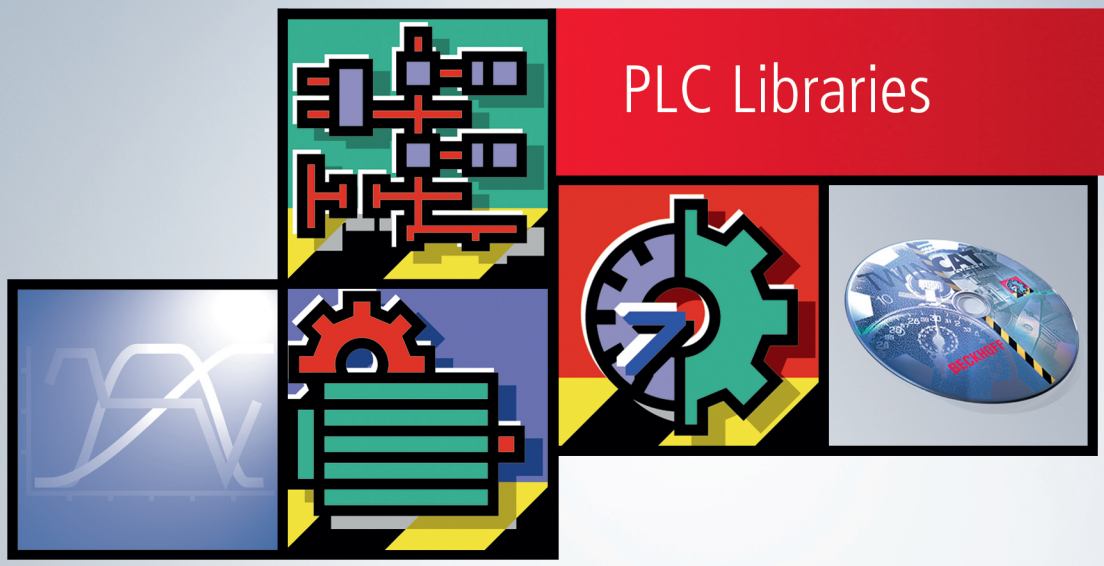
BECKHOFF New Automation Technology

Handbuch | DE

TX1200

TwinCAT 2 | PLC-Bibliothek: TcDMX



24.10.2022 | Version: 1.1

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, für jede Installation und Inbetriebnahme die zu dem betreffenden Zeitpunkt veröffentliche 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 weiter entwickelt.
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®

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

Copyright

1.2 Sicherheitshinweise

Sicherheitsbestimmungen

Beachten Sie die folgenden Sicherheitshinweise und Erklärungen!
Produktspezifische Sicherheitshinweise finden Sie auf den folgenden Seiten oder in den Bereichen Montage, Verdrahtung, Inbetriebnahme usw.

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.

Erklärung der Symbole

In der vorliegenden Dokumentation werden die folgenden Symbole mit einem nebenstehenden Sicherheitshinweis oder Hinweistext verwendet. Die Sicherheitshinweise sind aufmerksam zu lesen und unbedingt zu befolgen!

⚠ GEFAHR

Akute Verletzungsgefahr!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht unmittelbare Gefahr für Leben und Gesundheit von Personen!

⚠ WARNUNG

Verletzungsgefahr!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, besteht Gefahr für Leben und Gesundheit von Personen!

⚠ VORSICHT

Schädigung von Personen!

Wenn der Sicherheitshinweis neben diesem Symbol nicht beachtet wird, können Personen geschädigt werden!

HINWEIS

Schädigung von Umwelt oder Geräten

Wenn der Hinweis neben diesem Symbol nicht beachtet wird, können Umwelt oder Geräte geschädigt werden.

Tipp oder Fingerzeig

Dieses Symbol kennzeichnet Informationen, die zum besseren Verständnis beitragen.

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 Zielgruppen

Für den Nutzer dieser Bibliothek werden folgende Grundkenntnisse vorausgesetzt:

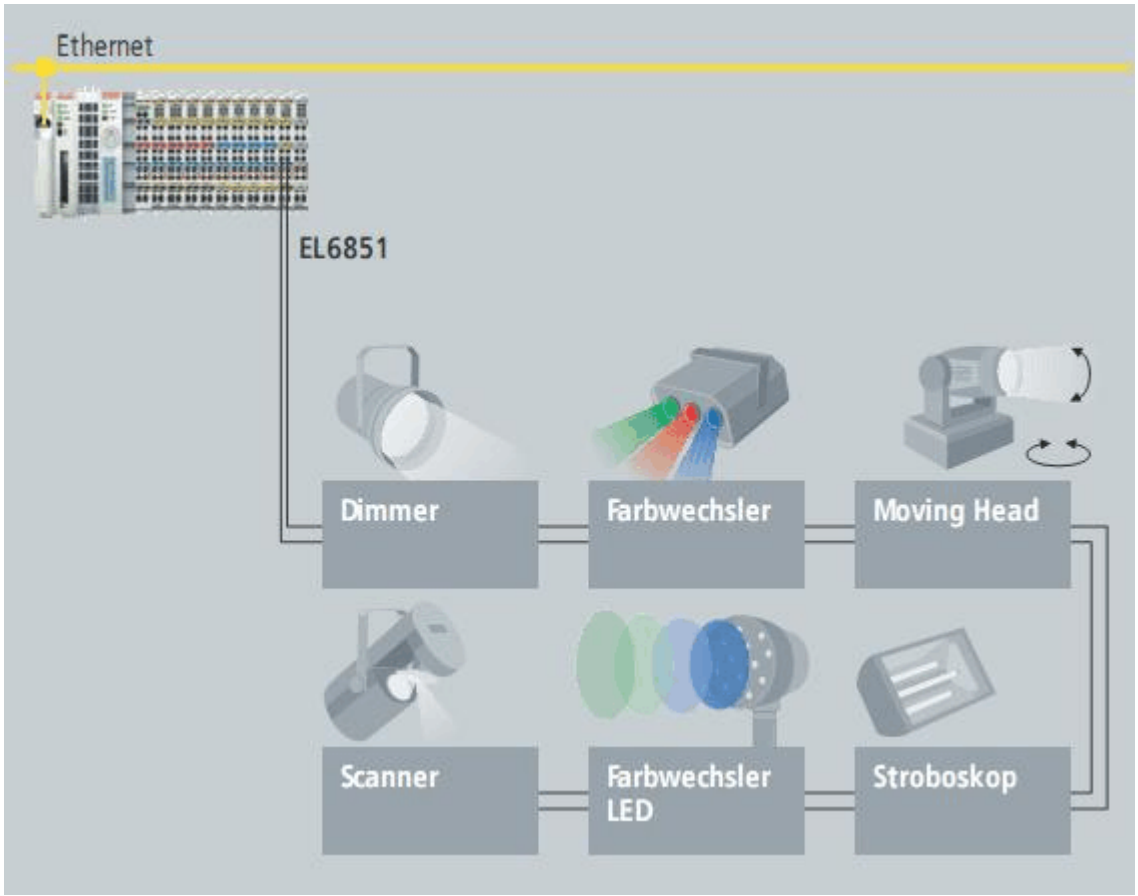
- TwinCAT PLC Control
- TwinCAT System Manager
- PC und Netzwerkkenntnisse
- Aufbau und Eigenschaften der Beckhoff Embedded-PC und deren Busklemmensystem
- Technologie von DMX Geräten
- Einschlägige Sicherheitsvorschriften der technischen Gebäudeausrüstung

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.

3 DMX

DMX ist das Standardprotokoll fÜr die Ansteuerung von Geräten der professionellen Bühnen- und Effektbeleuchtung, die beispielsweise zur dynamischen Beleuchtung von Show- und Verkaufsräumen, sowie für exklusive Licht- und Farbspiele in Prestige trächtigen Gebäuden, wie Hotels und Veranstaltungszentren, eingesetzt werden. Bei DMX-Geräten in Form von statischen Lichtquellen werden Farbmischungs- und Helligkeitswerte Übermittelt, bei bewegten Lichtquellen werden zusätzlich Raumkoordinaten kommuniziert. Die hohe Übertragungsrate von EtherCAT ermöglicht eine höhere Aktualisierungsrate der Lichteinstellungen, und bewirkt, dass die Licht- und Farbwechsel vom Auge harmonischer wahrgenommen werden. Mit der EL6851 [▶ 76] können DMX-Geräte mit drei Achsen wie z. B. Scanner, Moving Heads oder Spotlights angesteuert werden, die Implementierung des RDM-Protokolls (**R**emote **D**evice **M**anagement) für DMX-interne Diagnose und Parametrierung ist mit TwinCAT Bausteinen mÖglich.

Der DMX-Master sendet zyklisch mit 250 kBaud neue Einstellungen an Slaves, um dynamische Lichtwechsel und Farbspiele zu generieren. Im DMX-Protokoll sind maximal 32 Slaves an einem Strang ohne Repeater möglich. Der 512 Byte lange Frame im DMX-Protokoll wird als „Universe“ bezeichnet. In ihm stehen 512 Kanäle zur Verfügung, von denen jeweils einer mit 8 Bit Auflösung, also in 256 Stufen, eine Geräte-Einstellung repräsentiert z.B. Dimmen, Farbe, Fokus etc. Bei bewegten Lichtquellen nehmen weitere Einstellungen wie Neigung, Schwenken und Geschwindigkeit (in 8- oder 16 Bit Auflösung) zusätzliche Kanäle in Anspruch, so dass die 512 Kanäle nur indirekt für 32 Teilnehmer ausreichen. Weiterhin benötigt ein Frame bei voller Ausnutzung der Universe 22 ms zur internen DMX-Zirkulation, was eine Refresh-Rate von 44 Hz bedeutet. Licht-Änderungen bei dieser Frequenz werden als unharmonisch wahrgenommen, erst ab einer Frequenz von >200Hz erscheinen die Übergänge harmonisch. Durch Verringerung der Nutzdatenmenge kann die Zirkulationsdauer des Frames reduziert werden, optimal hat sich eine Ausnutzung von 64 Byte (Frequenz > 300 Hz) erwiesen, damit stehen 64 Kanäle pro Universe zu Verfügung.



Mit der EL6851 wird die Einbindung mehrerer Universen in eine Steuerung einfach: EtherCAT kann große Datenmengen schnell Übermitteln, das EtherCAT-Protokoll bleibt bis in die Klemme erhalten und die Klemme unterstützt verschieden große Mappings (64 bis 512 Byte). So kann bei Anschaltung mehrerer Masterklemmen, jeweils als eigene Universe, der Zeitversatz bei der Übertragung von der Steuerung zum Master deutlich minimiert werden.

4 Integration in TwinCAT (CX9020)

Dieses Beispiel beschreibt, wie ein einfaches SPS-Programm für DMX in TwinCAT geschrieben werden kann und wie es mit der Hardware verknüpft wird. Es soll nach DMX-Geräten gesucht werden.

Beispieldateien entpacken https://infosys.beckhoff.com/content/1031/tcplclibdmx/Resources/

11977737739.zip

Hardware

Einrichtung der Komponenten

Es wird folgende Hardware benötigt:

- 1x Embedded-PC CX9020
- 1x DMX-Master-Klemme EL6851 [▶ 76]
- 1x Abschlusskappe EL9111

Richten Sie die Hardware sowie die DMX-Komponenten wie in den entsprechenden Dokumentationen beschrieben ein.

Software

Erstellung des SPS-Programms

Erstellen Sie ein neues SPS-Projekt für PC-basierte Systeme (ARM) und fügen die Bibliothek *TcDMX.lib* hinzu.

Erzeugen Sie als Nächstes die folgenden globalen Variablen:

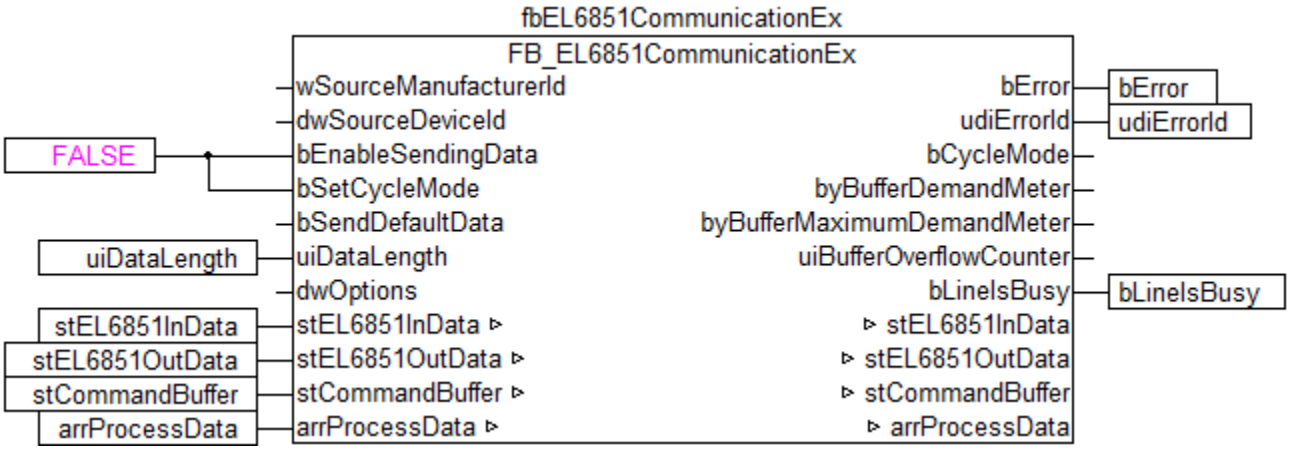
```
VAR_GLOBAL
    stEL6851InData   AT %I* : ST_EL6851InDataEx;
    stEL6851OutData  AT %Q* : ST_EL6851OutData;
    stCommandBuffer         : ST_DMXCommandBuffer;
END_VAR
```

stEL6851InData: Eingangsvariable für die DMX-Klemme.

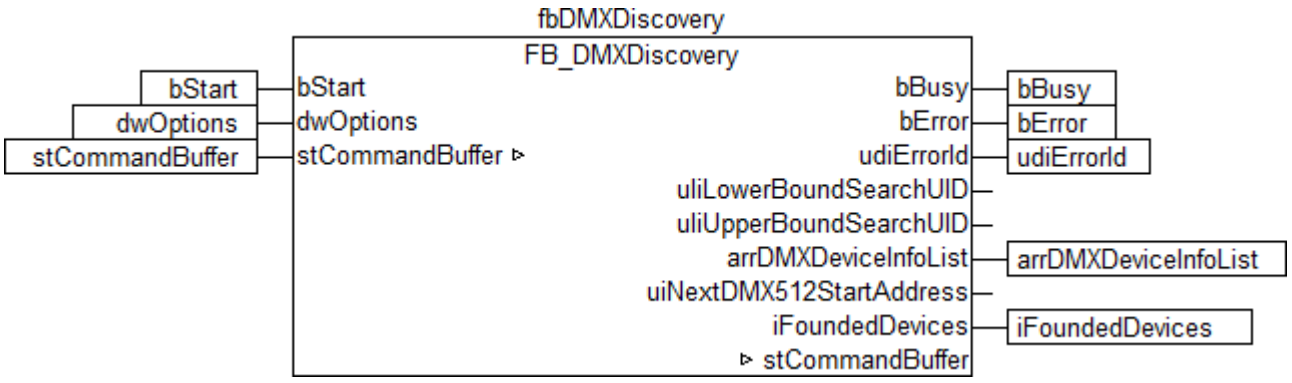
stEL6851OutData: Ausgangsvariable für die DMX-Klemme.

stCommandBuffer: Wird für die Kommunikation mit DMX benötigt.

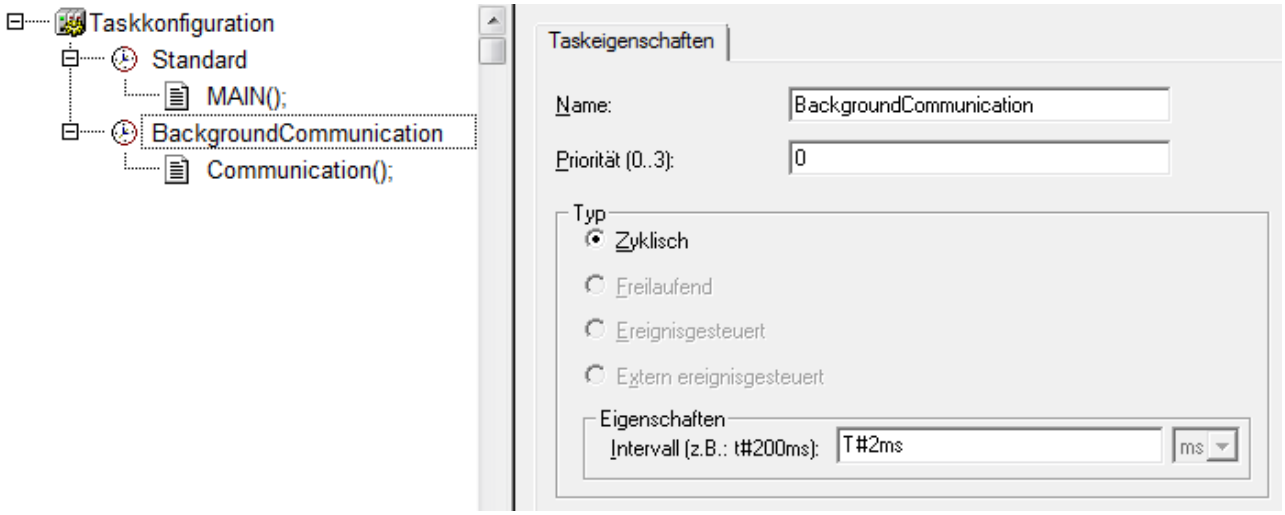
Legen Sie anschließend ein Programm (CFC) für die Hintergrundkommunikation mit DMX an. In diesem wird der Baustein FB_EL6851CommunicationEx() [▶ 23] aufgerufen. Achten Sie beim Kommunikationsbaustein darauf, mit *stEL6851InData*, *stEL6851OutData* und *stCommandBuffer* zu verknüpfen.



Legen Sie ein MAIN-Programm (CFC) an in dem der Baustein FB_DMXDiscovery() [▶ 16] aufgerufen wird. Der Eingang *stCommandBuffer* des Bausteins wird mit der globalen Variable *stCommandBuffer* verknüpft.



Gehen Sie in die Taskkonfiguration und legen Sie eine neue Task für die Hintergrundkommunikation an. Fügen Sie dieser Task das Programm für die Kommunikation hinzu. Geben Sie dieser Task eine höhere Priorität (kleinere Zahl) und eine niedrigere Intervall-Zeit als der Standard-Task. Genauere Informationen dazu finden Sie in der Beschreibung des Bausteins FB_EL6851CommunicationEx() [▶ 23].

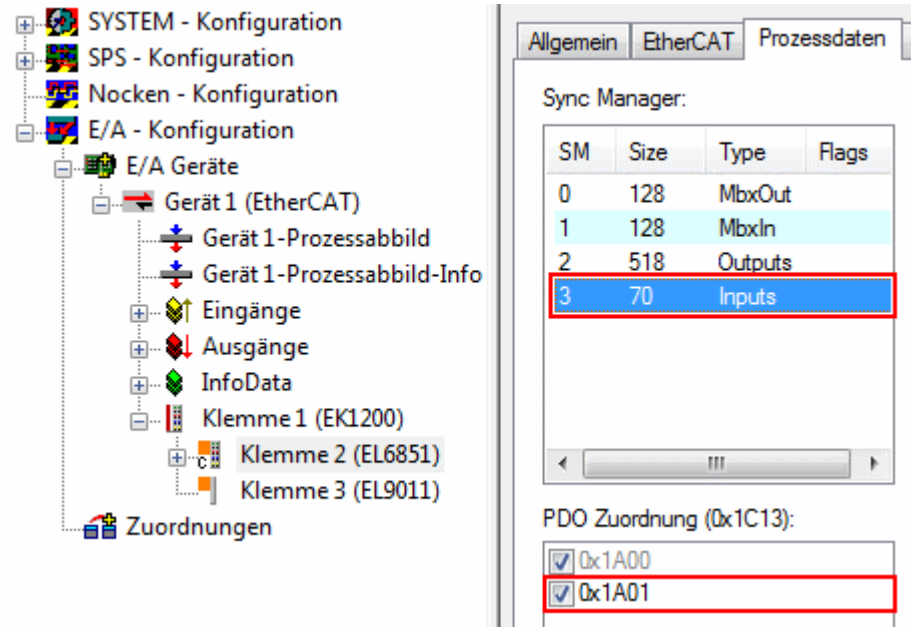


Laden Sie das Projekt als Bootprojekt auf den CX und speichern Sie es ab.

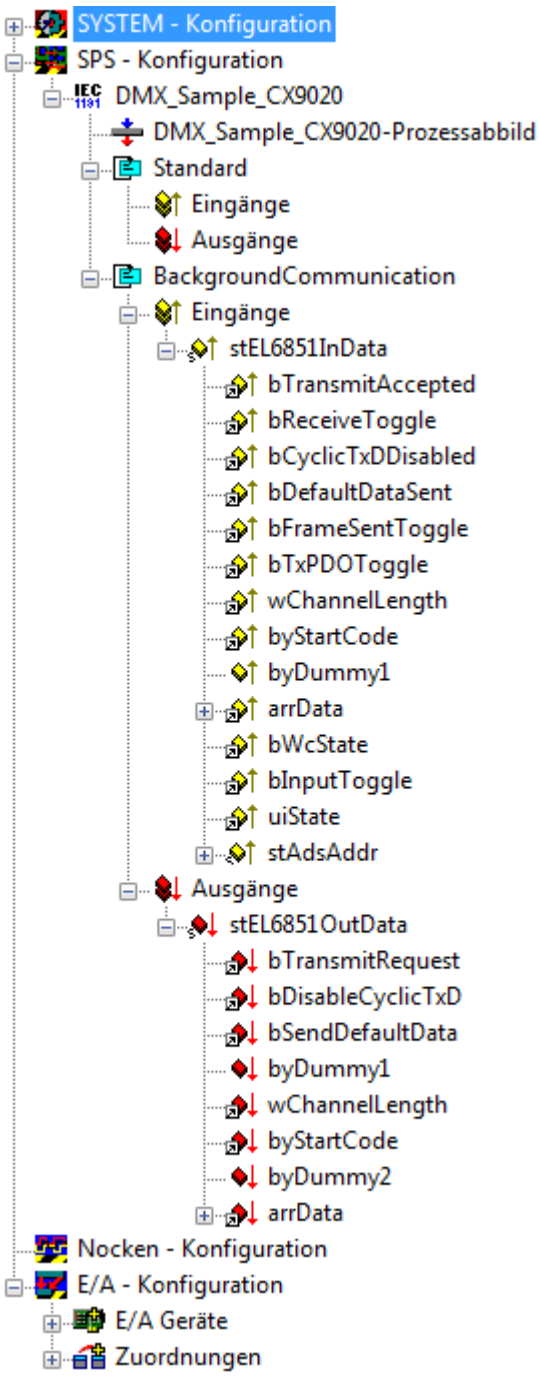
Konfiguration im System Manager

Legen Sie ein neues TwinCAT System-Manager-Projekt an, wählen Sie als Zielsystem den CX und lassen Sie nach dessen Hardware suchen.

Stellen Sie die DMX-Eingänge 1 bis 64 zur Verfügung, indem Sie die Registerkarte *Prozessdaten* der EL6851 öffnen und unter *Inputs* die Option *0x1A01* anwählen.



Fügen Sie das oben angelegte SPS-Programm unter SPS-Konfiguration hinzu. Durch Aufklappen des SPS-Projekts in der Baumansicht werden die beiden Tasks aufgelistet. Beim Erweitern der Tasks ist sicherzustellen, dass die globalen Ein- und Ausgangsvariablen *stEL6851InData* und *stEL6851OutData* der Hintergrund-Kommunikations-Task zugeordnet sind da diese schneller abgearbeitet werden sollen. Mit Drag & Drop können die Variablen verschoben werden.



Verknüpfen Sie die globalen Variablen des SPS-Programms nun mit den Ein- und Ausgängen der Busklemmen, erzeugen Sie die Zuordnungen und aktivieren Sie die Konfiguration. Starten Sie dann das Gerät im Run-Modus.

Ihr CX ist jetzt einsatzbereit.

Wenn die Variable *bStart* TRUE ist wird die Suche nach DMX-Geräten gestartet. Die Anzahl gefundener Geräte steht in der Variable *iFoundedDevices* und zusätzliche Informationen dazu in *arrDMXDeviceInfoList*.

Sehen Sie dazu auch

- ST_EL6851InDataEx [▶ 70]
- ST_EL6851OutData [▶ 70]
- ST_DMXCommandBuffer [▶ 66]

5 Programmierung

Installation

Ab TwinCAT 2.11 Build 2229 (R3 und x64 Engineering) wird die Bibliothek "TcDMX.lib" standard-mäßig mitinstalliert.

Name der Bibliothek

Diese Bibliothek ersetzt die "TcEL6851.lib".

Hardware Dokumentation im Beckhoff Information System: EL6851 - DMX-Master-Slaveklemme [▶ 76]

Weitere erforderliche Bibliotheken

Für PC-Systeme (x86) und Embedded-PCs (CXxxxx):

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

Für Busklemmen-Controller der Serie BCxx00, BCxx50, BCxx20, BC9191 und BXxx00:

- nicht verfügbar

Speicherauslastung

Durch Einbinden der Bibliothek wird bereits SPS-Programmspeicher verbraucht. Abhängig vom Applikationsprogramm kann daher der verbleibende Speicher nicht ausreichend sein.

Es empfiehlt sich der Einsatz der TwinCAT-Bibliotheken sowohl für das RDM-Protokoll als auch für das Versenden der zyklischen Prozessdaten zu den DMX-Slaves. Zu beiden Varianten finden Sie Beispiele im Anhang.

Beschreibung der Bibliothek

Es ist grundsätzlich nur ein Baustein für die Grundkommunikation notwendig. Der FB_EL6851CommunicationEx übernimmt die Kommunikation mit der EL6851. Mit diesem Baustein kann zwischen RDM- und dem DMX-Protokoll umgeschaltet werden. Wird das DMX-Protokoll gefahren, kann keine RDM-Kommunikation stattfinden und umgekehrt.

5.1 Übersicht Funktionsbausteine

High Level

Name	Beschreibung
FB_DMXDiscovery [▶ 16]	Sucht nach bis zu 50 DMX-Geräten und stellt optional die Startadresse automatisch ein.
FB_DMXDiscovery512 [▶ 18]	Sucht nach bis zu 512 DMX-Geräten und stellt optional die Startadresse automatisch ein.

Low Level

Base

Name	Beschreibung
FB_DMXSendRDMCommand [▶ 19]	Dient zum allgemeinen Senden eines RDM-Kommandos.
FB_EL6851Communication [▶ 21]	Zugriff auf die EL6851 [▶ 76].

Name	Beschreibung
FB_EL6851CommunicationEx [▶ 23]	Zugriff auf die EL6851 [▶ 76].

Device Control Parameter Messages

Name	Beschreibung
FB_DMXGetIdentifyDevice [▶ 25]	Fragt ab, ob von einem DMX-Gerät die Identifizierung aktiv ist.
FB_DMXSetIdentifyDevice [▶ 26]	Aktiviert oder deaktiviert die Identifizierung eines DMX-Gerätes.
FB_DMXSetResetDevice [▶ 27]	Aktiviert einen Reset bei einem DMX-Gerät.

Discovery Messages

Name	Beschreibung
FB_DMXDiscMute [▶ 28]	Setzt das Mute-Flag von einem DMX-Gerät.
FB_DMXDiscUniqueBranch [▶ 30]	Fragt ab, ob sich innerhalb eines bestimmten Adressbereichs DMX-Geräte befinden.
FB_DMXDiscUnMute [▶ 31]	Setzt das Mute-Flag von einem DMX-Gerät zurück.

Power/Lamp Setting Parameter Messages

Name	Beschreibung
FB_DMXGetLampHours [▶ 32]	Liest die Anzahl der Stunden aus, in denen die Lampe an war.
FB_DMXGetLampOnMode [▶ 33]	Liest den Parameter aus, der das Einschaltverhalten des DMX-Gerätes definiert.
FB_DMXSetLampHours [▶ 35]	Setzt den Betriebsstundenzähler der Lampe.
FB_DMXSetLampOnMode [▶ 36]	Legt das Einschaltverhalten des DMX-Gerätes fest.

Product Information Messages

Name	Beschreibung
FB_DMXGetDeviceInfo [▶ 37]	Fragt alle relevanten Informationen von einem DMX-Gerät ab.
FB_DMXGetDeviceLabel [▶ 38]	Liest aus dem DMX-Gerät einen Text aus, der das Gerät genauer beschreibt.
FB_DMXGetDeviceModelDescription [▶ 39]	Fragt die Beschreibung des Gerätetyps ab.
FB_DMXGetManufacturerLabel [▶ 40]	Fragt die Beschreibung des DMX-Geräteherstellers ab.
FB_DMXGetProductDetailIdList [▶ 42]	Fragt die Kategorien ab, zu denen das DMX-Gerätes zugehörig ist.
FB_DMXGetSoftwareVersionLabel [▶ 43]	Fragt die Beschreibung der Softwareversion des DMX-Gerätes ab.
FB_DMXSetDeviceLabel [▶ 44]	Schreibt in das DMX-Gerät einen Beschreibungstext.

Queued and Status Messages

Name	Beschreibung
FB_DMXClearStatusId [▶ 45]	Löscht den Nachrichtenbuffer im DMX-Gerät.
FB_DMXGetStatusIdDescription [▶ 46]	Liest den Text einer bestimmten Status-Id aus dem DMX-Gerät aus.
FB_DMXGetStatusMessages [▶ 47]	Sammelt die Status- oder Fehlerinformation von meinem DMX-Gerät.

RDM Information Messages

Name	Beschreibung
FB_DMXGetParameterDescription [▶ 48]	Fragt die Definition von herstellerspezifischen PIDs ab.
FB_DMXGetSupportedParameters [▶ 49]	Fragt alle unterstützen Parameter von einem DMX-Gerät ab.

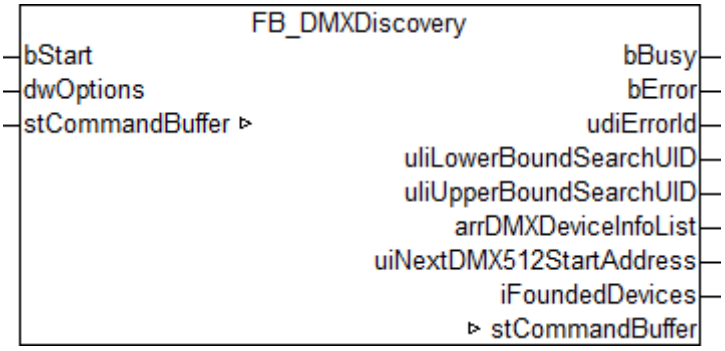
Sensor Parameter Messages

Name	Beschreibung
FB_DMXGetSensorDefinition [▶ 50]	Fragt die Definition eines bestimmten Sensors ab.
FB_DMXGetSensorValue [▶ 52]	Fragt den aktuellen Wert eines Sensors ab.

Setup Messages

Name	Beschreibung
FB_DMXGetDMX512PersonalityDescription [▶ 53]	Liest aus dem DMX-Gerät weitere Informationen einer *Personality* aus.
FB_DMXGetDMX512StartAddress [▶ 54]	Erfragt die DMX512-Startadresse.
FB_DMXGetSlotDescription [▶ 55]	Fragt die Textbeschreibung für Slot Offsets ab.
FB_DMXGetSlotInfo [▶ 56]	Fragt die Basisinformationen zur Funktionalität der DMX512 Slots eines DMX-Gerätes ab.
FB_DMXSetDMX512StartAddress [▶ 57]	Setzt die DMX512-Startadresse.

5.2 FB_DMXDiscovery



Dieser Funktionsbaustein sucht nach bis zu 50 DMX-Geräten und stellt optional die Startadresse automatisch ein. Die wichtigsten Informationen der gefundenen Geräte werden in einer Struktur angezeigt.

VAR_INPUT

```
bStart          : BOOL;
dwOptions       : DWORD;
```

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

dwOptions: Optionen (siehe Tabelle). Die einzelnen Konstanten müssen miteinander "ODER" verknüpft werden.

Konstante	Beschreibung
DMX_OPTION_COMPLETE_NEW_DISCOVERY	Es werden alle DMX-Geräte berücksichtigt.
DMX_OPTION_SET_START_ADDRESS	Bei allen DMX-Geräten die gefunden werden, wird die Startadresse gesetzt. Fortlaufend beginnend mit 1.

Konstante	Beschreibung
DMX_OPTION_OPTICAL_FEEDBACK	Nachdem ein DMX-Gerät gefunden wurde, wird für zwei Sekunden die Funktion IDENTIFY_DEVICE aufgerufen.

VAR_OUTPUT

```
bBusy                   : BOOL;
bError                  : BOOL;
udiErrorId              : UDINT;
uliLowerBoundSearchUID  : T_ULARGE_INTEGER;
uliUpperBoundSearchUID  : T_ULARGE_INTEGER;
arrDMXDeviceInfoList    : ARRAY[1..50] OF ST_DMXDeviceInfo;
uiNextDMX512StartAddress : UINT;
iFoundedDevices         : INT;
```

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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

uliLowerBoundSearchUID: Während der Suche wird an diesem Ausgang die untere Suchadresse ausgegeben.

uliUpperBoundSearchUID: Während der Suche wird an diesem Ausgang die obere Suchadresse ausgegeben.

arrDMXDeviceInfoList: Array mit den wichtigsten Informationen der gefundenen DMX-Geräte.

uiNextDMX512StartAddress: Ist die Option DMX_OPTION_SET_START_ADDRESS aktiviert, so wird an diesem Ausgang die Startadresse angezeigt, die dem nächsten DMX-Gerät zugewiesen wird.

iFoundedDevices: Während der Suche wird an diesem Ausgang die aktuelle Anzahl der gefunden Geräte ausgegeben.

VAR_IN_OUT

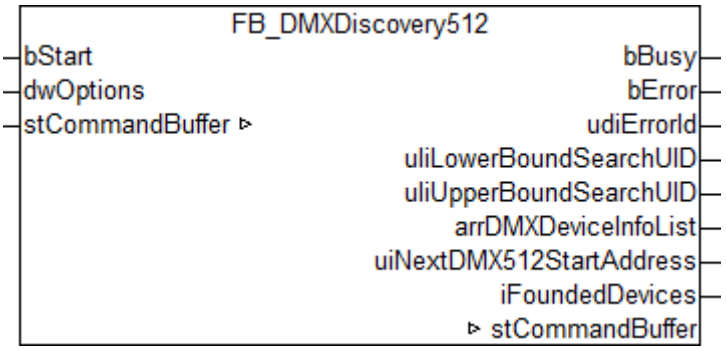
```
stCommandBuffer      : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

Sehen Sie dazu auch

- ST_DMXDeviceInfo [▶ 66]
- ST_DMXCommandBuffer [▶ 66]

5.3 FB_DMXDiscovery512



Dieser Funktionsbaustein sucht nach bis zu 512 DMX-Geräten und stellt optional die Startadresse automatisch ein. Die wichtigsten Informationen der gefundenen Geräte werden in einer Struktur angezeigt.

VAR_INPUT

```
bStart          : BOOL;
dwOptions       : DWORD;
```

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

dwOptions: Optionen (siehe Tabelle). Die einzelnen Konstanten müssen miteinander "ODER" verknüpft werden.

Konstante	Beschreibung
DMX_OPTION_COMPLETE_NEW_DISCOVERY	Es werden alle DMX-Geräte berücksichtigt.
DMX_OPTION_SET_START_ADDRESS	Bei allen DMX-Geräten die gefunden werden, wird die Startadresse gesetzt. Fortlaufend beginnend mit 1.
DMX_OPTION_OPTICAL_FEEDBACK	Nachdem ein DMX-Gerät gefunden wurde, wird für zwei Sekunden die Funktion IDENTIFY_DEVICE aufgerufen.

VAR_OUTPUT

```
bBusy                   : BOOL;
bError                  : BOOL;
udiErrorId              : UDINT;
uliLowerBoundSearchUID  : T_ULARGE_INTEGER;
uliUpperBoundSearchUID  : T_ULARGE_INTEGER;
arrDMXDeviceInfoList    : ARRAY[1..512] OF ST_DMXDeviceInfo;
uiNextDMX512StartAddress : UINT;
iFoundedDevices         : INT;
```

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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

uliLowerBoundSearchUID: Während der Suche wird an diesem Ausgang die untere Suchadresse ausgegeben.

uliUpperBoundSearchUID: Während der Suche wird an diesem Ausgang die obere Suchadresse ausgegeben.

arrDMXDeviceInfoList: Array mit den wichtigsten Informationen der gefundenen DMX-Geräte.

uiNextDMX512StartAddress: Ist die Option DMX_OPTION_SET_START_ADDRESS aktiviert, so wird an diesem Ausgang die Startadresse angezeigt, die dem nächsten DMX-Gerät zugewiesen wird.

iFoundedDevices: Während der Suche wird an diesem Ausgang die aktuelle Anzahl der gefunden Geräte ausgegeben.

VAR_IN_OUT

```
stCommandBuffer      : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

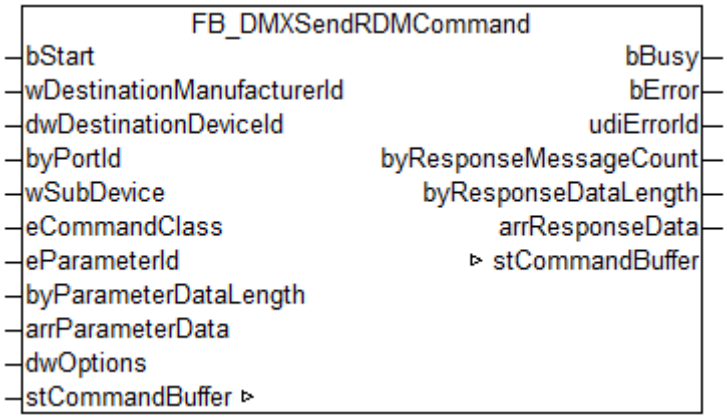
Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2240	PC/CX	TcDMX-Bibliothek ab V1.1.0

Sehen Sie dazu auch

- ST_DMXDeviceInfo [▶ 66]
- ST_DMXCommandBuffer [▶ 66]

5.4 FB_DMXSendRDMCommand



Dieser Funktionsbaustein dient zum allgemeinen Senden eines RDM-Kommandos, definiert per Befehlsnummer und, falls erforderlich, Übergabeparameter.

VAR_INPUT

```
bStart                      : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                    : BYTE;
wSubDevice                  : WORD;
eCommandClass               : E_DMXCommandClass;
eParameterId                : E_DMXParameterId;
byParameterDataLength       : BYTE;
arrParameterData            : ARRAY[0..255] OF BYTE;
dwOptions                   : DWORD := 0;
```

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

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

wSubDevice: Untergeräte (Sub-Devices) sollten in Geräten mit sich wiederholender Anzahl von Modulen verwendet werden, wie bei einem Dimmer Rack. Der Untergeräte (Sub-Device) Eingang erlaubt es Parameternachrichten an ein bestimmtes Modul innerhalb des Gerätes zu senden, um Eigenschaften von dem Modul zu lesen oder zu setzen.

eCommandClass: Command Class (CC) gibt die Aktion der Nachricht an.

eParameterId: Parameter Id ist eine 16-Bit Nummer, die einen bestimmten Typ von Parameterdaten identifiziert.

byParameterDataLength: Die Datenlänge der Parameter (PDL) ist die vorausgegangene Anzahl an Slots enthalten im Parameterdatenbereich. Ist dieser Eingang 0x00, so folgen keine Parameterdaten.

arrParameterData: Parameterdaten von variabler Länge. Das Format vom Inhalt ist PID abhängig.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

```
bBusy                   : BOOL;
bError                  : BOOL;
udiErrorId              : UDINT;
byResponseMessageCount  : BYTE;
byResponseDataLength    : BYTE;
arrResponseData         : ARRAY[0..255] OF 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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

byResponseMessageCount: Dieser Ausgang zeigt an, dass sich in den DMX-Slave weitere Nachrichten befinden. Mit dem RDM-Befehl Get: QUEUED_MESSAGE werden diese Nachrichten ausgelesen.

byResponseDataLength: Enhält die Anzahl der Bytes, die durch den RDM-Befehl zurückgegeben werden.

arrResponseData: Dieser Ausgang enthält die Daten der Rückantwort vom RDM-Befehl. Die Länge is variabel und das Format der Daten ist abhängig vom RDM-Befehl.

VAR_IN_OUT

```
stCommandBuffer          : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

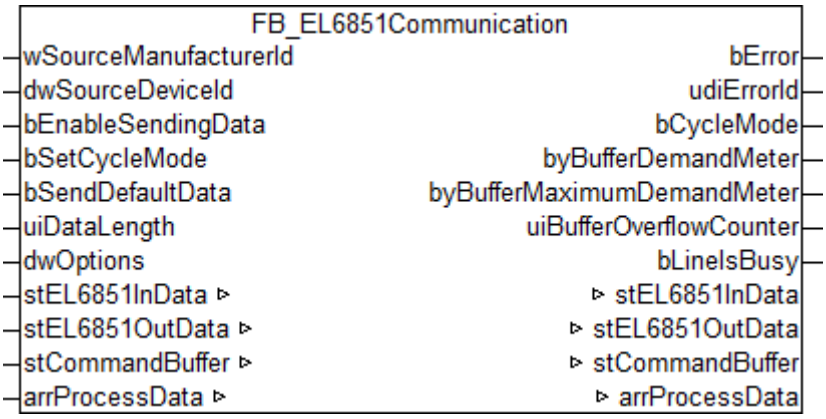
Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

Sehen Sie dazu auch

- E_DMXCommandClass [▶ 58]
- E_DMXParameterId [▶ 59]
- ST_DMXCommandBuffer [▶ 66]

5.5 FB_EL6851Communication



Dieser Baustein ist veraltet. Nutzen Sie stattdessen den FB_EL6851CommunicationEx() [▶ 23].

Der Zugriff auf die EL6851 [▶ 76] sollte immer über diesen Baustein erfolgen. Dieses betrifft sowohl das Versenden der zyklischen DMX-Daten, als auch das Versenden der RDM-Befehle.

Sollen Daten zyklisch zu den DMX-Geräten versendet werden, so setzen sie den Eingang *bEnableSendingData* auf TRUE, den Eingang *bSetCycleMode* auf TRUE, den Eingang *bSendDefaultData* auf FALSE und den Eingang *uiDataLength* auf die entsprechende Länge (jn Byte). Die Daten, die versendet werden sollen, können über die Variable *arrProcessData* vorgegeben werden.

Sollen RDM-Befehle versendet werden, so setzen sie den Eingang *bEnableSendingData* auf FALSE und den Eingang *bSetCycleMode* auf FALSE. Die Bausteine für die DMX-RDM-Befehle greifen nicht direkt auf das Prozessabbild der EL6851 zu, sondern legen die einzelnen DMX-RDM-Befehle in einen Puffer ab. Der Baustein FB_EL6851Communication() liest sequentiell die Befehle aus diesen Puffer aus und gibt diese zu der EL6851 weiter. Hierdurch wird sichergestellt, dass nicht mehrere Bausteine gleichzeitig auf das Prozessabbild der EL6851 zugreifen. Der Puffer, in dem die DMX-RDM-Befehle abgelegt werden, ist in einer Variablen vom Typ ST_DMXCommandBuffer enthalten. Pro EL6851 gibt es eine Instanz vom Baustein FB_EL6851Communication() und eine Variable vom Typ ST_DMXCommandBuffer.

Über die Ausgänge des Bausteins kann ermittelt werden, wie stark der Puffer ausgelastet ist. Sollten Sie feststellen, dass der Puffer regelmäßig überläuft, so sollten mit Hilfe des TwinCAT System Managers die Auslastung der SPS-Task analysieren.

Der Baustein FB_EL6851Communication() kann, wenn nötig in einer separaten, schnelleren Task aufgerufen werden. In diesem Fall sollte die schnellere Task, in der der Baustein FB_EL6851Communication() aufgerufen wird, eine höhere Priorität haben, als die TASK, in der die Bausteine für die RDM-Befehle aufgerufen werden.

Zu beiden Betriebsarten finden sie auch entsprechende Beispiele im Anhang.

Anmerkung zu den IDs von DMX-Geräten

Jedes DMX-Gerät hat eine eindeutige, feste, 48-Bit lange Adresse, auch Unique ID oder kurz UID genannt. Diese Adresse besteht aus der Hersteller-Id (16 Bit) und der Geräte-Id (32 Bit). Die Hersteller-Id identifiziert den Hersteller des Gerätes und wird von der ESTA (Entertainment Services and Technology Association) vergeben. Eine Liste mit allen bekannten Hersteller-Ids ist zu finden unter http://www.esta.org/tsp/working_groups/CP/mfctrIDs.php. Die Geräte-Id wird frei vom Hersteller festgelegt. Hierdurch soll sichergestellt werden, dass jede UID weltweit einmalig vorhanden ist. Die UID kann in der Regel nicht verändert werden. Von der ESTA wurde Beckhoff Automation die Hersteller-Id 0x4241 zugeordnet. Da auch ein DMX-Master eine UID besitzt, sollte diese entsprechend der ESTA angegeben werden (Eingang *wSourceManufacturerId*).

VAR_INPUT

```
wSourceManufacturerId     : WORD := 16#42_41;
dwSourceDeviceId          : DWORD := 16#12_13_14_15;
bEnableSendingData        : BOOL := TRUE;
bSetCycleMode             : BOOL := TRUE;
bSendDefaultData          : BOOL;
uiDataLength              : UINT;
dwOptions                 : DWORD;
```

wSourceManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät. Sollte gemäß der ESTA 0x4241 sein.

dwSourceDeviceId: Eindeutige Geräte-Id vom DMX-Gerät. Kann frei vergeben werden.

bEnableSendingData: Ist die Klemme im Cycle-Mode (Ausgang *bCycleMode* = TRUE), so kann das Versenden mit diesem Baustein aktiviert (TRUE) oder gesperrt (FALSE) werden..

bSetCycleMode: Aktiviert den Cycle-Mode. Im Cycle-Mode können die zyklischen Prozessdaten an die DMX-Geräte versendet werden. Zum Versenden der RDM-DMX-Befehle muss der Cycle-Mode deaktiviert werden..

bSendDefaultData: Ist dieser Eingang aktive (TRUE), so werden im Cycle-Mode die Standardwerte versendet.

uiDataLength: Dieser Eingang ist nur relevant, wenn der Cycle-Mode aktiv ist. Er gibt die Länge des DMX512-Frames in Bytes an.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

```
bError                    : BOOL;
udiErrorId                : UDINT;
bCycleMode                : BOOL;
byBufferDemandMeter       : BYTE;
byBufferMaximumDemandMeter  : BYTE;
uiBufferOverflowCounter   : UINT;
bLineIsBusy               : BOOL;
```

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

bCycleMode: Ist auf TRUE, wenn der Cycle-Mode aktiv ist (siehe auch Eingang *bSetCycleMode*).

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

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

uiBufferOverflowCounter: Bisherige Anzahl der Pufferüberläufe.

bLineIsBusy: Solange der Baustein FB_EL6851Communication() DMX-RDM-Befehle bearbeitet, ist dieser Ausgang gesetzt.

VAR_IN_OUT

```
stEL6851InData        : ST_EL6851InData;
stEL6851OutData       : ST_EL6851OutData;
stCommandBuffer       : ST_DMXCommandBuffer;
arrProcessData        : ARRAY[1..512] OF BYTE;
```

stEL6851InData: Struktur im Eingangsprozessabbild der EL6851. Sie dient zur Kommunikation von der EL6851 [▶ 76] zur SPS.

stEL6851OutData: Struktur im Ausgangsprozessabbild der EL6851. Sie dient zur Kommunikation von der SPS zur EL6851 [▶ 76].

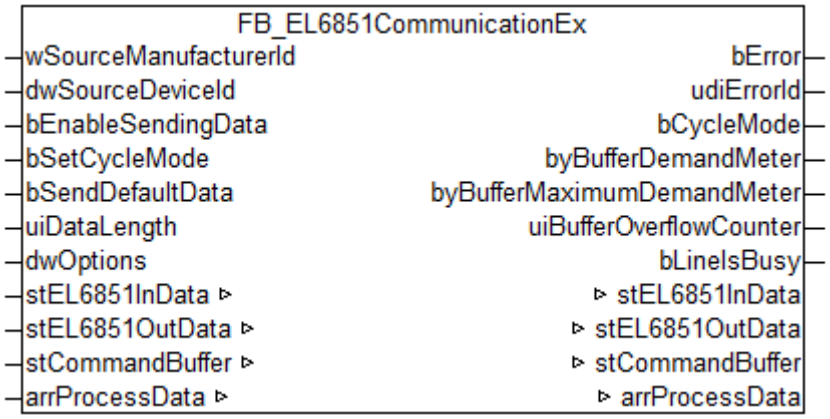
stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851Communication()-Baustein.

arrProcessData: Über diese Variable werden dem Baustein die Daten übergeben, die zyklisch an die DMX-Geräte versendet werden sollen. Hierzu muss der CycleMode aktiv sein (siehe auch Eingang *bSetCycleMode*).

Sehen Sie dazu auch

- ST_EL6851InData [▶ 70]
- ST_EL6851OutData [▶ 70]
- ST_DMXCommandBuffer [▶ 66]

5.6 FB_EL6851CommunicationEx



Der Zugriff auf die EL6851 [▶ 76] sollte immer über diesen Baustein erfolgen. Dieses betrifft sowohl das Versenden der zyklischen DMX-Daten, als auch das Versenden der RDM-Befehle.

Sollen Daten zyklisch zu den DMX-Geräten versendet werden, so setzen sie den Eingang *bEnableSendingData* auf TRUE, den Eingang *bSetCycleMode* auf TRUE, den Eingang *bSendDefaultData* auf FALSE und den Eingang *uiDataLength* auf die entsprechende Länge (jn Byte). Die Daten, die versendet werden sollen, können über die Variable *arrProcessData* vorgegeben werden.

Sollen RDM-Befehle versendet werden, so setzen sie den Eingang *bEnableSendingData* auf FALSE und den Eingang *bSetCycleMode* auf FALSE. Die Bausteine für die DMX-RDM-Befehle greifen nicht direkt auf das Prozessabbild der EL6851 zu, sondern legen die einzelnen DMX-RDM-Befehle in einen Puffer ab. Der Baustein FB_EL6851CommunicationEx() liest sequentiell die Befehle aus diesen Puffer aus und gibt diese zu der EL6851 weiter. Hierdurch wird sichergestellt, dass nicht mehrere Bausteine gleichzeitig auf das Prozessabbild der EL6851 zugreifen. Der Puffer, in dem die DMX-RDM-Befehle abgelegt werden, ist in einer Variablen vom Typ ST_DMXCommandBuffer enthalten. Pro EL6851 gibt es eine Instanz vom Baustein FB_EL6851CommunicationEx() und eine Variable vom Typ ST_DMXCommandBuffer.

Über die Ausgänge des Bausteins kann ermittelt werden, wie stark der Puffer ausgelastet ist. Sollten Sie feststellen, dass der Puffer regelmäßig überläuft, so sollten mit Hilfe des TwinCAT System Managers die Auslastung der SPS-Task analysieren.

Der Baustein FB_EL6851CommunicationEx() kann, wenn nötig in einer separaten, schnelleren Task aufgerufen werden. In diesem Fall sollte die schnellere Task, in der der Baustein FB_EL6851CommunicationEx() aufgerufen wird, eine höhere Priorität haben, als die TASK, in der die Bausteine für die RDM-Befehle aufgerufen werden.

Zu beiden Betriebsarten finden sie auch entsprechende Beispiele im Anhang.

Anmerkung zu den IDs von DMX-Geräten

Jedes DMX-Gerät hat eine eindeutige, feste, 48-Bit lange Adresse, auch Unique ID oder kurz UID genannt. Diese Adresse besteht aus der Hersteller-Id (16 Bit) und der Geräte-Id (32 Bit). Die Hersteller-Id identifiziert den Hersteller des Gerätes und wird von der ESTA (Entertainment Services and Technology Association) vergeben. Eine Liste mit allen bekannten Hersteller-Ids ist zu finden unter http://www.esta.org/tsp/working_groups/CP/mfctrIDs.php. Die Geräte-Id wird frei vom Hersteller festgelegt. Hierdurch soll sichergestellt werden, dass jede UID weltweit einmalig vorhanden ist. Die UID kann in der Regel nicht verändert werden. Von der ESTA wurde Beckhoff Automation die Hersteller-Id 0x4241 zugeordnet. Da auch ein DMX-Master eine UID besitzt, sollte diese entsprechend der ESTA angegeben werden (Eingang *wSourceManufacturerId*).

VAR_INPUT

```
wSourceManufacturerId  : WORD := 16#42_41;
dwSourceDeviceId       : DWORD := 16#12_13_14_15;
bEnableSendingData     : BOOL := TRUE;
bSetCycleMode          : BOOL := TRUE;
bSendDefaultData       : BOOL;
uiDataLength           : UINT;
dwOptions              : DWORD;
```

wSourceManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät. Sollte gemäß der ESTA 0x4241 sein.

dwSourceDeviceId: Eindeutige Geräte-Id vom DMX-Gerät. Kann frei vergeben werden.

bEnableSendingData: Ist die Klemme im Cycle-Mode (Ausgang *bCycleMode* = TRUE), so kann das Versenden mit diesem Baustein aktiviert (TRUE) oder gesperrt (FALSE) werden.

bSetCycleMode: Aktiviert den Cycle-Mode. Im Cycle-Mode können die zyklischen Prozessdaten an die DMX-Geräte versendet werden. Zum Versenden der RDM-DMX-Befehle muss der Cycle-Mode deaktiviert werden.

bSendDefaultData: Ist dieser Eingang aktive (TRUE), so werden im Cycle-Mode die Standardwerte versendet.

uiDataLength: Dieser Eingang ist nur relevant, wenn der Cycle-Mode aktiv ist. Er gibt die Länge des DMX512-Frames in Bytes an.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

```
bError                      : BOOL;
udiErrorId                  : UDINT;
bCycleMode                  : BOOL;
byBufferDemandMeter         : BYTE;
byBufferMaximumDemandMeter  : BYTE;
uiBufferOverflowCounter     : UINT;
bLineIsBusy                 : BOOL;
```

bError: Dieser Ausgang wird auf TRUE geschaltet, wenn bei der Ausführung eines Befehls ein Fehler aufgetreten ist. Der befehlsspezifische Fehlercode ist in *udiErrorId* enthalten. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

bCycleMode: Ist auf TRUE, wenn der Cycle-Mode aktiv ist (siehe auch Eingang *bSetCycleMode*).

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

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

uiBufferOverflowCounter: Bisherige Anzahl der Pufferüberläufe.

bLineIsBusy: Solange der Baustein FB_EL6851CommunicationEx() DMX-RDM-Befehle bearbeitet, ist dieser Ausgang gesetzt.

VAR_IN_OUT

```
stEL6851InData          : ST_EL6851InDataEx;
stEL6851OutData         : ST_EL6851OutData;
stCommandBuffer         : ST_DMXCommandBuffer;
arrProcessData          : ARRAY[1..512] OF BYTE;
```

stEL6851InData: Struktur im Eingangsprozessabbild der EL6851. Sie dient zur Kommunikation von der EL6851 [▶ 76] zur SPS.

stEL6851OutData: Struktur im Ausgangsprozessabbild der EL6851. Sie dient zur Kommunikation von der SPS zur EL6851 [▶ 76].

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx()-Baustein.

arrProcessData: Über diese Variable werden dem Baustein die Daten übergeben, die zyklisch an die DMX-Geräte versendet werden sollen. Hierzu muss der CycleMode aktiv sein (siehe auch Eingang *bSetCycleMode*).

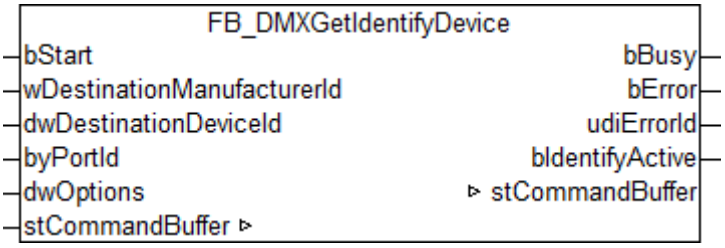
Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

Sehen Sie dazu auch

- ST_EL6851InDataEx [▶ 70]
- ST_EL6851OutData [▶ 70]
- ST_DMXCommandBuffer [▶ 66]

5.7 FB_DMXGetIdentifyDevice



Dieser Funktionsbaustein fragt ab, ob von einem DMX-Gerät die Identifizierung aktiv ist.

Durch eine positive Flanke an dem Eingang *bStart* wird der Baustein gestartet und der Ausgang *bBusy* geht auf TRUE. Die Eingänge *wDestinationManufacturerId* und *dwDestinationDeviceId* adressieren das DMX-Gerät. Der Eingang *byPortId* definiert den Kanal innerhalb des adressierten DMX-Gerätes. Ist die Ausführung des Befehls abgeschlossen, geht der Ausgang *bBusy* wieder auf FALSE. Die Ausgänge *bError*, *udiErrorId* und *bIdentifyActive* können jetzt ausgewertet werden. Solange der Baustein aktiv ist (*bBusy* ist TRUE) werden weitere positive Flanken am Eingang *bStart* ignoriert.

VAR_INPUT

```
bStart                      : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortID                    : BYTE;
dwOptions                   : DWORD := 0;
```

bStart: Über eine positive Flanke an diesem Eingang wird der Befehl gestartet.

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

```
bBusy            : BOOL;
bError           : BOOL;
udiErrorId       : UDINT;
bIdentifyActive  : 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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

bIdentifyActive: Ist die Abarbeitung des Befehls abgeschlossen (*bBusy* ist FALSE), so wird an diesem Ausgang der Zustand der Identifizierung von dem DMX-Gerät angezeigt.

VAR_IN_OUT

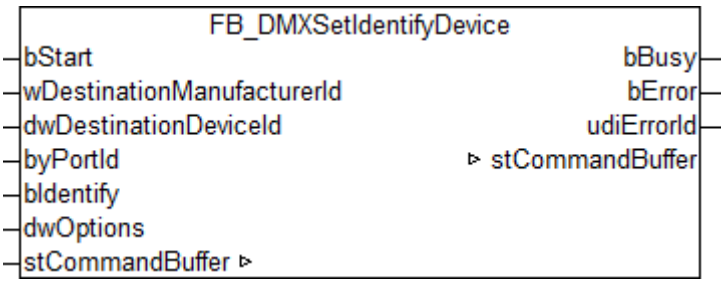
```
stCommandBuffer          : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

Sehen Sie dazu auch

ST_DMXCommandBuffer [▶ 66]

5.8 FB_DMXSetIdentifyDevice



Dieser Funktionsbaustein aktiviert oder deaktiviert die Identifizierung eines DMX-Gerätes.

Durch eine positive Flanke an dem Eingang *bStart* wird der Baustein gestartet und der Ausgang *bBusy* geht auf TRUE. Die Eingänge *wDestinationManufacturerId* und *dwDestinationDeviceId* adressieren das DMX-Gerät. Der Eingang *byPortId* definiert den Kanal innerhalb des adressierten DMX-Gerätes. Ist die Ausführung des Befehls abgeschlossen, so geht der Ausgang *bBusy* wieder auf FALSE. Die Ausgänge *bError* und *udiErrorId* können jetzt ausgewertet werden. Solange der Baustein aktiv ist (*bBusy* ist TRUE) werden weitere positive Flanken am Eingang *bStart* ignoriert.

VAR_INPUT

```
bStart                    : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
```

```
byPortId                    : BYTE;
bIdentify                   : BOOL := FALSE;
dwOptions                   : DWORD := 0;
```

bStart: Über eine positive Flanke an diesem Eingang wird der Befehl gestartet.

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

bIdentify: Legt fest, ob die Identifizierung aktiviert (TRUE) oder deaktiviert (FALSE) werden soll.

dwOptions: Optionen (wird derzeit nicht benutzt).

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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

VAR_IN_OUT

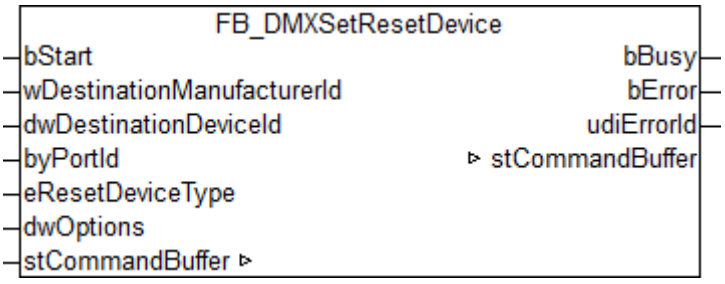
```
stCommandBuffer         : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

Sehen Sie dazu auch

ST_DMXCommandBuffer [▶ 66]

5.9 FB_DMXSetResetDevice



Dieser Funktionsbaustein aktiviert einen Reset bei einem DMX-Gerät.

Durch eine positive Flanke an dem Eingang *bStart* wird der Baustein gestartet und der Ausgang *bBusy* geht auf TRUE. Die Eingänge *wDestinationManufacturerId* und *dwDestinationDeviceId* adressieren das DMX-Gerät. Der Eingang *byPortId* definiert den Kanal innerhalb des adressierten DMX-Gerätes. Ist die Ausführung des Befehls abgeschlossen, geht der Ausgang *bBusy* wieder auf FALSE. Die Ausgänge *bError* und *udiErrorId* können jetzt ausgewertet werden. Solange der Baustein aktiv ist (*bBusy* ist TRUE) werden weitere positive Flanken am Eingang *bStart* ignoriert.

VAR_INPUT

```
bStart                      : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                    : BYTE;
eResetDeviceType            : E_DMXResetDeviceType := eDMXResetDeviceTypeWarm;
dwOptions                   : DWORD := 0;
```

bStart: Über eine positive Flanke an diesem Eingang wird der Befehl gestartet.

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

eResetDeviceType: Legt fest, ob ein Warmstart (eDMXResetDeviceTypeWarm) oder ein Kaltstart (eDMXResetDeviceTypeCold) ausgeführt werden soll. Andere Werte sind für diesen Eingang nicht möglich.

dwOptions: Optionen (wird derzeit nicht benutzt).

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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

VAR_IN_OUT

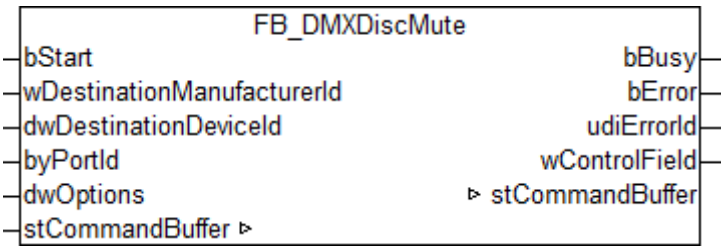
```
stCommandBuffer          : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

Sehen Sie dazu auch

- E_DMXResetDeviceType [▶ 62]
- ST_DMXCommandBuffer [▶ 66]

5.10 FB_DMXDiscMute



Dieser Funktionsbaustein setzt das Mute-Flag von einem DMX-Gerät. Das Mute-Flag legt fest, ob ein DMX-Gerät auf den Befehl FB_DMXDiscUniqueBranch() [▶ 30] reagiert (Mute-Flag ist nicht gesetzt) oder nicht (Mute-Flag ist gesetzt).

Durch eine positive Flanke an dem Eingang *bStart* wird der Baustein gestartet und der Ausgang *bBusy* geht auf TRUE. Die Eingänge *wDestinationManufacturerId* und *dwDestinationDeviceId* adressieren das DMX-Gerät. Der Eingang *byPortId* definiert den Kanal innerhalb des adressierten DMX-Gerätes. Ist die Ausführung des Befehls abgeschlossen, geht der Ausgang *bBusy* wieder auf FALSE. Die Ausgänge *bError*, *udiErrorId* und *wControlField* können jetzt ausgewertet werden. Solange der Baustein aktiv ist (*bBusy* ist TRUE) werden weitere positive Flanken am Eingang *bStart* ignoriert.

VAR_INPUT

```
bStart                      : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                    : BYTE;
dwOptions                   : DWORD := 0;
```

bStart: Über eine positive Flanke an diesem Eingang wird der Befehl gestartet.

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

```
bBusy               : BOOL;
bError              : BOOL;
udiErrorId          : UDINT;
wControlField       : 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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

wControlField: Ist die Abarbeitung des Befehls abgeschlossen (*bBusy* ist FALSE), so werden an diesem Ausgang weitere Informationen über das DMX-Gerät ausgegeben. Hierbei ist die Bedeutung der einzelnen Bits wie folgt definiert:

Bit	Beschreibung
0 - Managed Proxy Flag	Dieses Bit ist gesetzt, wenn das DMX-Gerät ein Proxygerät ist.
1 - Sub-Device Flag	Dieses Bit ist gesetzt, wenn das DMX-Gerät Untergeräte (Sub-Devices) unterstützt.
2 - Boot-Loader Flag	Dieses Bit ist gesetzt, wenn das DMX-Gerät keine Befehle empfangen kann (z.B. während die Firmware geladen wird).
3 - Proxied Device Flag	Dieses Bit ist gesetzt, wenn die Antwort von einem Proxygerät gesendet wurde.
4 - 15	Reserve (immer 0).

VAR_IN_OUT

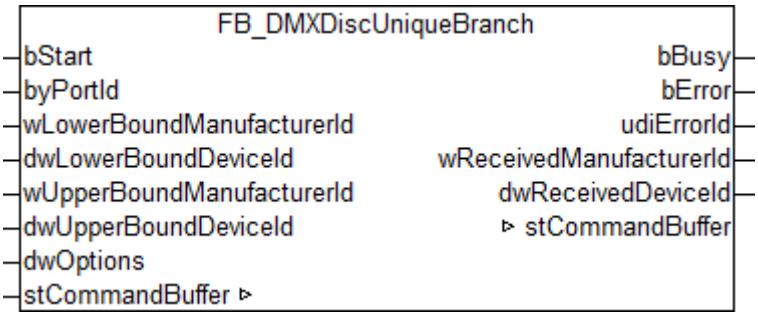
```
stCommandBuffer          : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

Sehen Sie dazu auch

ST_DMXCommandBuffer [▶ 66]

5.11 FB_DMXDiscUniqueBranch



Dieser Funktionsbaustein fragt ab, ob sich innerhalb eines bestimmten Adressbereichs DMX-Geräte befinden. Dieser Befehl findet Verwendung beim Ermitteln (Discovery) der angeschlossenen DMX-Geräte.

Durch eine positive Flanke am Eingang *bStart* wird der Baustein gestartet und der Ausgang *bBusy* geht auf TRUE. Die Eingänge *wLowerBoundManufacturerId*, *dwLowerBoundDeviceId*, *wUpperBoundManufacturerId* und *dwUpperBoundDeviceId* definieren den Adressbereich, in dem DMX-Geräte gesucht werden. Der Eingang *byPortId* definiert den Kanal innerhalb des adressierten DMX-Gerätes. Ist die Ausführung des Befehls abgeschlossen, geht der Ausgang *bBusy* wieder auf FALSE. Die Ausgänge *bError*, *udiErrorId*, *wReceivedManufacturerId* und *dwReceivedDeviceId* können jetzt ausgewertet werden. Solange der Baustein aktiv ist (*bBusy* ist TRUE) werden weitere positive Flanken am Eingang *bStart* ignoriert.

Liegt im definierten Adressbereich nur ein DMX-Gerät, so werden über die Ausgänge *wReceivedManufacturerId* und *dwReceivedDeviceId* die 48-Bit UID des DMX-Gerätes zurückgegeben. Liegen in diesem Bereich keine DMX-Geräte, so ist der Ausgang *bError* auf TRUE und *udiErrorId* ist auf 0x8002 (keine Antwort vom DMX-Gerät). Befinden sich in dem Adressbereich zwei oder mehr DMX-Geräte, so ist *bError* auf TRUE und *udiErrorId* enthält eine 0x8006 (Checksummenfehler).

VAR_INPUT

```
bStart                  : BOOL;
byPortId                : BYTE;
wLowerBoundManufacturerId  : WORD;
dwLowerBoundDeviceId       : DWORD;
wUpperBoundManufacturerId  : WORD;
dwUpperBoundDeviceId       : DWORD;
dwOptions               : DWORD := 0;
```

bStart: Über eine positive Flanke an diesem Eingang wird der Befehl gestartet.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

wLowerBoundManufacturerId: Eindeutige Hersteller-Id vom unteren Adressbereichs.

dwLowerBoundDeviceId: Eindeutige Geräte-Id vom unteren Adressbereichs.

wUpperBoundManufacturerId: Eindeutige Hersteller-Id vom oberen Adressbereichs.

dwUpperBoundDeviceId: Eindeutige Geräte-Id vom oberen Adressbereichs.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

```
bBusy                   : BOOL;
bError                  : BOOL;
udiErrorId              : UDINT;
wReceivedManufacturerId  : WORD;
dwReceivedDeviceId       : 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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

wReceivedManufacturerId: Ist die Abarbeitung des Befehls abgeschlossen (*bBusy* ist FALSE), so wird an diesem Ausgang der Zustand der Identifizierung von dem DMX-Gerät angezeigt.

dwReceivedDeviceId: Ist die Abarbeitung des Befehls abgeschlossen (*bBusy* ist FALSE), so wird an diesem Ausgang der Zustand der Identifizierung von dem DMX-Gerät angezeigt.

VAR_IN_OUT

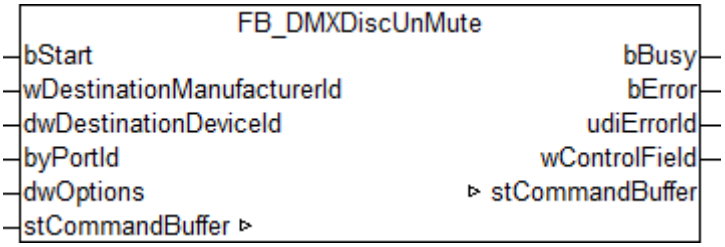
```
stCommandBuffer         : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

Sehen Sie dazu auch

- ST_DMXCommandBuffer [▶ 66]

5.12 FB_DMXDiscUnMute



Dieser Funktionsbaustein setzt das Mute-Flag von einem DMX-Gerät zurück. Das Mute-Flag legt fest, ob ein DMX-Gerät auf den Befehl FB_DMXDiscUniqueBranch() [▶ 30] reagiert (Mute-Flag ist nicht gesetzt) oder nicht (Mute-Flag ist gesetzt).

Durch eine positive Flanke an dem Eingang *bStart* wird der Baustein gestartet und der Ausgang *bBusy* geht auf TRUE. Die Eingänge *wDestinationManufacturerId* und *dwDestinationDeviceId* adressieren das DMX-Gerät. Der Eingang *byPortId* definiert den Kanal innerhalb des adressierten DMX-Gerätes. Ist die Ausführung des Befehls abgeschlossen, geht der Ausgang *bBusy* wieder auf FALSE. Die Ausgänge *bError*, *udiErrorId* und *wControlField* können jetzt ausgewertet werden. Solange der Baustein aktiv ist (*bBusy* ist TRUE) werden weitere positive Flanken am Eingang *bStart* ignoriert.

VAR_INPUT

```
bStart                      : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                    : BYTE;
dwOptions                   : DWORD := 0;
```

bStart: Über eine positive Flanke an diesem Eingang wird der Befehl gestartet.

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

```
bBusy            : BOOL;
bError           : BOOL;
udiErrorId       : UDINT;
wControlField    : 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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

wControlField: Ist die Abarbeitung des Befehls abgeschlossen (*bBusy* ist FALSE), so werden an diesem Ausgang weitere Informationen über das DMX-Gerät ausgegeben. Hierbei ist die Bedeutung der einzelnen Bits wie folgt definiert:

Bit	Beschreibung
0 - Managed Proxy Flag	Dieses Bit ist gesetzt, wenn das DMX-Gerät ein Proxygerät ist.
1 - Sub-Device Flag	Dieses Bit ist gesetzt, wenn das DMX-Gerät Untergeräte (Sub-Devices) unterstützt.
2 - Boot-Loader Flag	Dieses Bit ist gesetzt, wenn das DMX-Gerät keine Befehle empfangen kann (z.B. während die Firmware geladen wird).
3 - Proxied Device Flag	Dieses Bit ist gesetzt, wenn die Antwort von einem Proxygerät gesendet wurde.
4 - 15	Reserve (immer 0).

VAR_IN_OUT

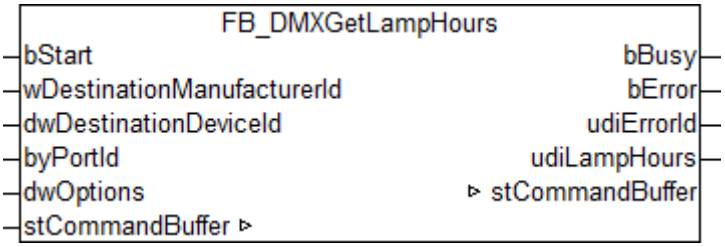
```
stCommandBuffer          : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

Sehen Sie dazu auch

ST_DMXCommandBuffer [▶ 66]

5.13 FB_DMXGetLampHours



Dieser Funktionsbaustein liest die Anzahl der Stunden aus, in denen die Lampe an war. Durch den Baustein FB_DMXSetLampHours() [▶ 35] kann dieser Stundenzähler verändert werden.

VAR_INPUT

```
bStart                      : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                    : BYTE;
dwOptions                   : DWORD := 0;
```

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

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

```
bBusy               : BOOL;
bError              : BOOL;
udiErrorId          : UDINT;
udiLampHours        : 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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

udiLampHours: Anzahl der Stunden, in denen die Lampe eingeschaltet war.

VAR_IN_OUT

```
stCommandBuffer         : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

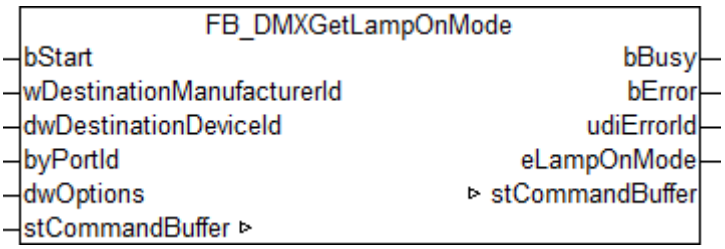
Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2251	PC/CX	TcDMX-Bibliothek ab V1.2.0

Sehen Sie dazu auch

ST_DMXCommandBuffer [▶ 66]

5.14 FB_DMXGetLampOnMode



Dieser Funktionsbaustein liest den Parameter aus, der das Einschaltverhalten des DMX-Gerätes definiert. Mit dem Baustein FB_DMXSetLampOnMode() [▶ 36] kann der Wert verändert werden.

VAR_INPUT

```
bStart                      : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                    : BYTE;
dwOptions                   : DWORD := 0;
```

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

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

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

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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

eLampOnMode: Enthält den aktuellen Parameter, der das Einschaltverhalten des DMX-Gerätes definiert.

VAR_IN_OUT

```
stCommandBuffer         : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

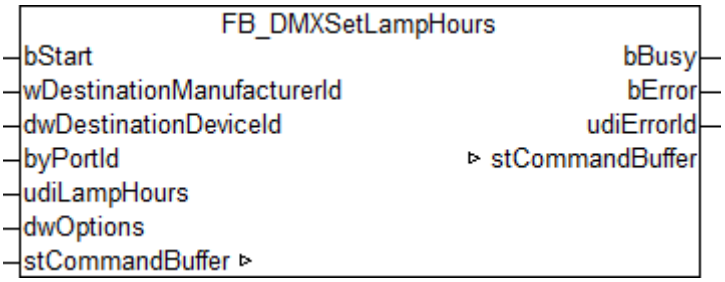
Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2251	PC/CX	TcDMX-Bibliothek ab V1.2.0

Sehen Sie dazu auch

- E_DMXLampOnMode [▶ 59]
- ST_DMXCommandBuffer [▶ 66]

5.15 FB_DMXSetLampHours



Dieser Funktionsbaustein setzt den Betriebsstundenzähler der Lampe. Durch den Baustein FB_DMXGetLampHours() [▶ 32] kann der Zähler ausgelesen werden.

VAR_INPUT

```
bStart                      : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                    : BYTE;
udiLampHours                : UDINT := 0;
dwOptions                   : DWORD := 0;
```

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

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

udiLampHours: Neuer Wert für den Betriebsstundenzähler.

dwOptions: Optionen (wird derzeit nicht benutzt).

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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

VAR_IN_OUT

```
stCommandBuffer         : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

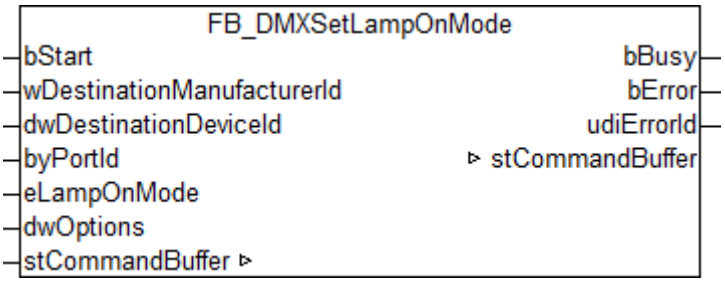
Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2251	PC/CX	TcDMX-Bibliothek ab V1.2.0

Sehen Sie dazu auch

ST_DMXCommandBuffer [▶ 66]

5.16 FB_DMXSetLampOnMode



Dieser Funktionsbaustein legt das Einschaltverhalten des DMX-Gerätes fest. Mit dem Baustein FB_DMXGetLampOnMode() [▶ 33] kann der eingestellte Wert ausgelesen werden.

VAR_INPUT

```
bStart                      : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                    : BYTE;
eLampOnMode                 : E_DMXLampOnMode := eDMXLampOnModeOff;
dwOptions                   : DWORD := 0;
```

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

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

eLampOnMode: Dieser Parameter legt das Einschaltverhalten des DMX-Gerätes fest.

dwOptions: Optionen (wird derzeit nicht benutzt).

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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

VAR_IN_OUT

```
stCommandBuffer         : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

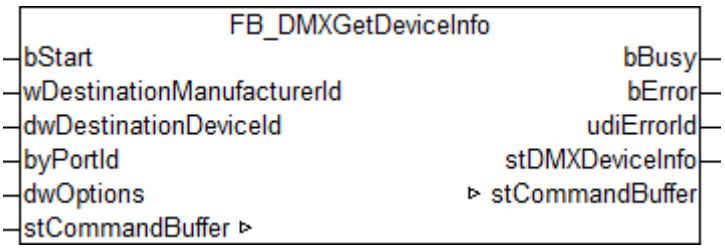
Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2251	PC/CX	TcDMX-Bibliothek ab V1.2.0

Sehen Sie dazu auch

- E_DMXLampOnMode [▶ 59]
- ST_DMXCommandBuffer [▶ 66]

5.17 FB_DMXGetDeviceInfo



Dieser Funktionsbaustein fragt alle relevanten Informationen von einem DMX-Gerät ab.

Durch eine positive Flanke an dem Eingang *bStart* wird der Baustein gestartet und der Ausgang *bBusy* geht auf TRUE. Die Eingänge *wDestinationManufacturerId* und *dwDestinationDeviceId* adressieren das DMX-Gerät. Der Eingang *byPortId* definiert den Kanal innerhalb des adressierten DMX-Gerätes. Ist die Ausführung des Befehls abgeschlossen, geht der Ausgang *bBusy* wieder auf FALSE. Die Ausgänge *bError*, *udiErrorId* und *stDMXDeviceInfo* können jetzt ausgewertet werden. Solange der Baustein aktiv ist (*bBusy* ist TRUE) werden weitere positive Flanken am Eingang *bStart* ignoriert.

VAR_INPUT

```
bStart                      : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                    : BYTE;
dwOptions                   : DWORD := 0;
```

bStart: Über eine positive Flanke an diesem Eingang wird der Befehl gestartet.

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

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

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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

stDMXDeviceInfo: Ist die Abarbeitung des Befehls abgeschlossen (*bBusy* ist FALSE), so werden an diesem Ausgang in einer Struktur alle relevanten Informationen des DMX-Gerätes ausgegeben. Siehe auch Beschreibung der Struktur ST_DMXDeviceInfo [▶ 66].

uliUID: Eindeutige 48-Bit Adresse des DMX-Gerätes. Die unteren 32 Bit definieren die Geräte-Id und die oberen 16 Bit die Hersteller-Id.

tRDMProtocolVersion: Versionsnummer des unterstützen RDM-Standard.

uiDeviceModelId: Die Gerätemodel-Id wird von dem Gerätehersteller definiert.

stProductCategory: Gerätetyp des DMX-Gerätes. Die Gerätetypen sind in der RDM-Norm definiert.

udiSoftwareVersionId: Versionsnummer der Software (Firmware) in dem DMX-Gerät.

uiDMX512Footprint: Anzahl der DMX512-Slots, die das Gerät belegt. Jedes Untergerät (Sub-Device) und das Hauptgerät (Root-Device) belegen eigene DMX512-Slots.

stDMX512Personality: DMX-Geräte können verschiedene *Personalities* (Persönlichkeiten) besitzen. *Personalities* sind mit Profilen vergleichbar. Das aktuelle Profil und die Anzahl der Profile wird in diesem Element angezeigt.

uiDMX512StartAddress: Die DMX512-Startadresse. Diese liegt im Bereich von 1 bis 512. Ist der Parameter *uiDMX512Footprint* 0, so ist die DMX512-Startadresse 0xFFFF (65535). Jedes Untergerät (Sub-Device) und das Hauptgerät (Root-Device) belegen unterschiedliche DMX512-Startadressen.

uiSubDeviceCount: Anzahl der Untergeräte (Sub-Devices).

bySensorCount: Anzahl der Sensoren, die in dem Untergerät (Sub-Device) oder dem Hauptgerät (Root-Device) enthalten sind.

VAR_IN_OUT

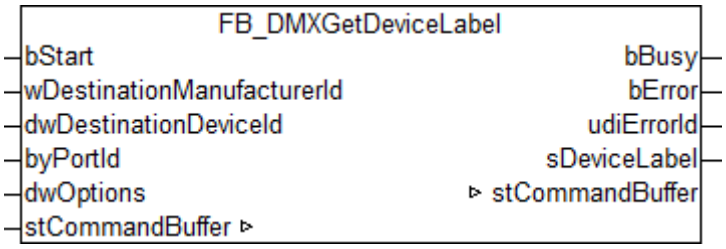
```
stCommandBuffer          : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

Sehen Sie dazu auch

- ST_DMXCommandBuffer [▶ 66]

5.18 FB_DMXGetDeviceLabel



Dieser Funktionsbaustein liest aus dem DMX-Gerät einen Text aus, der das Gerät genauer beschreibt. Mit dem Baustein FB_DMXSetDeviceLabel() [▶ 44] kann der Text verändert werden.

VAR_INPUT

```
bStart                   : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId    : DWORD;
byPortId                 : BYTE;
dwOptions                : DWORD := 0;
```

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

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

```
bBusy            : BOOL;
bError           : BOOL;
udiErrorId       : UDINT;
sDeviceLabel     : 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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

sDeviceLabel: Beschreibungstext des DMX-Gerätes.

VAR_IN_OUT

```
stCommandBuffer          : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

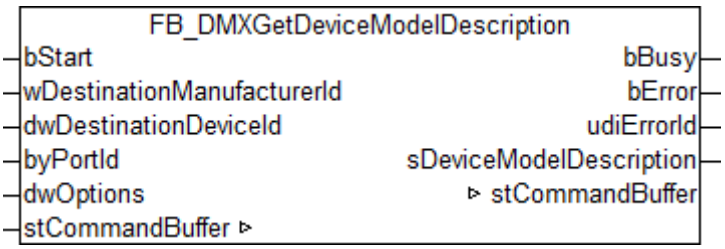
Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2251	PC/CX	TcDMX-Bibliothek ab V1.2.0

Sehen Sie dazu auch

ST_DMXCommandBuffer [▶ 66]

5.19 FB_DMXGetDeviceModelDescription



Dieser Funktionsbaustein fragt die Beschreibung des Gerätetyps ab.

Durch eine positive Flanke an dem Eingang *bStart* wird der Baustein gestartet und der Ausgang *bBusy* geht auf TRUE. Die Eingänge *wDestinationManufacturerId* und *dwDestinationDeviceId* adressieren das DMX-Gerät. Der Eingang *byPortId* definiert den Kanal innerhalb des adressierten DMX-Gerätes. Ist die

Ausführung des Befehls abgeschlossen, geht der Ausgang *bBusy* wieder auf FALSE. Die Ausgänge *bError*, *udiErrorId* und *sDeviceModelDescription* können jetzt ausgewertet werden. Solange der Baustein aktiv ist (*bBusy* ist TRUE) werden weitere positive Flanken am Eingang *bStart* ignoriert.

VAR_INPUT

```
bStart                      : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                    : BYTE;
dwOptions                   : DWORD := 0;
```

bStart: Über eine positive Flanke an diesem Eingang wird der Befehl gestartet.

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

```
bBusy                   : BOOL;
bError                  : BOOL;
udiErrorId              : UDINT;
sDeviceModelDescription : 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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

sDeviceModelDescription: Ist die Abarbeitung des Befehls abgeschlossen (*bBusy* ist FALSE), so wird an diesem Ausgang eine Beschreibung (maximal 32 Zeichen) des Gerätetyps ausgegeben. Der Inhalt wird vom DMX-Gerätehersteller festgelegt.

VAR_IN_OUT

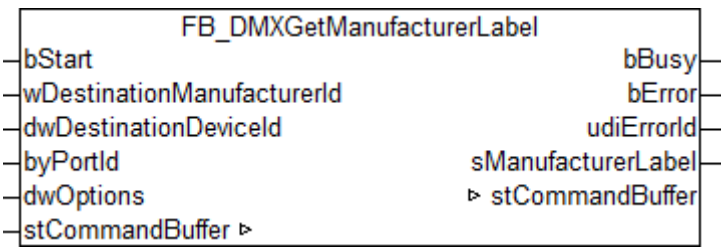
```
stCommandBuffer         : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

Sehen Sie dazu auch

ST_DMXCommandBuffer [▶ 66]

5.20 FB_DMXGetManufacturerLabel



Dieser Funktionsbaustein fragt die Beschreibung des DMX-Geräteherstellers ab.

Durch eine positive Flanke an dem Eingang *bStart* wird der Baustein gestartet und der Ausgang *bBusy* geht auf TRUE. Die Eingänge *wDestinationManufacturerId* und *dwDestinationDeviceId* adressieren das DMX-Gerät. Der Eingang *byPortId* definiert den Kanal innerhalb des adressierten DMX-Gerätes. Ist die Ausführung des Befehls abgeschlossen, geht der Ausgang *bBusy* wieder auf FALSE. Die Ausgänge *bError*, *udiErrorId* und *sManufacturerLabel* können jetzt ausgewertet werden. Solange der Baustein aktiv ist (*bBusy* ist TRUE) werden weitere positive Flanken am Eingang *bStart* ignoriert.

VAR_INPUT

```
bStart                      : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                    : BYTE;
dwOptions                   : DWORD := 0;
```

bStart: Über eine positive Flanke an diesem Eingang wird der Befehl gestartet.

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

```
bBusy               : BOOL;
bError              : BOOL;
udiErrorId          : UDINT;
sManufacturerLabel  : 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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

sManufacturerLabel: Ist die Abarbeitung des Befehls abgeschlossen (*bBusy* ist FALSE), so wird an diesem Ausgang eine Beschreibung (maximal 32 Zeichen) des DMX-Geräteherstellers ausgegeben. Der Inhalt wird vom DMX-Gerätehersteller festgelegt.

VAR_IN_OUT

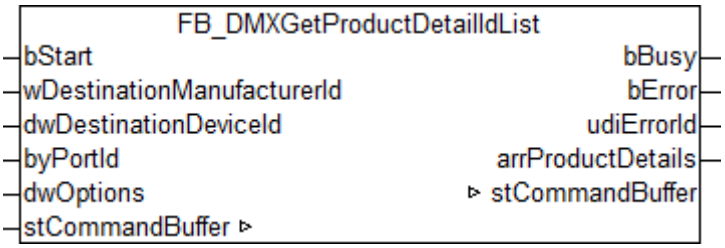
```
stCommandBuffer          : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

Sehen Sie dazu auch

ST_DMXCommandBuffer [▶ 66]

5.21 FB_DMXGetProductDetailIdList



Dieser Funktionsbaustein fragt die Kategorien ab, zu denen das DMX-Gerät zugehörig ist.

RDM definiert verschiedene Gerätekategorien. Jedes DMX-Gerät kann bis zu 6 Kategorien zugeordnet werden. Die Zuordnung erfolgt durch den Gerätehersteller und kann nicht per RDM geändert werden.

VAR_INPUT

```
bStart                      : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                    : BYTE;
dwOptions                   : DWORD := 0;
```

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

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

```
bBusy               : BOOL;
bError              : BOOL;
udiErrorId          : UDINT;
arrProductDetails   : ARRAY[1..6] OF E_DMXProductDetail;
```

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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

arrProductDetails: Enthält die Liste mit bis zu 6 Gerätekategorien.

VAR_IN_OUT

```
stCommandBuffer         : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

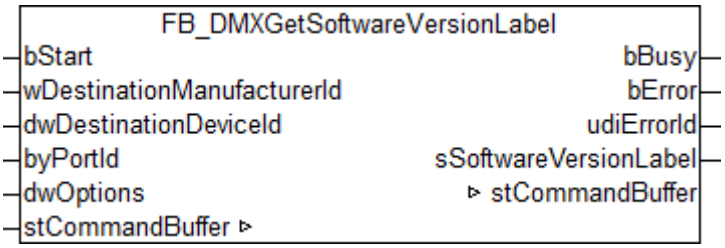
Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2251	PC/CX	TcDMX-Bibliothek ab V1.2.0

Sehen Sie dazu auch

- E_DMXProductDetail [▶ 61]
- ST_DMXCommandBuffer [▶ 66]

5.22 FB_DMXGetSoftwareVersionLabel



Dieser Funktionsbaustein fragt die Beschreibung der Softwareversion des DMX-Gerätes ab.

Durch eine positive Flanke an dem Eingang *bStart* wird der Baustein gestartet und der Ausgang *bBusy* geht auf TRUE. Die Eingänge *wDestinationManufacturerId* und *dwDestinationDeviceId* adressieren das DMX-Gerät. Der Eingang *byPortId* definiert den Kanal innerhalb des adressierten DMX-Gerätes. Ist die Ausführung des Befehls abgeschlossen, geht der Ausgang *bBusy* wieder auf FALSE. Die Ausgänge *bError*, *udiErrorId* und *sSoftwareVersionLabel* können jetzt ausgewertet werden. Solange der Baustein aktiv ist (*bBusy* ist TRUE) werden weitere positiven Flanke am Eingang *bStart* ignoriert.

VAR_INPUT

```
bStart                  : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                : BYTE;
dwOptions               : DWORD := 0;
```

bStart: Über eine positive Flanke an diesem Eingang wird der Befehl gestartet.

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

```
bBusy                : BOOL;
bError               : BOOL;
udiErrorId           : UDINT;
sSoftwareVersionLabel  : 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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

sSoftwareVersionLabel: Ist die Abarbeitung des Befehls abgeschlossen (*bBusy* ist FALSE), so wird an diesem Ausgang eine Beschreibung (maximal 32 Zeichen) der Softwareversion des DMX-Gerätes ausgegeben. Der Inhalt wird vom DMX-Gerätehersteller festgelegt.

VAR_IN_OUT

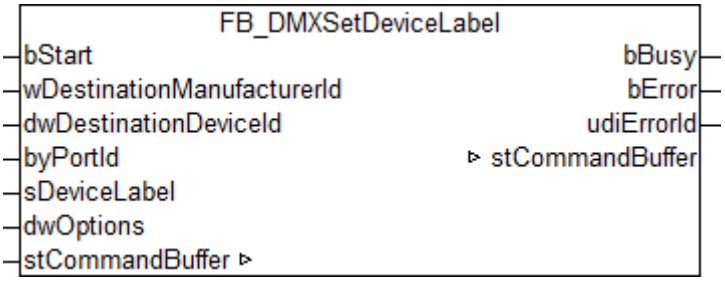
```
stCommandBuffer          : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

Sehen Sie dazu auch

ST_DMXCommandBuffer [▶ 66]

5.23 FB_DMXSetDeviceLabel



Dieser Funktionsbaustein schreibt in das DMX-Gerät einen Beschreibungstext. Mit dem Baustein FB_DMXGetDeviceLabel() [▶ 38] kann der Text ausgelesen werden.

VAR_INPUT

```
bStart                       : BOOL;
wDestinationManufacturerId   : WORD;
dwDestinationDeviceId        : DWORD;
byPortId                     : BYTE;
sDeviceLabel                 : STRING := '';
dwOptions                    : DWORD := 0;
```

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

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

sDeviceLabel: Neuer Beschreibungstext für das DMX-Gerät.

dwOptions: Optionen (wird derzeit nicht benutzt).

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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

VAR_IN_OUT

```
stCommandBuffer          : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

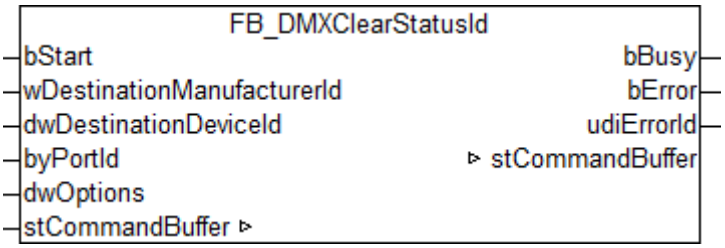
Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2251	PC/CX	TcDMX-Bibliothek ab V1.2.0

Sehen Sie dazu auch

ST_DMXCommandBuffer [▶ 66]

5.24 FB_DMXClearStatusId



Dieser Funktionsbaustein löscht den Nachrichtenbuffer im DMX-Gerät.

VAR_INPUT

```
bStart                      : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                    : BYTE;
dwOptions                   : DWORD := 0;
```

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

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

dwOptions: Optionen (wird derzeit nicht benutzt).

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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

VAR_IN_OUT

```
stCommandBuffer         : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

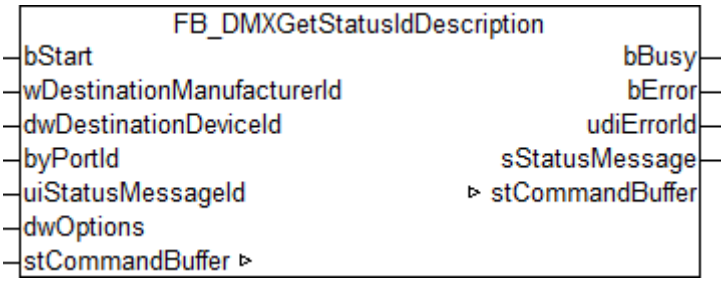
Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2251	PC/CX	TcDMX-Bibliothek ab V1.2.0

Sehen Sie dazu auch

ST_DMXCommandBuffer [▶ 66]

5.25 FB_DMXGetStatusIdDescription



Dieser Funktionsbaustein liest den Text einer bestimmten Status-Id aus dem DMX-Gerät aus.

RDM definiert einige Standardmeldungen. Jede dieser Meldungen hat eine eindeutige Status-Id. Der zugehörige Text kann mit diesem Baustein aus dem DMX-Gerät ausgelesen werden.

VAR_INPUT

```
bStart                      : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                    : BYTE;
uiStatusMessageId           : UINT := 1;
dwOptions                   : DWORD := 0;
```

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

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

uiStatusMessageId: Status-Id zu dem der Text ausgelesen werden soll.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

```
bBusy               : BOOL;
bError              : BOOL;
udiErrorId          : UDINT;
sStatusMessage      : 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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

sStatusMessage: Statusmeldung.

VAR_IN_OUT

```
stCommandBuffer          : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

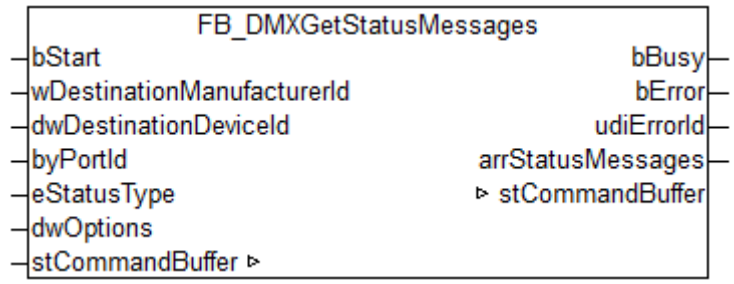
Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2251	PC/CX	TcDMX-Bibliothek ab V1.2.0

Sehen Sie dazu auch

- ST_DMXCommandBuffer [▶ 66]

5.26 FB_DMXGetStatusMessages



Dieser Funktionsbaustein sammelt die Status- oder Fehlerinformation von meinem DMX-Gerät.

VAR_INPUT

```
bStart                      : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                    : BYTE;
eStatusType                 : E_DMXStatusType := eDMXStatusTypeNone;
dwOptions                   : DWORD := 0;
```

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

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

eStatusType: Statustyp.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

```
bBusy              : BOOL;
bError             : BOOL;
udiErrorId         : UDINT;
arrStatusMessages  : ARRAY[0..24] OF ST_DMXStatusMessage;
```

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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

arrStatusMessages: Wenn die Ausführung des Befehls abgeschlossen wurde (*bBusy* ist FALSE), dann stehen alle Informationen zum Status/Fehler an diesem Ausgang an.

VAR_IN_OUT

```
stCommandBuffer           : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

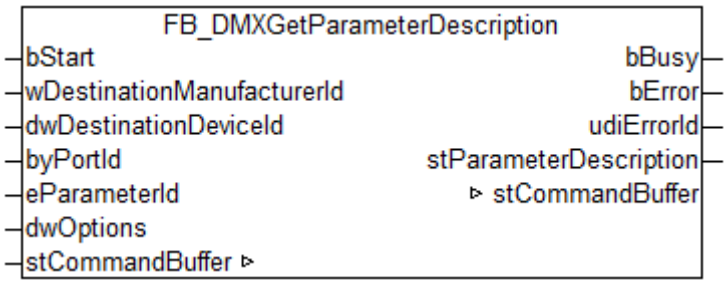
Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

Sehen Sie dazu auch

- E_DMXStatusType [▶ 65]
- ST_DMXStatusMessage [▶ 69]
- ST_DMXCommandBuffer [▶ 66]

5.27 FB_DMXGetParameterDescription



Dieser Funktionsbaustein fragt die Definition von herstellerspezifischen PIDs ab.

VAR_INPUT

```
bStart                    : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                  : BYTE;
eParameterId              : E_DMXParameterId;
dwOptions                 : DWORD := 0;
```

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

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

eParameterId: Angeforderte herstellerspezifische PID.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

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

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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

stParameterDescription: Wenn die Ausführung des Befehls abgeschlossen wurde (*bBusy* ist FALSE), dann stehen die Informationen zum PID an diesem Ausgang an.

VAR_IN_OUT

```
stCommandBuffer           : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

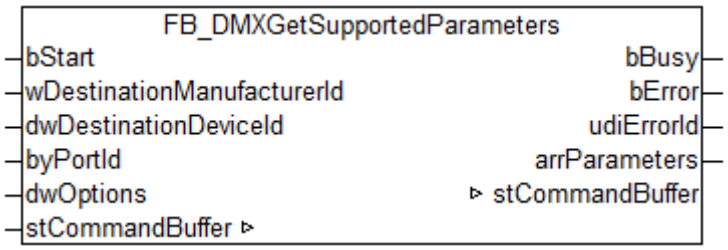
Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

Sehen Sie dazu auch

- E_DMXParameterId [▶ 59]
- ST_DMXParameterDescription [▶ 67]
- ST_DMXCommandBuffer [▶ 66]

5.28 FB_DMXGetSupportedParameters



Dieser Funktionsbaustein fragt alle unterstützen Parameter von einem DMX-Gerät ab.

VAR_INPUT

```
bStart                      : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                    : BYTE;
dwOptions                   : DWORD := 0;
```

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

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

```
bBusy             : BOOL;
bError            : BOOL;
udiErrorId        : UDINT;
arrParameters     : ARRAY[0..114] OF E_DMXParameterId;
```

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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

arrParameters: Wenn die Ausführung des Befehls abgeschlossen wurde (*bBusy* ist FALSE), dann stehen alle unterstützen Parameter für das DMX-Gerät an diesem Ausgang an.

VAR_IN_OUT

```
stCommandBuffer           : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

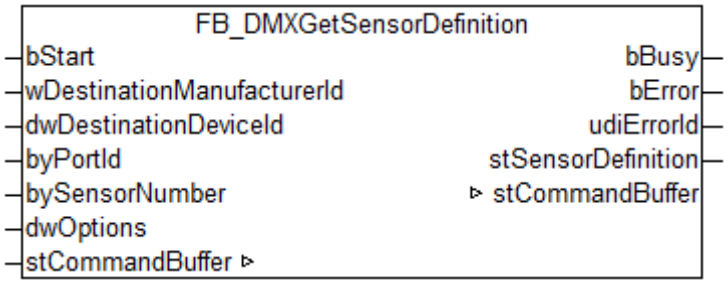
Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

Sehen Sie dazu auch

- E_DMXParameterId [▶ 59]
- ST_DMXCommandBuffer [▶ 66]

5.29 FB_DMXGetSensorDefinition



Dieser Funktionsbaustein fragt die Definition eines bestimmten Sensors ab.

VAR_INPUT

```
bStart                      : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                    : BYTE;
bySensorNumber              : BYTE := 0;
dwOptions                   : DWORD := 0;
```

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

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

bySensorNumber: DMX512 Sensornummer (0 - 254).

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

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

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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

stSensorDefinition: Wenn die Ausführung des Befehls abgeschlossen wurde (*bBusy* ist FALSE), dann steht die Definition des Sensors an diesem Ausgang an.

VAR_IN_OUT

```
stCommandBuffer           : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

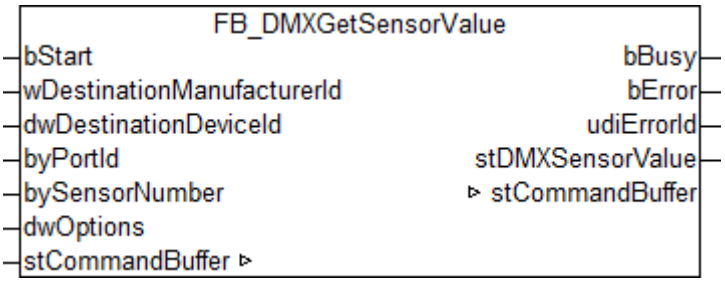
Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

Sehen Sie dazu auch

- ST_DMXSensorDefinition [▶ 68]
- ST_DMXCommandBuffer [▶ 66]

5.30 FB_DMXGetSensorValue



Dieser Funktionsbaustein fragt den aktuellen Wert eines Sensors ab.

VAR_INPUT

```
bStart                      : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                    : BYTE;
bySensorNumber              : BYTE := 0;
dwOptions                   : DWORD := 0;
```

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

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

bySensorNumber: Nummer des Sensors, dessen Werte abgefragt werden.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

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

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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

stDMXSensorValue: Struktur mit Informationen über den aktuellen Zustand des Sensors.

VAR_IN_OUT

```
stCommandBuffer         : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

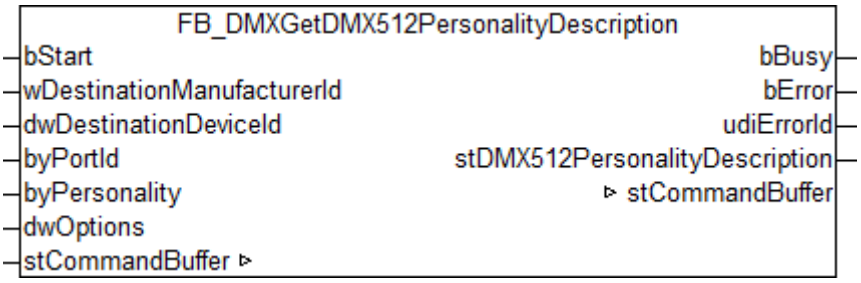
Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2251	PC/CX	TcDMX-Bibliothek ab V1.2.0

Sehen Sie dazu auch

- ST_DMXSensorValue [▶ 69]
- ST_DMXCommandBuffer [▶ 66]

5.31 FB_DMXGetDMX512PersonalityDescription



Dieser Funktionsbaustein liest aus dem DMX-Gerät weitere Informationen einer *Personality* aus. Manche DMX-Geräte unterstützen sogenannte *Personalities*. Ein verändern der *Personality* kann Einfluß auf bestimmte RDM-Parameter haben.

VAR_INPUT

```
bStart                    : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                  : BYTE;
byPersonality             : BYTE := 0;
dwOptions                 : DWORD := 0;
```

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

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

byPersonality: Die *Personality* dessen Informationen abgefragt werden.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

```
bBusy                          : BOOL;
bError                         : BOOL;
udiErrorId                     : UDINT;
stDMX512PersonalityDescription : ST_DMX512PersonalityDescription;
```

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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

stDMX512PersonalityDescription: Struktur mit Informationen der *Personality*.

VAR_IN_OUT

```
stCommandBuffer        : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

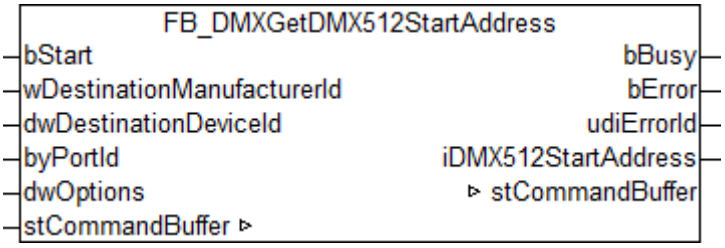
Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2251	PC/CX	TcDMX-Bibliothek ab V1.2.0

Sehen Sie dazu auch

- ST_DMX512PersonalityDescription [▶ 65]
- ST_DMXCommandBuffer [▶ 66]

5.32 FB_DMXGetDMX512StartAddress



Dieser Funktionsbaustein erfragt die DMX512-Startadresse. Diese liegt im Bereich von 1 bis 512. Belegt das DMX-Gerät keinen DMX-Slot, so ist die DMX512-Startadresse 0xFFFF (65535). Jedes Untergerät (Sub-Device) und das Hauptgerät (Root-Device) belegen unterschiedliche DMX512-Startadressen.

Durch eine positive Flanke an dem Eingang *bStart* wird der Baustein gestartet und der Ausgang *bBusy* geht auf TRUE. Die Eingänge *wDestinationManufacturerId* und *dwDestinationDeviceId* adressieren das DMX-Gerät. Der Eingang *byPortId* definiert den Kanal innerhalb des adressierten DMX-Gerätes. Ist die Ausführung des Befehls abgeschlossen, geht der Ausgang *bBusy* wieder auf FALSE. Die Ausgänge *bError*, *udiErrorId* und *iDMX512StartAddress* können jetzt ausgewertet werden. Solange der Baustein aktiv ist (*bBusy* ist TRUE) werden weitere positive Flanken am Eingang *bStart* ignoriert.

VAR_INPUT

```
bStart                      : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                    : BYTE;
dwOptions                   : DWORD := 0;
```

bStart: Über eine positive Flanke an diesem Eingang wird der Befehl gestartet.

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

```
bBusy               : BOOL;
bError              : BOOL;
udiErrorId          : UDINT;
iDMX512StartAddress : INT;
```

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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

iDMX512StartAddress: Ist die Abarbeitung des Befehls abgeschlossen (*bBusy* ist FALSE), so wird an diesem Ausgang die DMX512-Startadresse des DMX-Gerätes ausgegeben.

VAR_IN_OUT

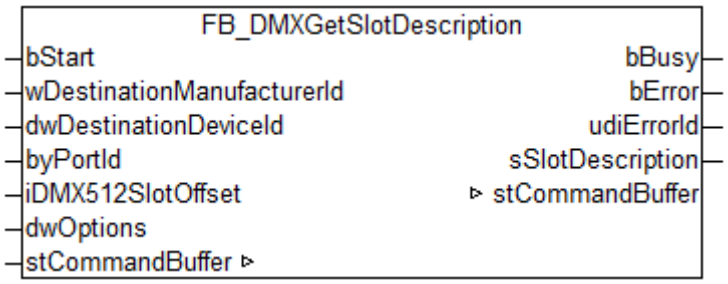
```
stCommandBuffer          : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

Sehen Sie dazu auch

ST_DMXCommandBuffer [▶ 66]

5.33 FB_DMXGetSlotDescription



Dieser Funktionsbaustein fragt die Textbeschreibung für Slot Offsets ab.

VAR_INPUT

```
bStart                      : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                    : BYTE;
iDMX512SlotOffset           : INT := 0;
dwOptions                   : DWORD := 0;
```

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

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

iDMX512SlotOffset: DMX512 Slot Offset (0 - 511).

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

```
bBusy             : BOOL;
bError            : BOOL;
udiErrorId        : UDINT;
sSlotDescription  : 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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

sSlotDescription: Wenn die Ausführung des Befehls abgeschlossen wurde (*bBusy* ist FALSE), dann steht die Beschreibung (maximal 32 Zeichen) vom Slot an diesem Ausgang an.

VAR_IN_OUT

```
stCommandBuffer           : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

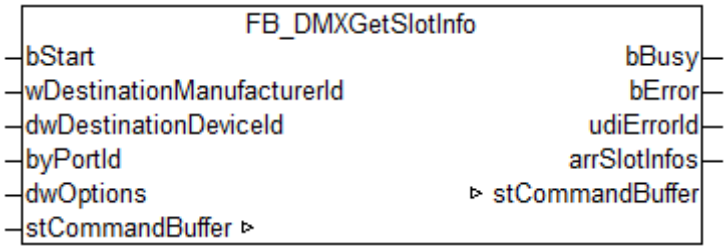
Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

Sehen Sie dazu auch

ST_DMXCommandBuffer [▶ 66]

5.34 FB_DMXGetSlotInfo



Dieser Funktionsbaustein fragt die Basisinformationen zur Funktionalität der DMX512 Slots eines DMX-Gerätes ab.

VAR_INPUT

```
bStart                    : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                  : BYTE;
dwOptions                 : DWORD := 0;
```

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

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: Eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

dwOptions: Optionen (wird derzeit nicht benutzt).

VAR_OUTPUT

```
bBusy             : BOOL;
bError            : BOOL;
udiErrorId        : UDINT;
arrSlotInfos      : ARRAY[0..45] OF ST_DMXSlotInfo;
```

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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

arrSlotInfos: Wenn die Ausführung des Befehls abgeschlossen wurde (*bBusy* ist FALSE), dann stehen alle relevanten Informationen der DMX512 Slots als Array an diesem Ausgang an.

VAR_IN_OUT

```
stCommandBuffer           : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

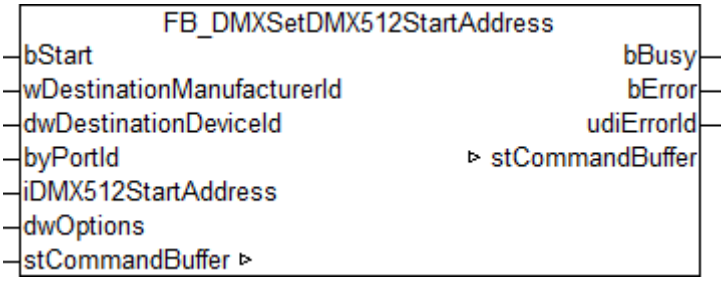
Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

Sehen Sie dazu auch

- ST_DMXSlotInfo [▶ 69]
- ST_DMXCommandBuffer [▶ 66]

5.35 FB_DMXSetDMX512StartAddress



Dieser Funktionsbaustein setzt die DMX512-Startadresse. Diese liegt im Bereich von 1 bis 512. Jedes Untergerät (Sub-Device) und das Hauptgerät (Root-Device) belegen unterschiedliche DMX512-Startadressen.

Durch eine positive Flanke an dem Eingang *bStart* wird der Baustein gestartet und der Ausgang *bBusy* geht auf TRUE. Die Eingänge *wDestinationManufacturerId* und *dwDestinationDeviceId* adressieren das DMX-Gerät. Der Eingang *byPortId* definiert den Kanal innerhalb des adressierten DMX-Gerätes. Ist die Ausführung des Befehls abgeschlossen, geht der Ausgang *bBusy* wieder auf FALSE. Die Ausgänge *bError* und *udiErrorId* können jetzt ausgewertet werden. Solange der Baustein aktiv ist (*bBusy* ist TRUE) werden weitere positive Flanken am Eingang *bStart* ignoriert.

VAR_INPUT

```
bStart                      : BOOL;
wDestinationManufacturerId  : WORD;
dwDestinationDeviceId       : DWORD;
byPortId                    : BYTE;
iDMX512StartAddress         : INT := 1;
dwOptions                   : DWORD := 0;
```

bStart: Über eine positive Flanke an diesem Eingang wird der Befehl gestartet.

wDestinationManufacturerId: Eindeutige Hersteller-Id vom DMX-Gerät.

dwDestinationDeviceId: eindeutige Geräte-Id vom DMX-Gerät.

byPortId: Kanal innerhalb des adressierten DMX-Gerätes. Untergeräte (Sub-Devices) werden durch die Port-Id adressiert. Das Hauptgerät (Root-Device) hat immer die Port-Id 0.

iDMX512StartAddress: DMX512-Startadresse (1 bis 512).

dwOptions: Optionen (wird derzeit nicht benutzt).

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. Nur gültig, wenn *bBusy* auf FALSE ist.

udiErrorId: Enthält den befehlsspezifischen Fehlercode des zuletzt ausgeführten Befehls. Nur gültig, wenn *bBusy* auf FALSE ist. Siehe Fehlercodes [▶ 70].

VAR_IN_OUT

```
stCommandBuffer         : ST_DMXCommandBuffer;
```

stCommandBuffer: Verweis auf die Struktur zur Kommunikation (Puffer) mit dem FB_EL6851CommunicationEx() [▶ 23]-Baustein.

Sehen Sie dazu auch

ST_DMXCommandBuffer [▶ 66]

5.36 Datentypen

5.36.1 E_DMXCommandClass

```
TYPE E_DMXCommandClass : (* Table A-1 *)
(
  eDMXCommandClassNotDefined          := 16#0000,     (* command class is not defined *)
  eDMXCommandClassDiscoveryCommand        := 16#0010,
  eDMXCommandClassDiscoveryCommandResponse    := 16#0011,
  eDMXCommandClassGetCommand          := 16#0020,
  eDMXCommandClassGetCommandResponse      := 16#0021,
  eDMXCommandClassSetCommand          := 16#0030,
  eDMXCommandClassSetCommandResponse      := 16#0031
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

5.36.2 E_DMXDataType

```
TYPE E_DMXDataType : (* Table A-15 *)
(
  eDMXDataTypeNotDefined          := 16#0000,     (* Data type is not defined *)
  eDMXDataTypeBitField        := 16#0001,     (* Data is bit packed *)
  eDMXDataTypeASCII           := 16#0002,     (* Data is a string *)
  eDMXDataTypeUnsignedByte        := 16#0003,     (* Data is an array of unsigned bytes *)
  eDMXDataTypeSignedByte          := 16#0004,     (* Data is an array of signed bytes *)
  eDMXDataTypeUnsignedWord        := 16#0005,     (* Data is an array of unsigned 16-bit words *)
  eDMXDataTypeSignedWord          := 16#0006,     (* Data is an array of signed 16-bit words *)
  eDMXDataTypeUnsignedDWord       := 16#0007,     (* Data is an array of unsigned 32-bit words *)
  eDMXDataTypeSignedDWord         := 16#0008      (* Data is an array of signed 32-bit words *)
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

5.36.3 E_DMXLampOnMode

```
TYPE E_DMXLampOnMode :
(
  eDMXLampOnModeOff    := 0,   (* Lamp stays off until directly instructed to strike *)
  eDMXLampOnModeDMX    := 1,   (* Lamp strikes upon receiving a DMX512 signal *)
  eDMXLampOnModeOn     := 2,   (* Lamp strikes automatically at power-up *)
  eDMXLampOnModeAfterCal   := 3    (* Lamp strikes after calibration or homing procedure *)
);
END_TYPE
```

5.36.4 E_DMXParameterDescriptionCommandClass

```
TYPE E_DMXParameterDescriptionCommandClass : (* Table A-16 *)
(
  eDMXParameterDescriptionCommandClassGet     := 16#0001,  (* PID supports GET only *)
  eDMXParameterDescriptionCommandClassSet     := 16#0002,  (* PID supports SET only *)
  eDMXParameterDescriptionCommandClassGetSet  := 16#0003   (* PID supports GET & SET *)
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

5.36.5 E_DMXParameterId

```
TYPE E_DMXParameterId : (* Table A-3 *)
(
  eDMXParameterIdNone                 := 16#0000,

  (* Network Management *)
  eDMXParameterIdDiscUniqueBranch             := 16#0001,
  eDMXParameterIdDiscMute                 := 16#0002,
  eDMXParameterIdDiscUnMute               := 16#0003,
```

```
  eDMXParameterIdProxiedDevices               := 16#0010,
  eDMXParameterIdProxiedDeviceCount           := 16#0011,
  eDMXParameterIdCommsStatus              := 16#0015,

  (* Status Collection *)
  eDMXParameterIdQueuedMessage            := 16#0020,
  eDMXParameterIdStatusMessages               := 16#0030,
  eDMXParameterIdStatusIdDescription          := 16#0031,
  eDMXParameterIdClearStatusId            := 16#0032,
  eDMXParameterIdSubDeviceStatusReportThreshold       := 16#0033,

  (* RDM Information *)
  eDMXParameterIdSupportedParamaters          := 16#0050,
  eDMXParameterIdParameterDescription         := 16#0051,

  (* Product Information *)
  eDMXParameterIdDeviceInfo               := 16#0060,
  eDMXParameterIdProductDetailIdList          := 16#0070,
  eDMXParameterIdDeviceModelDescription           := 16#0080,
  eDMXParameterIdManufacturerLabel            := 16#0081,
  eDMXParameterIdDeviceLabel              := 16#0082,
  eDMXParameterIdFactoryDefaults              := 16#0090,
  eDMXParameterIdLanguageCapabilities         := 16#00A0,
  eDMXParameterIdLanguage                 := 16#00B0,
  eDMXParameterIdSoftwareVersionLabel         := 16#00C0,
  eDMXParameterIdBootSoftwareVersionId        := 16#00C1,
  eDMXParameterIdBootSoftwareVersionLabel         := 16#00C2,

  (* DMX512 Setup *)
  eDMXParameterIdDMXPersonality               := 16#00E0,
  eDMXParameterIdDMXPersonalityDescription        := 16#00E1,
  eDMXParameterIdDMXStartAddress              := 16#00F0,
  eDMXParameterIdSlotInfo                 := 16#0120,
  eDMXParameterIdSlotDescription              := 16#0121,
  eDMXParameterIdDefaultSlotValue             := 16#0122,

  (* Sensors *)
  eDMXParameterIdSensorDefinition             := 16#0200,
  eDMXParameterIdSensorValue              := 16#0201,
  eDMXParameterIdRecordSensors            := 16#0202,

  (* Power/Lamp Settings *)
  eDMXParameterIdDeviceHours              := 16#0400,
  eDMXParameterIdLampHours                := 16#0401,
  eDMXParameterIdLampStrikes              := 16#0402,
  eDMXParameterIdLampState                := 16#0403,
  eDMXParameterIdLampOnMode               := 16#0404,
  eDMXParameterIdDevicePowerCycles            := 16#0405,

  (* Display Settings *)
  eDMXParameterIdDisplayInvert            := 16#0500,
  eDMXParameterIdDisplayLevel             := 16#0501,

  (* Configuration *)
  eDMXParameterIdPanInvert                := 16#0600,
  eDMXParameterIdTiltInvert               := 16#0601,
  eDMXParameterIdPanTiltSwap              := 16#0602,
  eDMXParameterIdRealTimeClock            := 16#0603,

  (* Control *)
  eDMXParameterIdIdentifyDevice               := 16#1000,
  eDMXParameterIdResetDevice              := 16#1001,
  eDMXParameterIdPowerState               := 16#1010,
  eDMXParameterIdPerformSelftest              := 16#1020,

  eDMXParameterIdSelfTestDescription          := 16#1021,
  eDMXParameterIdCapturePreset            := 16#1030,
  eDMXParameterIdPresetPlayBack               := 16#1031
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

5.36.6 E_DMXProductDetail

```
TYPE E_DMXProductDetail :
(
  eDMXProductDetailNotDeclared        := 16#0000,

  (* Generally applied to fixtures / 0x00xx *)
  eDMXProductDetailArc             := 16#0001,
  eDMXProductDetailMetalHalide        := 16#0002,
  eDMXProductDetailIncandescent          := 16#0003,
  eDMXProductDetailLED             := 16#0004,
  eDMXProductDetailFluroescent        := 16#0005,
  eDMXProductDetailColdcathode        := 16#0006,
  eDMXProductDetailElectroluminescent     := 16#0007,
  eDMXProductDetailLaser              := 16#0008,
  eDMXProductDetailFlashtube          := 16#0009,

  (* Generally applied to fixture accessories / 0x01xx *)
  eDMXProductDetailColorscroller          := 16#0100,
  eDMXProductDetailColorwheel         := 16#0101,
  eDMXProductDetailColorchange        := 16#0102,
  eDMXProductDetailIrisDouser         := 16#0103,
  eDMXProductDetailDimmingShutter         := 16#0104,
  eDMXProductDetailProfileShutter         := 16#0105,
  eDMXProductDetailBarndoorShutter        := 16#0106,
  eDMXProductDetailEffectsDisc        := 16#0107,
  eDMXProductDetailGoboRotator        := 16#0108,

  (* Generally applied to projectors / 0x02xx *)
  eDMXProductDetailVideo              := 16#0200,
  eDMXProductDetailSlide              := 16#0201,
  eDMXProductDetailFilm               := 16#0202,
  eDMXProductDetailOilwheel           := 16#0203,
  eDMXProductDetailLCDGate            := 16#0204,

  (* Generally applied to atmospheric effects / 0x03xx *)
  eDMXProductDetailFoggerGlycol           := 16#0300,
  eDMXProductDetailFoggerMineraloil       := 16#0301,
  eDMXProductDetailFoggerWater        := 16#0302,
  eDMXProductDetailCO2             := 16#0303,
  eDMXProductDetailLN2             := 16#0304,
  eDMXProductDetailBubble             := 16#0305,
  eDMXProductDetailFlamePropane           := 16#0306,
  eDMXProductDetailFlameOther         := 16#0307,
  eDMXProductDetailOlefactoryStimulator       := 16#0308,
  eDMXProductDetailSnow               := 16#0309,
  eDMXProductDetailWaterJet           := 16#030A,
  eDMXProductDetailWind               := 16#030B,
  eDMXProductDetailConfetti           := 16#030C,
  eDMXProductDetailHazard             := 16#030D,

  (* Generally applied to dimmers/power controllers / 0x04xx *)
  eDMXProductDetailPhaseControl           := 16#0400,
  eDMXProductDetailReversePhaseControl    := 16#0401,
  eDMXProductDetailSine               := 16#0402,
  eDMXProductDetailPWM             := 16#0403,
  eDMXProductDetailDC              := 16#0404,
  eDMXProductDetailHfballast          := 16#0405,
  eDMXProductDetailHfhvNeonballast        := 16#0406,
  eDMXProductDetailHfhvEl             := 16#0407,
  eDMXProductDetailMhrBallast         := 16#0408,
  eDMXProductDetailBitangleModulation     := 16#0409,
  eDMXProductDetailFrequencyModulation    := 16#040A,
  eDMXProductDetailHighfrequency12V       := 16#040B,
  eDMXProductDetailRelayMechanical        := 16#040C,
  eDMXProductDetailRelayElectronic        := 16#040D,
  eDMXProductDetailSwitchElectronic       := 16#040E,
  eDMXProductDetailContactor          := 16#040F,

  (* Generally applied to scenic drive / 0x05xx *)
  eDMXProductDetailMirrorballRotator      := 16#0500,
  eDMXProductDetailOtherRotator           := 16#0501,
  eDMXProductDetailKabukiDrop         := 16#0502,
  eDMXProductDetailCurtain            := 16#0503,
  eDMXProductDetailLineset            := 16#0504,
  eDMXProductDetailMotorControl           := 16#0505,
  eDMXProductDetailDamperControl          := 16#0506,

  (* Generally applied to data distribution / 0x06xx *)
```

```
  eDMXProductDetailSplitter           := 16#0600,
  eDMXProductDetailEthernetNode           := 16#0601,
  eDMXProductDetailMerge              := 16#0602,
  eDMXProductDetailDatapatch          := 16#0603,
  eDMXProductDetailWirelessLink           := 16#0604,

  (* Generally applied to data conversion and interfaces / 0x07xx *)
  eDMXProductDetailProtocolConvertor      := 16#0701,
  eDMXProductDetailAnalogDemultiplex      := 16#0702,
  eDMXProductDetailAnalogMultiplex        := 16#0703,
  eDMXProductDetailSwitchPanel        := 16#0704,

  (* Generally applied to audio or video (AV) devices / 0x08xx *)
  eDMXProductDetailRouter             := 16#0800,
  eDMXProductDetailFader              := 16#0801,
  eDMXProductDetailMixer              := 16#0802,

  (* Generally applied to controllers, backup devices and test equipment / 0x09xx *)
  eDMXProductDetailChangeoverManual       := 16#0900,
  eDMXProductDetailChangeoverAuto         := 16#0901,
  eDMXProductDetailTest               := 16#0902,

  (* Generally applied to any category / 0x0Axx *)
  eDMXProductDetailGfiRcd             := 16#0A00,
  eDMXProductDetailBattery            := 16#0A01,
  eDMXProductDetailControllableBreaker    := 16#0A02,

  (* Manufacturer Specific Types / 0x8000 - 0xDFFF *)
  eDMXProductDetailOther              := 16#7FFF  (* for use where the Manufacturer believes that no
ne of the defined details apply *)
);
END_TYPE
```

5.36.7 E_DMXResetDeviceType

```
TYPE E_DMXResetDeviceType :
(
  eDMXResetDeviceTypeWarm     := 1,
  eDMXResetDeviceTypeCold     := 255
);
END_TYPE
```

5.36.8 E_DMXSensorType

```
TYPE E_DMXSensorType : (* Table A-12 *)
(
  eDMXSensorTypeTemperature       := 16#00,
  eDMXSensorTypeVoltage       := 16#01,
  eDMXSensorTypeCurrent       := 16#02,
  eDMXSensorTypeFrequency     := 16#03,
  eDMXSensorTypeResistance    := 16#04,
  eDMXSensorTypePower         := 16#05,
  eDMXSensorTypeMass          := 16#06,
  eDMXSensorTypeLength        := 16#07,
  eDMXSensorTypeArea          := 16#08,
  eDMXSensorTypeVolume        := 16#09,
  eDMXSensorTypeDensity       := 16#0A,
  eDMXSensorTypeVelocity      := 16#0B,
  eDMXSensorTypeAcceleration      := 16#0C,
  eDMXSensorTypeForce         := 16#0D,
  eDMXSensorTypeEnergy        := 16#0E,
  eDMXSensorTypePressure      := 16#0F,
  eDMXSensorTypeTime          := 16#10,
  eDMXSensorTypeAngle         := 16#11,
  eDMXSensorTypePositionX     := 16#12,
  eDMXSensorTypePositionY     := 16#13,
  eDMXSensorTypePositionZ     := 16#14,
  eDMXSensorTypeAngularVelocity   := 16#15,
  eDMXSensorTypeLuminousIntensity := 16#16,
  eDMXSensorTypeLuminousFlux      := 16#17,
  eDMXSensorTypeIlluminance       := 16#18,
  eDMXSensorTypeChrominanceRed    := 16#19,
  eDMXSensorTypeChrominanceGreen  := 16#1A,
  eDMXSensorTypeChrominanceBlue   := 16#1B,
  eDMXSensorTypeContacts      := 16#1C,
  eDMXSensorTypeMemory        := 16#1D,
```

```
  eDMXSensorTypeItems          := 16#1E,
  eDMXSensorTypeHumidity       := 16#1F,
  eDMXSensorTypeCounter16Bit       := 16#20,
  eDMXSensorTypeOther          := 16#7F
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

5.36.9 E_DMXSensorUnit

```
TYPE E_DMXSensorUnit : (* Table A-13 *)
(
  eDMXSensorUnitNone               := 16#00,   (* CONTACTS *)
  eDMXSensorUnitCentigrade         := 16#01,   (* TEMPERATURE *)
  eDMXSensorUnitVoltsDC            := 16#02,   (* VOLTAGE *)
  eDMXSensorUnitVoltsACPeak            := 16#03,   (* VOLTAGE *)
  eDMXSensorUnitVoltsACRms         := 16#04,   (* VOLTAGE *)
  eDMXSensorUnitAmpereDC           := 16#05,   (* CURRENT *)
  eDMXSensorUnitAmpereACPeak           := 16#06,   (* CURRENT *)
  eDMXSensorUnitAmpereACRms            := 16#07,   (* CURRENT *)
  eDMXSensorUnitHertz              := 16#08,   (* FREQUENCY / ANG_VEL *)
  eDMXSensorUnitOhm                := 16#09,   (* RESISTANCE *)
  eDMXSensorUnitWatt               := 16#0A,   (* POWER *)
  eDMXSensorUnitKilogram           := 16#0B,   (* MASS *)
  eDMXSensorUnitMeters             := 16#0C,   (* LENGTH / POSITION *)
  eDMXSensorUnitMetersSquared          := 16#0D,   (* AREA *)
  eDMXSensorUnitMetersCubed            := 16#0E,   (* VOLUME *)
  eDMXSensorUnitKilogrammesPerMeterCubed  := 16#0F,   (* DENSITY *)
  eDMXSensorUnitMetersPerSecond        := 16#10,   (* VELOCITY *)
  eDMXSensorUnitMetersPerSecondSquared    := 16#11,   (* ACCELERATION *)
  eDMXSensorUnitNewton             := 16#12,   (* FORCE *)
  eDMXSensorUnitJoule              := 16#13,   (* ENERGY *)
  eDMXSensorUnitPascal             := 16#14,   (* PRESSURE *)
  eDMXSensorUnitSecond             := 16#15,   (* TIME *)
  eDMXSensorUnitDegree             := 16#16,   (* ANGLE *)
  eDMXSensorUnitSteradian          := 16#17,   (* ANGLE *)
  eDMXSensorUnitCandela            := 16#18,   (* LUMINOUS_INTENSITY *)
  eDMXSensorUnitLumen              := 16#19,   (* LUMINOUS_FLUX *)
  eDMXSensorUnitLux                := 16#1A,   (* ILLUMINANCE *)
  eDMXSensorUnitIre                := 16#1B,   (* CHROMINANCE *)
  eDMXSensorUnitByte               := 16#1C    (* MEMORY *)
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

5.36.10 E_DMXSensorUnitPrefix

```
TYPE E_DMXSensorUnitPrefix : (* Table A-14 *)
(
  eDMXSensorUnitPrefixNone           := 16#00,   (* Multiply by 1 *)
  eDMXSensorUnitPrefixDeci           := 16#01,   (* Multiply by 10^-1 *)
  eDMXSensorUnitPrefixCenti          := 16#02,   (* Multiply by 10^-2 *)
  eDMXSensorUnitPrefixMilli          := 16#03,   (* Multiply by 10^-3 *)
  eDMXSensorUnitPrefixMicro          := 16#04,   (* Multiply by 10^-6 *)
  eDMXSensorUnitPrefixNano           := 16#05,   (* Multiply by 10^-9 *)
  eDMXSensorUnitPrefixPico           := 16#06,   (* Multiply by 10^-12 *)
  eDMXSensorUnitPrefixFempto         := 16#07,   (* Multiply by 10^-15 *)
  eDMXSensorUnitPrefixAtto           := 16#08,   (* Multiply by 10^-18 *)
  eDMXSensorUnitPrefixZepto          := 16#09,   (* Multiply by 10^-21 *)
  eDMXSensorUnitPrefixYocto          := 16#0A,   (* Multiply by 10^-24 *)
  eDMXSensorUnitPrefixDeca           := 16#11,   (* Multiply by 10^1 *)
```

```
  eDMXSensorUnitPrefixHecto          := 16#12,    (* Multiply by 10^2 *)
  eDMXSensorUnitPrefixKilo           := 16#13,    (* Multiply by 10^3 *)
  eDMXSensorUnitPrefixMega           := 16#14,    (* Multiply by 10^6 *)
  eDMXSensorUnitPrefixGiga           := 16#15,    (* Multiply by 10^9 *)
  eDMXSensorUnitPrefixTerra          := 16#16,    (* Multiply by 10^12 *)
  eDMXSensorUnitPrefixPeta           := 16#17,    (* Multiply by 10^15 *)
  eDMXSensorUnitPrefixExa            := 16#18,    (* Multiply by 10^18 *)
  eDMXSensorUnitPrefixZetta          := 16#19,    (* Multiply by 10^21 *)
  eDMXSensorUnitPrefixYotta          := 16#1A     (* Multiply by 10^24 *)
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

5.36.11 E_DMXSlotDefinition

```
TYPE E_DMXSlotDefinition : (* Table C-2 *)
(
  (* Intensity Functions / 0x00xx *)
  eDMXSlotDefinitionIntensity          := 16#0001,  (* Intensity *)
  eDMXSlotDefinitionIntensityMaster        := 16#0002,  (* Intensity Master *)

  (* Movement Functions / 0x01xx *)
  eDMXSlotDefinitionPan            := 16#0101,  (* Pan *)
  eDMXSlotDefinitionTilt           := 16#0102,  (* Tilt *)

  (* Color Functions / 0x02xx *)
  eDMXSlotDefinitionColorWheel         := 16#0201,  (* Color Wheel *)
  eDMXSlotDefinitionColorSubCyan       := 16#0202,  (* Subtractive Color Mixer - Cyan/Blue *)
  eDMXSlotDefinitionColorSubYellow     := 16#0203,  (* Subtractive Color Mixer - Yellow/Amber *)
  eDMXSlotDefinitionColorSubMagenta        := 16#0204,  (* Subtractive Color Mixer - Magenta *)
  eDMXSlotDefinitionColorAddRed        := 16#0205,  (* Additive Color Mixer - Red *)
  eDMXSlotDefinitionColorAddGreen      := 16#0206,  (* Additive Color Mixer - Green *)
  eDMXSlotDefinitionColorAddBlue       := 16#0207,  (* Additive Color Mixer - Blue *)
  eDMXSlotDefinitionColorCorrection        := 16#0208,  (* Color Temperature Correction *)
  eDMXSlotDefinitionColorScroll        := 16#0209,  (* Color Scroll *)
  eDMXSlotDefinitionColorSemaphore     := 16#0210,  (* Color Semaphore *)

  (* Image Functions / 0x03xx *)
  eDMXSlotDefinitionStaticGoboWheel        := 16#0301,  (* Static gobo wheel *)
  eDMXSlotDefinitionRotoGoboWheel      := 16#0302,  (* Rotating gobo wheel *)
  eDMXSlotDefinitionPrismWheel         := 16#0303,  (* Prism wheel *)
  eDMXSlotDefinitionEffectsWheel       := 16#0304,  (* Effects wheel *)

  (* Beam Functions / 0x04xx *)
  eDMXSlotDefinitionBeamSizeIris       := 16#0401,  (* Beam size iris *)
  eDMXSlotDefinitionEdge           := 16#0402,  (* Edge/Lens focus *)
  eDMXSlotDefinitionFrost          := 16#0403,  (* Frost/Diffusion *)
  eDMXSlotDefinitionStrobe         := 16#0404,  (* Strobe/Shutter *)
  eDMXSlotDefinitionZoom           := 16#0405,  (* Zoom lens *)
  eDMXSlotDefinitionFramingShutter     := 16#0406,  (* Framing shutter *)
  eDMXSlotDefinitionShutterRotate      := 16#0407,  (* Framing shutter rotation *)
  eDMXSlotDefinitionDouser         := 16#0408,  (* Douser *)
  eDMXSlotDefinitionBarnDoor           := 16#0409,  (* Barn Door *)

  (* Control Functions / 0x05xx *)
  eDMXSlotDefinitionLampControl        := 16#0501,  (* Lamp control functions *)
  eDMXSlotDefinitionFixtureControl     := 16#0502,  (* Fixture control channel *)
  eDMXSlotDefinitionFixtureSpeed       := 16#0503,  (* Overall speed setting applied to multiple or a
ll parameters *)
  eDMXSlotDefinitionMacro          := 16#0504,  (* Macro control *)

  eDMXSlotDefinitionUndefined          := 16#FFFF   (* No definition *)
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

5.36.12 E_DMXSlotType

```
TYPE E_DMXSlotType : (* Table C-1 *)
(
  eDMXSlotTypePrinary     := 0,   (* Slot directly controls parameter (represents Coarse for 16-
bit parameters) *)
  eDMXSlotTypeSecFine     := 1,   (* Fine, for 16-bit parameters *)
  eDMXSlotTypeSecTiming       := 2,   (* Slot sets timing value for associated parameter *)
  eDMXSlotTypeSecSpeed    := 3,   (* Slot sets speed/velocity for associated parameter *)
  eDMXSlotTypeSecControl      := 4,   (* Slot provides control/mode info for parameter *)
  eDMXSlotTypeSecIndex    := 5,   (* Slot sets index position for associated parameter *)
  eDMXSlotTypeSecRotation     := 6,   (* Slot sets rotation speed for associated parameter *)
  eDMXSlotTypeSecIndexRotate  := 7,   (* Combined index/rotation control *)
  eDMXSlotTypeSecUndefined    := 255  (* Undefined secondary type *)
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

5.36.13 E_DMXStatusType

```
TYPE E_DMXStatusType : (* Table A-4 *)
(
  eDMXStatusTypeNone          := 16#00,
  eDMXStatusTypeGetLastMessage    := 16#01,
  eDMXStatusTypeAdvisory      := 16#02,
  eDMXStatusTypeWarning       := 16#03,
  eDMXStatusTypeError         := 16#04,
  eDMXStatusTypeAdvisoryCleared   := 16#12,
  eDMXSensorTypeWarningCleared    := 16#13,
  eDMXSensorTypeErrorCleared      := 16#14
);
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

5.36.14 ST_DMX512Personality

```
TYPE ST_DMX512Personality :
STRUCT
  byCurrentPersonality      : BYTE;
  byTotalPersonalities      : BYTE;
END_STRUCT
END_TYPE
```

5.36.15 ST_DMX512PersonalityDescription

```
TYPE ST_DMX512PersonalityDescription :
STRUCT
  iDMX512SlotsRequired       : INT;
```

```
  sDescription           : STRING;
END_STRUCT
END_TYPE
```

5.36.16 ST_DMXCommandBuffer

```
TYPE ST_DMXCommandBuffer :
STRUCT
  arrMessageQueue      : ST_DMXMessageQueue;
  stResponseTable      : ST_DMXResponseTable;
  byTransactionNumber      : BYTE;
END_STRUCT
END_TYPE
```

Sehen Sie dazu auch

- ST_DMXMessageQueue [▶ 66]
- ST_DMXResponseTable [▶ 68]

5.36.17 ST_DMXDeviceInfo

```
TYPE ST_DMXDeviceInfo :
STRUCT
  uliUID               : ST_DMXMac;
  stRDMProtocolVersion     : ST_DMXRDMProtocolVersion;
  uiDeviceModelId          : UINT;
  stProductCategory        : ST_DMXProductCategory;
  udiSoftwareVersionId     : UDINT;
  uiDMX512Footprint        : UINT;
  stDMX512Personality      : ST_DMX512Personality;
  uiDMX512StartAddress     : UINT;
  uiSubDeviceCount         : UINT;
  bySensorCount        : BYTE;
END_STRUCT
END_TYPE
```

Sehen Sie dazu auch

- ST_DMXMac [▶ 66]
- ST_DMXRDMProtocolVersion [▶ 68]
- ST_DMXProductCategory [▶ 67]
- ST_DMX512Personality [▶ 65]

5.36.18 ST_DMXMac

```
TYPE ST_DMXMac :
STRUCT
  wHighPart     : WORD;   (* Manufacturer ID / Higher word *)
  dwLowPart     : DWORD;  (* Device ID / Lower double word *)
END_STRUCT
END_TYPE
```

5.36.19 ST_DMXMessageQueue

```
TYPE ST_DMXMessageQueue :
STRUCT
  arrBuffer              : ARRAY [1..20] OF ST_DMXMessageQueueItem;
  byBufferReadPointer        : BYTE;
  byBufferWritePointer       : BYTE;
  byBufferDemandCounter      : BYTE;
  byBufferMaximumDemandCounter   : BYTE;
  uiBufferOverflowCounter    : UINT;
  bLockSemaphore         : BOOL;
END_STRUCT
END_TYPE
```

Sehen Sie dazu auch

ST_DMXMessageQueueItem [▶ 67]

5.36.20 ST_DMXMessageQueueItem

```
TYPE ST_DMXMessageQueueItem :
STRUCT
  bEntryIsEngaged              : BOOL;
  byMessageLength              : BYTE;
  wDestinationManufacturerId   : WORD;
  dwDestinationDeviceId        : DWORD;
  byTransactionNumber          : BYTE;
  byPortId                     : BYTE;
  byMessageCount               : BYTE;
  wSubDevice                   : WORD;
  byCommandClass               : BYTE;
  wParameterId                 : WORD;
  byParameterDataLength        : BYTE;
  arrParameterData             : ARRAY [0..255] OF BYTE;
  bWaitingForDMXSlaveResponse  : BOOL;
END_STRUCT
END_TYPE
```

5.36.21 ST_DMXParameterDescription

```
TYPE ST_DMXParameterDescription :
STRUCT
  byParameterDataLength  : BYTE;
  eDataType              : E_DMXDataType;
  ePDCommandClass        : E_DMXParameterDescriptionCommandClass;
  eType                  : E_DMXSensorType;
  eUnit                  : E_DMXSensorUnit;
  eUnitPrefix            : E_DMXSensorUnitPrefix;
  dwMinValidValue        : DWORD;
  dwMaxValidValue        : DWORD;
  dwDefaultValue         : DWORD;
  sDescription           : STRING;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

Sehen Sie dazu auch

- E_DMXDataType [▶ 59]
- E_DMXParameterDescriptionCommandClass [▶ 59]
- E_DMXSensorType [▶ 62]
- E_DMXSensorUnit [▶ 63]
- E_DMXSensorUnitPrefix [▶ 63]

5.36.22 ST_DMXProductCategory

```
TYPE ST_DMXProductCategory :
STRUCT
  byCoarse  : BYTE;
  byFine    : BYTE;
END_STRUCT
END_TYPE
```

5.36.23 ST_DMXRDMProtocolVersion

```
TYPE ST_DMXRDMProtocolVersion :
STRUCT
  byMajorVersion      : BYTE;
  byMinorVersion      : BYTE;
END_STRUCT
END_TYPE
```

5.36.24 ST_DMXResponseTable

```
TYPE ST_DMXResponseTable :
STRUCT
  arrResponseTable               : ARRAY [1..20] OF ST_DMXResponseTableItem;
  byResponseTableCounter         : BYTE;
  byResponseTableMaxCounter      : BYTE;
  uiResponseTableOverflowCounter : UINT;
  bLockSemaphore                 : BOOL;
END_STRUCT
END_TYPE
```

Sehen Sie dazu auch

ST_DMXResponseTableItem [▶ 68]

5.36.25 ST_DMXResponseTableItem

```
TYPE ST_DMXResponseTableItem :
STRUCT
  bEntryIsEngaged           : BOOL;
  uiErrorId                 : UINT;
  iErrorParameter           : INT;
  byMessageLength           : BYTE;
  wDestinationManufacturerId: WORD;
  dwDestinationDeviceId     : DWORD;
  wSourceManufacturerId     : WORD;
  dwSourceDeviceId          : DWORD;
  byTransactionNumber       : BYTE;
  byResponseType            : BYTE;
  byMessageCount            : BYTE;
  wSubDevice                : WORD;
  byCommandClass            : BYTE;
  wParameterId              : WORD;
  byParameterDataLength     : BYTE;
  arrParameterData          : ARRAY [0..255] OF BYTE;
END_STRUCT
END_TYPE
```

5.36.26 ST_DMXSensorDefinition

```
TYPE ST_DMXSensorDefinition :
STRUCT
  eSensorType           : E_DMXSensorType;
  eSensorUnit           : E_DMXSensorUnit;
  eSensorUnitPrefix     : E_DMXSensorUnitPrefix;
  iRangeMinimumValue    : INT;
  iRangeMaximumValue    : INT;
  iNormalMinimumValue   : INT;
  iNormalMaximumValue   : INT;
  byRecordValueSupport  : BYTE;
  sDescription          : STRING;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

Sehen Sie dazu auch

- E_DMXSensorType [▶ 62]
- E_DMXSensorUnit [▶ 63]
- E_DMXSensorUnitPrefix [▶ 63]

5.36.27 ST_DMXSensorValue

```
TYPE ST_DMXSensorValue :
STRUCT
  iPresentValue         : INT;
  iLowestDetectedValue  : INT;
  iHighestDetectedValue : INT;
  iRecordedValue        : INT;
END_STRUCT
END_TYPE
```

5.36.28 ST_DMXSlotInfo

```
TYPE ST_DMXSlotInfo :
STRUCT
  bEntryIsValid    : BOOL;
  eSlotType        : E_DMXSlotType;
  eSlotDefinition  : E_DMXSlotDefinition;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

Sehen Sie dazu auch

- E_DMXSlotType [▶ 65]
- E_DMXSlotDefinition [▶ 64]

5.36.29 ST_DMXStatusMessage

```
TYPE ST_DMXStatusMessage :
STRUCT
  bEntryIsValid       : BOOL;
  iSubDeviceId        : INT;
  eStatusType         : E_DMXStatusType;
  iStatusMessageId    : INT;
  iDataValue01        : INT;
  iDataValue02        : INT;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

Sehen Sie dazu auch

- E_DMXStatusType [▶ 65]

5.36.30 ST_EL6851InData

```
TYPE ST_EL6851InData :
STRUCT
  bTransmitAccepted    : BOOL;
  bReceiveToggle       : BOOL;
  bCyclicTxDDisabled   : BOOL;
  bDefaultDataSent     : BOOL;
  bFrameSentToggle     : BOOL;
  bTxPDOToggle         : BOOL;
  wChannelLength       : WORD;
  byStartCode          : BYTE;
  byDummy              : BYTE;
  arrData              : ARRAY [1..64] OF BYTE;
END_STRUCT
END_TYPE
```

5.36.31 ST_EL6851InDataEx

```
TYPE ST_EL6851InDataEx :
STRUCT
  bTransmitAccepted    : BOOL;
  bReceiveToggle       : BOOL;
  bCyclicTxDDisabled   : BOOL;
  bDefaultDataSent     : BOOL;
  bFrameSentToggle     : BOOL;
  bTxPDOToggle         : BOOL;
  wChannelLength       : WORD;
  byStartCode          : BYTE;
  byDummy1             : BYTE;
  arrData              : ARRAY [1..64] OF BYTE;
  bWcState             : BOOL;
  bInputToggle         : BOOL;
  uiState              : UINT;
  stAdsAddr            : ST_EL6851AdsAddr;
END_STRUCT
END_TYPE
```

Voraussetzungen

Entwicklungsumgebung	Zielsystem	erforderliche Bibliotheken
TwinCAT 2.11 R3/x64 ab Build 2256	PC/CX	TcDMX-Bibliothek ab V1.3.0

5.36.32 ST_EL6851OutData

```
TYPE ST_EL6851OutData :
STRUCT
  bTransmitRequest     : BOOL;
  bDisableCyclicTxD    : BOOL;
  bSendDefaultData     : BOOL;
  byDummy1             : BYTE;
  wChannelLength       : WORD;
  byStartCode          : BYTE;
  byDummy2             : BYTE;
  arrData              : ARRAY [1..512] OF BYTE;
END_STRUCT
END_TYPE
```

5.37 Fehlercodes

Fehlercodes der TwinCAT PLC-Bibliothek TcDMX

Wert (hex)	Wert (dez)	Beschreibung
0x0000	0	Kein Fehler.
0x8001	32769	Keine Antwort von der DMX-Klemme.
0x8002	32770	Keine Antwort vom DMX-Gerät.

Wert (hex)	Wert (dez)	Beschreibung
0x8003	32771	Überlauf Kommunikationspuffer.
0x8004	32772	Keine Antwort des Kommunikationsbausteines.
0x8005	32773	Parameter *byPortId* liegt außerhalb des gültigen Bereichs.
0x8006	32774	Fehler Checksumme.
0x8007	32775	Parameter *eResetDeviceType* liegt außerhalb des gültigen Bereichs.
0x8008	32776	Zeit Überschreitung.
0x8009	32777	Parameter *uliLowerBoundUID* ist grÖßer als *uliUpperBoundUID.*
0x800A	32778	Es können keine RDM-Befehle versendet werden, da sich die Klemme im Cycle-Mode befindet.
0x800B	32779	Parameter *iDMX512StartAddress* liegt außerhalb des gültigen Bereichs (1-512).
0x800C	32780	Fehler beim Setzen der DMX512 Start Adresse.
0x800D	32781	Es können keine Prozessdaten versendet werden, da sich die Klemme nicht im Cycle-Mode befindet.
0x800E	32782	Es wurde ein RDM-Telegramm mit der Datenlänge 0 empfangen.
0x800F	32783	RDM-Response: Antwort vom RDM-Telegramm ist ungültig.
0x8010	32784	RDM-Response: Der RDM-Befehl ist nicht im DMX-Gerät implementiert.
0x8011	32785	RDM-Response: Das DMX-Gerät kann den gesendeten RDM-Befehl nicht auswerten, da diese nicht richtig formatiert sind.
0x8012	32786	RDM-Response: Das DMX-Gerät kann auf den RDM-Befehl nicht richtig reagieren, da ein interner Hardwarefehler im DMX-Gerät vorliegt.
0x8013	32787	RDM-Response: Der Proxy ist nicht der RDM-Linienmaster und kann deshalb nicht auf den RDM-Befehl reagieren.
0x8014	32788	RDM-Response: RDM-Schreibbefehle sind im DMX-Gerät derzeit blockiert.
0x8015	32789	RDM-Response: Die *Command Class* (CC) ist nicht korrekt. Evtl. wurde ein Schreibbefehl versendet, obwohl nur Lesebefehle erlaubt sind.
0x8016	32790	RDM-Response: Der Parameter, der mit dem RDM-Befehl versendet wurde, liegt außerhalb des gültigen Bereichs oder ist ungültig.
0x8017	32791	RDM-Response: interner Buffer-Überlauf im DMX-Gerät.
0x8018	32792	RDM-Response: Der RDM-Befehl konnte nicht vollständig empfangen werden, da nicht ausreichend Speicher im DMX-Gerät vorhanden ist.
0x8019	32793	RDM-Response: *Sub Device* (SD) ist nicht korrekt. Evtl. liegt der Wert außerhalb des gültigen Bereichs oder ist ungültig.
0x801A	32794	Parameter *iDMX512SlotOffset* liegt außerhalb des gültigen Bereichs (0-511).
0x801B	32795	Parameter *bySensorNumber* liegt außerhalb des gültigen Bereichs (0-254).
0x801C	32796	RDM-Response: Das Feld *Parameter Data* (PD) ist zu lang. Es können nicht alle Daten der Rückantwort empfangen werden. Hierzu muss der Baustein FB_EL6851CommunicationEx() [▶ 23] eingesetzt werden.
0x801D	32797	Die ADS-Adresse für den Zugriff auf die PDOs ist ungültig. Wurde die Struktur *AdsAddr* der KL6851 auf die entsprechende SPS-Variable gemappt?
0x801E	32798	Beim Lesezugriff auf die PDOs ist ein ADS-Fehler aufgetreten.

6 Anhang

6.1 Versenden der zyklischen Prozessdaten als DMX-Master (EL6851)

Beispieldateien entpacken https://infosys.beckhoff.com/content/1031/tcplclibdmx/Resources/

11977739147.zip 

Starten des Beispielprogramms

Die Applikationsbeispiele sind mit einem Prüfaufbau getestet und entsprechend beschrieben worden. Etwaige Abweichungen bei der Einrichtung an realen Applikationen sind möglich.

Für den Prüfaufbau wurde folgende Hardware und Software verwendet:

- TwinCAT-Master-PC mit TwinCAT Version 2.11 (Build 2229) oder neuer und INTEL PRO/100 VE Ethernet-Adapter.
- Beckhoff EtherCAT Koppler EK1100, Klemmen EL6851 [▶ 76] und EL9011.
- RGB-LED DMX-Slave mit 3 Kanälen (je Farbe einer). Pro Kanal wird ein Slot belegt.

Die Verdrahtung des DMX-Slave ist entsprechend des Anschlussschemas [▶ 76] vorzunehmen.

Vorgehensweise zum Starten des Programms

- Speichern Sie die TSM-Datei für den TwinCAT System Manager und die PRO-Datei für das TwinCAT PLC Control https://infosys.beckhoff.com/content/1031/tcplclibdmx/Resources/11977739147.zip

   lokal auf Ihrer Festplatte.
- Start der *.TSM-Datei und *.PRO-Datei; der TwinCAT System Manger und das TwinCAT PLC Control öffnen sich.
- Schließen Sie die Hardware entsprechend Abb. 1 an und verbinden Sie den Ethernet-Adapter ihres PCs mit dem EtherCAT-Koppler (weitere Hinweise hierzu finden Sie in den entsprechenden Kopplerhandbüchern)
- Auswahl des lokalen Ethernet-Adapters (ggf. mit Echtzeit-Treiber) unter Systemkonfiguration, E/A - Konfiguration, E/A -Geräte, Gerät (EtherCAT); wählen Sie dann unter Karteireiter "Adapter", "Suchen..." den entsprechenden Adapter aus und bestätigen Sie das(siehe Abb. 2a + 2 b)

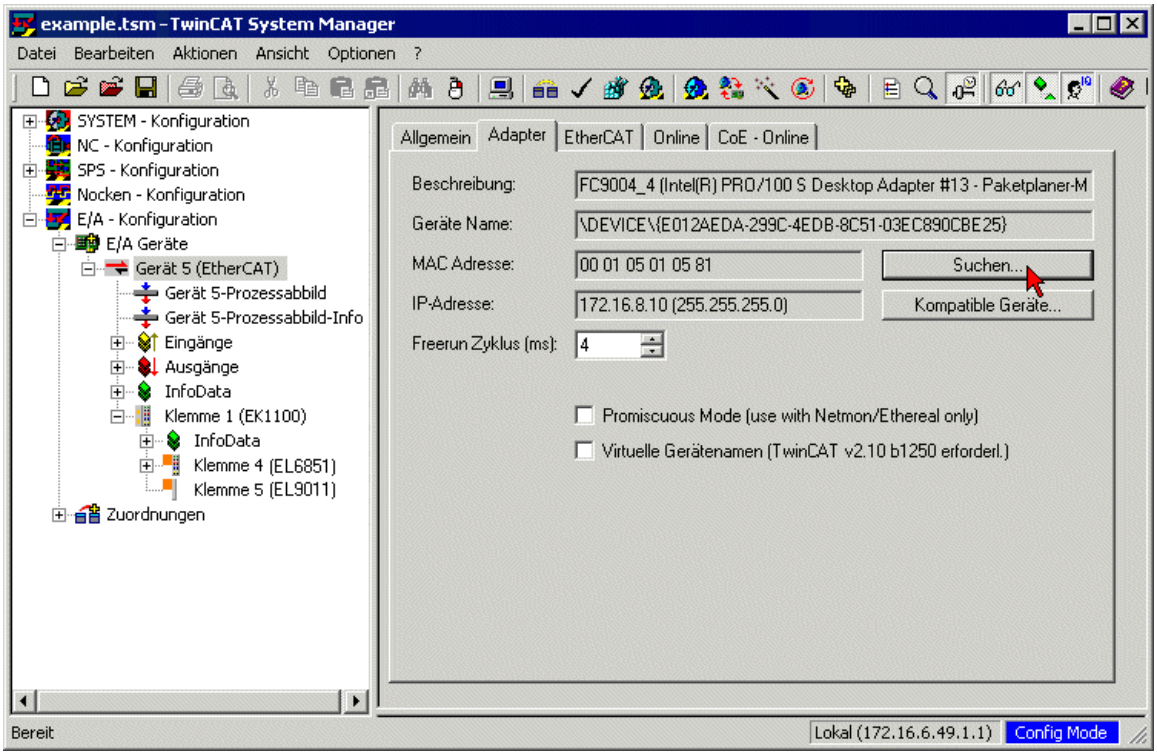


Abb. 2a: Suchen des Ethernet-Adapters

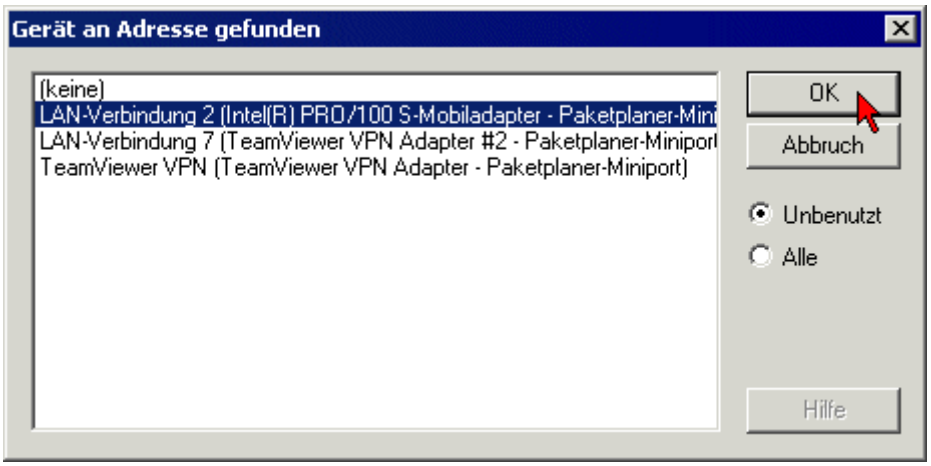


Abb. 2b: Auswahl und Bestätigung des Ethernet-Adapters

- Aktivierung der Konfiguration und bestätigen (Abb. 3a +3b)

Abb. 3a: Aktivierung der Konfiguration

Abb. 3b: Konfigurationsaktivierung bestätigen

- Neue Variablenzuordnung bestätigen, Neustart im RUN-Modus (Abb. 4a + 4b)

Abb. 4a: Variablenzuordnung erzeugen

Abb. 4b: Neustart TwinCAT im RUN-Modus

- Im TwinCAT PLC Control unter Menü "Projekt" -> "Alles Übersetzen" das Projekt übersetzen (Abb. 5)

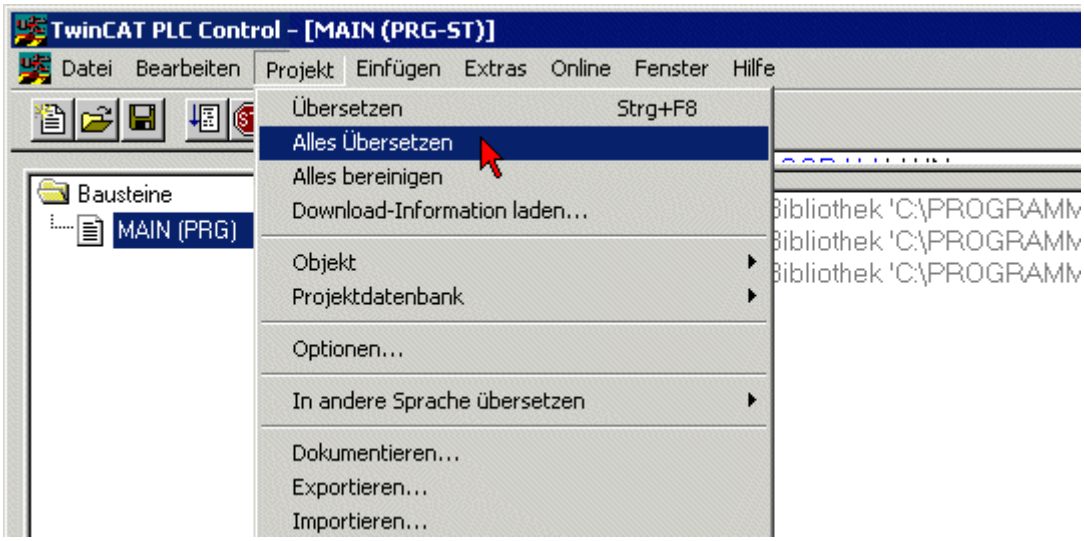


Abb. 5: Projekt übersetzen

- Im TwinCAT PLC Control: Einloggen mit der Taste "F11", Laden des Programms bestätigen (Abb. 6), Start des Programms mit Taste "F5"

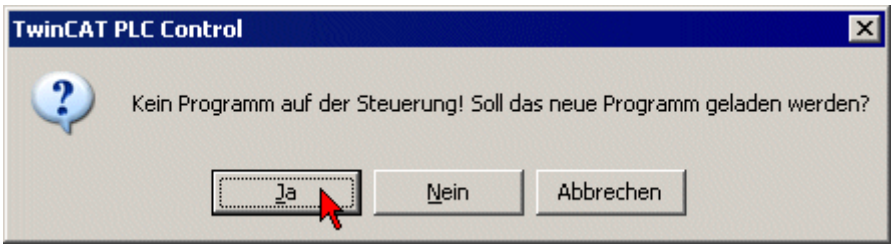


Abb. 6: Programmstart bestätigen

Visualisierung

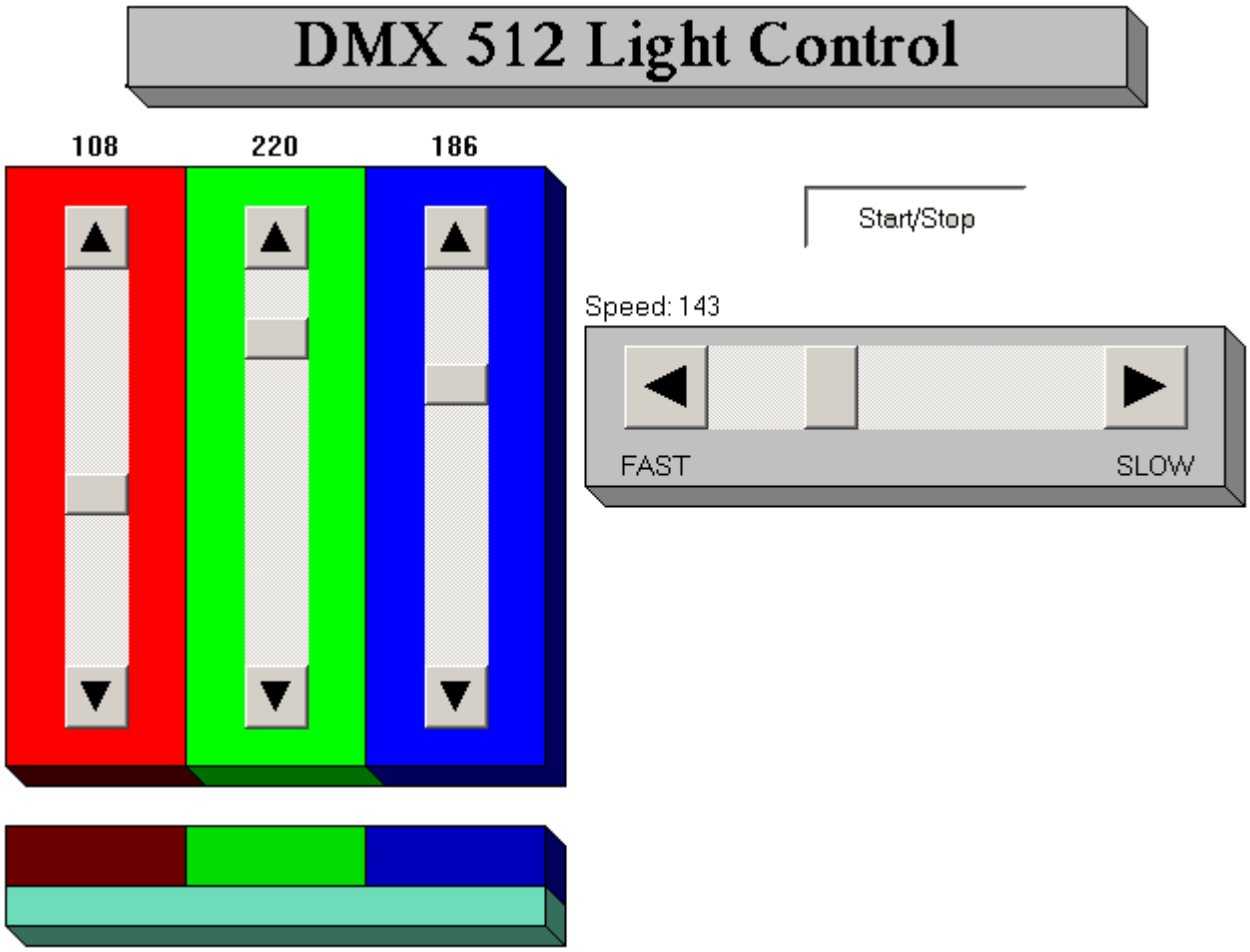


Abb. 7: Vorgabe der Stellgrößen für die drei Farben des DMX-Slaves im TwinCAT PLC Control

Das Beispiel sendet zyklisch die DMX-Daten zu einem DMX-Slave. Das hier verwendete DMX-Gerät belegt drei Slots (Bytes) im DMX512-Frame. Jeder Slot adressiert eine der drei Farben. Ist die Schaltfläche '*Start/Stop*' gedrückt, so werden automatisch generierte Daten an das DMX-Gerät gesendet. Die Geschwindigkeit der Änderungen kann durch den waagerechten Schieberegler verändert werden. Ist die Schaltfläche '*Start/Stop*' nicht gedrückt, so können Sie durch die drei senkrechten Schieberegler die Werte manuell verändern.

6.2 Empfangen von jeweils 64 Byte Daten an zwei DMX-Slaves (EL6851-0010)

Beispieldateien entpacken https://infosys.beckhoff.com/content/1031/tcplclibdmx/Resources/

11977740555.zip

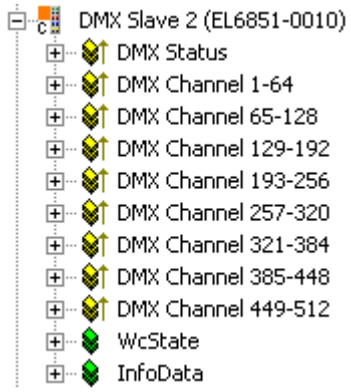


Abb. 3: 8 Arrays zu je 64 Byte im Vollausbau (alle PDO angewählt)

Eine EL6851-0010 [▶ 76] kann max. 512 Byte lesen (jeweils 64 Byte in acht Arrays, siehe Abb. 3). Die Arrays können im TwinCAT System Manger (Reiter "Prozessdaten") über das PDO 0x1C13 zugeordnet werden.

Beispiel:

DMX Channel 1 - 64 - > Index 0x1A01
DMX Channel 65 - 128 -> Index 0x1A02

....

DMX Channel 449 - 512 -> Index 0x1A08

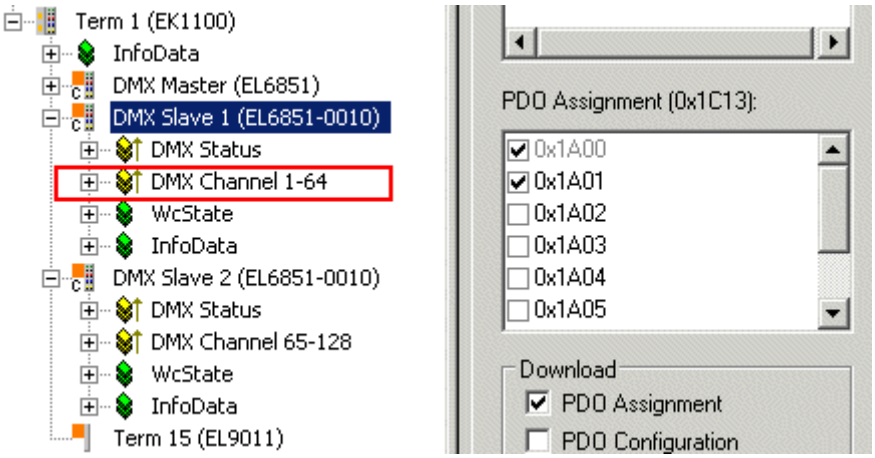


Abb. 4: DMX Channel 1 - 64 (default) durch Anwahl PDO 0x1A01

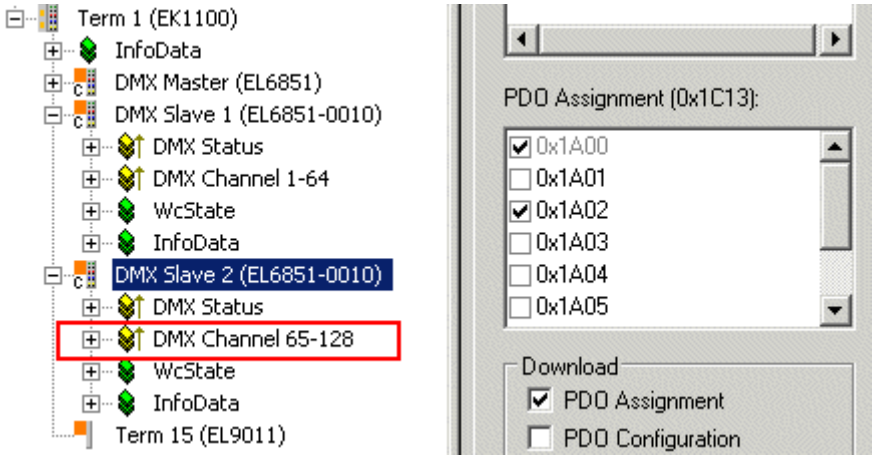


Abb. 5: DMX Channel 65 - 128 durch Anwahl PDO 0x1A02

Im Beispiel-Programm empfängt der erste DMX-Slave die ersten gesendeten 64 Byte und der zweite die nächsten 64 Byte (Abb. 4 + 5; der Empfang von den gesamten 128 Byte mit einer EL6851-0010 ist auch möglich, im Beispiel ist die Aufteilung bewusst gewählt).

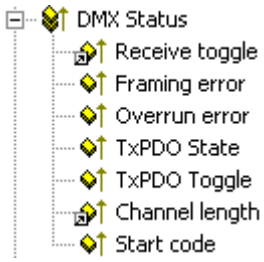


Abb. 6: DMX Status Objekt

Im DMX Status Objekt (Index 0x6000, "DMX-Status", Abb. 6) ist mit Index 0x6000:11 ("Channel length") ein Copy Counter angelegt.

Beispiel:

Bei aktiviertem PDO 0x1A01 beträgt der Wert von "Channel length" 64_{dez}. Bei aktiviertem PDO 0x1A02 ist der Wert 128_{dez}. Sind beide PDO aktiviert (0x1A01 und 0x1A02) betragt der Wert ebenfalls 128_{dez}.

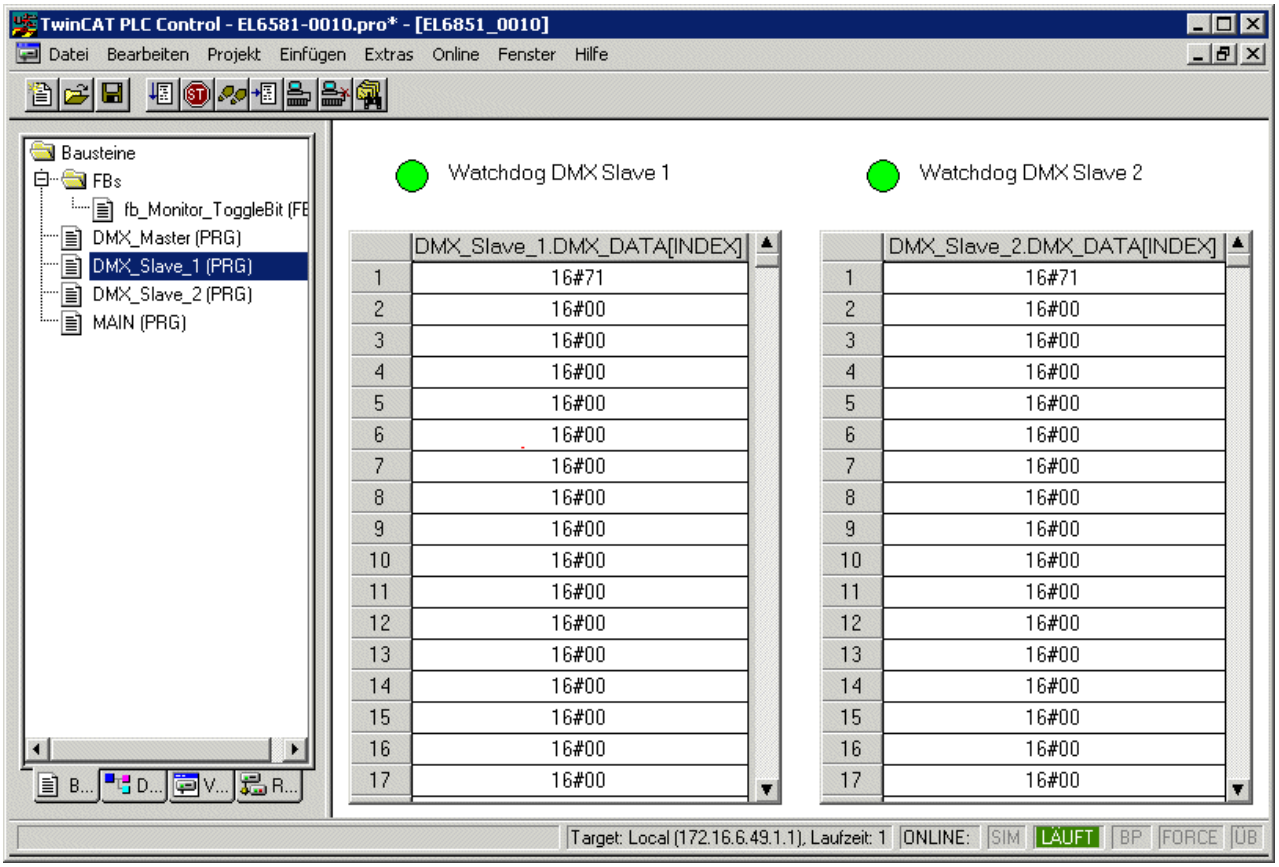


Abb. 6: Visualisierung Beispiel 3 in der TwinCAT PLC

DMX Slave 1 empfängt 64 Byte Daten auf Kanal 1 des ersten Arrays ("DMX Channel 1 - 64")
DMX Slave 2 empfängt 64 Byte Daten auf Kanal 1 des zweiten Arrays ("DMX Channel 65 - 128")

Das "Receive toggle" Bit (Index 0x6000:02) wird jeweils über den FB "fb_Monitor_ToggleBit" ausgewertet und angezeigt (Watchdog DMX Slave).

6.3 Konfigurieren von DMX-Slaves per Remote Device Management (RDM)

Beispieldateien entpacken https://infosys.beckhoff.com/content/1031/tcplclibdmx/Resources/

11977741963.zip

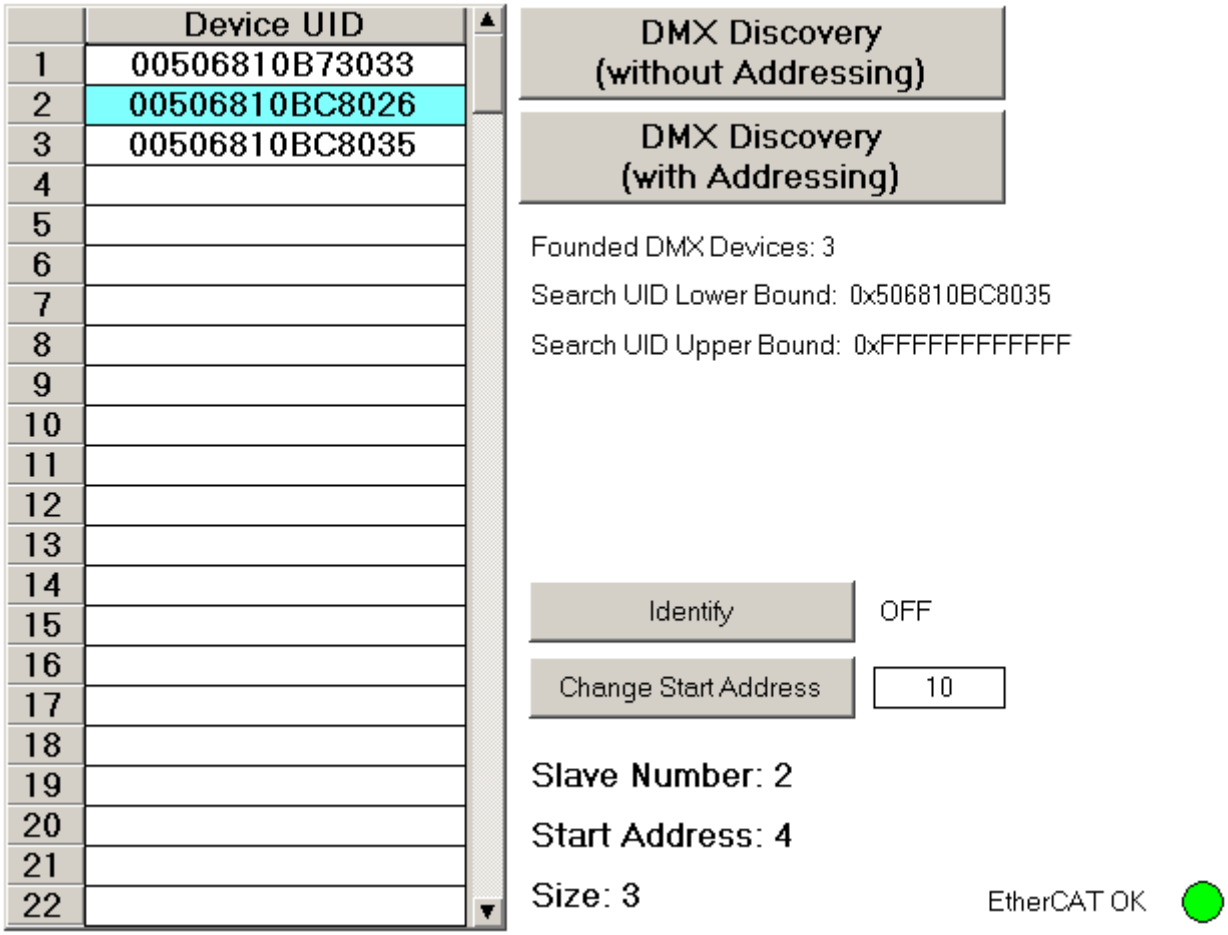


Abb. 2: Visualisierung Beispiel 2 in der TwinCAT PLC

EtherCAT-Funktionalität

Der Dialog ist nur dann in Funktion, wenn die "EtherCAT OK"- LED grün leuchtet. Eine rote LED zeigt eine EtherCAT-Kommunikationsstörung an.

Mit der Schaltfläche '*DMX Discovery (without Addressing)*' wird an der DMX-Linie nach DMX-Slaves gesucht. Alle gefundenen DMX-Slaves werden in der linken Liste angezeigt. Durch Anklicken eines Eintrages wird dieser selektiert. Nach Betätigen von '*Identify*' wird der RDM-Befehl zum Identifizieren des jeweiligen DMX-Gerätes versendet. Die Startadresse kann in dem Eingabefeld neben der Schaltfläche '*Change Start Address*' eingegeben werden. Nach Drücken der Schaltfläche wird die neue Startadresse an das selektierte DMX-Gerät gesendet. Im unteren Bereich wird die Nummer des DMX-Gerätes, die Startadresse im DMX512-Frame und die Slotgröße (Anzahl der Bytes im DMX-Frame) für das selektierte Gerät angezeigt.

6.4 DMX-Master mit BC9191-0100

Dieses Beispiel zeigt ein einfaches SPS-Programm, wie mit DMX-Teilnehmern kommuniziert werden kann. Es werden die Analogwerte der drei Analogeingänge des BC9191-0100 eingelesen und auf die DMX Kanäle 1-3 über die DMX-Schnittstelle an Klemmleiste X1 wieder ausgegeben.

Beispieldateien entpacken https://infosys.beckhoff.com/content/1031/tcplclibdmx/Resources/

11977743371.zip

Hardware

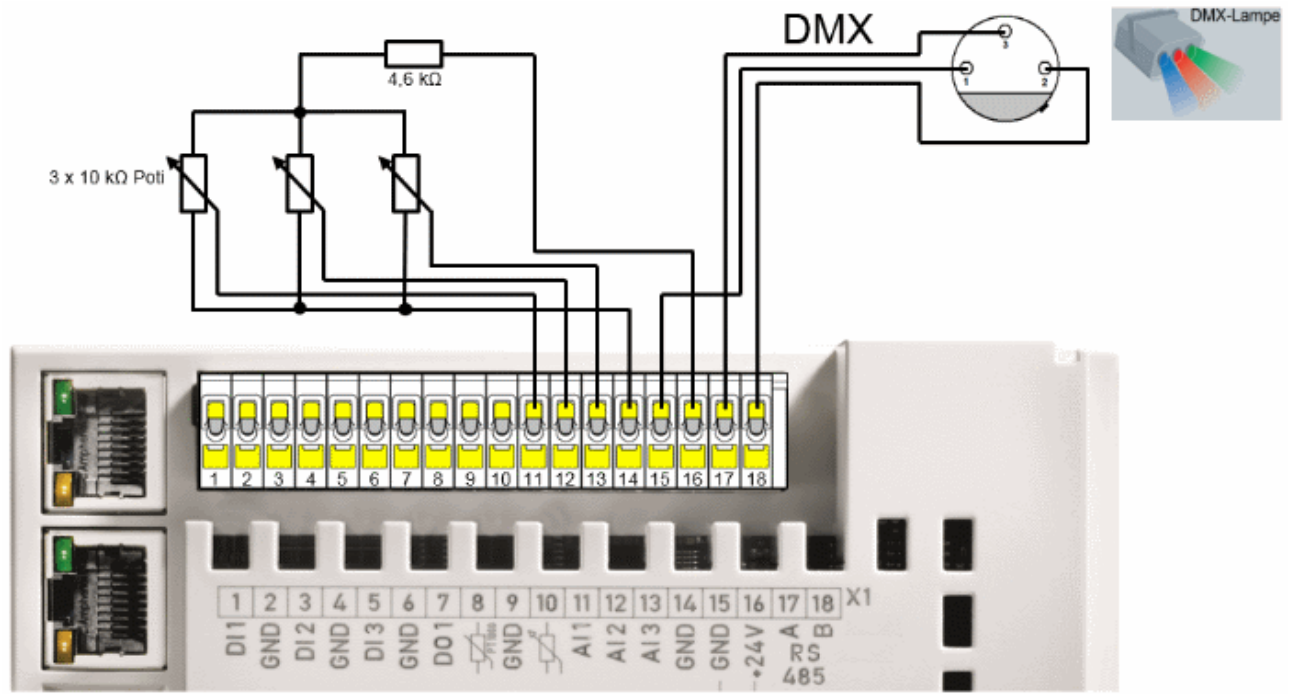
Einrichtung der Komponenten

Es wird folgende Hardware benötigt:

- 1x Busklemmen Controller BC9191 mit RS485
- 1x Endklemme KL9010
- 1x DMX Gerät

DMX-Belegung

Klemmleiste X1	3-pol. XLR Buchse	5-pol. XLR Buchse
X1 / 15 (Masse/Schirm)	Pin 1	Pin 1
X1 / 17 (DMX +)	Pin 3	Pin 3
X1 / 18 (DMX -)	Pin 2	Pin 2



Richten Sie die DMX-Teilnehmer, wie in den entsprechenden Dokumentationen beschrieben, ein.

Software

Erläuterungen zum SPS-Programm sind als Kommentare im Quellcode zu finden.

6.5 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.

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 unseren Internetseiten: https://www.beckhoff.de

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(0)5246 963 157
Fax:	+49(0)5246 963 9157
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(0)5246 963 460
Fax:	+49(0)5246 963 479
E-Mail:	service@beckhoff.com

Beckhoff Firmenzentrale

Beckhoff Automation GmbH & Co. KG

Hülshorstweg 20
33415 Verl
Deutschland

Telefon:	+49(0)5246 963 0
Fax:	+49(0)5246 963 198
E-Mail:	info@beckhoff.com
Internet:	https://www.beckhoff.de

Mehr Informationen:
www.beckhoff.de/tx1200

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