI line. All SMI devices found are displayed in the list on the left by the manufacturer code [▶ 65] and the slave ID. An entry is selected by clicking it. The other buttons are always related to the selected entry. This allows all important SMI commands to be sent to every addressed SMI device. If an error is detected with an SMI command, this is indicated in the bottom right-hand corner by the error code [▶ 57].

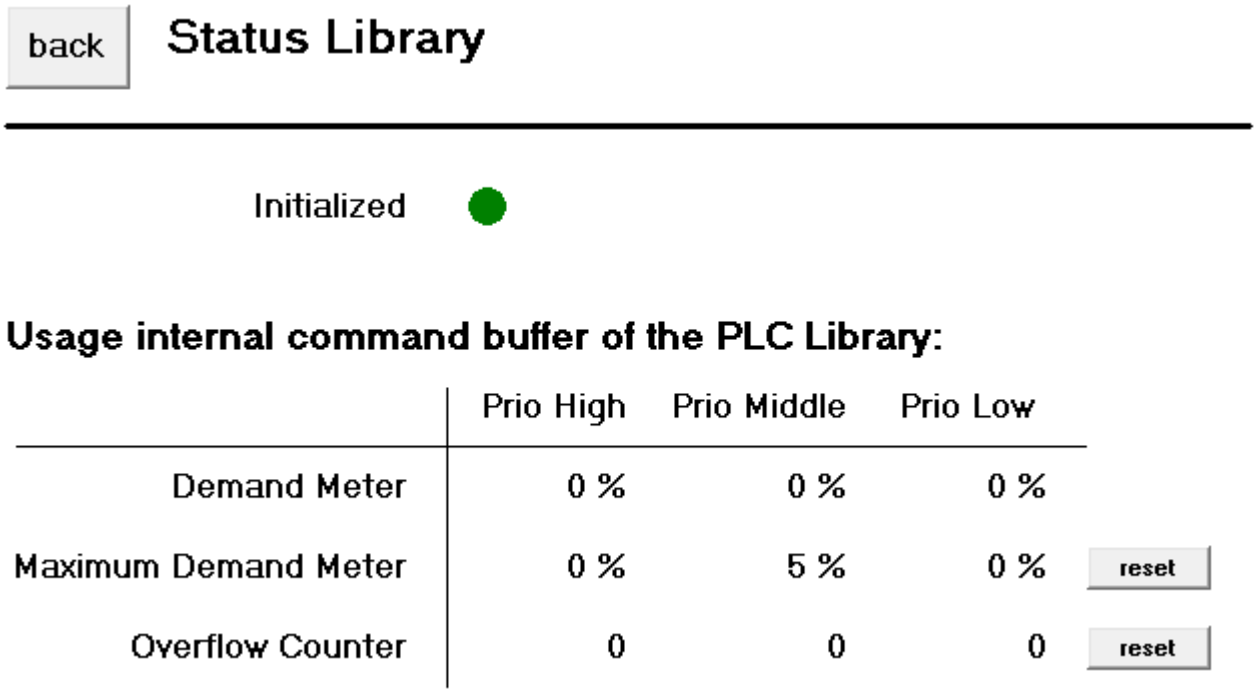
Addressing



Each SMI device can be assigned an address from 0 to 15. Each SMI device can be addressed via this address. Although there are other methods of addressing SMI devices (see here [▶ 9]), a unique address is necessary for group addressing. Therefore it is advisable to assign an address to each SMI device. The *Start Scanning* button is used to search for all addressed SMI devices. If an address is to be changed, then the corresponding SMI device must be selected in the list. The desired address can be entered in the input box on the right next to the *Change Address* button and accepted by actuating the button.

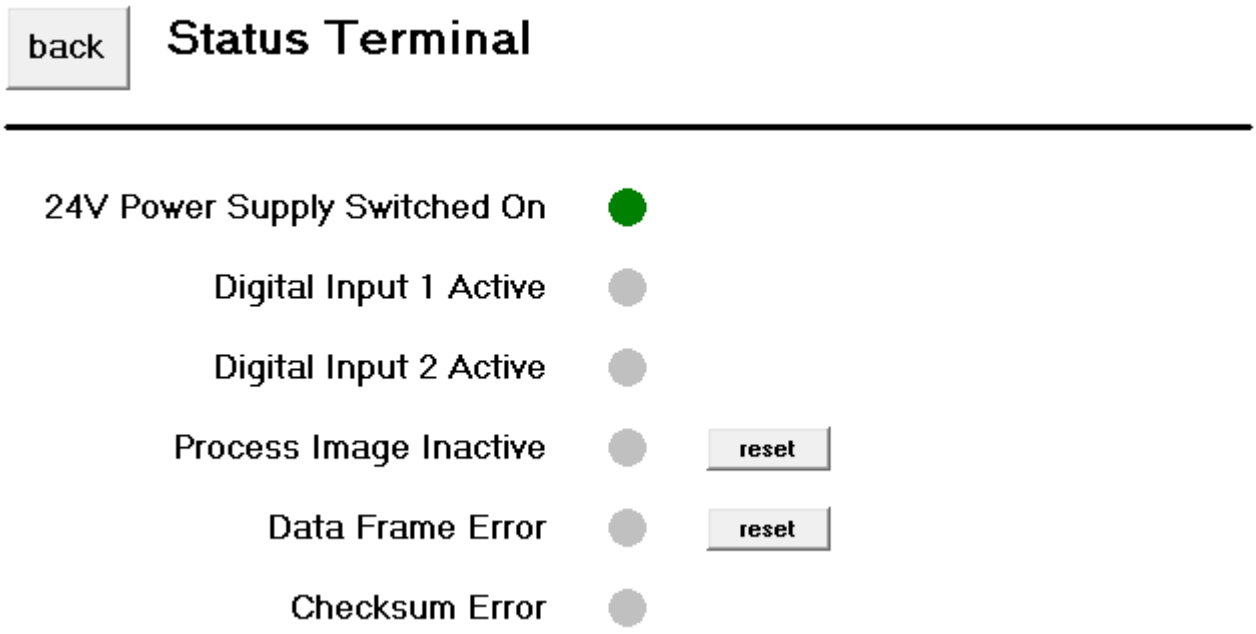
If SMI devices do not have an address yet, all SMI devices are assigned an address by pressing the *Start Addressing* button. The user has no influence over which address is assigned to which SMI device. The addresses are assigned in descending order, starting with the address specified by the *HighestAddress* parameter. The *Current Manufacturer Id*, *Current Address* and *Current Search Slave-Id* fields provide information about the status of the addressing. The addressing can be prematurely cancelled by pressing *Cancel Addressing*.

Library Status



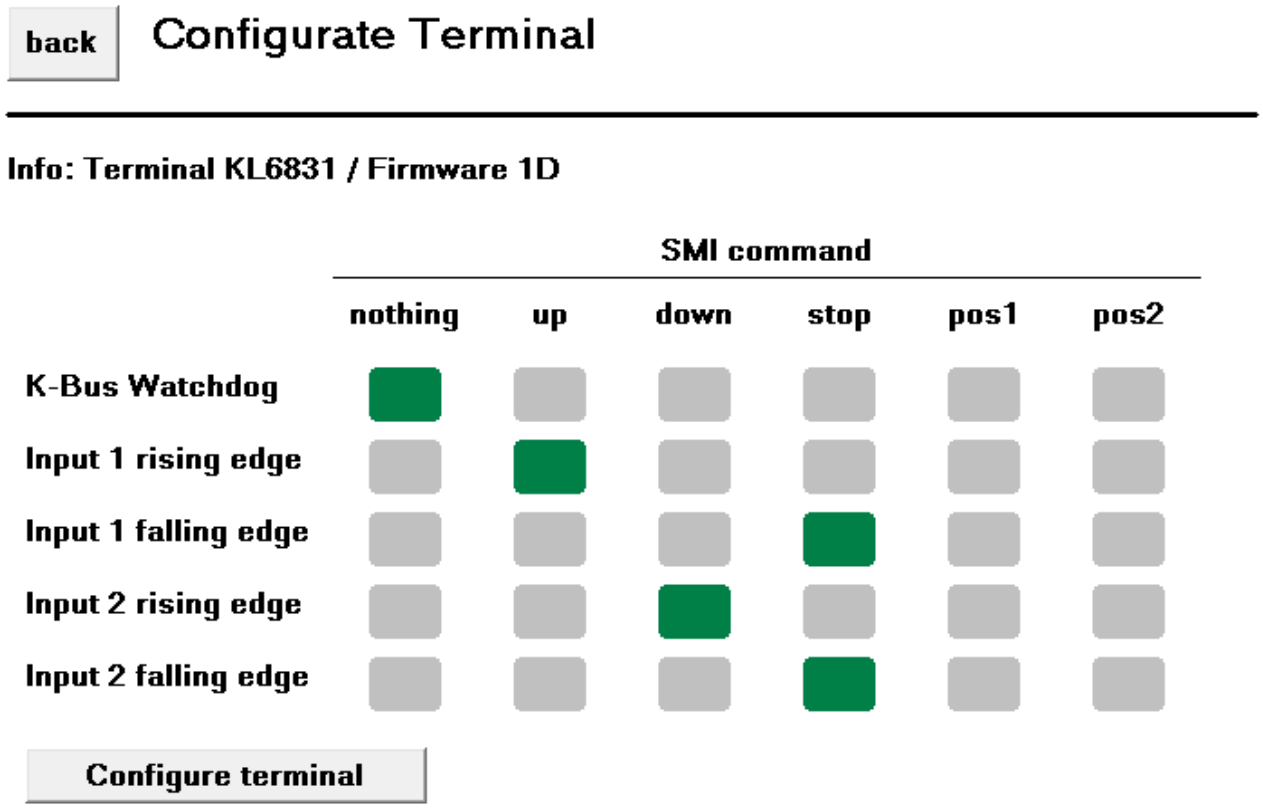
Communication between the individual PLC blocks and the Bus Terminal takes place within the PLC library via three central buffers (for each SMI terminal). The extent of utilisation of the buffers and possible overflows can be determined from the illustrated table.

Terminal Status



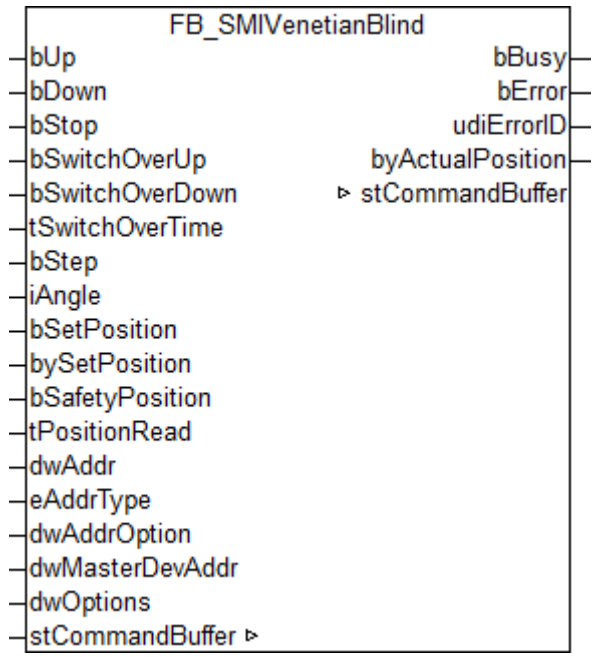
The status information from the process image of the terminal is displayed in this dialogue. In addition, messages requiring acknowledgement can be reset in this dialogue.

Configurate Terminal



SMI commands can be sent via two digital inputs of the SMI terminal. This dialog can be used to specify which SMI commands are triggered in the event of a rising or falling edge. An SMI command can also be sent, if communication has ceased on the K-bus of the terminal (e.g. the PLC is stopped, or the connection to the fieldbus master is interrupted).

6.2 Example: FB_SMIVenetianBlind



Venetian blind control.

https://infosys.beckhoff.com/content/1033/tcplclibsmi/Resources/12074365323/.zip

VAR_INPUT

```
bUp             : BOOL;
bDown           : BOOL;
bStop           : BOOL;
bSwitchOverUp   : BOOL;
bSwitchOverDown : BOOL;
tSwitchOverTime : TIME := t#400ms;
bStep           : BOOL;
iAngle          : INT := 0;
bSetPosition    : BOOL;
bySetPosition   : BYTE := 0;
bSafetyPosition : BOOL;
tPositionRead   : TIME := t#1s;
dwAddr          : DWORD := 0;
eAddrType       : E_SMIAddrType := eSMIAddrTypeAddress;
dwAddrOption    : DWORD := 0;
dwMasterDevAddr : DWORD := 0;
dwOptions       : DWORD := 0;
```

bUp: driving upwards. Starts with creep velocity and raises to rapid velocity.

bDown: driving downwards. Starts with creep velocity and raises to rapid velocity.

bStop: stops the blind.

bSwitchOverUp: stepping the blind up. If the signal remains present for longer than *tSwitchOverTime*, the blind remains latched.

bSwitchOverDown: stepping the blind down. If the signal remains present for longer than *tSwitchOverTime*, the blind remains latched.

tSwitchOverTime: gives the time for which the *bSwitchOverUp* and *bSwitchOverDown* inputs must remain asserted before the blind gets latched. If the value is 0, the blind is latched immediately.

bStep: moves the blind to the specified position.

iAngle: angle (-510° ... +510°) to which the blind is to be moved, after a rising edge has been applied to input *bStep*. Negative value corresponds to move up, positive value to move down.

bSetPosition: moves the blind to the position specified under input *bySetPosition*.

bySetPosition: position in percentage to which the blind is to be moved, after a rising edge has been applied to input *bSetPosition*. 0 % corresponds to fully up, 100 % to fully down.

bSafetyPosition: the safety position is approached. It is not possible to operate the blind while this input is set.

tPositionRead: interval in which the current position is read from the SMI motor if no positioning commands are processed.

dwAddr: manufacturer code [▶ 65] (0-15), address of a device (0-15), bit field (16 bits) for the group addressing [▶ 53] or slave ID (32-bit key ID).

eAddrType: defines whether the *dwAddr* input is to be evaluated as a manufacturer code [▶ 65], the address of a device, for group addressing or as a slave ID.

dwAddrOption: if the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 65] must be specified via this input.

dwMasterDevAddr: address (0-15) of the SMI motor of which the current position is to be read when *eAddrType* != *eSMIAddrTypeAddress*.

dwOptions: reserved for future expansions.

VAR_OUTPUT

```
bBusy           : BOOL;
bError          : BOOL;
udiErrorId      : UDINT;
byActualPosition: BYTE;
```

bBusy: this output is set as soon as the function block processes a command and remains active until the command has been processed.

bError: this output is switched to TRUE as soon as an error occurs during the execution of a command.

udiErrorId: contains the command-specific error code [▶ 57] of the most recently executed command.

byActualPosition: the position read out in percentage. 0 % corresponds to the upper final position and 100 % to the lower final position.

VAR_IN_OUT

```
stCommandBuffer     : ST_SMICommandBuffer;
```

stCommandBuffer: reference to the structure [▶ 56] for communication (buffer) with the FB_KL6831KL6841Communication() [▶ 19] block.

Requirements

Development environment	Target system	Required libraries
TwinCAT 2.11 R3/x64 from build 2238	PC/CX, BX or BC	TcSMI library from V1.0.0

6.3 Manufacturer codes

Value (hex)	Value (dec)	Description
0x00	0	all manufacturers
0x01	1	Dunkermotoren GmbH
0x02	2	Becker Antriebe GmbH
0x03	3	elero GmbH
0x04	4	Selve GmbH & Co. KG
0x05	5	Unused
0x06	6	Vestamatic GmbH
0x07	7	WAREMA Renkhoff GmbH
0x08	8	Groeninger Antriebstechnik GmbH
0x09	9	Gerhard Geiger GmbH & Co. KG
0x0A	10	Griesser AG
0x0B	11	Unused
0x0C	12	Unused
0x0D	13	Unused
0x0E	14	Unused
0x0F	15	reserved for extensions

6.4 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