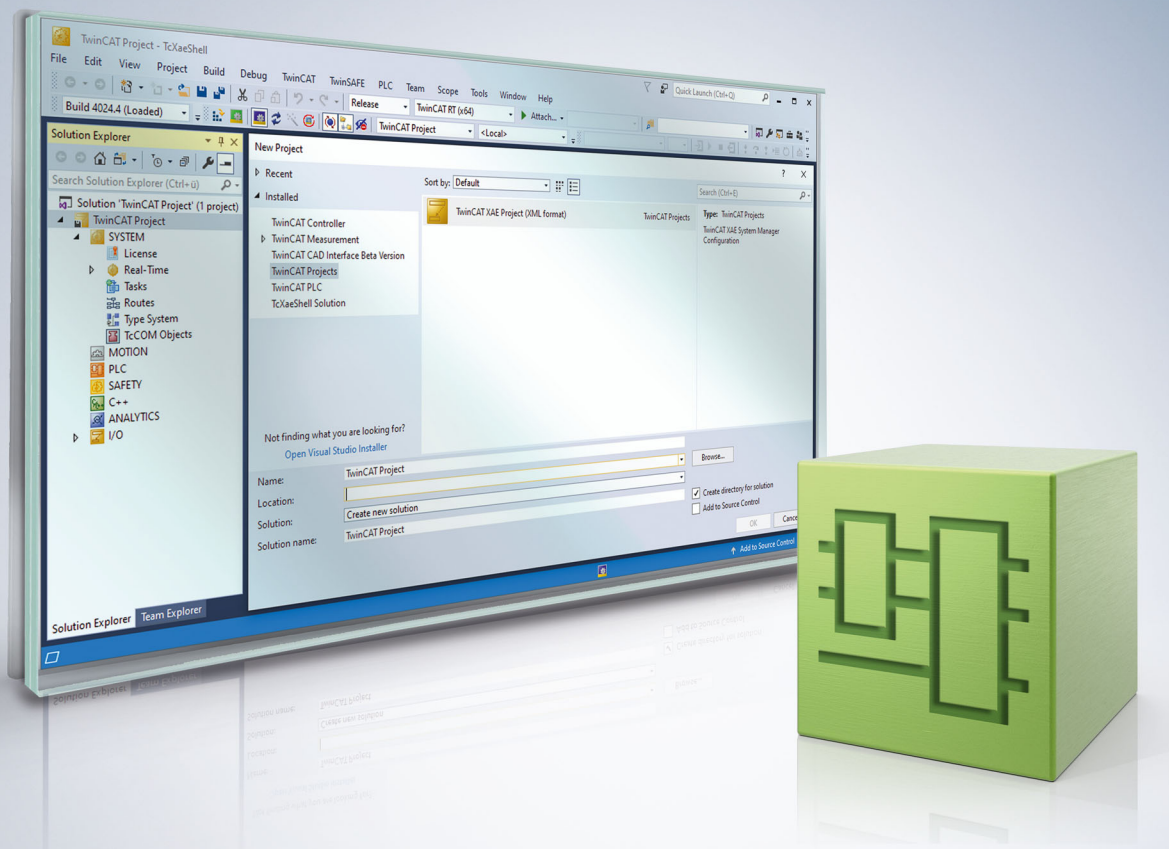
BECKHOFF New Automation Technology

Handbuch | DE

TE1000

TwinCAT 3 | PLC-Bibliothek: Tc2_SMI



15.11.2023 | Version: 1.7.1

Inhaltsverzeichnis

1 Vorwort

1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.
Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.
Das Fachpersonal ist verpflichtet, stets die aktuell gültige Dokumentation zu verwenden.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbaren Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiterentwickelt.
Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.
Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Marken

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® und XPlanar® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.
Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.

Patente

Die EtherCAT-Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

EtherCAT® ist eine eingetragene Marke und patentierte Technologie lizenziert durch die Beckhoff Automation GmbH, Deutschland

Copyright

1.2 Zu Ihrer Sicherheit

Sicherheitsbestimmungen

Lesen Sie die folgenden Erklärungen zu Ihrer Sicherheit.
Beachten und befolgen Sie stets produktspezifische Sicherheitshinweise, die Sie gegebenenfalls an den entsprechenden Stellen in diesem Dokument vorfinden.

Haftungsausschluss

Die gesamten Komponenten werden je nach Anwendungsbestimmungen in bestimmten Hard- und Software-Konfigurationen ausgeliefert. Änderungen der Hard- oder Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig und bewirken den Haftungsausschluss der Beckhoff Automation GmbH & Co. KG.

Qualifikation des Personals

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen vertraut ist.

Signalwörter

Im Folgenden werden die Signalwörter eingeordnet, die in der Dokumentation verwendet werden. Um Personen- und Sachschäden zu vermeiden, lesen und befolgen Sie die Sicherheits- und Warnhinweise.

Warnungen vor Personenschäden

⚠ GEFAHR

Es besteht eine Gefährdung mit hohem Risikograd, die den Tod oder eine schwere Verletzung zur Folge hat.

⚠ WARNUNG

Es besteht eine Gefährdung mit mittlerem Risikograd, die den Tod oder eine schwere Verletzung zur Folge haben kann.

⚠ VORSICHT

Es besteht eine Gefährdung mit geringem Risikograd, die eine mittelschwere oder leichte Verletzung zur Folge haben kann.

Warnung vor Umwelt- oder Sachschäden

HINWEIS

Es besteht eine mögliche Schädigung für Umwelt, Geräte oder Daten.

Information zum Umgang mit dem Produkt

Diese Information beinhaltet z. B.:
Handlungsempfehlungen, Hilfestellungen oder weiterführende Informationen zum Produkt.

1.3 Hinweise zur Informationssicherheit

Die Produkte der Beckhoff Automation GmbH & Co. KG (Beckhoff) sind, sofern sie online zu erreichen sind, mit Security-Funktionen ausgestattet, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen. Trotz der Security-Funktionen sind die Erstellung, Implementierung und ständige Aktualisierung eines ganzheitlichen Security-Konzepts für den Betrieb notwendig, um die jeweilige Anlage, das System, die Maschine und die Netzwerke gegen Cyber-Bedrohungen zu schützen. Die von Beckhoff verkauften Produkte bilden dabei nur einen Teil des gesamtheitlichen Security-Konzepts. Der Kunde ist dafür verantwortlich, dass unbefugte Zugriffe durch Dritte auf seine Anlagen, Systeme, Maschinen und Netzwerke verhindert werden. Letztere sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn entsprechende Schutzmaßnahmen eingerichtet wurden.

Zusätzlich sollten die Empfehlungen von Beckhoff zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Informationssicherheit und Industrial Security finden Sie in unserem https://www.beckhoff.de/secguide.

Die Produkte und Lösungen von Beckhoff werden ständig weiterentwickelt. Dies betrifft auch die Security-Funktionen. Aufgrund der stetigen Weiterentwicklung empfiehlt Beckhoff ausdrücklich, die Produkte ständig auf dem aktuellen Stand zu halten und nach Bereitstellung von Updates diese auf die Produkte aufzuspielen. Die Verwendung veralteter oder nicht mehr unterstützter Produktversionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Hinweise zur Informationssicherheit zu Produkten von Beckhoff informiert zu sein, abonnieren Sie den RSS Feed unter https://www.beckhoff.de/secinfo.

2 Einleitung

Für den Nutzer dieser Bibliothek werden folgende Grundkenntnisse vorausgesetzt:

- TwinCAT XAE
- PC und Netzwerkkenntnisse
- Aufbau und Eigenschaften der Beckhoff Embedded-PC und deren Busklemmensystem
- Technologie von SMI-Geräten
- Einschlägige Sicherheitsvorschriften der technischen Gebäudeausrüstung

Diese Softwarebibliothek ist für Gebäudeautomation-Systempartner der Beckhoff Automation GmbH & Co. KG. Die Systempartner sind tätig in dem Bereich Gebäudeautomation und beschäftigen sich mit Errichtung, Inbetriebsetzung, Erweiterung, Wartung und Service von mess-, steuer- und regelungstechnischen Anlagen der technischen Gebäudeausrüstung.

Die Tc2_SMI-Bibliothek ist auf allen Hardware-Plattformen einsetzbar, die TwinCAT 3.1 oder höher unterstützen.

Hardware Dokumentation KL6831_KL6841 im Beckhoff Information System.

3 SMI

Mit dem SMI-Bussystem (Standard Motor Interface) wird in der Gebäudeautomatisierung die Rollladen- und Sonnenschutzautomation vereinfacht. Mit SMI-Antrieben können exakte Positionen bei Rollladen und gradgenaue Winkelpositionen bei Jalousieantrieben angefahren werden. Die SMI-Antriebe können Istpositionen, Fehlermeldungen und Serviceinformationen an die SMI-Masterklemme zurücksenden.

Bedeutende europäische Hersteller haben sich zum SMI-Arbeitskreis zusammengeschlossen und das digitale Interface entwickelt. Über diese einheitliche Schnittstelle werden Antriebe mittels Telegrammen angesteuert. Mit Standard-Befehlen lassen sich Funktionen realisieren, die bei konventionellen Antrieben nicht so leicht möglich sind. Beispiele sind das präzise Anfahren von Positionen, die Rückmeldung der aktuellen Position so wie die Diagnose. Für die Verstellung von Lamellen der Verschattungsanlage können z. B. Winkelauflösungen von 2° erreicht werden. Damit ist eine sonnenstandsabhängige Nachführung der Lamellen zur Konstantlichtregelung möglich. In der TwinCAT-HVAC-Bibliothek stehen leistungsfähige SPS-Bausteine für die Raumautomation nach VDI 3813 zur Verfügung.

Es sind Distanzen bis 350 Meter zwischen Steuerung und Antrieb möglich. Zur Verkabelung kann ein normales 5-adriges Stromkabel verwendet werden (mit PE, N, L sowie dem SMI spezifischen I+ und I-), wobei I+ und I- verpolungssicher sind. Bis zu 16 Antriebe können parallelgeschaltet und einzeln adressiert werden. SMI-Antriebe gibt es für Netzspannung (230 V AC) und für Kleinspannung (24 V DC).

Um die Kompatibilität der SMI-Produkte untereinander zu gewährleisten, müssen alle Produkte, die mit dem SMI-Logo gekennzeichnet werden sollen, zertifiziert werden. Eine positive Zertifizierung lässt sich auf der SMI-Group-Homepage (https://standard-motor-interface.com) nachlesen. Dort finden sie auch weitere Informationen über den SMI-Bus und SMI-Antriebe.

3.1 Geräte Adressierung

SMI definiert verschiedene Arten der Teilnehmeradressierung. Grundsätzlich kann unterschieden werden zwischen der Einzeladressierung, Gruppenadressierung und dem Sammelruf (Broadcast). Die meisten SPS-Bausteine besitzen hierfür den Eingang *dwAddr* und *eAddrType*. Während der Eingang *dwAddr* die notwendige Angabe der Adresse enthält, definiert *eAddrType* die Art der Adressierung. Die einzelnen Arten der Adressierung werden im Folgenden beschrieben. Beachten Sie, dass nicht jeder Befehl alle Adressierungsarten unterstützt. Details hierzu finden Sie in der Beschreibung des jeweiligen SPS-Bausteins.

per Adresse eines Teilnehmers (eAddrType := eSMIAddrTypeAddress)

Jedem SMI-Teilnehmer kann eine Adresse von 0 bis 15 zugewiesen werden. Die Adresse wird im SMI-Teilnehmer abgespeichert und muss bei einem Austausch des Antriebes wieder korrekt eingestellt werden. Da jede Adresse nur einmal zugewiesen werden sollte, lässt sich jeder SMI-Teilnehmer einzeln ansprechen. Der Eingang *dwAddr* enthält bei dieser Adressierungsart die Adresse im Bereich von 0 bis 15. Wird ein Wert außerhalb des gültigen Bereichs angegeben, so gibt der jeweilige Baustein einen Fehler [▶ 48] aus.

Gelegentlich wird diese Adresse auch Slave-Adresse genannt. Die Slave-Adresse darf nicht mit der Slave-Id verwechselt werden (siehe unten).

per Slave-Id (eAddrType := eSMIAddrTypeSlaveId)

Die einzelnen Gerätehersteller hinterlegen in jedem SMI-Gerät eine eindeutige 32-Bit große Nummer. Diese Slave-Id, auch Key-Id genannt, kann ebenfalls für die Adressierung eines Teilnehmers genutzt werden. Der Eingang *dwAddr* enthält bei dieser Adressierungsart die 32-Bit große Slave-Id und der Eingang *dwAddrOption* den Herstellercode (siehe unten). Hierdurch wird sichergestellt, dass SMI-Geräte weltweit eindeutig adressiert werden können, ohne dass diese eine Adresse benötigen.

Bei einigen SMI-Geräten ist die Slave-Id auf dem Typenschild aufgedruckt oder durch eine Beschriftung am Kabel sichtbar.

Die Adressierung per Slave-Id wird von den meisten Lese-Befehlen nicht unterstützt.

per Herstellercode (eAddrType := eSMIAddrTypeManufacturer)

SMI definiert für jeden Hersteller neben der Slave-Id eine weitere eindeutige Id, den sogenannten Herstellercode [▶ 61]. Der Herstellercode ist fest im SMI-Gerät hinterlegt und kann nicht verändert werden. Der mögliche Wertebereich ist von 0 bis 15, wobei der Wert 0 und 14 eine Sonderbedeutung haben. Der Herstellercode 0 adressiert alle Teilnehmer, unabhängig vom Hersteller. Somit kann diese Adressierungsart auch zum Versenden von Broadcast-Befehlen verwendet werden. Der Wert 15 ist reserviert für zukünftige Erweiterungen und darf nicht verwendet werden. Sehr häufig ist hier die englische Bezeichnung *Manufacturer Code* oder auch die Abkürzung *M-ID* zu finden. Durch diese Adressierung werden immer alle Geräte eines Herstellers angesprochen. Der Eingang *dwAddr* enthält bei dieser Adressierungsart den Herstellercode im Bereich von 0 bis 14. Wird ein Wert außerhalb des gültigen Bereichs angegeben, so gibt der jeweilige Baustein einen Fehler [▶ 48] aus.

Bei einigen SMI-Geräten ist der Herstellercode auf dem Typenschild aufgedruckt oder durch eine Beschriftung am Kabel sichtbar.

per Gruppenadressierung (eAddrType := eSMIAddrTypeGroup)

Jeder Teilnehmer der über die Gruppenadressierung angesteuert werden soll, muss eine Adresse von 0 bis 15 besitzen. Jedes Bit des Eingangs *dwAddr* entspricht bei der Gruppenadressierung einer Adresse. Wird Bit 0 von *dwAddr* gesetzt, so wird der Teilnehmer mit der Adresse 0 angesteuert. Wird Bit 1 gesetzt, so wird Teilnehmer 1 angesprochen usw. Somit belegt die Gruppenadressierung die Bits 0 bis 15, was dem Wertebereich von 0 bis 65535 entspricht. Wird ein Wert außerhalb des gültigen Bereichs angegeben, so gibt der jeweilige Baustein einen Fehler [▶ 48] aus.

Beispiel: Es sollen die Antriebe mit der Adresse 1, 4, 7 und 12 angesprochen werden. Somit muss an *dwAddr* der Wert 2#00010000_10010010 oder 16#1092 oder 4242 übergeben werden.

per Broadcast (eAddrType := eSMIAddrTypeBroadcast)

Bei der Adressierung per Broadcast werden immer alle Teilnehmer angesprochen, unabhängig von der eingestellten Adresse am Gerät. Der Eingang *dwAddr* wird bei dieser Adressierung nicht benötigt und auch nicht ausgewertet. Intern verwenden die SPS-Bausteine die Adressierung per Herstellercode, wobei als Herstellercode 0 verwendet wird.

4 Programmierung

4.1 POUs

Basisbefehle

Name	Beschreibung
FB_KL6831KL6841Communication [▶ 12]	Liest sequentiell die SMI-Befehle aus den internen Puffern der SPS-Bibliothek aus und gibt diese zu der KL6831/KL6841.
FB_KL6831KL6841Config [▶ 14]	Mit diesem Baustein kann die KL6831/KL6841 konfiguriert werden.
FB_SMISendSMICommand [▶ 17]	Dieser Baustein dient zum allgemeinen Senden eines SMI-Kommandos.

Grundbefehle

Name	Beschreibung
FB_SMIDiagAll [▶ 18]	Diagnosetelegramm wird versendet.
FB_SMIDown [▶ 20]	Motorlauf bis zur unteren Endlage.
FB_SMIDownStep [▶ 21]	Motorlauf nach unten um einen vorgegebenen Winkelgrad.
FB_SMIPos1 [▶ 22]	Fahrt zur motorseitig konfigurierten Fixposition *Pos1*.
FB_SMIPos1Read [▶ 23]	Lesen der motorseitig konfigurierten Fixposition *Pos1*.
FB_SMIPos1Write [▶ 25]	Schreiben der motorseitig konfigurierten Fixposition *Pos1*.
FB_SMIPos2 [▶ 26]	Fahrt zur motorseitig konfigurierten Fixposition *Pos2*.
FB_SMIPos2Read [▶ 27]	Lesen der motorseitig konfigurierten Fixposition *Pos2*.
FB_SMIPos2Write [▶ 28]	Schreiben der motorseitig konfigurierten Fixposition *Pos2*.
FB_SMIPosRead [▶ 29]	Aktuelle Position lesen.
FB_SMIPosWrite [▶ 30]	Anfahren einer Position.
FB_SMIStop [▶ 31]	Stopp des Motorlaufs.
FB_SMISyn [▶ 32]	Abfragen Herstellercode und Antriebstyp.
FB_SMIUp [▶ 33]	Motorlauf bis zur oberen Endlage.
FB_SMIUpStep [▶ 34]	Motorlauf nach oben um einen vorgegebenen Winkelgrad.

Adressierungsbefehle

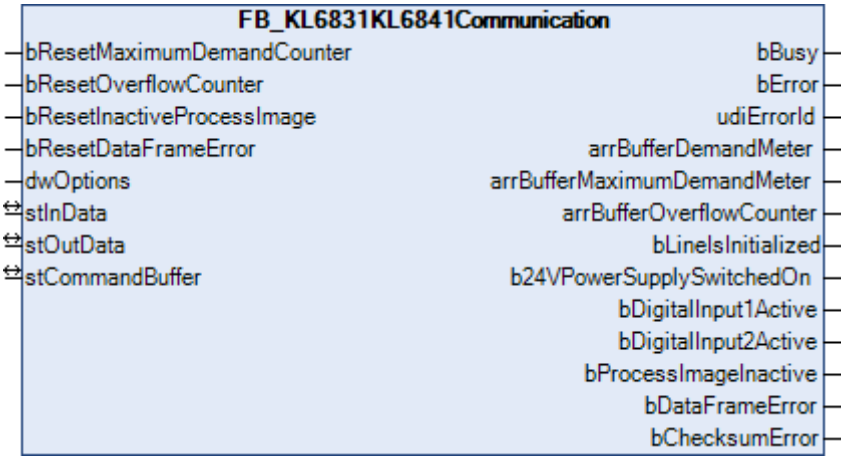
Name	Beschreibung
FB_SMIAddressing [▶ 35]	SMI-Geräte adressieren.
FB_SMIDiscoverySlaveId [▶ 36]	SMI-Geräte nach Herstellercode suchen.
FB_SMISlaveAddrRead [▶ 38]	Adresse (0-15) eines Antriebes auslesen.
FB_SMISlaveAddrWrite [▶ 39]	Adresse (0-15) in einen oder mehrere Antriebe schreiben.
FB_SMISlaveIdCompare [▶ 40]	Vergleichen einer vorgegebenen Slave-Id (32 Bit Key-Id) mit der motorseitig definierten Slave-Id (32 Bit Key-Id) eines oder mehrerer Antriebe.
FB_SMISlaveIdRead [▶ 42]	Lesen der Slave-Id (32 Bit Key-Id).

Systembefehle

Name	Beschreibung
FB_SMIParValueReadByte [▶ 44]	Lesen eines motorseitig gespeicherten Byte-Parameters (1-Byte).
FB_SMIParValueReadWord [▶ 45]	Lesen eines motorseitig gespeicherten Word-Parameters (2-Bytes).
FB_SMIParValueReadDWord [▶ 46]	Lesen eines motorseitig gespeicherten DWord-Parameters (4-Bytes).

4.1.1 Basisbefehle

4.1.1.1 FB_KL6831KL6841Communication



Die Bausteine für die SMI-Befehle greifen nicht direkt auf das Prozessabbild der KL6831/KL6841 zu, sondern legen die einzelnen SMI-Befehle in drei verschiedene Puffer ab. Der Baustein *FB_KL6831KL6841Communication()* liest sequentiell die SMI-Befehle aus diesen drei Puffern aus und gibt die SMI-Befehle zu der KL6831/KL6841 weiter. Hierdurch wird sichergestellt, dass nicht mehrere Bausteine gleichzeitig auf das Prozessabbild der KL6831/KL6841 zugreifen. Jeder dieser drei Puffer wird mit einer anderen Priorität (hoch, mittel oder niedrig) abgearbeitet. Durch den Parameter eCommandPriority [▶ 50], den es bei den meisten Bausteinen gibt, können Sie beeinflussen, mit welcher Priorität der jeweilige SMI-Befehl von dem Baustein *FB_KL6831KL6841Communication()* bearbeitet werden soll.

Die Puffer, in denen die SMI-Befehle abgelegt werden, sind alle in einer Variablen vom Typ ST_SMICommandBuffer [▶ 53] enthalten. Pro KL6831/KL6841 gibt es eine Instanz vom Baustein *FB_KL6831KL6841Communication()* und eine Variable vom Typ *ST_SMICommandBuffer*. Der Baustein *FB_KL6831KL6841Communication()* sollte, wenn möglich, in einer separaten, schnelleren Task aufgerufen werden.

Über die Ausgänge des Bausteins kann ermittelt werden, wie stark die Puffer ausgelastet sind. Hierzu werden drei Arrays ausgegeben, bei dem jedes Element (0, 1 oder 2) für einen der drei Puffer (hoch, mittel oder niedrig) steht. Sollten Sie feststellen, dass einer der drei Puffer regelmäßig überläuft, so sollten Sie folgende Maßnahmen in Betracht ziehen:

- Wie stark sind die einzelnen SPS-Task ausgelastet? TwinCAT XAE bietet zur Analyse entsprechende Hilfsmittel an.
- Versuchen Sie die Zykluszeit der Task, in der der Baustein *FB_KL6831KL6841Communication()* aufgerufen wird, zu verringern. Der Wert sollte nicht größer als 6ms sein, optimal sind 2ms.
- Überprüfen Sie die Zykluszeit der SPS-Task, in der die Bausteine für die einzelnen SMI-Befehle aufgerufen werden. Dieser Wert sollte zwischen 10ms und 60ms liegen.
- Vermeiden Sie möglichst das Pollen (regelmäßiges Auslesen) von Werten. Lesen Sie nur dann Werte aus, wenn diese auch benötigt werden.
- Verteilen Sie die einzelnen Antriebe gleichmäßig auf mehrere SMI-Linien. Da pro SPS-Zyklus mehrere SMI-Linien gleichzeitig bearbeitet werden, erhöht sich hierdurch der Datendurchsatz insgesamt.

VAR_INPUT

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

bResetMaximumDemandCounter: Eine positive Flanke setzt den gespeicherten Wert der maximalen Befehlspuffer-Auslastung *(arrBufferMaximumDemandMeter)* zurück.

bResetOverflowCounter: Eine positive Flanke setzt den gespeicherten Wert der Anzahl der Befehlspuffer-Überläufe *(arrBufferOverflowCounter)* zurück.

bResetInactiveProcessImage: Eine positive Flanke hebt die Sperrung des Prozessabbildes der Klemme wieder auf. Die Ausgänge *bProcessImageInactive*, *bDigitalInput1Active* und *bDigitalInput2Active* werden wieder auf FALSE gesetzt. Die Sperrung wird aktiviert, sobald einer der digitalen Eingänge *Input1* oder *Input2* an der Klemme betätigt wurde.

bResetDataFrameError: Eine positive Flanke setzt die Meldung für einen Telegrammfehler wieder zurück. Der Ausgang *bDatafRameError* wird wieder auf FALSE gesetzt. Die Sperrung wird aktiviert, sobald von der Klemme ein Telegrammfehler erkannt wurde.

dwOptions: reserviert für zukünftige Erweiterungen.

VAR_OUTPUT

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

bBusy: Der Ausgang wird gesetzt, sobald der Baustein einen Befehl verarbeitet und bleibt so lange aktiv, bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

arrBufferDemandMeter: Belegung des jeweiligen Puffers (0 - 100%).

arrBufferMaximumDemandMeter: Bisherige maximale Auslastung des jeweiligen Puffers (0 - 100%).

arrBufferOverflowCounter: Bisherige Anzahl der Pufferüberläufe.

bLineIsInitialized: Wird der Baustein das erste Mal aufgerufen (z.B. beim Starten der Steuerung), so wird eine Initialisierung durchgeführt. Während dieser Zeit können keine SMI-Befehle bearbeitet werden. Ist die Initialisierung abgeschlossen, wird dieser Ausgang auf TRUE gesetzt.

b24VPowerSupplySwitchedOn: Die KL6831/KL6841 muss über zwei Anschlüsse mit 24 V DC versorgt werden. Der Ausgang wird gesetzt, sobald die 24 V DC erkannt wurden. Fehlen die 24 V DC geht der Ausgang auf FALSE und es können keine SMI-Befehle über die Steuerung bearbeitet werden, solang die 24 V DC nicht vorhanden sind.

bDigitalInput1Active: Der digitale Eingang *Input1* an der Klemme wurde betätigt oder ist betätigt (siehe auch Klemmendokumentation). Der Ausgang *bProcessImageInactive* wird gesetzt und es können keine weiteren SMI-Befehle über die Steuerung bearbeitet werden.

bDigitalInput2Active: Der digitale Eingang *Input2* an der Klemme wurde betätigt oder ist betätigt (siehe auch Klemmendokumentation). Der Ausgang *bProcessImageInactive* wird gesetzt und es können keine weiteren SMI-Befehle über die Steuerung bearbeitet werden.

bProcessImageInactive: Einer der digitalen Eingänge *Input1* oder *Input2* wurde an der Klemme betätigt. Es können keine weiteren SMI-Befehle über die Steuerung bearbeitet werden. Über den Eingang *bResetInactiveProcessImage* muss die Sperrung wieder freigeschaltet werden.

bDataFrameError: Die Klemme hat einen Telegrammfehler auf dem SMI-Bus erkannt. Über den Eingang *bResetDataFrameError* muss der Fehler wieder zurückgesetzt werden.

bChecksumError: Die Klemme hat einen Checksummenfehler auf dem SMI-Bus erkannt. Die Meldung wird automatisch zurückgesetzt, sobald ein Telegramm wieder fehlerfrei übertragen wurde.

VAR_IN_OUT

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

stInData: Verweis auf die Struktur zur Kommunikation mit der KL6831/KL6841 (siehe ST_KL6831KL6841InData [▶ 53]).

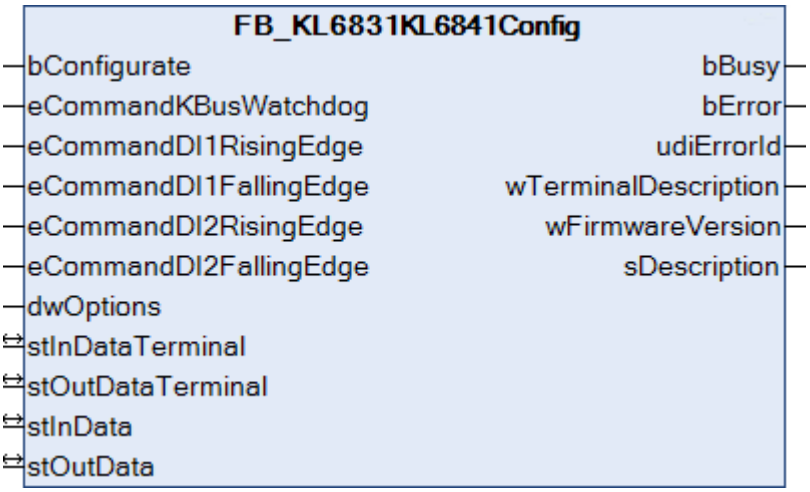
stOutData: Verweis auf die Struktur zur Kommunikation mit der KL6831/KL6841 (siehe ST_KL6831KL6841OutData [▶ 53]).

stCommandBuffer: Verweis auf die Struktur zur Kommunikation mit den SMI-Bausteinen (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

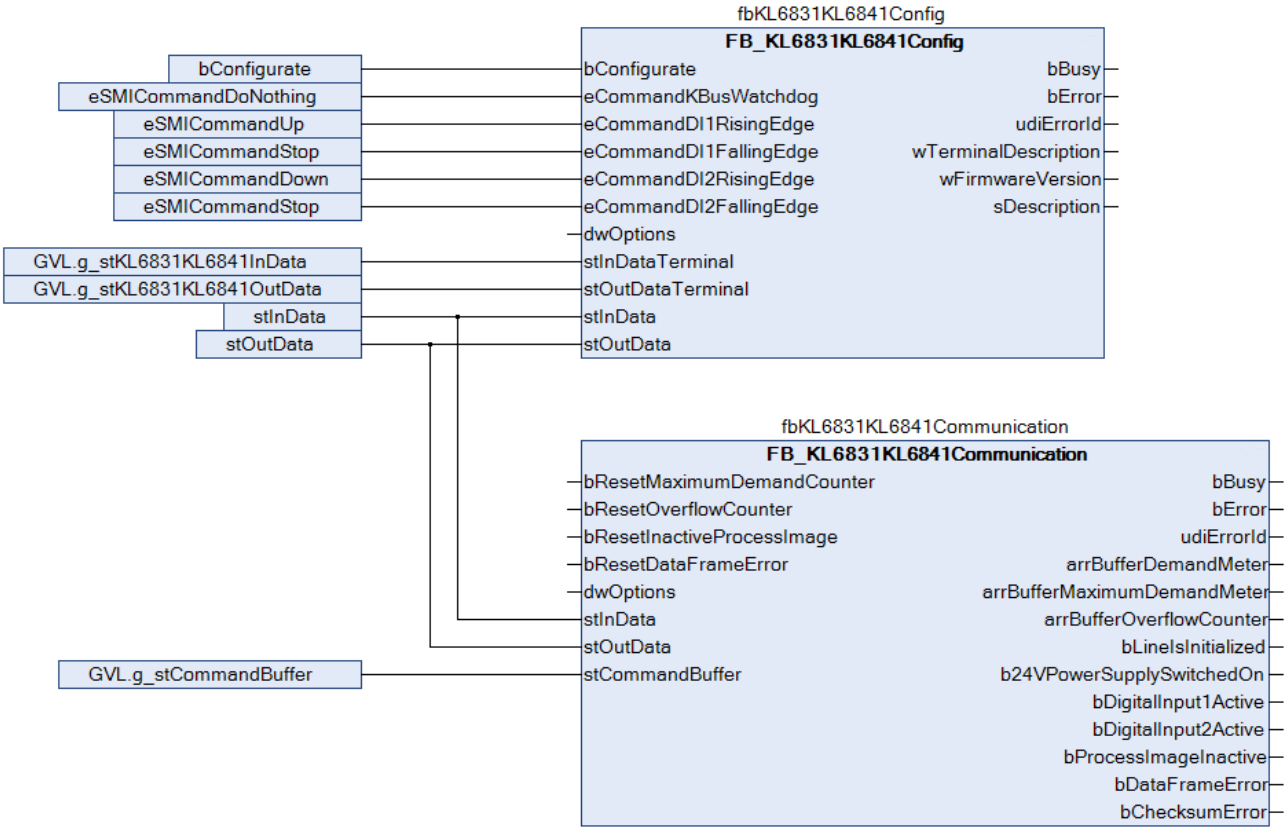
4.1.1.2 FB_KL6831KL6841Config



Dieser Baustein dient zum Konfigurieren der KL6831/KL6841. Das Konfigurieren wird beim Aufstarten des SPS-Programms ausgeführt oder durch eine positive Flanke am Eingang *bConfigurate*. Die Parameter werden in den jeweiligen Registern der KL6831/KL6841 spannungsausfallsicher abgespeichert. Des Weiteren werden aus der KL6831/KL6841 einige allgemeine Informationen, wie die Version der Firmware, ausgelesen.

Beispiel:

Der Baustein wird in der gleichen Task, wie der Baustein FB_KL6831KL6841Communication() [▶ 12] aufgerufen.



Der Baustein FB_KL6831KL6841Config() ist mit dem Prozessabbild der KL6831/KL6841 verbunden. Nach Abschluss der Konfiguration erhält der Baustein FB_KL6831KL6841Communication() [▶ 12] die Prozesswerte der KL6831/KL6841. Während des Konfigurieren können keine SMI-Befehle versendet werden.

https://infosys.beckhoff.com/content/1031/tcplclib_tc2_smi/Resources/3248663563.zip

VAR_INPUT

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

bConfigurate: Durch eine positive Flanke an diesem Eingang wird das Konfigurieren der Busklemme gestartet.

eCommandKBusWatchdog: Definiert den SMI-Befehl, der versendet wird, sobald die Busklemme über den K-Bus nicht mehr angesprochen wird (siehe E_SMIConfigurationCommands [▶ 50]). Entspricht Register 33 bis 35 der Busklemme.

eCommandDI1RisingEdge: Definiert den SMI-Befehl, der versendet wird, sobald am Eingang 1 der Busklemme eine steigende Flanke erkannt wird (siehe E_SMIConfigurationCommands [▶ 50]). Entspricht Register 36 bis 38 der Busklemme.

eCommandDI1FallingEdge: Definiert den SMI-Befehl, der versendet wird, sobald am Eingang 1 der Busklemme eine fallende Flanke erkannt wird (siehe E_SMIConfigurationCommands [▶ 50]). Entspricht Register 39 bis 41 der Busklemme.

eCommandDI2RisingEdge: Definiert den SMI-Befehl, der versendet wird, sobald am Eingang 2 der Busklemme eine steigende Flanke erkannt wird (siehe E_SMIConfigurationCommands [▶ 50]). Entspricht Register 42 bis 44 der Busklemme.

eCommandDI2FallingEdge: Definiert den SMI-Befehl, der versendet wird, sobald am Eingang 2 der Busklemme eine fallende Flanke erkannt wird (siehe E_SMIConfigurationCommands [▶ 50]). Entspricht Register 45 bis 47 der Busklemme.

dwOptions: Reserviert für zukünftige Erweiterungen.

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bConfigurate* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bConfigurate* wieder auf 0 zurückgesetzt. Siehe Fehlercodes.

wTerminalDescription: Enthält die Klemmenbezeichnung (z.B. 6831). Entspricht Register 8 der Busklemme.

wFirmwareVersion: Enthält die Version der Firmware. Entspricht Register 9 der Busklemme.

sDescription: Klemmenbezeichnung und die Version der Firmware als String (z.B. 'Terminal KL6831 / Firmware 1D').

VAR_IN_OUT

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

stInDataTerminal: Verweis auf die Struktur zur Kommunikation mit der KL6831/KL6841 (siehe ST_KL6831KL6841InData [▶ 53]).

stOutDataTerminal: Verweis auf die Struktur zur Kommunikation mit der KL6831/KL6841 (siehe ST_KL6831KL6841OutData [▶ 53]).

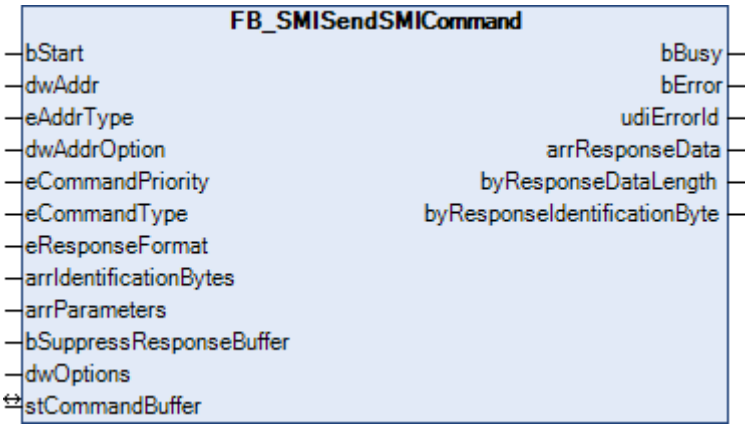
stInData: Verweis auf die Struktur zur Kommunikation mit dem Baustein FB_KL6831KL6841Communication() [▶ 12] (siehe ST_KL6831KL6841InData [▶ 53]).

stOutData: Verweis auf die Struktur zur Kommunikation mit dem Baustein FB_KL6831KL6841Communication() [▶ 12] (siehe ST_KL6831KL6841OutData [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.32	Tc2_SMI ab 3.3.6.0

4.1.1.3 FB_SMISendSMICommand



Dieser Baustein dient zum allgemeinen Senden eines SMI-Kommandos. Hierzu muss der genaue Aufbau eines SMI-Befehls und die Funktionsweise der KL6831/KL6841 bekannt sein. Der Einsatz dieses Bausteins ist nur notwendig, wenn ein SMI-Befehl versendet werden soll, der nicht durch die anderen SPS-Bausteine abgedeckt wird.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]), zur Gruppenadressierung oder als Slave-Id ausgewertet werden soll.

dwAddrOption: Wird das SMI-Gerät per Slave-Id adressiert (*eAddrType* = *eSMIAddrTypeSlaveId*), so muss über diesen Eingang der Herstellercode [▶ 61] angegeben werden.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

eCommandType: Kommandoart: Schreiben/Lesen (siehe E_SMICommandType [▶ 51]). Dieser Parameter beeinflusst das Bit 5 vom Startbyte des SMI-Telegramms.

eResponseFormat: Antwortformat: Diagnose-Sonderformat/Standard (siehe E_SMIResponseFormat [▶ 53]). Dieser Parameter beeinflusst das Bit 6 vom Startbyte des SMI-Telegramms.

arrIdentificationBytes: Ein SMI-Telegramm kann aus bis zu 3 Blöcken bestehen. Jeder Block besitzt ein Kennungsbyte. Über dieses Array werden die drei Kennungsbytes definiert.

arrParameters: Ein SMI-Telegramm kann aus bis zu 3 Blöcken bestehen. Jeder Block besitzt bis zu vier Wertebytes. Über dieses Array werden die Wertebytes der einzelnen Blöcke definiert.

bSuppressResponseBuffer: Wird dieser Eingang auf TRUE gesetzt, so wird der interne Software-Puffer nicht mit den Antworten des Bausteins FB_KL6831KL6841Communication() [▶ 12] gefüllt.

dwOptions: reserviert für zukünftige Erweiterungen.

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

arrResponseData: Die empfangenen Daten von den SMI-Geräten.

byResponseDataLength: Die Länge der empfangenen Daten, in Bytes.

byResponseIdentificationByte: Das empfangene Kennungsbyte.

VAR_IN_OUT

```
stCommandBuffer : ST_SMICommandBuffer;
```

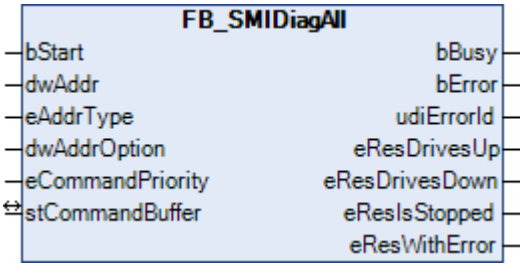
stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.2 Grundbefehle

4.1.2.1 FB_SMIDiagAll



Mit diesem Befehl kann ermittelt werden, in welche Richtung die Antriebe fahren, ob diese gestoppt sind oder ob ein Motorfehler vorliegt. Der Befehl kann auch an mehrere SMI-Slaves gesendet werden. Dadurch lassen sich die Zustände aller SMI-Slaves mit einem Befehl abfragen.

Das Ergebnis der Abfrage wird durch vier Ausgänge weitergegeben. Jeder dieser Ausgänge kann drei Zustände annehmen:

- Die Bedingung trifft auf mindestens einen Antrieb zu.
- Die Bedingung trifft auf keinen Antrieb zu.
- Die Bedingung konnte nicht ermittelt werden.

Weiter unten werden hierzu einige Beispiele erläutert.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]) oder zur Gruppenadressierung ausgewertet werden soll. Eine Adressierung per Slave-Id (*eAddrType* = *eSMIAddrTypeSlaveId*) ist nicht zulässig.

dwAddrOption: Reserviert für zukünftige Erweiterungen.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

eResDrivesUp: Mindestens ein Motor fährt hoch / Kein Motor fährt hoch / Der Wert ist undefiniert (siehe E_SMIDiagResDrivesUp [▶ 52]).

eResDrivesDown: Mindestens ein Motor fährt runter / Kein Motor fährt runter / Der Wert ist undefiniert (siehe E_SMIDiagResDrivesDown [▶ 52]).

eResIsStopped: Mindestens ein Motor ist gestoppt / Kein Motor ist gestoppt / Der Wert ist undefiniert (siehe E_SMIDiagResIsStopped [▶ 52]).

eResWithError: Mindestens ein Motor ist in Störung / Kein Motor ist in Störung / Der Wert ist undefiniert (siehe E_SMIDiagResWithError [▶ 52]).

VAR_IN_OUT

```
stCommandBuffer : ST_SMICommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Beispiele

Alle Antriebe sind gestoppt:

Ausgänge	Bedeutung
eResDrivesUp = eSMIDiagResNoMotorDrivesUp	Kein Antrieb fährt hoch.
eResDrivesDown = eSMIDiagResNoMotorDrivesDown	Kein Antrieb fährt runter.
eResIsStopped = eSMIDiagResAtLeastOneMotorIsStopped	Mindestens ein Antrieb ist gestoppt.
eResWithError = eSMIDiagResNoMotorWithError	Kein Antrieb mit Motorfehler.

Alle Antriebe fahren hoch:

Ausgänge	Bedeutung
eResDrivesUp = eSMIDiagResAtLeastOneMotorDrivesUp	Mindestens ein Antrieb fährt hoch.
eResDrivesDown = eSMIDiagResNoMotorDrivesDown	Kein Antrieb fährt runter.
eResIsStopped = eSMIDiagResNoMotorIsStopped	Kein Antrieb ist gestoppt.
eResWithError = eSMIDiagResNoMotorWithError	Kein Antrieb mit Motorfehler.

Ein Antrieb ist gestoppt und ein Antrieb fährt hoch:

Ausgänge	Bedeutung
eResDrivesUp = eSMIDiagResAtLeastOneMotorDrivesUp	Mindestens ein Antrieb fährt hoch.
eResDrivesDown = eSMIDiagResNoMotorDrivesDown	Kein Antrieb fährt runter.
eResIsStopped = eSMIDiagResAtLeastOneMotorIsStopped	Mindestens ein Antrieb ist gestoppt.
eResWithError = eSMIDiagResNoMotorWithError	Kein Antrieb mit Motorfehler.

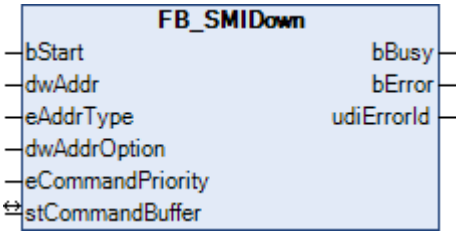
Ein Antrieb ist gestoppt, ein Antrieb fährt hoch und ein Antrieb fährt runter:

Ausgänge	Bedeutung
eResDrivesUp = eSMIDiagResAtLeastOneMotorDrivesUp	Mindestens ein Antrieb fährt hoch.
eResDrivesDown = eSMIDiagResAtLeastOneMotorDrivesDown	Mindestens ein Antrieb fährt runter.
eResIsStopped = eSMIDiagResAtLeastOneMotorIsStopped	Mindestens ein Antrieb ist gestoppt.
eResWithError = eSMIDiagResNoMotorWithError	Kein Antrieb mit Motorfehler.

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.2.2 FB_SMIDown



Motorlauf bis zur unteren Endlage.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]), zur Gruppenadressierung oder als Slave-Id ausgewertet werden soll.

dwAddrOption: Wird das SMI-Gerät per Slave-Id adressiert (*eAddrType* = *eSMIAddrTypeSlaveId*), so muss über diesen Eingang der Herstellercode [▶ 61] angegeben werden.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

VAR_IN_OUT

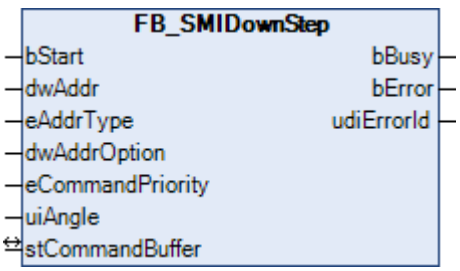
```
stCommandBuffer : ST_SMICommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.2.3 FB_SMIDownStep



Motorlauf nach unten um einen vorgegebenen Winkelgrad (0-510 Grad).

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]), zur Gruppenadressierung oder als Slave-Id ausgewertet werden soll.

dwAddrOption: Wird das SMI-Gerät per Slave-Id adressiert (*eAddrType* = *eSMIAddrTypeSlaveId*), so muss über diesen Eingang der Herstellercode [▶ 61] angegeben werden.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

uiAngle: Der vorgegebene Winkelgrad. Der Wertebereich ist 0...510 Grad. Der SMI-Standard reduziert die Genauigkeit auf eine Auflösung von 2 Grad.

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

VAR_IN_OUT

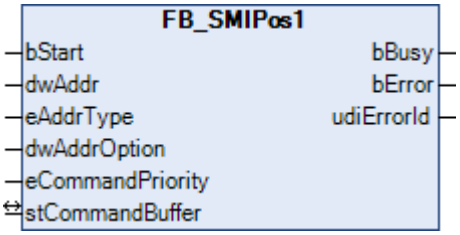
```
stCommandBuffer : ST_SMICommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.2.4 FB_SMIPos1



Fahrt zur motorseitig konfigurierten Fixposition *Pos1*. Auslesen und verändern von *Pos1* ist mit den Bausteinen FB_SMIPos1Read() [▶ 23] und FB_SMIPos1Write() [▶ 25] möglich.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]), zur Gruppenadressierung oder als Slave-Id ausgewertet werden soll.

dwAddrOption: Wird das SMI-Gerät per Slave-Id adressiert (*eAddrType* = *eSMIAddrTypeSlaveId*), so muss über diesen Eingang der Herstellercode [▶ 61] angegeben werden.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

VAR_IN_OUT

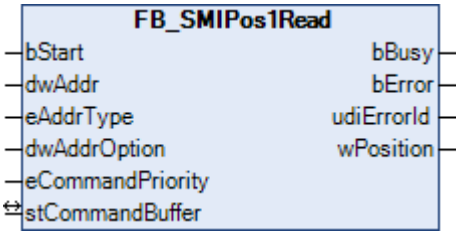
```
stCommandBuffer : ST_SMICommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.2.5 FB_SMIPos1Read



Lesen der motorseitig konfigurierten Fixposition *Pos1*. Mit dem Baustein FB_SMIPos1Write() [▶ 25] kann *Pos1* verändert werden.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]) oder zur Gruppenadressierung ausgewertet werden soll. Eine Adressierung per Slave-Id (*eAddrType* = *eSMIAddrTypeSlaveId*) ist nicht zulässig.

dwAddrOption: Reserviert für zukünftige Erweiterungen.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

wPosition: Die ausgelesene Fixposition *Pos1*. Hierbei entspricht der Wert 0 der oberen Endlage und der Wert 65535 (0xFFFF) der unteren Endlage.

VAR_IN_OUT

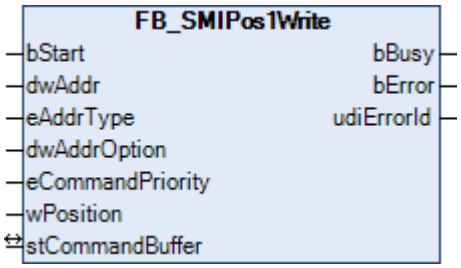
```
stCommandBuffer : ST_SMICommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.2.6 FB_SMIPos1Write



Schreiben der motorseitig konfigurierbaren Fixposition *Pos1*. Mit dem Baustein FB_SMIPos1Read() [▶ 23] kann *Pos1* ausgelesen werden.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]), zur Gruppenadressierung oder als Slave-Id ausgewertet werden soll.

dwAddrOption: Wird das SMI-Gerät per Slave-Id adressiert (*eAddrType* = *eSMIAddrTypeSlaveId*), so muss über diesen Eingang der Herstellercode [▶ 61] angegeben werden.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

wPosition: Die neue Fixposition *Pos1*. Hierbei entspricht der Wert 0 der oberen Endlage und der Wert 65535 (0xFFFF) der unteren Endlage.

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

VAR_IN_OUT

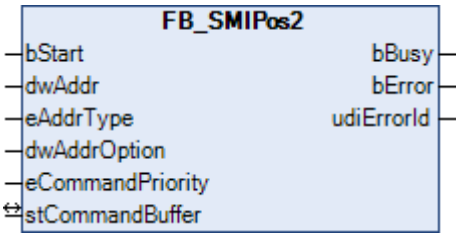
```
stCommandBuffer : ST_SMICommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.2.7 FB_SMIPos2



Fahrt zur motorseitig konfigurierten Fixposition *Pos2*. Auslesen und verändern von *Pos2* ist mit den Bausteinen FB_SMIPos2Read() [▶ 27] und FB_SMIPos2Write() [▶ 28] möglich.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]), zur Gruppenadressierung oder als Slave-Id ausgewertet werden soll.

dwAddrOption: Wird das SMI-Gerät per Slave-Id adressiert (*eAddrType* = *eSMIAddrTypeSlaveId*), so muss über diesen Eingang der Herstellercode [▶ 61] angegeben werden.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

VAR_IN_OUT

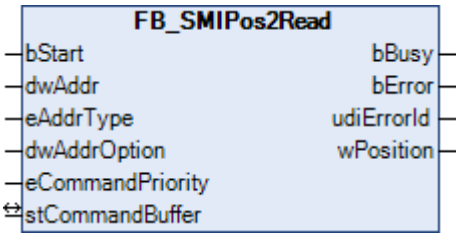
```
stCommandBuffer : ST_SMICommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.2.8 FB_SMIPos2Read



Lesen der motorseitig konfigurierten Fixposition *Pos2*. Mit dem Baustein FB_SMIPos2Write() [▶ 28] kann *Pos2* verändert werden.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]) oder zur Gruppenadressierung ausgewertet werden soll. Eine Adressierung per Slave-Id (*eAddrType* = *eSMIAddrTypeSlaveId*) ist nicht zulässig.

dwAddrOption: Reserviert für zukünftige Erweiterungen.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

wPosition: Die ausgelesene Fixposition *Pos2*. Hierbei entspricht der Wert 0 der oberen Endlage und der Wert 65535 (0xFFFF) der unteren Endlage.

VAR_IN_OUT

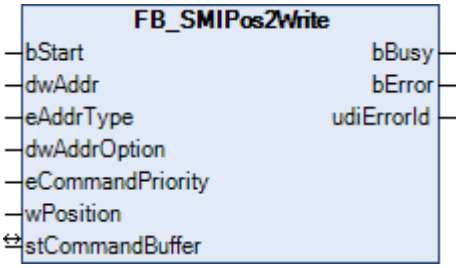
```
stCommandBuffer : ST_SMICommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.2.9 FB_SMIPos2Write



Schreiben der motorseitig konfigurierbaren Fixposition *Pos2*. Mit dem Baustein FB_SMIPos2Read() [▶ 27] kann *Pos2* ausgelesen werden.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]), zur Gruppenadressierung oder als Slave-Id ausgewertet werden soll.

dwAddrOption: Wird das SMI-Gerät per Slave-Id adressiert (*eAddrType* = *eSMIAddrTypeSlaveId*), so muss über diesen Eingang der Herstellercode [▶ 61] angegeben werden.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

wPosition: Die neue Fixposition *Pos2*. Hierbei entspricht der Wert 0 der oberen Endlage und der Wert 65535 (0xFFFF) der unteren Endlage.

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

VAR_IN_OUT

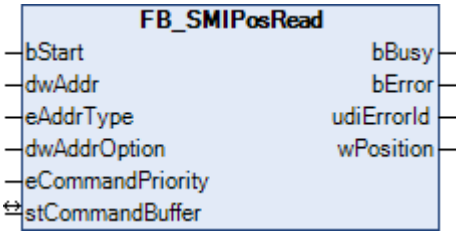
```
stCommandBuffer : ST_SMICommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.2.10 FB_SMIPosRead



Die aktuelle Position wird aus dem Antrieb ausgelesen.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]) oder zur Gruppenadressierung ausgewertet werden soll. Eine Adressierung per Slave-Id (*eAddrType* = *eSMIAddrTypeSlaveId*) ist nicht zulässig.

dwAddrOption: Reserviert für zukünftige Erweiterungen.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

wPosition: Die ausgelesene Position. Hierbei entspricht der Wert 0 der oberen Endlage und der Wert 65535 (0xFFFF) der unteren Endlage.

VAR_IN_OUT

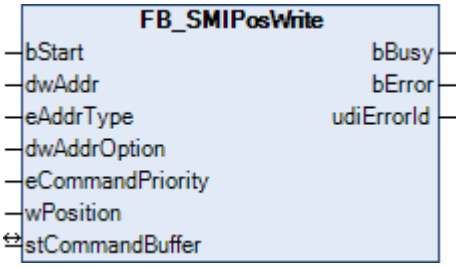
```
stCommandBuffer : ST_SMICommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.2.11 FB_SMIPosWrite



Der Antrieb wird auf die angegebene Position gefahren.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]), zur Gruppenadressierung oder als Slave-Id ausgewertet werden soll.

dwAddrOption: Wird das SMI-Gerät per Slave-Id adressiert (*eAddrType* = *eSMIAddrTypeSlaveId*), so muss über diesen Eingang der Herstellercode [▶ 61] angegeben werden.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

wPosition: Die neue Position. Hierbei entspricht der Wert 0 der oberen Endlage und der Wert 65535 (0xFFFF) der unteren Endlage.

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

VAR_IN_OUT

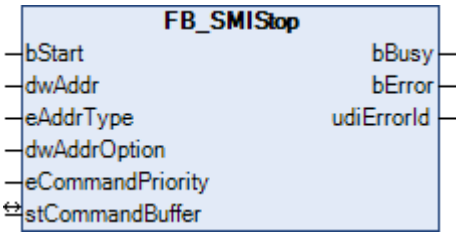
```
stCommandBuffer : ST_SMICommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.2.12 FB_SMIStop



Der Motorlauf wird gestoppt.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]), zur Gruppenadressierung oder als Slave-Id ausgewertet werden soll.

dwAddrOption: Wird das SMI-Gerät per Slave-Id adressiert (*eAddrType* = *eSMIAddrTypeSlaveId*), so muss über diesen Eingang der Herstellercode [▶ 61] angegeben werden.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

VAR_IN_OUT

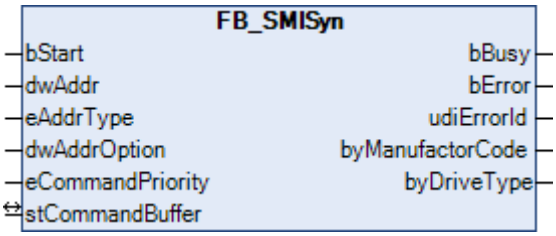
```
stCommandBuffer : ST_SMICommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.2.13 FB_SMISyn



Der Herstellercode und der Antriebstyp werden abgefragt.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]) oder zur Gruppenadressierung ausgewertet werden soll. Eine Adressierung per Slave-Id (*eAddrType* = *eSMIAddrTypeSlaveId*) ist nicht zulässig.

dwAddrOption: Reserviert für zukünftige Erweiterungen.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

byManufactorCode: Der Herstellercode [▶ 61] (1-14).

byDriveType: Der Typ des Antriebs (0-15). Die Bedeutung ist herstellerspezifisch.

VAR_IN_OUT

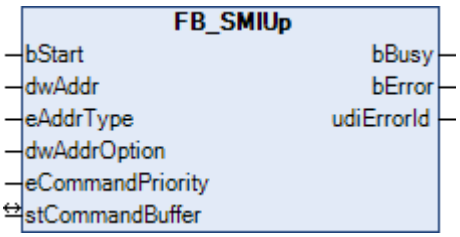
```
stCommandBuffer : ST_SMICommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.2.14 FB_SMIUp



Motorlauf bis zur oberen Endlage.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]), zur Gruppenadressierung oder als Slave-Id ausgewertet werden soll.

dwAddrOption: Wird das SMI-Gerät per Slave-Id adressiert (*eAddrType* = *eSMIAddrTypeSlaveId*), so muss über diesen Eingang der Herstellercode [▶ 61] angegeben werden.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

VAR_IN_OUT

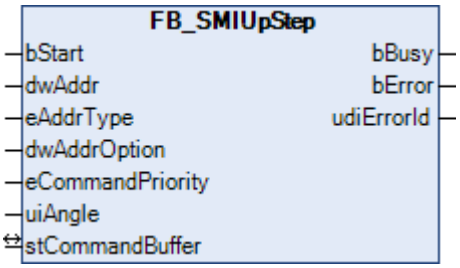
```
stCommandBuffer : ST_SMICommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.2.15 FB_SMIUpStep



Motorlauf nach oben um einen vorgegebenen Winkelgrad (0-510 Grad).

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]), zur Gruppenadressierung oder als Slave-Id ausgewertet werden soll.

dwAddrOption: Wird das SMI-Gerät per Slave-Id adressiert (*eAddrType* = *eSMIAddrTypeSlaveId*), so muss über diesen Eingang der Herstellercode [▶ 61] angegeben werden.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

uiAngle: Der vorgegebene Winkelgrad. Der Wertebereich ist 0...510 Grad. Der SMI-Standard reduziert die Genauigkeit auf eine Auflösung von 2 Grad.

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

VAR_IN_OUT

```
stCommandBuffer : ST_SMICommandBuffer;
```

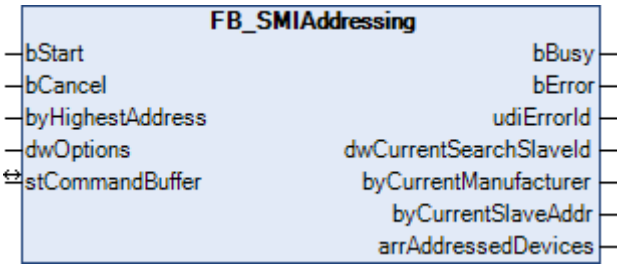
stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.3 Adressierungsbefehle

4.1.3.1 FB_SMIAddressing



Dieser Funktionsbaustein adressiert die angeschlossenen SMI-Geräte nach dem Zufallsprinzip. Der Anwender hat keinen Einfluss darauf, welches SMI-Gerät welche Adresse zugewiesen bekommt. Die Vergabe der Adressen erfolgt absteigend, beginnend bei der Adresse, die durch den Parameter *byHighestAddress* vorgegeben wird.

Durch eine positive Flanke an dem Eingang *bStart* wird der Baustein gestartet und der Ausgang *bBusy* geht auf TRUE. Der Baustein adressiert jetzt selbständig alle SMI-Geräte. Die Ausgangsvariable arrAddressedDevices gibt Auskunft darüber, welche SMI-Gerätes schon eine Adresse erhalten haben. Sind alle SMI-Geräte adressiert, so geht der Ausgang *bBusy* wieder auf FALSE. Die Adressierung kann vorzeitig durch eine positive Flanke am Eingang *bCancel* abgebrochen werden. Abhängig davon, wie viele SMI-Geräte angeschlossen sind, kann die Abarbeitung dieses Bausteines mehrere Minuten dauern.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

bCancel: Über eine positive Flanke an diesem Eingang wird der Baustein deaktiviert und die Suche abgebrochen.

byHighestAddress: Adresse, ab der absteigend die SMI-Geräte adressiert werden (0-15).

dwOptions: Reserviert für zukünftige Erweiterungen.

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

dwCurrentSearchSlaveId: Aktuelle Slave-Id, die im Such-Algorithmus verwendet wird.

byCurrentManufacturer: Aktueller Herstellercode [▶ 61], der im Such-Algorithmus verwendet wird.

byCurrentSlaveAddr: Aktueller Adresse, die im Such-Algorithmus verwendet wird.

arrAddressedDevices: Wird einem SMI-Gerät eine Adresse zugewiesen, so wird in dem Array das entsprechende Element auf TRUE gesetzt.

VAR_IN_OUT

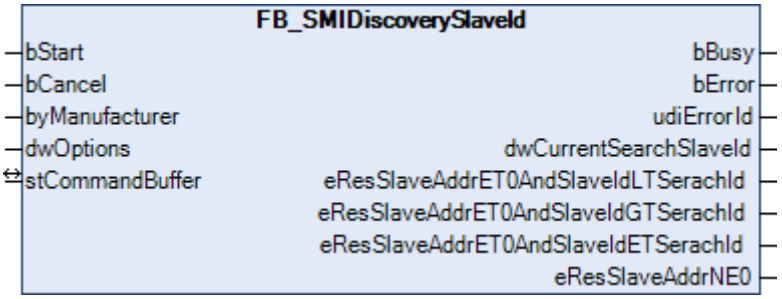
```
stCommandBuffer : ST_SMICommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.3.2 FB_SMIDiscoverySlaveId



Es wird der erste Antrieb gesucht, der dem vorgegebenen Herstellercode entspricht und bei dem die Adresse 0 ist. Dieser Baustein findet Verwendung bei der Adressierung von SMI-Geräten und wird im Baustein FB_SMIAddressing() [▶ 35] benutzt.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und die Suche gestartet.

bCancel: Über eine positive Flanke an diesem Eingang wird der Baustein deaktiviert und die Suche abgebrochen.

byManufacturer: Der vorgegebene Herstellercode [▶ 61] für die Suche nach dem SMI-Gerät. Einige SMI-Geräte erlauben nicht den Herstellercode 0.

dwOptions: Reserviert für zukünftige Erweiterungen.

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

dwCurrentSearchSlaveId: Sobald der Bausteine seine Ausführung beendet hat (*bBusy* wechselt von TRUE auf FALSE) zeigt dieser Ausgang die Slave-Id des gefundenen SMI-Gerätes an.

eResSlaveAddrET0AndSlaveIdLTSerachId: Bei mindestens einem Motor / Bei keinem Motor ist die Adresse gleich 0 und die Slave-Id ist kleiner als die gesuchte Slave-Id (*dwSlave-Id*) / Der Wert ist undefiniert (siehe E_SMICompResSlaveAddrET0AndSlaveIdLTSearchId [▶ 51]).

eResSlaveAddrET0AndSlaveIdGTSerachId: Bei mindestens einem Motor / Bei keinem Motor ist die Adresse gleich 0 und die Slave-Id ist größer als die gesuchte Slave-Id (*dwSlave-Id*) / Der Wert ist undefiniert (siehe E_SMICompResSlaveAddrET0AndSlaveIdGTSearchId [▶ 51]).

eResSlaveAddrET0AndSlaveIdETSerachId: Bei mindestens einem Motor / Bei keinem Motor ist die Adresse gleich 0 und die Slave-Id ist ebenfalls gleich der gesuchten Slave-Id (*dwSlave-Id*) / Der Wert ist undefiniert (siehe E_SMICompResSlaveAddrET0AndSlaveIdETSearchId [▶ 51]).

eResSlaveAddrNE0: Bei mindestens einem Motor / Bei keinem Motor ist die Adresse ungleich 0 / Der Wert ist undefiniert (siehe E_SMICompResSlaveAddrNE0 [▶ 51]).

VAR_IN_OUT

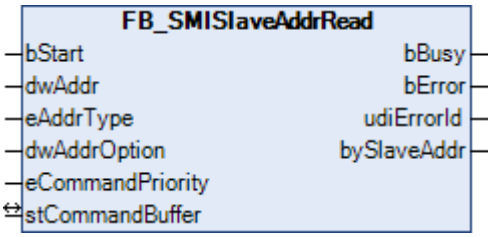
```
stCommandBuffer : ST_SMICommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.3.3 FB_SMISlaveAddrRead



Die Adresse (0-15) eines Antriebes wird ausgelesen.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]), zur Gruppenadressierung oder als Slave-Id ausgewertet werden soll.

dwAddrOption: Wird das SMI-Gerät per Slave-Id adressiert (*eAddrType* = *eSMIAddrTypeSlaveId*), so muss über diesen Eingang der Herstellercode [▶ 61] angegeben werden.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

bySlaveAddr: Die ausgelesene Slave-Adresse (0-15).

VAR_IN_OUT

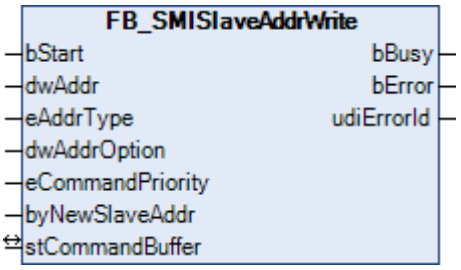
```
stCommandBuffer : ST_SMICommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.3.4 FB_SMISlaveAddrWrite



Die Adresse (0-15) einer oder mehrerer Antriebe schreiben.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]), zur Gruppenadressierung oder als Slave-Id ausgewertet werden soll.

dwAddrOption: Wird das SMI-Gerät per Slave-Id adressiert (*eAddrType* = *eSMIAddrTypeSlaveId*), so muss über diesen Eingang der Herstellercode [▶ 61] angegeben werden.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

byNewSlaveAddr: Die neue Slave-Adresse (0-15).

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

VAR_IN_OUT

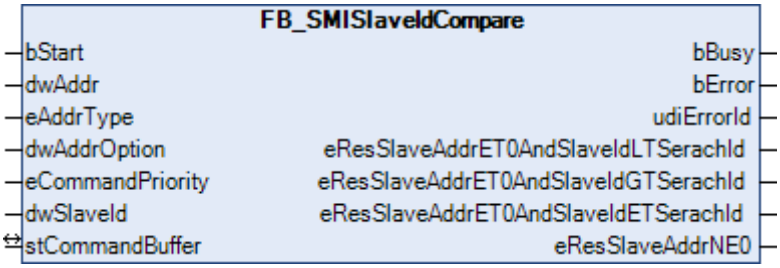
```
stCommandBuffer : ST_SMICommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.3.5 FB_SMISlaveIdCompare



Eine vorgegebene Slave-Id (32 Bit Key-Id) wird mit der motorseitig definierten Slave-Id (32 Bit Key-Id) eines oder mehrerer Antriebe verglichen. Der Befehl kann auch an mehrere SMI-Slaves gesendet werden.

Das Ergebnis der Abfrage wird durch vier Ausgänge weitergegeben. Jeder dieser Ausgänge kann drei Zustände annehmen:

- Die Bedingung trifft auf mindestens einen Antrieb zu.
- Die Bedingung trifft auf keinen Antrieb zu.
- Die Bedingung konnte nicht ermittelt werden.

Weiter unten werden hierzu einige Beispiele erläutert.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]), zur Gruppenadressierung oder als Slave-Id ausgewertet werden soll.

dwAddrOption: Wird das SMI-Gerät per Slave-Id adressiert (*eAddrType* = *eSMIAddrTypeSlaveId*), so muss über diesen Eingang der Herstellercode [▶ 61] angegeben werden.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

dwSlaveId: Die Slave-Id, mit der die motorseitige Slave-Id verglichen wird.

VAR_OUTPUT

```
bBusy                                : BOOL;
bError                               : BOOL;
udiErrorId                           : UDINT;
eResSlaveAddrET0AndSlaveIdLTSerachId : E_SMICompResSlaveAddrET0AndSlaveIdLTSearchId;
```

```
eResSlaveAddrET0AndSlaveIdGTSerachId : E_SMICompResSlaveAddrET0AndSlaveIdGTSearchId;
eResSlaveAddrET0AndSlaveIdETSerachId : E_SMICompResSlaveAddrET0AndSlaveIdETSearchId;
eResSlaveAddrNE0                    : E_SMICompResSlaveAddrNE0;
```

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

eResSlaveAddrET0AndSlaveIdLTSerachId: Bei mindestens einem Motor / Bei keinem Motor ist die Adresse gleich 0 und die Slave-Id ist kleiner als die gesuchte Slave-Id (*dwSlave-Id*) / Der Wert ist undefiniert (siehe E_SMICompResSlaveAddrET0AndSlaveIdLTSearchId [▶ 51]).

eResSlaveAddrET0AndSlaveIdGTSerachId: Bei mindestens einem Motor / Bei keinem Motor ist die Adresse gleich 0 und die Slave-Id ist größer als die gesuchte Slave-Id (*dwSlave-Id*) / Der Wert ist undefiniert (siehe E_SMICompResSlaveAddrET0AndSlaveIdGTSearchId [▶ 51]).

eResSlaveAddrET0AndSlaveIdETSerachId: Bei mindestens einem Motor / Bei keinem Motor ist die Adresse gleich 0 und die Slave-Id ist ebenfalls gleich der gesuchten Slave-Id (*dwSlave-Id*) / Der Wert ist undefiniert (siehe E_SMICompResSlaveAddrET0AndSlaveIdETSearchId [▶ 51]).

eResSlaveAddrNE0: Bei mindestens einem Motor / Bei keinem Motor ist die Adresse ungleich 0 / Der Wert ist undefiniert (siehe E_SMICompResSlaveAddrNE0 [▶ 51]).

VAR_IN_OUT

```
stCommandBuffer : ST_SMICommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Beispiele

Die folgenden Tabellen zeigen die Ergebnisse des Bausteins bei unterschiedlichen Ausgangssituationen. In allen Fällen sind zwei SMI-Geräte an einer SMI-Klemme angeschlossen und beide Adressen sind größer 0.

Die gesuchte Slave-Id (dwSlaveId) liegt zwischen den Slave-Ids der beiden Antriebe:

Ausgänge	Bedeutung
eResSlaveAddrET0AndSlaveIdLTSerachId = eSMIDiagResAtLeastOneSlaveAddrET0AndSlaveIdLTSearchId	Bei mindestens einem Motor ist die Slave-Adresse gleich 0 und die Slave-Id ist kleiner der gesuchten Slave-Id.
eResSlaveAddrET0AndSlaveIdGTSerachId = eSMIDiagResAtLeastOneSlaveAddrET0AndSlaveIdGTSearchId	Bei mindestens einem Motor ist die Slave-Adresse gleich 0 und die Slave-Id ist größer der gesuchten Slave-Id.
eResSlaveAddrET0AndSlaveIdETSerachId = eSMIDiagResNoSlaveAddrET0AndSlaveIdETSearchId	Bei keinem Motor ist die Slave-Adresse gleich 0 und die Slave-Id gleich der gesuchten Slave-Id.
eResSlaveAddrNE0 = eSMIDiagResNoSlaveAddrNE0	Bei keinem Motor ist die Slave-Adresse ungleich 0.

Die gesuchte Slave-Id (dwSlaveId) ist größer als die Slave-Ids der beiden Antriebe:

Ausgänge	Bedeutung
eResSlaveAddrET0AndSlaveIdLTSerachId = eSMIDiagResAtLeastOneSlaveAddrET0AndSlaveIdLTSearchId	Bei mindestens einem Motor ist die Slave-Adresse gleich 0 und die Slave-Id ist kleiner der gesuchten Slave-Id.
eResSlaveAddrET0AndSlaveIdGTSerachId = eSMIDiagResNoSlaveAddrET0AndSlaveIdGTSearchId	Bei keinem Motor ist die Slave-Adresse gleich 0 und die Slave-Id größer der gesuchten Slave-Id.
eResSlaveAddrET0AndSlaveIdETSerachId = eSMIDiagResNoSlaveAddrET0AndSlaveIdLTSearchId	Bei keinem Motor ist die Slave-Adresse gleich 0 und die Slave-Id kleiner der gesuchten Slave-Id.
eResSlaveAddrNE0 = eSMIDiagResNoSlaveAddrNE0	Bei keinem Motor ist die Slave-Adresse ungleich 0.

Die gesuchte Slave-Id (dwSlaveId) ist kleiner als die Slave-Ids der beiden Antriebe:

Ausgänge	Bedeutung
eResSlaveAddrET0AndSlaveIdLTSerachId = eSMIDiagResNoSlaveAddrET0AndSlaveIdLTSearchId	Bei keinem Motor ist die Slave-Adresse gleich 0 und die Slave-Id kleiner der gesuchten Slave-Id.
eResSlaveAddrET0AndSlaveIdGTSerachId = eSMIDiagResAtLeastOneSlaveAddrET0AndSlaveIdGTSearchId	Bei mindestens einem Motor ist die Slave-Adresse gleich 0 und die Slave-Id größer der gesuchten Slave-Id.
eResSlaveAddrET0AndSlaveIdETSerachId = eSMIDiagResNoSlaveAddrET0AndSlaveIdETSearchId	Bei keinem Motor ist die Slave-Adresse gleich 0 und die Slave-Id gleich der gesuchten Slave-Id.
eResSlaveAddrNE0 = eSMIDiagResNoSlaveAddrNE0	Bei keinem Motor ist die Slave-Adresse ungleich 0.

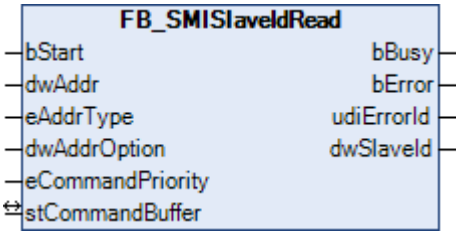
Die gesuchte Slave-Id (dwSlaveId) ist gleich der Slave-Id eines Antriebes:

Ausgänge	Bedeutung
eResSlaveAddrET0AndSlaveIdLTSerachId = eSMIDiagResNoSlaveAddrET0AndSlaveIdLTSearchId	Bei keinem Motor ist die Slave-Adresse gleich 0 und die Slave-Id kleiner der gesuchten Slave-Id.
eResSlaveAddrET0AndSlaveIdGTSerachId = eSMIDiagResAtLeastOneSlaveAddrET0AndSlaveIdGTSearchId	Bei mindestens einem Motor ist die Slave-Adresse gleich 0 und die Slave-Id größer der gesuchten Slave-Id.
eResSlaveAddrET0AndSlaveIdETSerachId = eSMIDiagResAtLeastOneSlaveAddrET0AndSlaveIdETSearchId	Bei mindestens einem Motor ist die Slave-Adresse gleich 0 und die Slave-Id gleich der gesuchten Slave-Id.
eResSlaveAddrNE0 = eSMIDiagResNoSlaveAddrNE0	Bei keinem Motor ist die Slave-Adresse ungleich 0.

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.3.6 FB_SMISlaveIdRead



Aus einem Antrieb wird die Slave-Id (32 Bit Key-Id) ausgelesen.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]), zur Gruppenadressierung oder als Slave-Id ausgewertet werden soll.

dwAddrOption: Wird das SMI-Gerät per Slave-Id adressiert (*eAddrType* = *eSMIAddrTypeSlaveId*), so muss über diesen Eingang der Herstellercode [▶ 61] angegeben werden.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

dwSlaveId: Die ausgelesene Slave-Id.

VAR_IN_OUT

```
stCommandBuffer : ST_SMICommandBuffer;
```

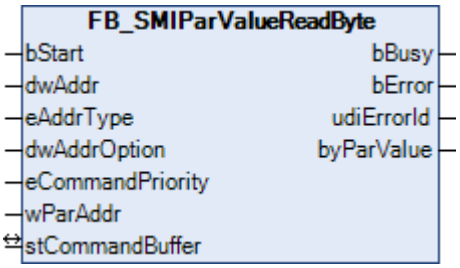
stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.4 Systembefehle

4.1.4.1 FB_SMIParValueReadByte



Lesen eines motorseitig gespeicherten Byte-Parameters (1-Byte). Die Bedeutung der einzelnen Parameter ist herstellerabhängig.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]) oder zur Gruppenadressierung ausgewertet werden soll. Eine Adressierung per Slave-Id (*eAddrType* = *eSMIAddrTypeSlaveId*) ist nicht zulässig.

dwAddrOption: Reserviert für zukünftige Erweiterungen.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

wParAddr: Adresse des Parameters (0-4095) der gelesen werden soll.

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

byParValue: Der ausgelesene Byte-Parameter.

VAR_IN_OUT

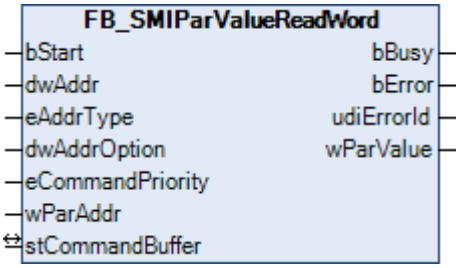
```
stCommandBuffer : ST_SMICommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.4.2 FB_SMIParValueReadWord



Lesen eines motorseitig gespeicherten Word-Parameters (2-Bytes). Die Bedeutung der einzelnen Parameter ist herstellerabhängig.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]) oder zur Gruppenadressierung ausgewertet werden soll. Eine Adressierung per Slave-Id (*eAddrType* = *eSMIAddrTypeSlaveId*) ist nicht zulässig.

dwAddrOption: Reserviert für zukünftige Erweiterungen.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

wParAddr: Adresse des Parameters (0-4095) der gelesen werden soll.

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

wParValue: Der ausgelesene Word-Parameter.

VAR_IN_OUT

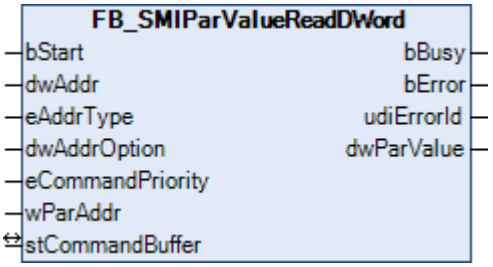
```
stCommandBuffer : ST_SMICommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.4.3 FB_SMIParValueReadDWord



Lesen eines motorseitig gespeicherten DWord-Parameters (4-Bytes). Die Bedeutung der einzelnen Parameter ist herstellerabhängig.

VAR_INPUT

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

bStart: Über eine positive Flanke an diesem Eingang wird der Baustein aktiviert und der Befehl versendet.

dwAddr: Herstellercode [▶ 61] (0-15), Adresse eines Teilnehmers (0-15), Bitfeld (16 Bit) für die Gruppenadressierung oder Slave-Id (32 Bit Key-Id). Wird ein Sammelruf (Broadcast) versendet, so hat dieser Eingang keine Bedeutung.

eAddrType: Legt fest, ob der Eingang *dwAddr* als Herstellercode [▶ 61], Adresse eines Teilnehmers (siehe E_SMIAddrType [▶ 50]) oder zur Gruppenadressierung ausgewertet werden soll. Eine Adressierung per Slave-Id (*eAddrType* = *eSMIAddrTypeSlaveId*) ist nicht zulässig.

dwAddrOption: Reserviert für zukünftige Erweiterungen.

eCommandPriority: Priorität (hoch, mittel oder niedrig), mit der der Befehl von der SPS-Bibliothek abgearbeitet wird (siehe E_SMICommandPriority [▶ 50]).

wParAddr: Adresse des Parameters (0-4095) der gelesen werden soll.

VAR_OUTPUT

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

bBusy: Bei der Aktivierung des Bausteins wird der Ausgang gesetzt und bleibt so lange aktiv bis der Befehl abgearbeitet wurde.

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wird der Ausgang wieder auf FALSE zurückgesetzt.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Wird durch das erneute Aktivieren des Bausteins über den Eingang *bStart* wieder auf 0 zurückgesetzt (siehe Fehlercodes [▶ 48]).

dwParValue: Der ausgelesene DWord-Parameter.

VAR_IN_OUT

```
stCommandBuffer : ST_SMICommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_KL6831KL6841Communication() [▶ 12]-Baustein (siehe ST_SMICommandBuffer [▶ 53]).

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.1.5 Fehlercodes

Wert (hex)	Wert (dez)	Beschreibung
0x0000	0	Kein Fehler.
0x8001	32769	Keine Rückantwort vom SMI-Antrieb.
0x8002	32770	Keine Klemmenrückmeldung für die Sendedaten von der SMI-Klemme.
0x8003	32771	Klemme hat ein Telegrammfehler erkannt (StatusWord.6 = true). Diese Meldung muss durch den Eingang bResetDataFrameError vom FB_KL6831KL6841Communication() quittiert werden.
0x8004	32772	NACK vom Antrieb empfangen.
0x8005	32773	Ungültige Rückmeldung vom Antrieb empfangen.
0x8006	32774	Überlauf Kommunikationspuffer.
0x8007	32775	Keine Antwort vom Kommunikationsbaustein.
0x8008	32776	Die Konstante SMI_COMMAND_BUFFER_ENTRIES liegt außerhalb des gültigen Bereichs (2-250).
0x8009	32777	Das empfangene Id Byte ist nicht korrekt.
0x800A	32778	Die empfangene Datenlänge ist nicht korrekt.
0x800B	32779	24V Versorgungsspannung an der KL6831/KL6841 fehlt (StatusWord.2 = false).
0x800C	32780	Prozessabbild wurde durch die Eingänge Switch1 oder Switch2 der Klemme deaktiviert (StatusWord.5 = true). Diese Meldung muss durch den Eingang bResetInactiveProcessImage vom FB_KL6831KL6841Communication() quittiert werden.
0x800D	32781	Klemme hat einen Checksummenfehler erkannt (StatusWord.8 = true). Sobald ein Telegramm erfolgreich übertragen wurde, wird diese Meldung wieder zurückgesetzt.
0x800E	32782	Der SMI-Befehl unterstützt nicht die Adressierung per Slave-Id (eAddrType = eSMIAddrTypeSlaveId).
0x800F	32783	Parameter wAddr (Bitfeld für Gruppenadressierung) ist außerhalb des gültigen Bereichs (0-65535).
0x8010	32784	Parameter wAddr (Adresse) ist außerhalb des gültigen Bereichs (0-15).
0x8011	32785	Parameter eCommandPriority ist ungültig.
0x8012	32786	Parameter eCommandType ist ungültig.
0x8013	32787	Parameter uiAngle ist außerhalb des gültigen Bereichs (0-510).
0x8014	32788	Parameter wParAddr ist außerhalb des gültigen Bereichs (0-4095).
0x8015	32789	Parameter eAddrType ist ungültig.
0x8016	32790	Parameter eResponseFormat ist ungültig.
0x8017	32791	Parameter wAddr (Herstellercode) ist außerhalb des gültigen Bereichs (0-15).
0x8018	32792	Das Kommando unterstützt nur Einzeladressierung.
0x8019	32793	Parameter-Option wAddrOption (Herstellercode) ist außerhalb des gültigen Bereichs (0-15).
0x801A	32794	Interner Fehler im Baustein FB_SMIDiscoverySlaveId aufgetreten.
0x801B	32795	Es wurden keine Geräte gefunden.
0x801C	32796	Alle 16 Adressen wurden schon vergeben. Evtl. sind mehr als 16 Geräte am SMI-Bus angeschlossen.
0x801D	32797	Ungültige Diagnoseantwort erhalten (weder NACK noch ACK).
0x801E	32798	Parameter byHighestAddress (höchste Adresse) ist außerhalb des gültigen Bereichs (0-15).

Wert (hex)	Wert (dez)	Beschreibung
0x801F	32799	Zeitüberschreitung bei der internen Adressierung. Die Klemme hat, nach dem Starten der internen Adressierung, keine Antwort zurückgeliefert.
0x8020	32800	Die interne Adressierung ist dreimal fehlgeschlagen.

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
```

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.32	Tc2_SMI ab 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
```

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.2.1.3 E_SMICommandPriority

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

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.2.1.4 E_SMICommandType

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

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.2.1.5 E_SMICompResSlaveAddrET0AndSlaveIdETSearchId

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

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.2.1.6 E_SMICompResSlaveAddrET0AndSlaveIdGTSearchId

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

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.2.1.7 E_SMICompResSlaveAddrET0AndSlaveIdLTSearchId

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

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.2.1.8 E_SMICompResSlaveAddrNE0

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

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.2.1.9 E_SMIDiagResDrivesDown

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

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.2.1.10 E_SMIDiagResDrivesUp

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

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.2.1.11 E_SMIDiagResIsStopped

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

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.2.1.12 E_SMIDiagResWithError

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

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.2.1.13 E_SMIResponseFormat

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

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 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
```

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 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
```

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 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
```

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 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
```

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 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
```

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 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
```

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 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
```

Voraussetzungen

Entwicklungsumgebung	erforderliche TC3 SPS-Bibliothek
TwinCAT ab v3.1.4020.14	Tc2_SMI ab 3.3.5.0

4.3 Integration in TwinCAT

4.3.1 KL6831 mit CX5120

Dieses Beispiel beschreibt, wie ein einfaches SPS-Programm für SMI in TwinCAT geschrieben werden kann und wie es mit der Hardware verknüpft wird.

Ein Motor wird stufenweise per Taster angesteuert. Ein Taster sendet den Auf-Befehl, ein anderer Taster den Ab-Befehl.

Beispiel: https://infosys.beckhoff.com/content/1031/tcplclib_tc2_smi/Resources/6012679435.zip

Hardware

Einrichtung der Komponenten

- 1x Embedded-PC CX5120
- 1x Digitale 4-Kanal-Eingangsklemme KL1104 (für die Auf-, Abfahr- und Reset-Funktion)
- 1x SMI-Klemme KL6831
- 1x Endklemme KL9010

Richten Sie die Hardware und die SMI-Komponenten, wie in den entsprechenden Dokumentationen beschrieben, ein.

Das Beispiel geht davon aus, dass ein Reset-Taster auf den ersten, ein Auf-Taster auf den zweiten und ein Ab-Taster auf den dritten Eingang der KL1104 gelegt wurde. An der SMI-Teilnehmeradresse 1 befindet sich ein Antrieb.

Software

Erstellung des SPS-Programms

Erstellen Sie ein neues „TwinCAT XAE Project“ und legen Sie ein „Standard PLC Project“ an.

Fügen Sie im SPS-Projekt unter „References“ die Bibliothek Tc2_SMI hinzu.

Erzeugen Sie die folgenden globalen Variablen:

```
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
```

bReset: Eingangsvariable für den Reset-Taster.

bUp: Eingangsvariable für den Auf-Taster.

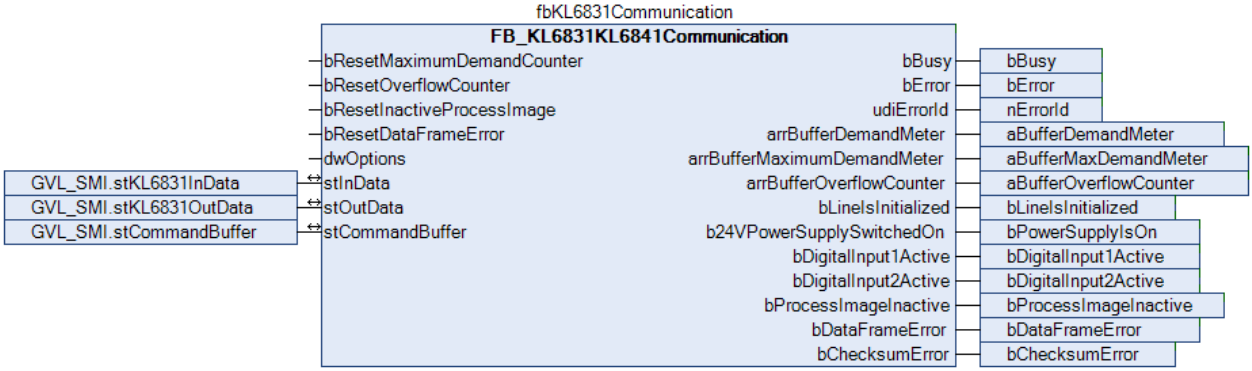
bDown: Eingangsvariable für den Ab-Taster.

stKL6831InData: Eingangsvariable für die SMI-Klemme (siehe ST_KL6831KL6841InData [▶ 53]).

stKL6831OutData: Ausgangsvariable für die SMI-Klemme (siehe ST_KL6831KL6841OutData [▶ 53]).

stCommandBuffer: Wird für die Kommunikation mit SMI benötigt (siehe ST_SMICommandBuffer [▶ 53]).

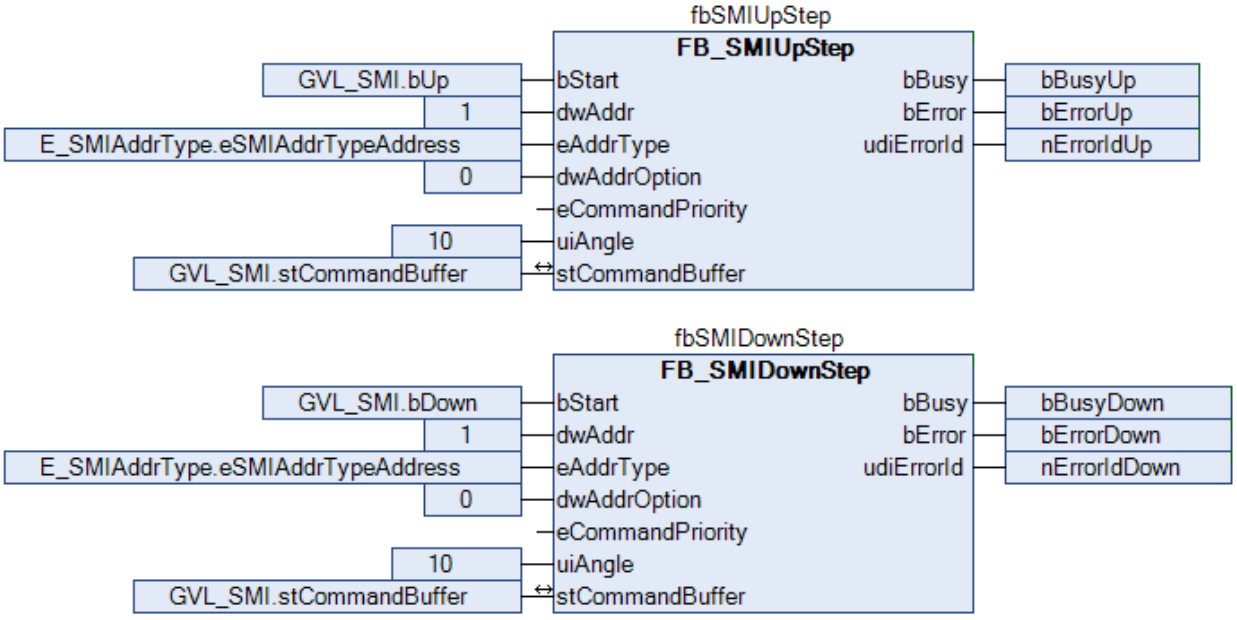
Legen Sie ein Programm (CFC) für die Hintergrundkommunikation mit SMI an. In dem Programm wird der Baustein FB_KL6831KL6841Communication [▶ 12] aufgerufen. Achten Sie beim Kommunikationsbaustein darauf, die Strukturen *stKL6381InData* und *stKL6831OutData* und *stCommandBuffer* zu verknüpfen.



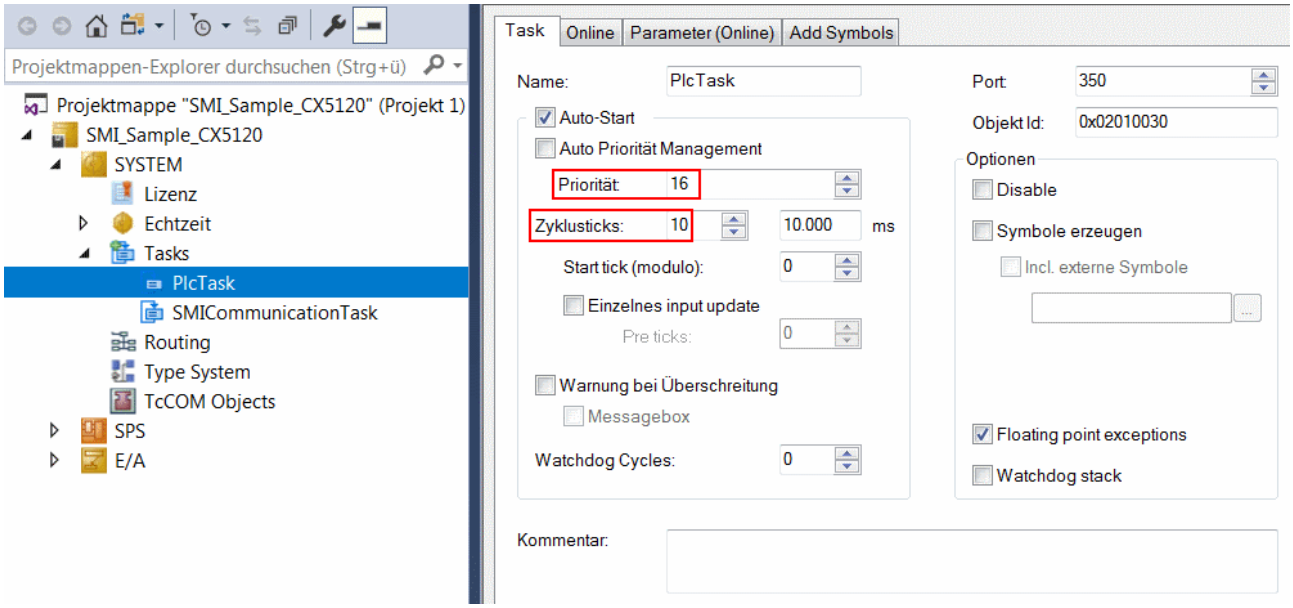
Legen Sie ein MAIN-Programm (CFC) an, in dem die Bausteine FB_SMIUpStep [▶ 34] und FB_SMIDownStep [▶ 21] aufgerufen werden.

Der Eingang *bStart* des Bausteins zum Senden des Auf-Befehls wird mit der globalen Variable *bUp* verknüpft.

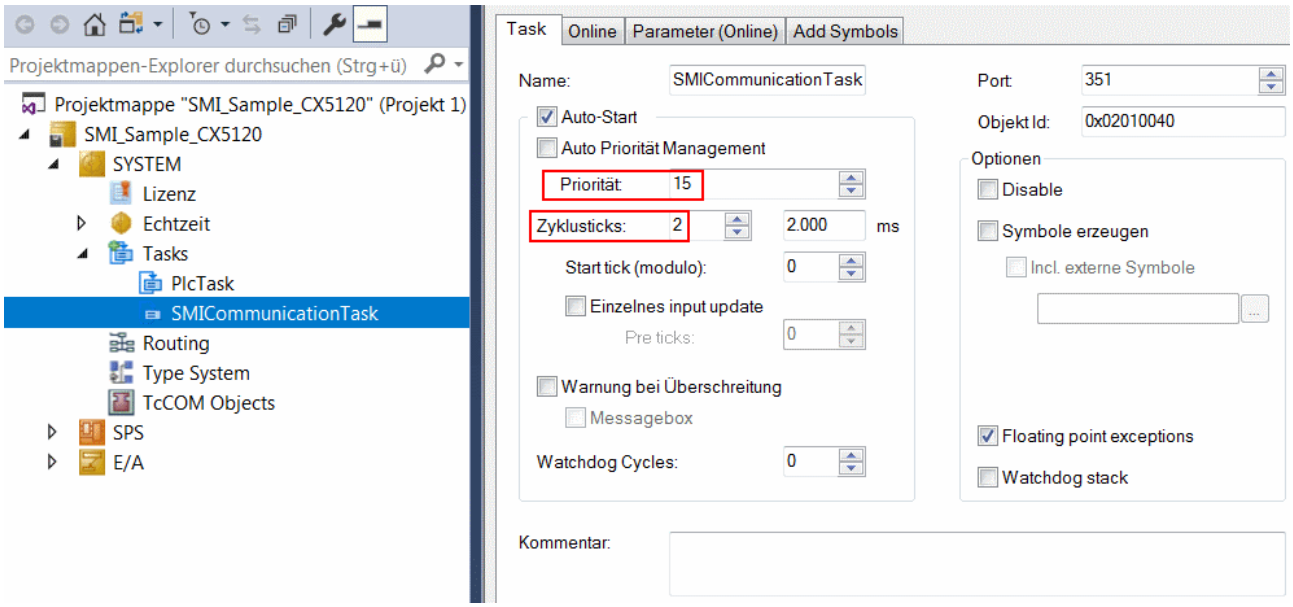
Der Eingang *bStart* des Bausteins zum Senden des Ab-Befehls wird mit der globalen Variable *bDown* verknüpft.



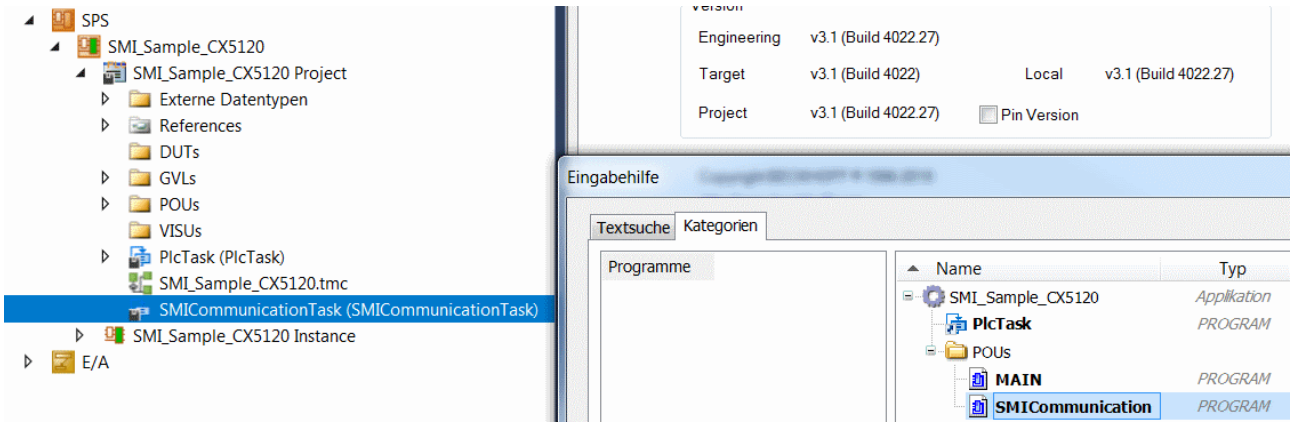
Navigieren Sie in den Bereich der Taskkonfiguration und konfigurieren die PlcTask. Exemplarisch erhält die Task die Priorität 16 und eine Zykluszeit von 10 ms.



Legen Sie eine weitere Task für die Hintergrundkommunikation an. Geben Sie dieser Task eine höhere Priorität (kleinere Zahl) und eine niedrigere Intervall-Zeit als der PlcTask.

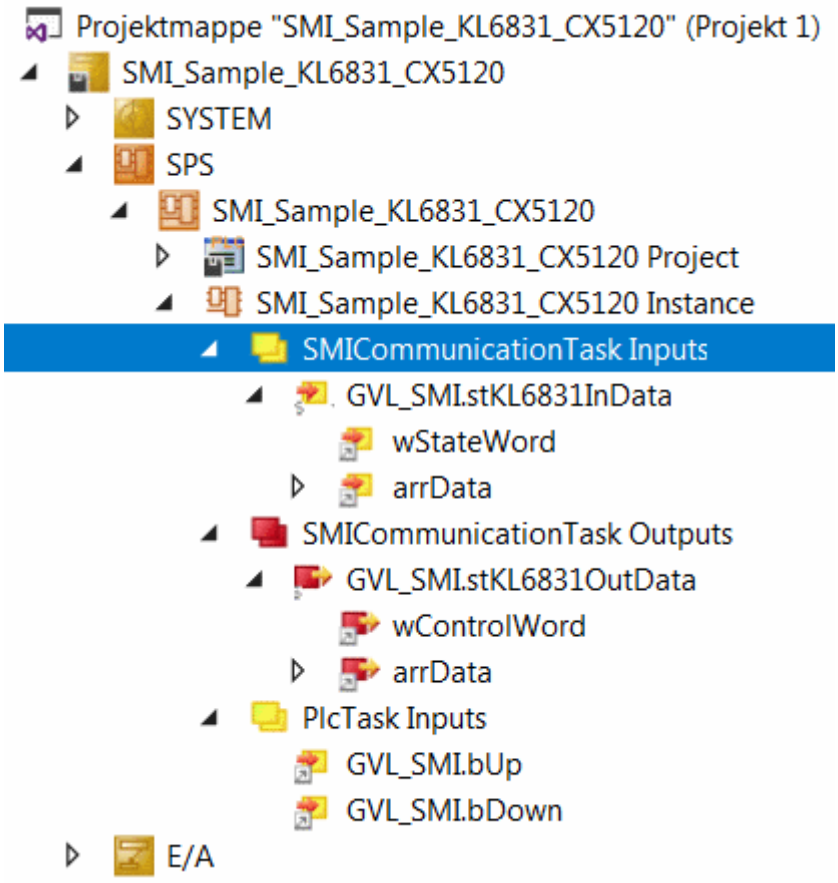


Fügen Sie dieser Task das Programm für die Kommunikation zu. Genauere Information zur Taskkonfiguration finden Sie in der Beschreibung des Bausteins.



E/A Konfiguration

Wählen Sie als Zielsystem den CX und lassen Sie nach dessen Hardware suchen. Im Bereich der SPS, in der Instanz des Projekts sehen Sie, dass die Ein- und Ausgangsvariablen den entsprechenden Tasks zugeordnet sind.



Verknüpfen Sie die globalen Variablen des SPS-Programms nun mit den Ein- und Ausgängen der Busklemmen. Erstellen Sie die Projektmappe und aktivieren Sie die Konfiguration.

Durch Betätigen des ersten Tasters wird die Lampe mit dem maximalen Helligkeitswert eingeschaltet. Mit dem zweiten Taster kann sie wieder ausgeschaltet werden.

Mit dem Reset-Taster können Sie die Eingänge in *arrBufferMaximumDemandMeter* und *arrBufferOverflowCounter* zurücksetzen.

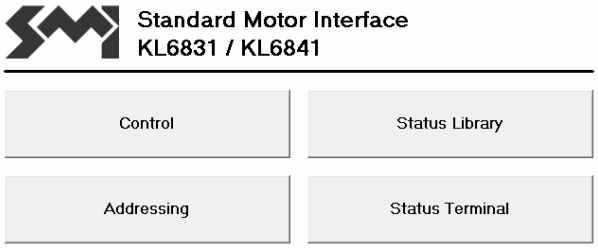
5 Anhang

5.1 Beispiel: Konfigurieren von SMI-Geräten

TwinCAT 3 Projekt: https://infosys.beckhoff.com/content/1031/tcplclib_tc2_smi/Resources/688934411.zip

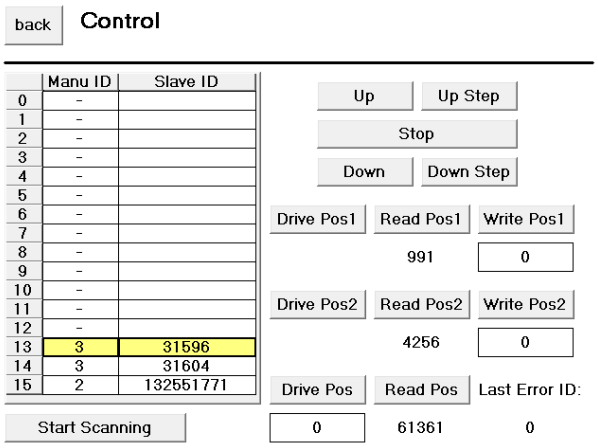
Mit dem Beispiel ist es möglich SMI-Geräte zu adressieren oder eine vorhandene Installation zu erweitern. Des Weiteren können die Dialoge zur Diagnose und Fehleranalyse genutzt werden. Insgesamt stehen 5 Dialoge zur Verfügung.

Start



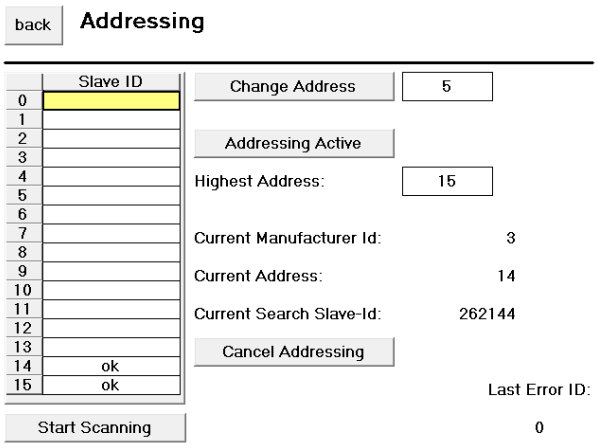
Unter *SMI_Start* befindet sich das Hauptmenü, über das die vier Untermenüs erreichbar sind.

Control



Mit der Schaltfläche *Start Scanning* wird an der SMI-Linie nach adressierten SMI-Geräten gesucht. Alle gefundenen SMI-Geräte werden in der linken Liste durch den Herstellercode [▶ 61] und der Slave-Id angezeigt. Durch Anklicken eines Eintrags wird dieser selektiert. Die anderen Schaltflächen beziehen sich immer auf den selektierten Eintrag. Hierdurch können alle wichtigen SMI-Kommandos an jedes adressierte SMI-Gerät versendet werden. Sollte bei einem SMI-Kommando ein Fehler erkannt werden, so wird dieser in der rechten unteren Ecke durch den Fehlercode [▶ 48] angezeigt.

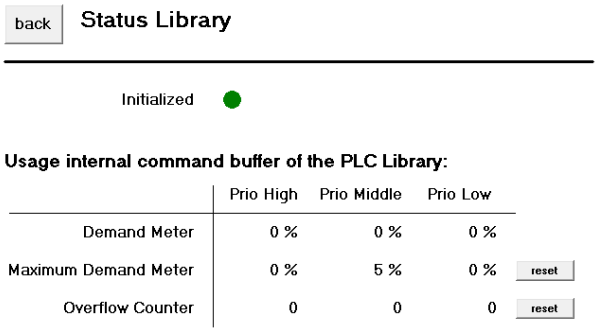
Addressing



Jedem SMI-Gerät kann eine Adresse von 0 bis 15 zugewiesen werden. Über diese Adresse kann jedes SMI-Gerät angesprochen werden. Es gibt zwar noch andere Methoden SMI-Geräte anzusprechen (siehe Geräte Adressierung [▶ 9]), jedoch ist für die Gruppenadressierung eine eindeutige Adresse notwendig. Von daher empfiehlt es sich, jedem SMI-Gerät eine Adresse zuzuweisen. Die Schaltfläche *Start Scanning* dient zum Suchen aller adressierten SMI-Geräte. Soll eine Adresse geändert werden, so muss das entsprechende SMI-Gerät in der Liste selektiert werden. Über das Eingabefeld rechts neben der Schaltfläche *Change Address* kann die gewünschte Adresse vorgegeben werden, die durch das Betätigen der Schaltfläche übernommen wird.

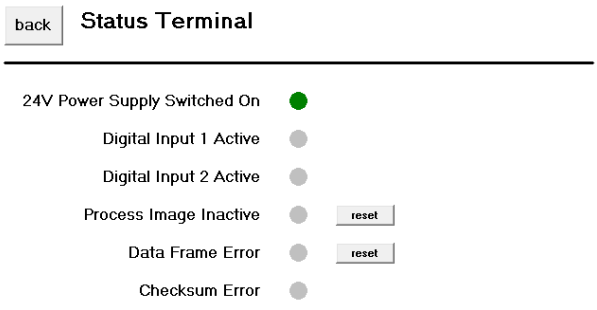
Besitzen SMI-Geräte noch keine Adresse, so bekommen alle SMI-Geräte durch das Betätigen der Schaltfläche *Start Addressing* eine eindeutige Adresse. Der Anwender hat keinen Einfluss darauf, welches SMI-Gerät welche Adresse zugewiesen bekommt. Die Vergabe der Adressen erfolgt absteigend, beginnend bei der Adresse, die durch den Parameter *Highest Address* vorgegeben wird. Die Felder *Current Manufacturer Id*, *Current Address* und *Current Search Slave-Id* geben Auskunft über den Status der Adressierung. Durch *Cancel Addressing* kann die Adressierung vorzeitig abgebrochen werden.

Status Library



Innerhalb der SPS-Bibliothek erfolgt die Kommunikation zwischen den einzelnen SPS-Bausteinen und der Busklemme über drei zentrale Puffer (je SMI-Klemme). Die Auslastung der Puffer und evtl. auftretende Überläufe können in der abgebildeten Tabelle ermittelt werden.

Status Terminal



Die Statusinformationen aus dem Prozessabbild der Klemme werden in diesem Dialog angezeigt. Auch lassen sich die quittierungspflichtigen Meldungen in diesen Dialog wieder zurücksetzen.

5.2 Herstellercodes

Wert (hex)	Wert (dez)	Beschreibung
0x00	0	alle Hersteller
0x01	1	Fa. Dunkermotoren GmbH
0x02	2	Fa. Becker Antriebe GmbH
0x03	3	Fa. elero GmbH
0x04	4	Fa. Selve GmbH & Co. KG
0x05	5	Fa. SUN-MASTER Sonnenschutz GmbH
0x06	6	Fa. Vestamatic GmbH
0x07	7	Fa. WAREMA Renkhoff GmbH
0x08	8	Fa. Groeninger Antriebstechnik GmbH
0x09	9	Fa. Gerhard Geiger GmbH & Co. KG
0x0A	10	Fa. Griesser AG
0x0B	11	unbenutzt
0x0C	12	unbenutzt
0x0D	13	unbenutzt
0x0E	14	unbenutzt
0x0F	15	reserviert für Erweiterungen

5.3 Support und Service

Beckhoff und seine weltweiten Partnerfirmen bieten einen umfassenden Support und Service, der eine schnelle und kompetente Unterstützung bei allen Fragen zu Beckhoff Produkten und Systemlösungen zur Verfügung stellt.

Downloadfinder

Unser Downloadfinder beinhaltet alle Dateien, die wir Ihnen zum Herunterladen anbieten. Sie finden dort Applikationsberichte, technische Dokumentationen, technische Zeichnungen, Konfigurationsdateien und vieles mehr.

Die Downloads sind in verschiedenen Formaten erhältlich.

Beckhoff Niederlassungen und Vertretungen

Wenden Sie sich bitte an Ihre Beckhoff Niederlassung oder Ihre Vertretung für den lokalen Support und Service zu Beckhoff Produkten!

Die Adressen der weltweiten Beckhoff Niederlassungen und Vertretungen entnehmen Sie bitte unserer Internetseite: www.beckhoff.com

Dort finden Sie auch weitere Dokumentationen zu Beckhoff Komponenten.

Beckhoff Support

Der Support bietet Ihnen einen umfangreichen technischen Support, der Sie nicht nur bei dem Einsatz einzelner Beckhoff Produkte, sondern auch bei weiteren umfassenden Dienstleistungen unterstützt:

- Support
- Planung, Programmierung und Inbetriebnahme komplexer Automatisierungssysteme
- umfangreiches Schulungsprogramm für Beckhoff Systemkomponenten

Hotline:	+49 5246 963-157
E-Mail:	support@beckhoff.com

Beckhoff Service

Das Beckhoff Service-Center unterstützt Sie rund um den After-Sales-Service:

- Vor-Ort-Service
- Reparaturservice
- Ersatzteilservice
- Hotline-Service

Hotline:	+49 5246 963-460
E-Mail:	service@beckhoff.com

Beckhoff Unternehmenszentrale

Beckhoff Automation GmbH & Co. KG

Hülshorstweg 20
33415 Verl
Deutschland

Telefon:	+49 5246 963-0
E-Mail:	info@beckhoff.com
Internet:	www.beckhoff.com

Mehr Informationen:
www.beckhoff.com/te1000

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