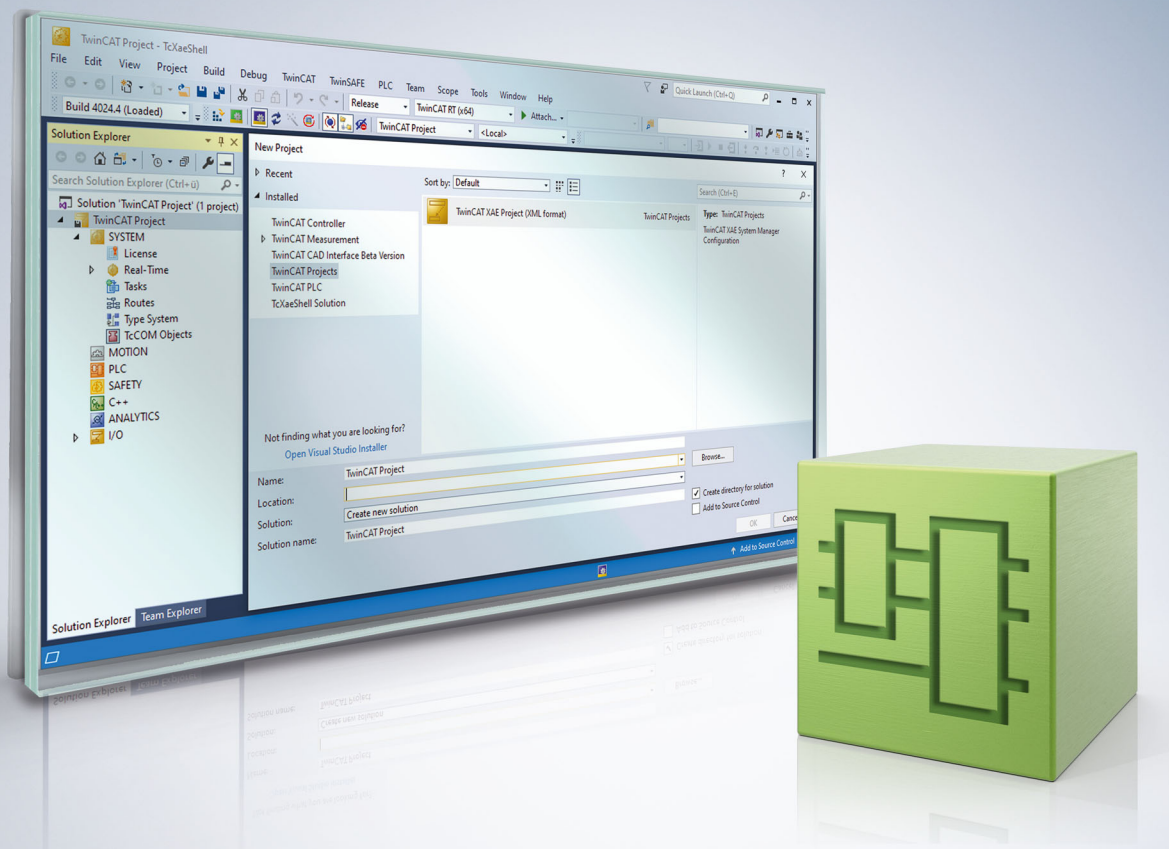
BECKHOFF New Automation Technology

Manual | EN

TE1000

TwinCAT 3 | PLC Library: Tc2_SMI



2025-05-15 | Version: 1.9.0

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.
The documentation and the following notes and explanations must be complied with when installing and commissioning the components.
The trained specialists must always use the current valid documentation.

The trained specialists must ensure that the application and use of the products described is in line with all safety requirements, including all relevant laws, regulations, guidelines, and standards.

Disclaimer

The documentation has been compiled 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.
Claims to modify products that have already been supplied may not be made on the basis of the data, diagrams, and descriptions in this documentation.

Trademarks

Beckhoff®, ATRO®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, MX-System®, Safety over EtherCAT®, TC/BSD®, TwinCAT®, TwinCAT/BSD®, TwinSAFE®, XFC®, XPlanar®, and XTS® are registered and licensed trademarks of Beckhoff Automation GmbH.
If third parties make use of the designations or trademarks contained in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.

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

Copyright

© Beckhoff Automation GmbH & Co. KG, Germany.
The distribution and reproduction of this document, as well as the use and communication of its contents without express authorization, are prohibited.
Offenders will be held liable for the payment of damages. All rights reserved in the event that a patent, utility model, or design are registered.

Third-party trademarks

Trademarks of third parties may be used in this documentation. You can find the trademark notices here: https://www.beckhoff.com/trademarks.

1.2 For your safety

Safety regulations

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

Exclusion of liability

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

Personnel qualification

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

Signal words

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

Personal injury warnings

⚠ DANGER

Hazard with high risk of death or serious injury.

⚠ WARNING

Hazard with medium risk of death or serious injury.

⚠ CAUTION

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

Warning of damage to property or environment

NOTICE

The environment, equipment, or data may be damaged.

Information on handling the product

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

1.3 Notes on information security

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

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

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

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

2 Introduction

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

- TwinCAT XAE
- 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.

The Tc2_SMI library is usable on all hardware platforms that support TwinCAT 3.1 or higher.

Hardware documentation KL6831 KL6841 in the Beckhoff information system.

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).

In order 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 [▶ 55].

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 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 [▶ 66]. 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 [▶ 55].

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 [▶ 55].

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*.

by 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 Programming

4.1 POUs

Basic commands

Name	Description
FB_KL6831KL6841Communication [▶ 12]	Reads the SMI commands sequentially from the internal buffer of the PLC library and forwards them to the KL6831/KL6841.
FB_KL6831KL6841Config [▶ 15]	This function block can be used to configure the KL6831/ KL6841.
FB_SMISendSMICommand [▶ 17]	This function block is for the general sending of an SMI command.

Basic commands

Name	Description
FB_SMIDiagAll [▶ 19]	Diagnostic telegram is sent.
FB_SMIDown [▶ 22]	Motor run to the lower end position.
FB_SMIDownStep [▶ 23]	Motor runs downwards by a specified angle.
FB_SMIPos1 [▶ 24]	Drives to the fixed position *Pos1* configured on the motor side.
FB_SMIPos1Read [▶ 26]	Reads the fixed position *Pos1* configured on the motor side.
FB_SMIPos1Write [▶ 27]	Writes the fixed position *Pos1* which is configured on the motor side.
FB_SMIPos2 [▶ 28]	Drives to the fixed position *Pos2* configured on the motor side.
FB_SMIPos2Read [▶ 30]	Reads the fixed position *Pos2* configured on the motor side.
FB_SMIPos2Write [▶ 31]	Writes the fixed position *Pos2* which is configured on the motor side.
FB_SMIPosRead [▶ 33]	Reads the current position.
FB_SMIPosWrite [▶ 34]	Drives to a position.
FB_SMIStop [▶ 35]	Stops the motor run.
FB_SMISyn [▶ 37]	Queries the manufacturer code and drive type.
FB_SMIUp [▶ 38]	Motor run to the upper end position.
FB_SMIUpStep [▶ 39]	Motor run upwards by a specified angle.

Addressing commands

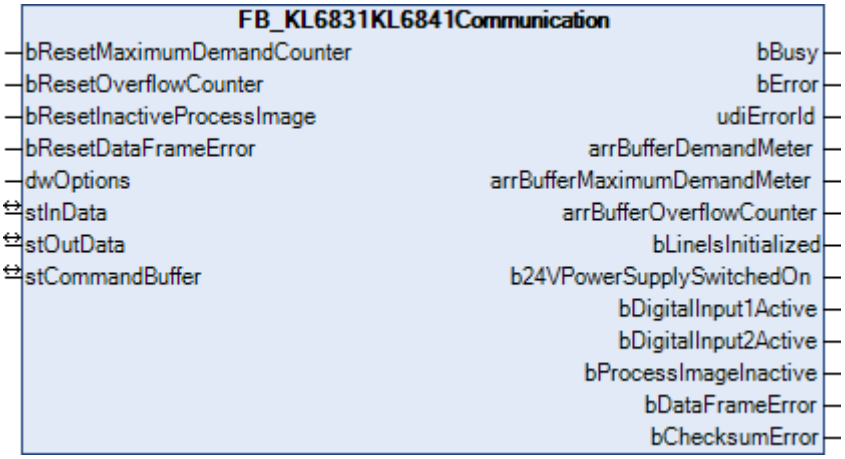
Name	Description
FB_SMIAddressing [▶ 41]	Addresses SMI devices.
FB_SMIDiscoverySlaveId [▶ 42]	Searches for SMI devices by manufacturer code.
FB_SMISlaveAddrRead [▶ 44]	Reads the address (0-15) of a drive.
FB_SMISlaveAddrWrite [▶ 45]	Writes an address (0-15) to one or more drives.
FB_SMISlaveIdCompare [▶ 46]	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 [▶ 51]	Reads a byte parameter (1 byte) stored on the motor side.
FB_SMIParValueReadWord [▶ 52]	Reads a Word parameter (2 bytes) stored on the motor side.
FB_SMIParValueReadDWord [▶ 53]	Reads a DWord parameter (4 bytes) stored on the motor side.

4.1.1 Base

4.1.1.1 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 function block *FB_KL6831KL6841Communication()* sequentially reads the SMI commands from these three buffers and forwards the SMI commands to the KL6831/KL6841. This prevents multiple function blocks accessing the KL6831/KL6841 process image simultaneously. Each of these three buffers is processed with a different priority (high, medium or low). The parameter eCommandPriority, which is available for most function blocks, can be used to influence the priority with which the respective SMI command is processed by the function block *FB_KL6831KL6841Communication()*.

The buffers in which the SMI commands are stored are all contained in a variable of the type ST_SMICommandBuffer. For each KL6831/KL6841 there is an instance of the function block *FB_KL6831KL6841Communication()* and a variable of the type *ST_SMICommandBuffer*. If possible, the function block *FB_KL6831KL6841Communication()* 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 utilized? TwinCAT XAE provides suitable analysis tools.
- Try reducing the cycle time of the task in which the function block *FB_KL6831KL6841Communication()* 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 function 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 actually 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.

Inputs

```
VAR_INPUT
  bResetMaximumDemandCounter : BOOL;
  bResetOverflowCounter      : BOOL;
  bResetInactiveProcessImage : BOOL;
```

```
  bResetDataFrameError       : BOOL;
  dwOptions                  : DWORD := 0;
END_VAR
```

Name	Type	Description
bResetMaximumDemandCounter	BOOL	A positive edge resets the stored value of the maximum command buffer demand *(arrBufferMaximumDemandMeter)*.
bResetOverflowCounter	BOOL	A positive edge resets the stored value of the number of command buffer overflows *(arrBufferOverflowCounter)*.
bResetInactiveProcessImage	BOOL	A positive edge cancels the blocking of the process image of the terminal. The outputs *bProcessImageInactive*, *bDigitalInput1Active* and *bDigitalInput2Active* are again set to FALSE. The blocking is activated as soon as one of the digital inputs *Input1* or *Input2* on the terminal is actuated.
bResetDataFrameError	BOOL	A positive edge resets the message for a telegram error. The output *bDataFrameError* is set again to FALSE. The blocking is activated as soon as the terminal detects a telegram error.
dwOptions	DWORD	Reserved for future extensions

Inputs/outputs

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

Name	Type	Description
stInData	ST_KL6831KL6841InData [▶ 59]	Reference to the structure for communication with the KL6831/KL6841
stOutData	ST_KL6831KL6841OutData [▶ 59]	Reference to the structure for communication with the KL6831/KL6841
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication with the SMI function blocks

Outputs

```
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;
END_VAR
```

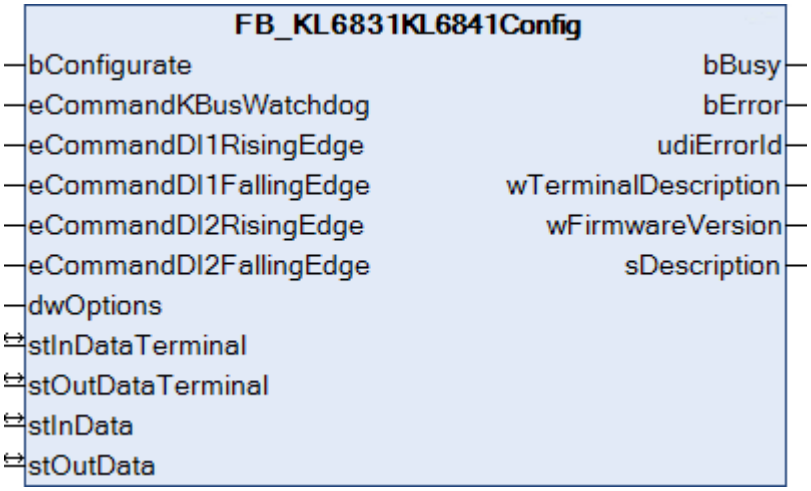
Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.

Name	Type	Description
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).
arrBufferDemandMeter	ARRAY OF BYTE	Demand of the respective buffer (0 - 100%)
arrBufferMaximumDemandMeter	ARRAY OF BYTE	Previous maximum demand of the respective buffer (0 - 100%)
arrBufferOverflowCounter	ARRAY OF BYTE	Number of buffer overflows to date
bLineIsInitialized	BOOL	If the function block is called for the first time (e.g. when starting the controller), initialization is carried out. No SMI commands can be processed during this time. This output is set to TRUE once the initialization has been completed.
b24VPowerSupplySwitchedOn	BOOL	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	BOOL	The digital input *Input1* on the terminal has been or is actuated (see also the terminal documentation). The output *bProcessImageInactive* is set and no further SMI commands can be processed by the controller.
bDigitalInput2Active	BOOL	The digital input *Input2* on the terminal has been or is actuated (see also the terminal documentation). The output *bProcessImageInactive* is set and no further SMI commands can be processed by the controller.
bProcessImageInactive	BOOL	One of the digital inputs *Input1* or *Input2* has been actuated on the terminal. No further SMI commands can be processed by the controller. The blockage must be released again via the input *bResetInactiveProcessImage*.
bDataFrameError	BOOL	The terminal has detected a telegram error on the SMI bus. The error must be reset via the input *bResetDataFrameError*.
bChecksumError	BOOL	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.

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

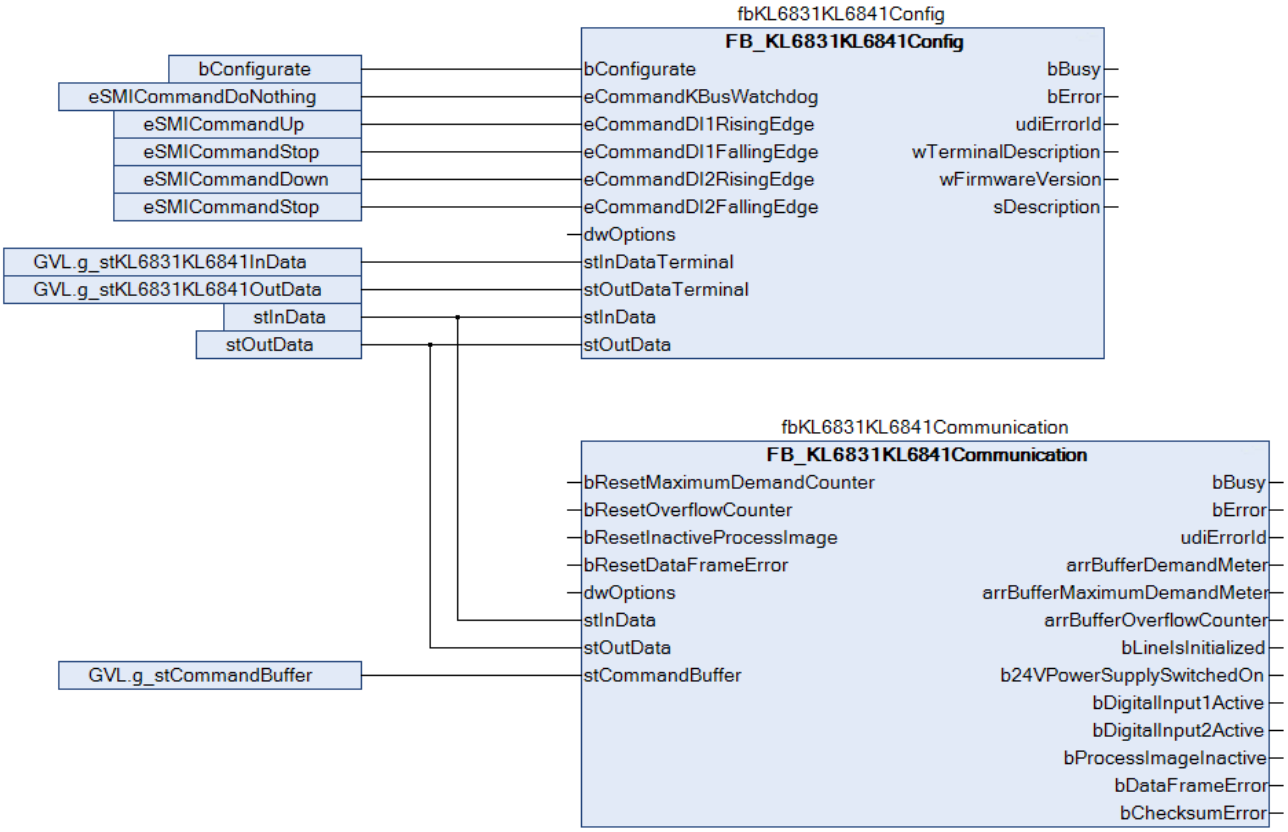
4.1.1.2 FB_KL6831KL6841Config



The function block FB_KL6831KL6841Config 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 function block is called in the same task as the function block FB_KL6831KL6841Communication().



The function block FB_KL6831KL6841Config() is linked to the process image of the KL6831/KL6841. Once the configuration is complete, the function block FB_KL6831KL6841Communication() receives the process values of the KL6831/KL6841. No SMI commands can be sent while the configuration is in progress.

Sample

See https://infosys.beckhoff.com/content/1033/tcplclib_tc2_smi/Resources/3248663563.zip

Inputs

```
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;
END_VAR
```

Name	Type	Description
bConfigurate	BOOL	Configuration of the bus terminal is started by a positive edge at this input.
eCommandKBusWatchdog	E_SMIConfigurationCommands [▶ 56]	Defines the SMI command 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	E_SMIConfigurationCommands [▶ 56]	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	E_SMIConfigurationCommands [▶ 56]	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	E_SMIConfigurationCommands [▶ 56]	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	E_SMIConfigurationCommands [▶ 56]	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	DWORD	Reserved for future extensions

Inputs/outputs

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

Name	Type	Description
stInDataTerminal	ST_KL6831KL6841InData [▶ 59]	Reference to the structure for communication with the KL6831/KL6841
stOutDataTerminal	ST_KL6831KL6841OutData [▶ 59]	Reference to the structure for communication with the KL6831/KL6841
stInData	ST_KL6831KL6841InData [▶ 59]	Reference to the structure for communication with the function block FB_KL6831KL6841Communication() [▶ 12]
stOutData	ST_KL6831KL6841OutData [▶ 59]	Reference to the structure for communication with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

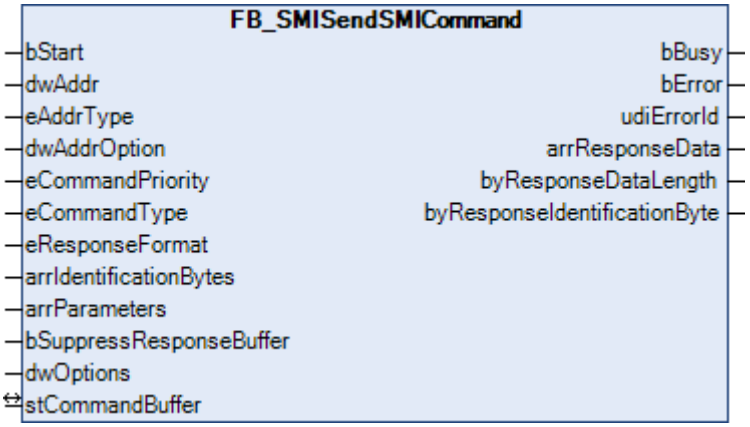
```
VAR_OUTPUT
  bBusy                : BOOL;
  bError               : BOOL;
  udiErrorId           : UDINT;
  wTerminalDescription : WORD;
  wFirmwareVersion     : WORD;
  sDescription         : STRING;
END_VAR
```

Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).
wTerminalDescription	WORD	Contains the terminal name (e.g. 6831). Corresponds to register 8 of the bus terminal.
wFirmwareVersion	WORD	Contains the firmware version. Corresponds to register 9 of the bus terminal.
sDescription	STRING	Terminal name and firmware version as string (e.g. 'Terminal KL6831 / Firmware 1D')

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.32	Tc2_SMI from 3.3.6.0

4.1.1.3 FB_SMISendSMICommand



The function block FB_SMISendSMICommand is used 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.

Inputs

```
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;
END_VAR
```

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.
eAddrType	E_SMIAddrType [▶ 56]	Defines whether the input *dwAddr* is to be evaluated as a manufacturer code [▶ 66], the address of a device, for group addressing or as a slave ID.
dwAddrOption	DWORD	If the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 66] must be specified via this input.
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.
eCommandType	E_SMICommandType [▶ 57]	Command type: Write/Read. This parameter affects bit 5 of the start byte of the SMI telegram.
eResponseFormat	E_SMIResponseFormat [▶ 59]	Response format: diagnostic special format/standard. This parameter affects bit 6 of the start byte of the SMI telegram.
arrIdentificationBytes	ARRAY OF BYTE	An SMI telegram can consist of up to 3 blocks. Each block possesses an identification byte. This array defines the three identification bytes.
arrParameters	ARRAY OF DWORD	An SMI telegram can consist of up to 3 blocks. Each block has up to four value bytes. This array defines the value bytes of each block.
bSuppressResponseBuffer	BOOL	If this input is set to TRUE, the internal software buffer is not filled with the responses of the function block FB_KL6831KL6841Communication() [▶ 12].
dwOptions	DWORD	Reserved for future extensions

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

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

Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.

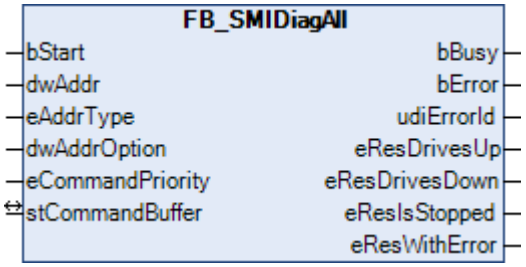
Name	Type	Description
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).
arrResponseData	ARRAY OF BYTE	The data received from the SMI devices
byResponseDataLength	BYTE	The length of the data received in bytes
byResponseIdentificationByte	BYTE	The received identification byte

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.2 Basic commands

4.1.2.1 FB_SMIDiagAll



The function block FB_SMIDiagAll can be used to determine in which direction the drives are moving, whether they are 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.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.

Name	Type	Description
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.
eAddrType	E_SMIAddrType [▶ 56]	Specifies whether the input *dwAddr* is to be evaluated as manufacturer code [▶ 66], address of a device or for group addressing. Addressing via slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*) is not permitted.
dwAddrOption	DWORD	Reserved for future extensions
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

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

Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).
eResDrivesUp	E_SMIDiagResDrivesUp [▶ 58]	At least one motor is driving up / no motor is driving up / the value is undefined
eResDrivesDown	E_SMIDiagResDrivesDown [▶ 58]	At least one motor is driving down. / no motor is driving down. / the value is undefined.
eResIsStopped	E_SMIDiagResIsStopped [▶ 58]	At least one motor is stopped. / no motor is stopped. / the value is undefined.
eResWithError	E_SMIDiagResWithError [▶ 58]	At least one motor is faulty. / no motor is faulty. / the value is undefined.

Examples

All drives have stopped:

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

One drive is 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 is stopped
eResWithError = eSMIDiagResNoMotorWithError	No drive has a motor error

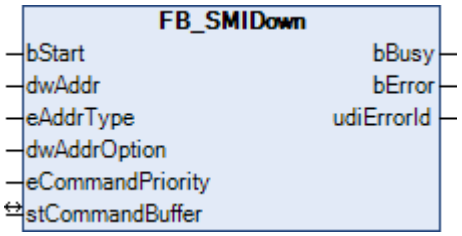
One drive is stopped, one drive is driving up and one drive is driving down:

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 is stopped
eResWithError = eSMIDiagResNoMotorWithError	No drive has a motor error

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.2.2 FB_SMIDown



The function block FB_SMIDown controls the motor run to the lower end position.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.
eAddrType	E_SMIAddrType [▶ 56]	Defines whether the input *dwAddr* is to be evaluated as a manufacturer code [▶ 66], the address of a device, for group addressing or as a slave ID.
dwAddrOption	DWORD	If the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 66] must be specified via this input.
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

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

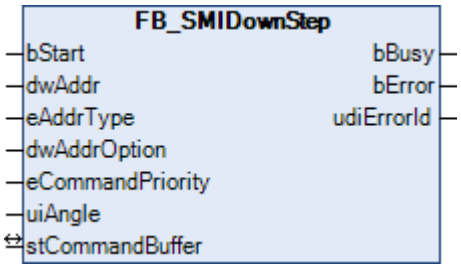
Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.

Name	Type	Description
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.2.3 FB_SMIDownStep



The function block FB_SMIDownStep controls the motor run downwards by a specified angle (0-510 degrees).

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.
eAddrType	E_SMIAddrType [▶ 56]	Defines whether the input *dwAddr* is to be evaluated as a manufacturer code [▶ 66], the address of a device, for group addressing or as a slave ID.
dwAddrOption	DWORD	If the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 66] must be specified via this input.
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.
uiAngle	UINT	The specified angle. The value range is 0...510 degrees. The SMI standard reduces the accuracy to a resolution of 2 degrees.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

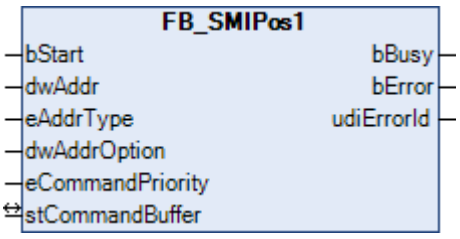
```
VAR_OUTPUT
  bBusy      : BOOL;
  bError     : BOOL;
  udiErrorId : UDINT;
END_VAR
```

Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.2.4 FB_SMIPos1



The function block FB_SMIPos1 realizes the movement to the fixed position *Pos1* configured on the motor side. *Pos1* can be read and changed using the function blocks FB_SMIPos1Read() [▶ 26] and FB_SMIPos1Write() [▶ 27].

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.
eAddrType	E_SMIAddrType [▶ 56]	Defines whether the input *dwAddr* is to be evaluated as a manufacturer code [▶ 66], the address of a device, for group addressing or as a slave ID.
dwAddrOption	DWORD	If the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 66] must be specified via this input.
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

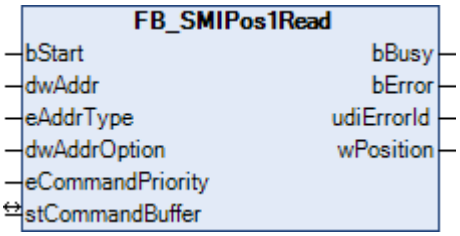
```
VAR_OUTPUT
  bBusy      : BOOL;
  bError     : BOOL;
  udiErrorId : UDINT;
END_VAR
```

Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.2.5 FB_SMIPos1Read



The function block FB_SMIPos1Read reads the fixed position *Pos1* configured on the motor side. *Pos1* can be changed with the function block FB_SMIPos1Write() [▶ 27].

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.
eAddrType	E_SMIAddrType [▶ 56]	Specifies whether the input *dwAddr* is to be evaluated as manufacturer code [▶ 66], address of a device or for group addressing. Addressing via slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*) is not permitted.
dwAddrOption	DWORD	Reserved for future extensions
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

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

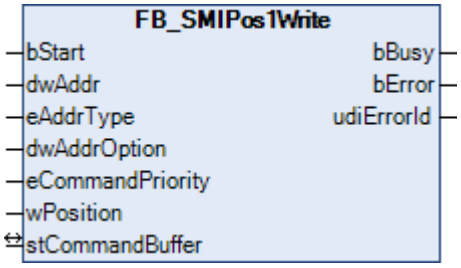
Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.

Name	Type	Description
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).
wPosition	WORD	The read fixed position *Pos1*. The value 0 corresponds here to the upper end position and the value 65535 (0xFFFF) to the lower end position.

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.2.6 FB_SMIPos1Write



The function block FB_SMIPos1Write writes the fixed position *Pos1* that can be configured on the motor side. *Pos1* can be read with the function block FB_SMIPos1Read() [▶ 26].

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.
eAddrType	E_SMIAddrType [▶ 56]	Defines whether the input *dwAddr* is to be evaluated as a manufacturer code [▶ 66], the address of a device, for group addressing or as a slave ID.
dwAddrOption	DWORD	If the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 66] must be specified via this input.
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.

Name	Type	Description
wPosition	WORD	The new fixed position *Pos1*. The value 0 corresponds here to the upper end position and the value 65535 (0xFFFF) to the lower end position.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

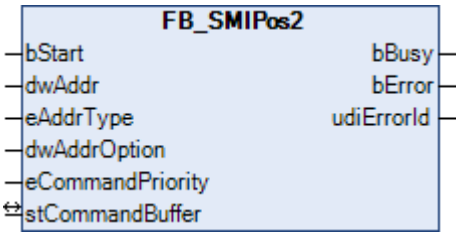
```
VAR_OUTPUT
  bBusy      : BOOL;
  bError     : BOOL;
  udiErrorId : UDINT;
END_VAR
```

Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.2.7 FB_SMIPos2



The function block FB_SMIPos2 controls the movement to the fixed position *Pos2* configured on the motor side. *Pos2* can be read and changed using the function blocks FB_SMIPos2Read() [▶ 30] and FB_SMIPos2Write() [▶ 31].

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.
eAddrType	E_SMIAddrType [▶ 56]	Defines whether the input *dwAddr* is to be evaluated as a manufacturer code [▶ 66], the address of a device, for group addressing or as a slave ID.
dwAddrOption	DWORD	If the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 66] must be specified via this input.
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

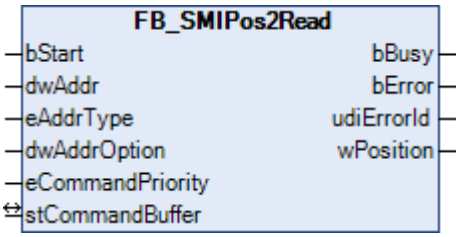
```
VAR_OUTPUT
  bBusy      : BOOL;
  bError     : BOOL;
  udiErrorId : UDINT;
END_VAR
```

Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.2.8 FB_SMIPos2Read



The function block FB_SMIPos2Read reads the fixed position *Pos2* configured on the motor side. *Pos2* can be changed with the function block FB_SMIPos2Write() [▶ 31].

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.
eAddrType	E_SMIAddrType [▶ 56]	Specifies whether the input *dwAddr* is to be evaluated as manufacturer code [▶ 66], address of a device or for group addressing. Addressing via slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*) is not permitted.
dwAddrOption	DWORD	Reserved for future extensions
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

```
VAR_OUTPUT
  bBusy      : BOOL;
  bError     : BOOL;
```

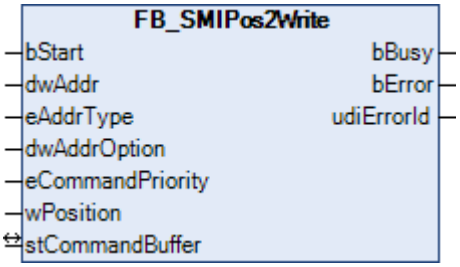
```
  udiErrorId : UDINT;
  wPosition  : WORD;
END_VAR
```

Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).
wPosition	WORD	The read fixed position *Pos1*. The value 0 corresponds here to the upper end position and the value 65535 (0xFFFF) to the lower end position.

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.2.9 FB_SMIPos2Write



The function block FB_SMIPos2Write writes the fixed position *Pos2* that can be configured on the motor side. *Pos2* can be read with the function block FB_SMIPos2Read() [▶ 30].

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.

Name	Type	Description
eAddrType	E_SMIAddrType [▶ 56]	Defines whether the input *dwAddr* is to be evaluated as a manufacturer code [▶ 66], the address of a device, for group addressing or as a slave ID.
dwAddrOption	DWORD	If the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 66] must be specified via this input.
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.
wPosition	WORD	The new fixed position *Pos2*. The value 0 corresponds here to the upper end position and the value 65535 (0xFFFF) to the lower end position.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

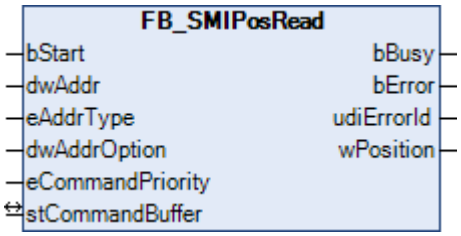
```
VAR_OUTPUT
  bBusy      : BOOL;
  bError     : BOOL;
  udiErrorId : UDINT;
END_VAR
```

Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.2.10 FB_SMIPosRead



The function block FB_SMIPosRead reads the current position from the drive.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.
eAddrType	E_SMIAddrType [▶ 56]	Specifies whether the input *dwAddr* is to be evaluated as manufacturer code [▶ 66], address of a device or for group addressing. Addressing via slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*) is not permitted.
dwAddrOption	DWORD	Reserved for future extensions
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

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

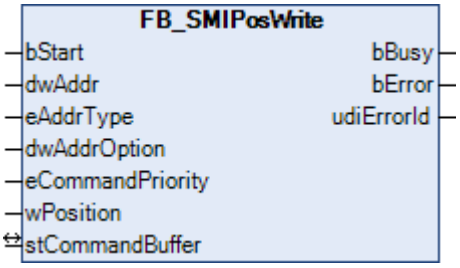
Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.

Name	Type	Description
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).
wPosition	WORD	The read fixed position *Pos1*. The value 0 corresponds here to the upper end position and the value 65535 (0xFFFF) to the lower end position.

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.2.11 FB_SMIPosWrite



The function block FB_SMIPosWrite moves the drive to the specified position.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.
eAddrType	E_SMIAddrType [▶ 56]	Defines whether the input *dwAddr* is to be evaluated as a manufacturer code [▶ 66], the address of a device, for group addressing or as a slave ID.
dwAddrOption	DWORD	If the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 66] must be specified via this input.
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.

Name	Type	Description
wPosition	WORD	The new position. The value 0 corresponds here to the upper end position and the value 65535 (0xFFFF) to the lower end position.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

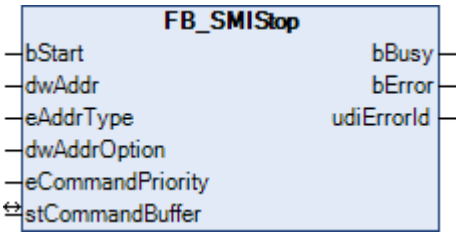
```
VAR_OUTPUT
  bBusy      : BOOL;
  bError     : BOOL;
  udiErrorId : UDINT;
END_VAR
```

Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.2.12 FB_SMIStop



The function block FB_SMIStop stops the motor run.

Inputs

```
VAR_INPUT
  bStart           : BOOL;
  dwAddr           : DWORD := 0;
```

```
  eAddrType        : E_SMIAddrType := eSMIAddrTypeAddress;
  dwAddrOption     : DWORD := 0;
  eCommandPriority : E_SMICommandPriority := eSMICommandPriorityMiddle;
END_VAR
```

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.
eAddrType	E_SMIAddrType [▶ 56]	Defines whether the input *dwAddr* is to be evaluated as a manufacturer code [▶ 66], the address of a device, for group addressing or as a slave ID.
dwAddrOption	DWORD	If the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 66] must be specified via this input.
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

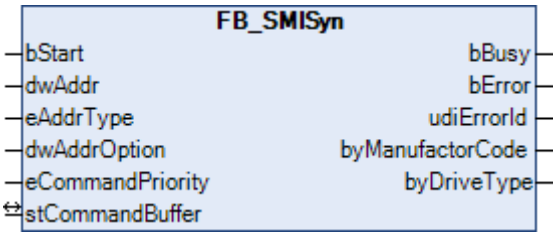
```
VAR_OUTPUT
  bBusy      : BOOL;
  bError     : BOOL;
  udiErrorId : UDINT;
END_VAR
```

Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.2.13 FB_SMISyn



The function block FB_SMISyn queries the manufacturer code and the drive type.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.
eAddrType	E_SMIAddrType [▶ 56]	Specifies whether the input *dwAddr* is to be evaluated as manufacturer code [▶ 66], address of a device or for group addressing. Addressing via slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*) is not permitted.
dwAddrOption	DWORD	Reserved for future extensions
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

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

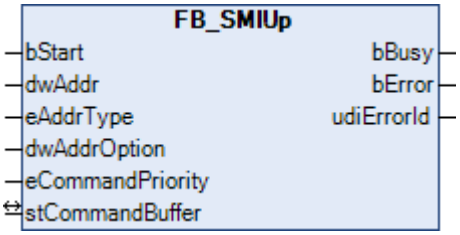
Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.

Name	Type	Description
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).
byManufactorCode	BYTE	The manufacturer code [▶ 66] (1-14)
byDriveType	BYTE	The drive type (0-15). The meaning is manufacturer-specific.

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.2.14 FB_SMIUp



The function block FB_SMIUp controls the motor run up to the upper end position.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.
eAddrType	E_SMIAddrType [▶ 56]	Defines whether the input *dwAddr* is to be evaluated as a manufacturer code [▶ 66], the address of a device, for group addressing or as a slave ID.
dwAddrOption	DWORD	If the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 66] must be specified via this input.
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

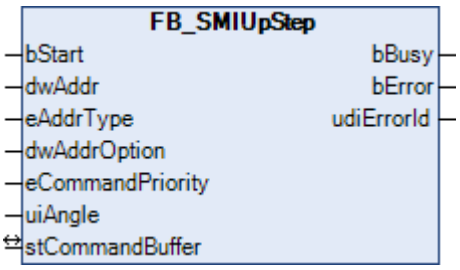
```
VAR_OUTPUT
  bBusy      : BOOL;
  bError     : BOOL;
  udiErrorId : UDINT;
END_VAR
```

Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.2.15 FB_SMIUpStep



The function block FB_SMIUpStep controls the motor run upwards by a specified angle (0-510 degrees).

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.
eAddrType	E_SMIAddrType [▶ 56]	Defines whether the input *dwAddr* is to be evaluated as a manufacturer code [▶ 66], the address of a device, for group addressing or as a slave ID.
dwAddrOption	DWORD	If the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 66] must be specified via this input.
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.
uiAngle	UINT	The specified angle. The value range is 0...510 degrees. The SMI standard reduces the accuracy to a resolution of 2 degrees.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

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

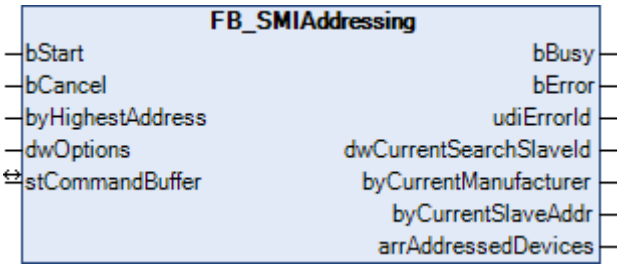
Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.3 Addressing commands

4.1.3.1 FB_SMIAddressing



This function block addresses the connected SMI devices according to the random principle. Which SMI device is assigned to which address is beyond the control of the user. The addresses are assigned in descending order, starting with the address specified by the parameter *byHighestAddress*.

Applying a positive edge to the input *bStart* starts the function block, and the output *bBusy* 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 output *bBusy* 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.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.
bCancel	BOOL	The function block is deactivated and the search is aborted on applying a positive edge to this input.
byHighestAddress	BYTE	Address from which the SMI devices are addressed in descending order (0-15).
dwOptions	DWORD	Reserved for future extensions

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

```
VAR_OUTPUT
  bBusy                : BOOL;
  bError               : BOOL;
  udiErrorId           : UDINT;
  dwCurrentSearchSlaveId : DWORD;
  byCurrentManufacturer  : BYTE;
```

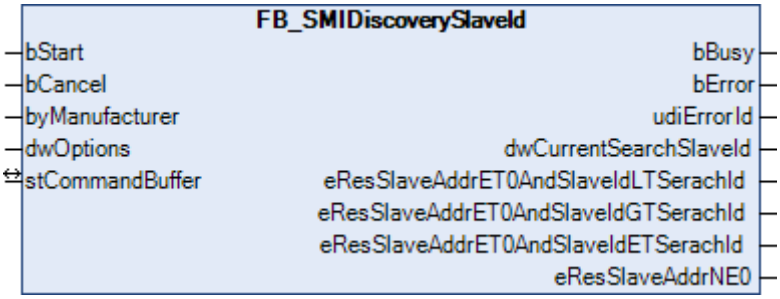
```
  byCurrentSlaveAddr     : BYTE;
  arrAddressedDevices    : ARRAY [0..15] OF BOOL;
END_VAR
```

Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).
dwCurrentSearchSlaveId	DWORD	Current slave ID used in the search algorithm.
byCurrentManufacturer	BYTE	Current manufacturer code [▶ 66] used in the search algorithm.
byCurrentSlaveAddr	BYTE	Current address used in the search algorithm.
arrAddressedDevices	ARRAY OF BOOL	If an address is assigned to an SMI device, then the corresponding element in the array is set to TRUE.

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.3.2 FB_SMIDiscoverySlaveId



The first drive is sought that corresponds to the specified manufacturer code and has the address 0. This function block is used for addressing SMI devices and is used in the function block FB_SMIAddressing() [▶ 41].

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the search is started by applying a positive edge to this input.

Name	Type	Description
bCancel	BOOL	The function block is deactivated and the search is aborted on applying a positive edge to this input.
byManufacturer	BYTE	The manufacturer code [▶ 66] specified for the search for the SMI device. Some SMI devices do not permit the manufacturer code 0.
dwOptions	DWORD	Reserved for future extensions

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

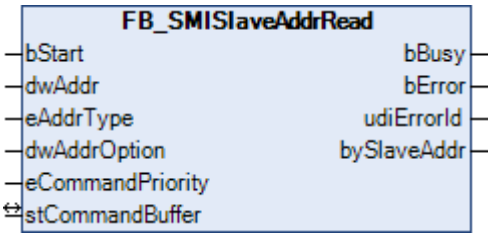
```
VAR_OUTPUT
  bBusy                                : BOOL;
  bError                               : BOOL;
  udiErrorId                           : UDINT;
  dwCurrentSearchSlaveId               : DWORD;
  eResSlaveAddrET0AndSlaveIdLTSerachId : E_SMICompResSlaveAddrET0AndSlaveIdLTSearchId;
  eResSlaveAddrET0AndSlaveIdGTSerachId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

Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).
dwCurrentSearchSlaveId	DWORD	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	E_SMICompResSlaveAddrET0AndSlaveIdLTSearchId [▶ 57]	At least one motor / no motor has the address 0 and the slave ID is smaller than the slave ID sought (*dwSlave-Id*) / the value is undefined.
eResSlaveAddrET0AndSlaveIdGTSerachId	E_SMICompResSlaveAddrET0AndSlaveIdGTSearchId [▶ 57]	At least one motor / no motor has the address 0 and the slave ID is larger than the slave ID sought (*dwSlave-Id*) / the value is undefined.
eResSlaveAddrET0AndSlaveIdETSerachId	E_SMICompResSlaveAddrET0AndSlaveIdETSearchId [▶ 57]	At least one motor / no motor has the address 0 and the slave ID is also the same as the slave ID sought (*dwSlave-Id*) / the value is undefined.
eResSlaveAddrNE0	E_SMICompResSlaveAddrNE0 [▶ 57]	At least one motor / no motor has an address unequal 0 / the value is undefined.

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.3.3 FB_SMISlaveAddrRead



The function block FB_SMISlaveAddrRead reads the address (0-15) of a drive.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.
eAddrType	E_SMIAddrType [▶ 56]	Defines whether the input *dwAddr* is to be evaluated as a manufacturer code [▶ 66], the address of a device, for group addressing or as a slave ID.
dwAddrOption	DWORD	If the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 66] must be specified via this input.
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

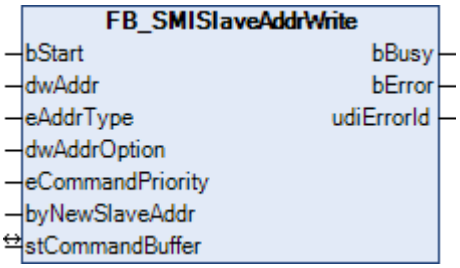
```
VAR_OUTPUT
  bBusy       : BOOL;
  bError      : BOOL;
  udiErrorId  : UDINT;
  bySlaveAddr : BYTE;
```

Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).
bySlaveAddr	BYTE	The read slave address (0-15)

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.3.4 FB_SMISlaveAddrWrite



The function block FB_SMISlaveAddrWrite writes the address (0-15) of one or more drives.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.
eAddrType	E_SMIAddrType [▶ 56]	Defines whether the input *dwAddr* is to be evaluated as a manufacturer code [▶ 66], the address of a device, for group addressing or as a slave ID.
dwAddrOption	DWORD	If the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 66] must be specified via this input.

Name	Type	Description
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.
byNewSlaveAddr	BYTE	The new slave address (0-15)

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

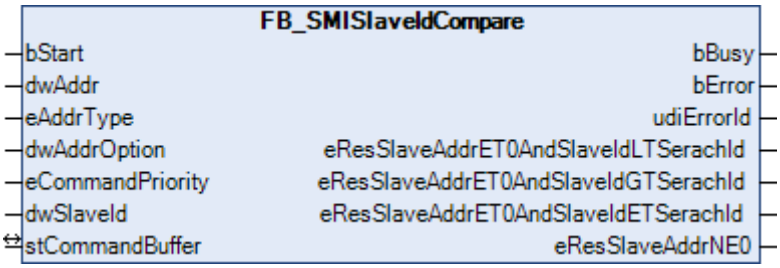
```
VAR_OUTPUT
  bBusy      : BOOL;
  bError     : BOOL;
  udiErrorId : UDINT;
END_VAR
```

Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.3.5 FB_SMISlaveIdCompare



The function block FB_SMISlaveIdCompare compares a specified slave ID (32-bit key ID) with the slave ID (32-bit key ID) defined on the motor side for one or more drives. 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.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.
eAddrType	E_SMIAddrType [▶ 56]	Defines whether the input *dwAddr* is to be evaluated as a manufacturer code [▶ 66], the address of a device, for group addressing or as a slave ID.
dwAddrOption	DWORD	If the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 66] must be specified via this input.
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.
dwSlaveId	DWORD	The Slave ID with which the Slave ID on the motor side is compared.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

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

Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.

Name	Type	Description
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).
eResSlaveAddrET0AndSlaveIdLTSerachId	E_SMICompResSlaveAddrET0AndSlaveIdLTSearchId [▶ 57]	At least one motor / no motor has the address 0 and the slave ID is smaller than the slave ID sought (*dwSlave-Id*) / the value is undefined.
eResSlaveAddrET0AndSlaveIdGTSerachId	E_SMICompResSlaveAddrET0AndSlaveIdGTSearchId [▶ 57]	At least one motor / no motor has the address 0 and the slave ID is larger than the slave ID sought (*dwSlave-Id*) / the value is undefined.
eResSlaveAddrET0AndSlaveIdETSerachId	E_SMICompResSlaveAddrET0AndSlaveIdETSearchId [▶ 57]	At least one motor / no motor has the address 0 and the slave ID is also the same as the slave ID sought (*dwSlave-Id*) / the value is undefined.
eResSlaveAddrNE0	E_SMICompResSlaveAddrNE0 [▶ 57]	At least one motor / no motor has an address unequal 0 / the value is undefined.

Examples

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

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

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

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

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 = 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.

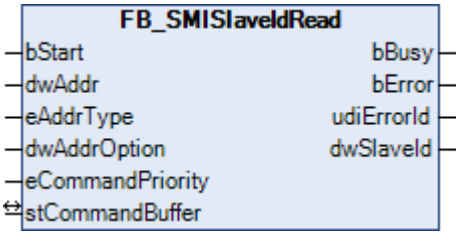
The slave ID (dwSlaveId) sought is the same as the slave ID of a drive:

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	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.3.6 FB_SMISlaveIdRead



The function block FB_SMISlaveIdRead reads the slave ID (32-bit key ID) from a drive.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.

Name	Type	Description
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.
eAddrType	E_SMIAddrType [▶ 56]	Defines whether the input *dwAddr* is to be evaluated as a manufacturer code [▶ 66], the address of a device, for group addressing or as a slave ID.
dwAddrOption	DWORD	If the SMI device is addressed by slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*), then the manufacturer code [▶ 66] must be specified via this input.
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

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

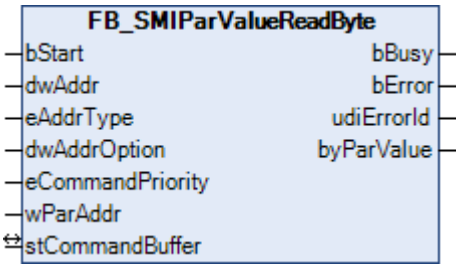
Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. The output is reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).
dwSlaveId	DWORD	The read slave ID

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.4 System commands

4.1.4.1 FB_SMIParValueReadByte



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

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.
eAddrType	E_SMIAddrType [▶ 56]	Specifies whether the input *dwAddr* is to be evaluated as manufacturer code [▶ 66], address of a device or for group addressing. Addressing via slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*) is not permitted.
dwAddrOption	DWORD	Reserved for future extensions
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.
wParAddr	WORD	Address of the parameter (0-4095) to be read.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

```
VAR_OUTPUT
  bBusy      : BOOL;
  bError     : BOOL;
```

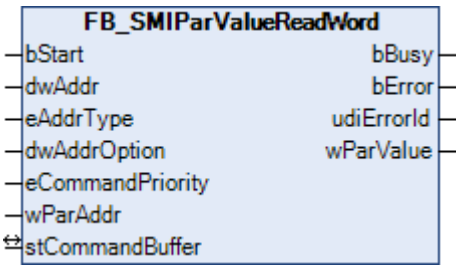
```
  udiErrorId : UDINT;
  byParValue : BYTE;
END_VAR
```

Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).
byParValue	BYTE	The read byte parameter

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.4.2 FB_SMIParValueReadWord



The function block FB_SMIParValueReadWord reads a word parameter (2 bytes) stored on the motor side. The meaning of the individual parameters is manufacturer-specific.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.
eAddrType	E_SMIAddrType [▶ 56]	Specifies whether the input *dwAddr* is to be evaluated as manufacturer code [▶ 66], address of a device or for group addressing. Addressing via slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*) is not permitted.

Name	Type	Description
dwAddrOption	DWORD	Reserved for future extensions
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.
wParAddr	WORD	Address of the parameter (0-4095) to be read.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

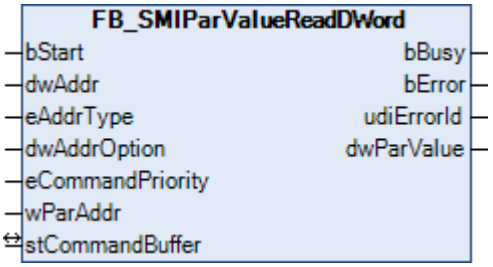
```
VAR_OUTPUT
  bBusy      : BOOL;
  bError     : BOOL;
  udiErrorId : UDINT;
  wParValue  : WORD;
END_VAR
```

Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).
wParValue	WORD	The read Word parameter

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.4.3 FB_SMIParValueReadDWord



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

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated and the command is sent by applying a positive edge to this input.
dwAddr	DWORD	Manufacturer code [▶ 66] (0-15), address of a device (0-15), bit field (16 bits) for group addressing or slave ID (32-bit key ID). This input has no meaning if a broadcast is sent.
eAddrType	E_SMIAddrType [▶ 56]	Specifies whether the input *dwAddr* is to be evaluated as manufacturer code [▶ 66], address of a device or for group addressing. Addressing via slave ID (*eAddrType* = *eSMIAddrTypeSlaveId*) is not permitted.
dwAddrOption	DWORD	Reserved for future extensions
eCommandPriority	E_SMICommandPriority [▶ 56]	Priority (high, medium or low) with which the command is processed by the PLC library.
wParAddr	WORD	Address of the parameter (0-4095) to be read.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Reference to the structure for communication (buffer) with the function block FB_KL6831KL6841Communication() [▶ 12]

Outputs

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

Name	Type	Description
bBusy	BOOL	This output is set as soon as the function block processes a command and remains active until the command has been processed.
bError	BOOL	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 reset to FALSE by the reactivation of the function block via the input *bStart*.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. It is reset to 0 by the reactivation of the function block via the input *bStart* (see error codes [▶ 55]).
dwParValue	DWORD	The read DWord parameter

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.1.5 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 send data from the SMI terminal.
0x8003	32771	The terminal has detected a telegram error (StatusWord.6 = true). This message must be acknowledged by the input bResetDataFrameError of FB_KL6831KL6841Communication().
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 incorrect.
0x800B	32779	No 24 V supply voltage to the KL6831/KL6841 (StatusWord.2 = false).
0x800C	32780	Process image was deactivated by the inputs Switch1 or Switch2 of the terminal (StatusWord.5 = true). This message must be acknowledged by the bResetInactiveProcessImage input of FB_KL6831KL6841Communication().
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 via slave ID (eAddrType = eSMIAddrTypeSlaveId).
0x800F	32783	Parameter wAddr (bit field for group addressing) is outside the valid range (0-65535).
0x8010	32784	Parameter wAddr (address) is outside the valid range (0-15).
0x8011	32785	Parameter eCommandPriority is invalid
0x8012	32786	Parameter eCommandType is invalid
0x8013	32787	Parameter uiAngle is outside the valid range (0-510)
0x8014	32788	Parameter wParAddr is outside the valid range (0-4095)
0x8015	32789	Parameter eAddrType is invalid
0x8016	32790	Parameter eResponseFormat is invalid
0x8017	32791	Parameter wAddr (manufacturer code) is outside the valid range (0-15)
0x8018	32792	The command supports only individual addressing.
0x8019	32793	Parameter wAddrOption (manufacturer code) is outside the valid range (0-15).
0x801A	32794	An internal error has occurred in the function block FB_SMIDiscoverySlaveId.
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).

Value (hex)	Value (dec)	Description
0x801E	32798	Parameter byHighestAddress (highest address) is outside the valid range (0-15).
0x801F	32799	Timeout for internal addressing. The terminal has not sent a response following the start of internal addressing.
0x8020	32800	The internal addressing failed three times.

4.2 DUTs

4.2.1 Enums

4.2.1.1 E_SMIConfigurationCommands

```
TYPE E_SMIConfigurationCommands :
(
  eSMICommandDoNothing := 0,
  eSMICommandUp        := 1,
  eSMICommandDown      := 2,
  eSMICommandStop      := 3,
  eSMICommandPos1      := 4,
  eSMICommandPos2      := 5
);
END_TYPE
```

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.32	Tc2_SMI from 3.3.6.0

4.2.1.2 E_SMIAddrType

```
TYPE E_SMIAddrType :
(
  eSMIAddrTypeManufacturer := 0,
  eSMIAddrTypeAddress      := 1,
  eSMIAddrTypeGroup        := 2,
  eSMIAddrTypeSlaveId      := 3,
  eSMIAddrTypeBroadcast    := 4
);
END_TYPE
```

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.2.1.3 E_SMICommandPriority

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

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.2.1.4 E_SMICommandType

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

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.2.1.5 E_SMICompResSlaveAddrET0AndSlaveIdETSearchId

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

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.2.1.6 E_SMICompResSlaveAddrET0AndSlaveIdGTSearchId

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

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.2.1.7 E_SMICompResSlaveAddrET0AndSlaveIdLTSearchId

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

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.2.1.8 E_SMICompResSlaveAddrNE0

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

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.2.1.9 E_SMIDiagResDrivesDown

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

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.2.1.10 E_SMIDiagResDrivesUp

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

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.2.1.11 E_SMIDiagResIsStopped

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

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.2.1.12 E_SMIDiagResWithError

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

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.2.1.13 E_SMIResponseFormat

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

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.2.2 Structures

4.2.2.1 ST_KL6831KL6841InData

```
TYPE ST_KL6831KL6841InData :
STRUCT
  wStateWord : WORD;
  arrData    : ARRAY [0..21] OF BYTE;
END_STRUCT
END_TYPE
```

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.2.2.2 ST_KL6831KL6841OutData

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

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.2.2.3 ST_SMICommandBuffer

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

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.2.2.4 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
```

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.2.2.5 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
```

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.2.2.6 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
```

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.2.2.7 ST_SMIResponseTableItem

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

Requirements

Development Environment	PLC library to include
TwinCAT from v3.1.4020.14	Tc2_SMI from 3.3.5.0

4.3 Integration into TwinCAT

4.3.1 KL6831 with CX5120

This sample describes how to write a simple PLC program for SMI in TwinCAT and how to link it with the hardware.

A motor is controlled stepwise with a button. One button sends the Up command, the other the Down command.

Sample: https://infosys.beckhoff.com/content/1033/tcplclib_tc2_smi/Resources/6012679435.zip

The TwinCAT project is available for download as *.zip file. This must first be unpacked locally so that the archive (*.tnzip file) is available for import into the TwinCAT project.

Hardware

Setting up the components

- 1x CX5120 Embedded PC
- 1 x KL1104 digital 4-channel input terminal (for the Up, Down and Reset function)
- 1x KL6831 SMI terminal
- 1x KL9010 end terminal

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

The sample assumes that a Reset button has been connected to the first input of the KL1104, an Up button to the second input and a Down button to the third input. There is a drive at the SMI device address 1.

Software

Creation of the PLC program

Create a new "TwinCAT XAE Project" and a "Standard PLC Project".

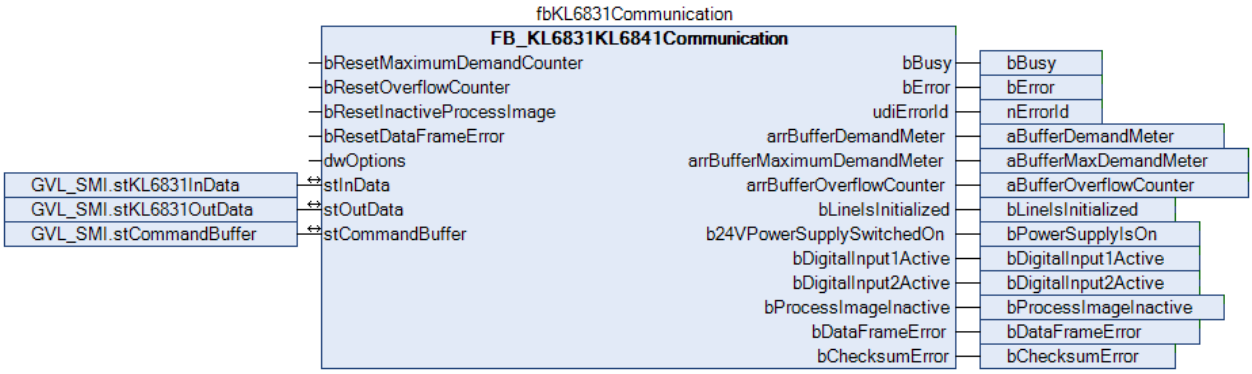
Add the library Tc2_SMI in the PLC project under "References".

Create the following global variables:

```
VAR_GLOBAL
    bReset              AT %I* : BOOL;
    bUp                 AT %I* : BOOL;
    bDown               AT %I* : BOOL;
    stKL6831InData      AT %I* : ST_KL6831KL6841InData;
    stKL6831OutData     AT %Q* : ST_KL6831KL6841OutData;
    stCommandBuffer            : ST_SMICommandBuffer;
END_VAR
```

Name	Type	Description
bReset	BOOL	Input variable for the Reset button
bUp	BOOL	Input variable for the Up button
bDown	BOOL	Input variable for the Down button
stKL6831InData	ST_KL6831KL6841InData [▶ 59]	Input variable for the SMI terminal
stKL6831OutData	ST_KL6831KL6841OutData [▶ 59]	Output variable for the SMI terminal
stCommandBuffer	ST_SMICommandBuffer [▶ 59]	Required for communication with SMI

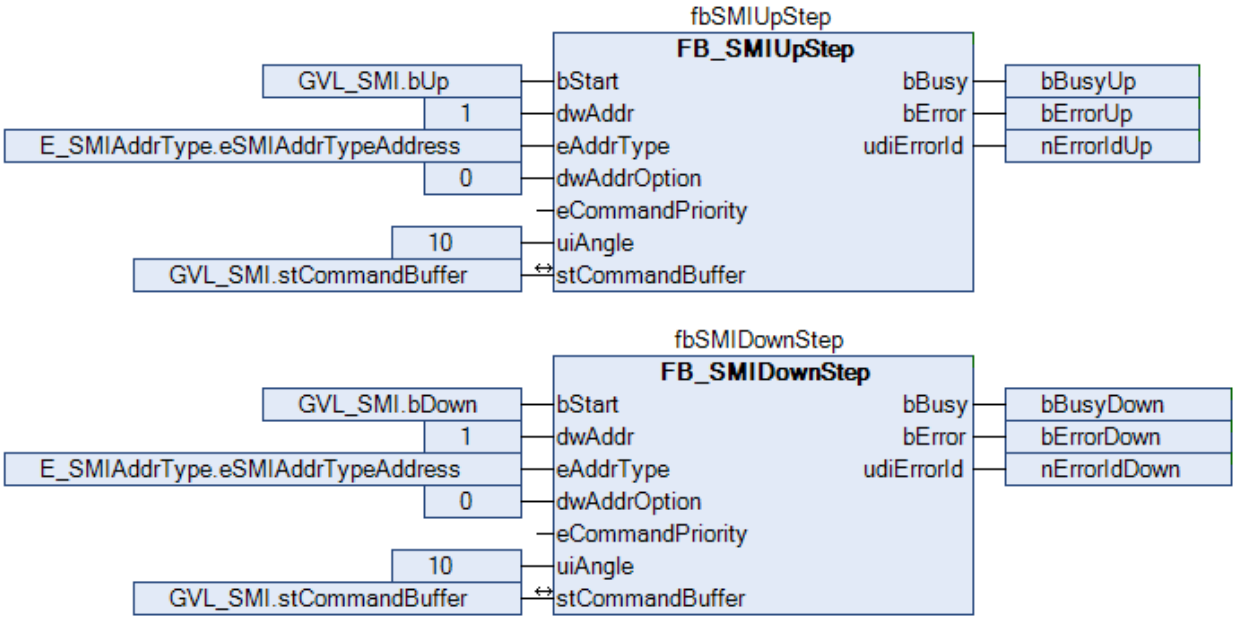
Create a program (CFC) for the background communication with SMI. The function block FB_KL6831KL6841Communication [▶ 12] is called in the program. With the communication block, ensure that the structures *stKL6381InData* and *stKL6831OutData* and *stCommandBuffer* are linked.



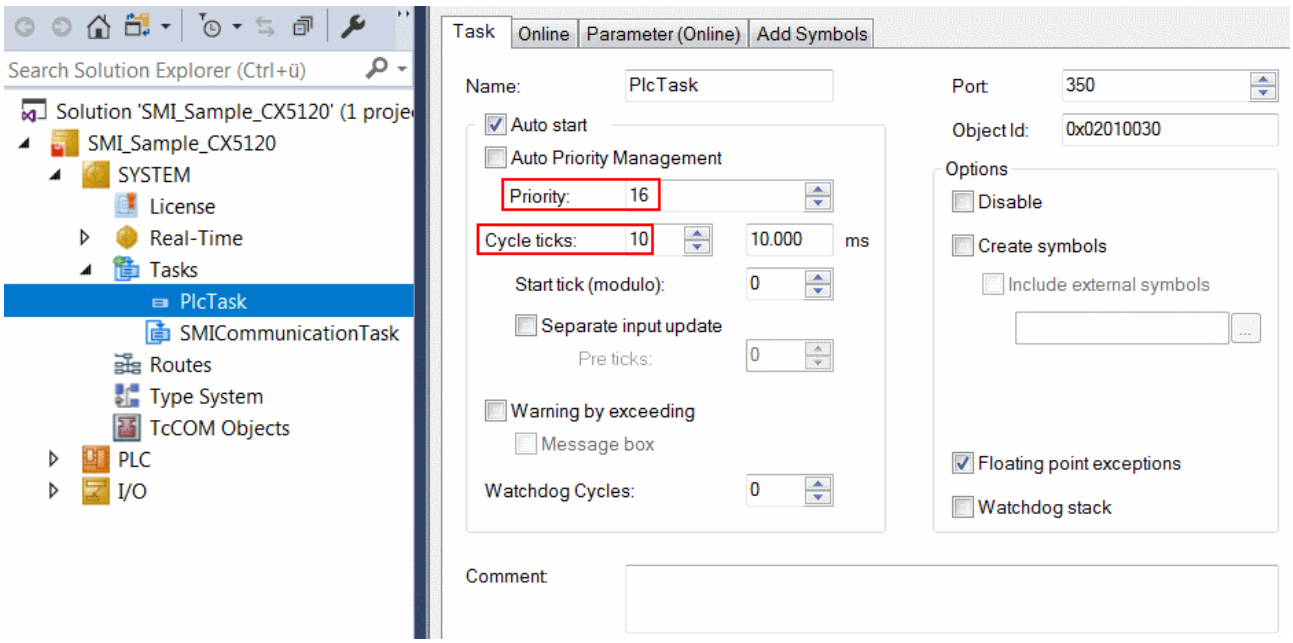
Create a MAIN program (CFC) in which the function blocks FB_SMIUpStep [▶ 39] and FB_SMIDownStep [▶ 23] are called.

The input *bStart* of the function block for sending the Up command is linked to the global variable *bUp*.

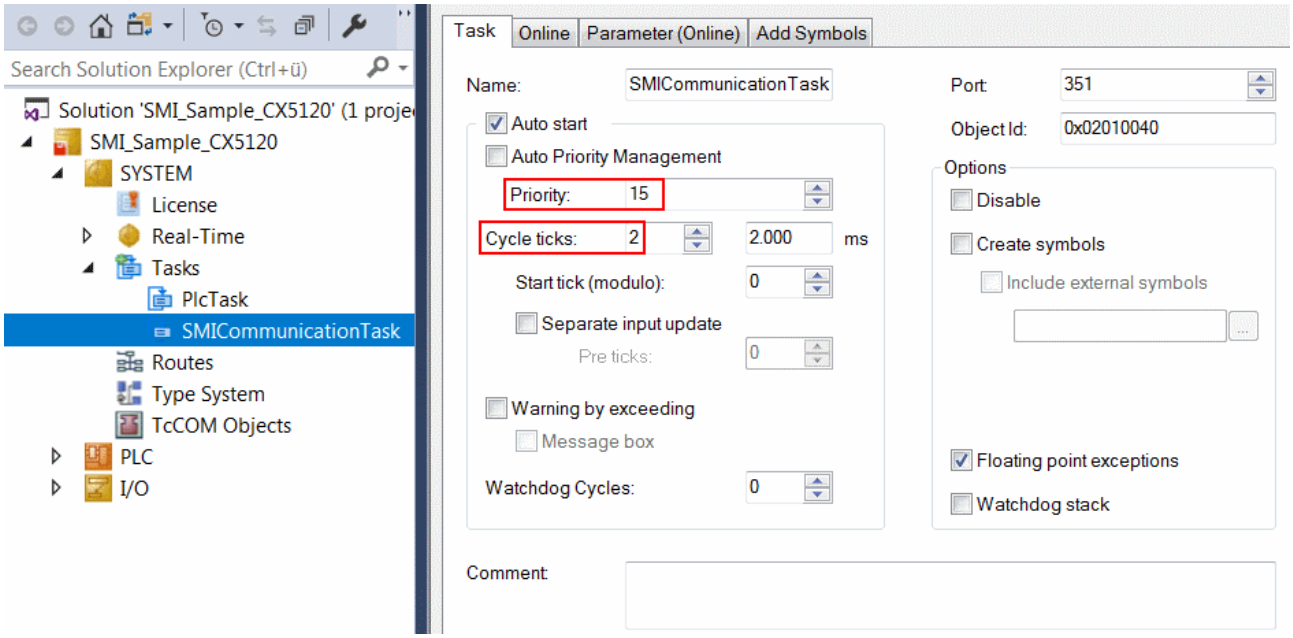
The input *bStart* of the function block for sending the Down command is linked to the global variable *bDown*.



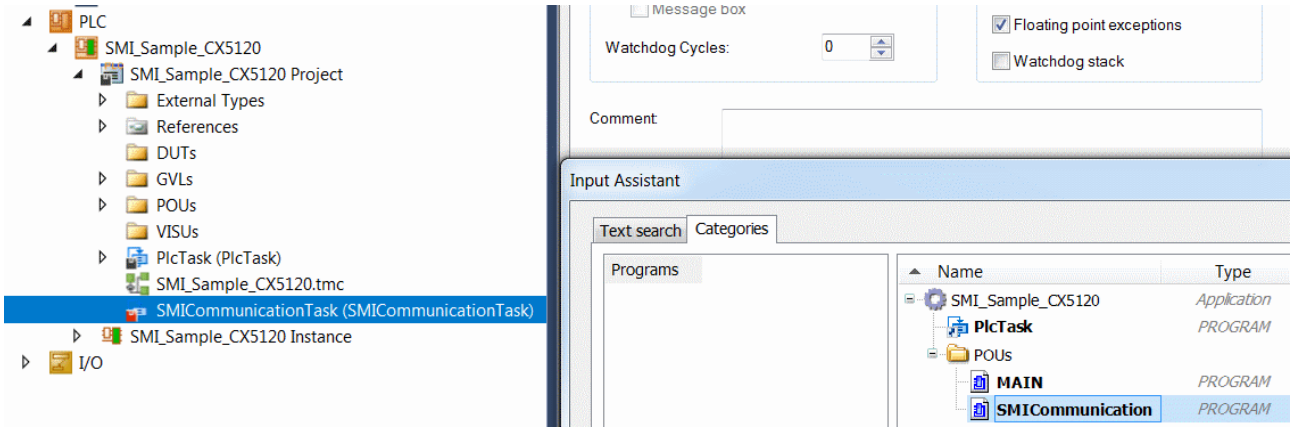
Navigate to the task configuration section and configure the PlcTask. By way of example, the task is assigned priority 16 and a cycle time of 10 ms.



Create a further task for the background communication. Assign a higher priority (smaller number) and a lower interval time to this task than the PlcTask.

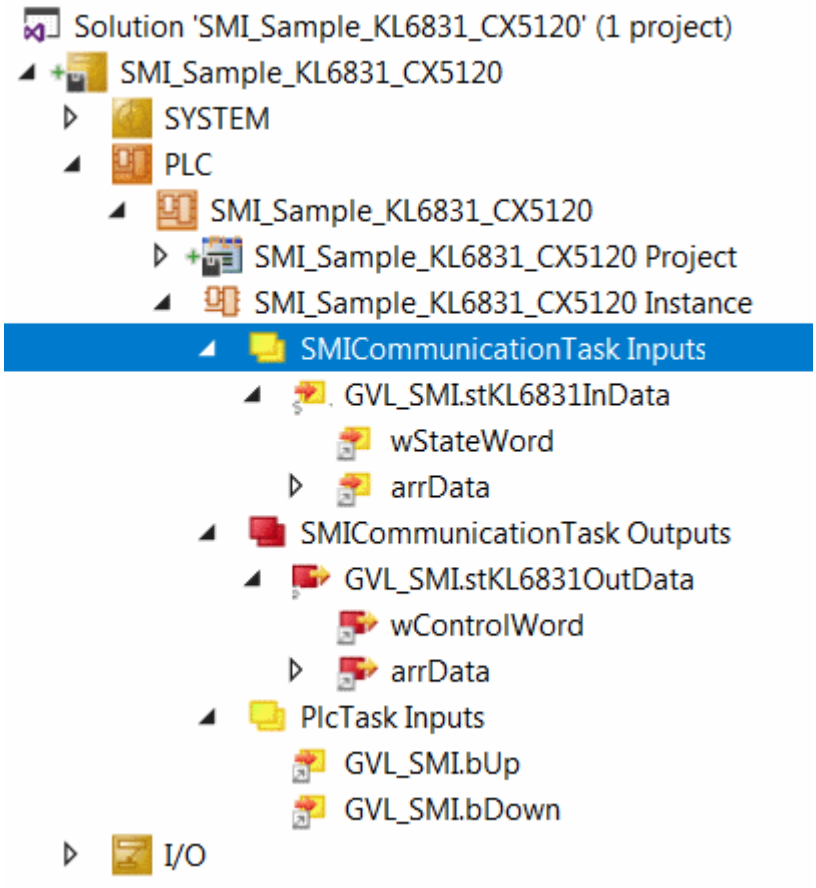


Add the program for the communication to this task. More precise information on the task configuration can be found in the function block description.



I/O configuration

Select the CX as target system and initiate a search for its hardware. In the project instance within the PLC section, you can see that the input and output variables are assigned to the corresponding tasks.



Now link the global variables of PLC program with the inputs and outputs of the bus terminals. Create the Solution and enable the configuration.

The lamp with the maximum brightness value is switched on by pressing the first push button. The second push button can be used to switch it off again.

You can use the Reset button to reset the inputs in *arrBufferMaximumDemandMeter* and *arrBufferOverflowCounter*.

Also see about this

- ST_KL6831KL6841InData [▶ 59]
- ST_KL6831KL6841OutData [▶ 59]
- ST_SMICommandBuffer [▶ 59]

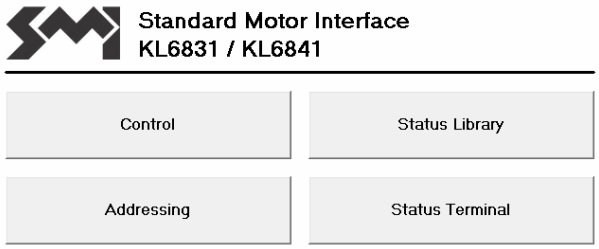
5 Appendix

5.1 Example: Configuration of SMI devices

TwinCAT 3 project: https://infosys.beckhoff.com/content/1033/tcplclib_tc2_smi/Resources/688934411.zip

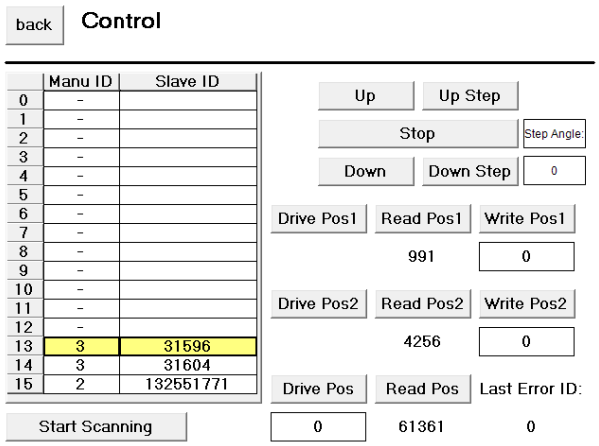
With the example it is possible to address SMI devices or to expand an existing installation. Moreover the dialogs for diagnostics and error analysis can be used. A total of 5 dialogs are available.

Start



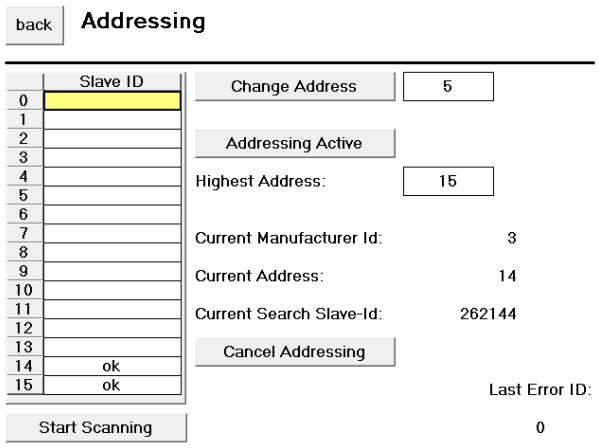
Under *SMI_Start* is the main menu, via which the four submenus can be accessed.

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 [▶ 66] 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 [▶ 55].

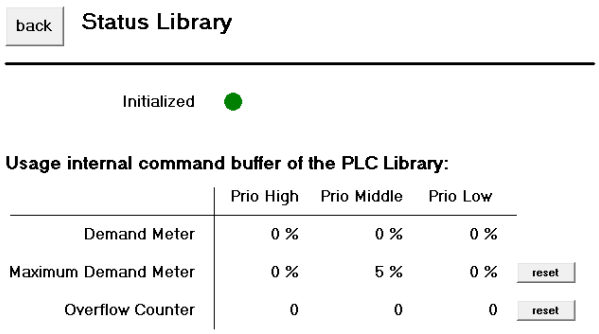
Addressing



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 Device Addressing [▶ 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.

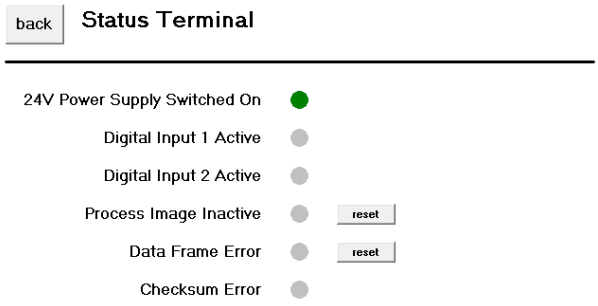
If SMI devices do not have an address yet, all SMI devices are assigned an address by pressing the *Start Addressing* button. The user has no influence over which address is assigned to which SMI device. The addresses are assigned in descending order, starting with the address specified by the *HighestAddress* 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.

5.2 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	Fa. SUN-MASTER Sonnenschutz GmbH
0x06	6	Vestamatic GmbH
0x07	7	WAREMA Renkhoff GmbH

Value (hex)	Value (dec)	Description
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

5.3 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

Trademark statements

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.

More Information:
www.beckhoff.com/te1000

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