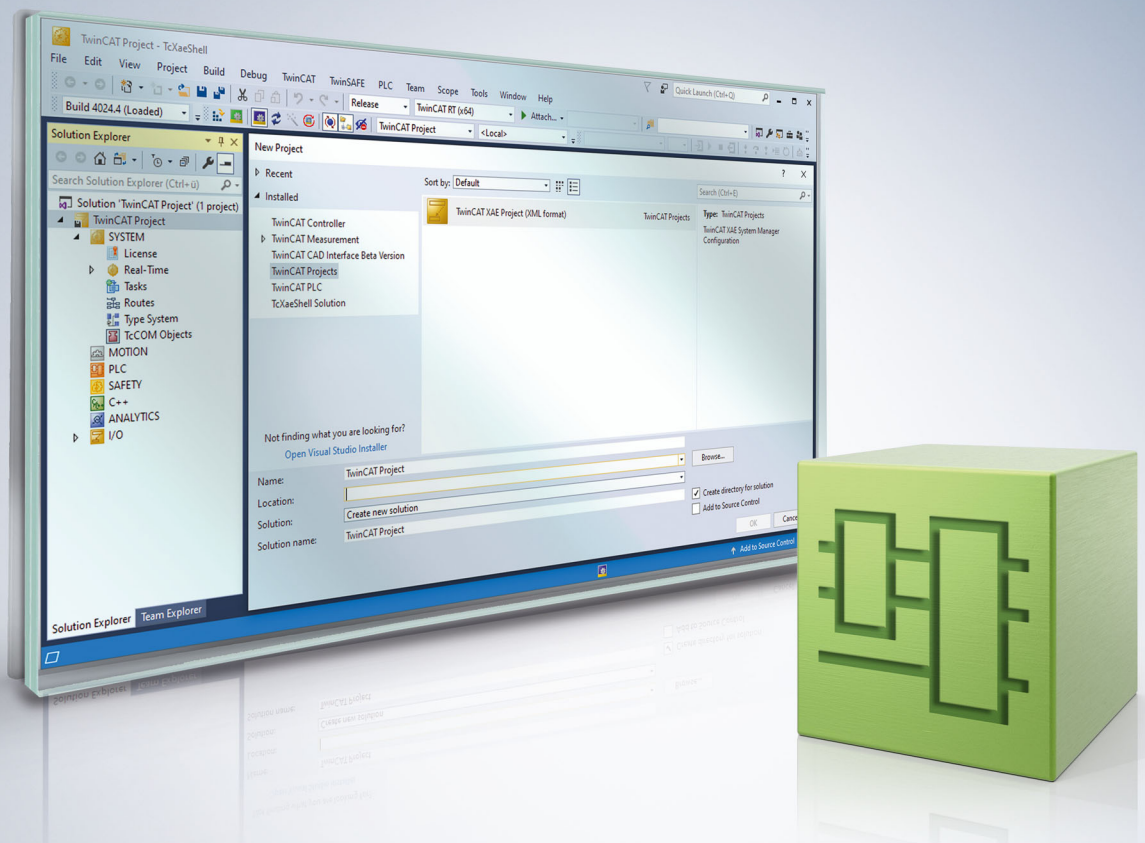
BECKHOFF New Automation Technology

Manual | EN

TE1000

TwinCAT 3 | PLC Library: Tc2_DMX



2023-09-13 | Version: 1.7.1

Table of contents

1 Foreword ... 5
1.1 Notes on the documentation ... 5
1.2 For your safety ... 6
1.3 Notes on information security ... 7
2 Introduction ... 8
3 DMX ... 9
4 Programming ... 10
4.1 POUs ... 10
4.1.1 High Level ... 10
4.1.2 Low Level ... 13
4.1.3 Error codes ... 61
4.2 DUTs ... 62
4.2.1 Enums ... 62
4.2.2 Structures ... 68
4.3 Integration into TwinCAT ... 74
4.3.1 EL6851 with CX5120 ... 74
5 Appendix ... 79
5.1 Example: Configuration by RDM ... 79
5.2 Example: DMX master ... 79
5.3 Example: DMX slave ... 83
5.4 Support and Service ... 85

1 Foreword

1.1 Notes on the documentation

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

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

Disclaimer

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

Trademarks

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

Patent Pending

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

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

Copyright

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

1.2 For your safety

Safety regulations

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

Exclusion of liability

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

Personnel qualification

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

Signal words

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

Personal injury warnings

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

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

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

Warning of damage to property or environment

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

Information on handling the product

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

1.3 Notes on information security

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

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

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

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

2 Introduction

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

- TwinCAT XAE
- PC and network knowledge
- Structure and properties of the Beckhoff Embedded PC and its Bus Terminal system
- Technology of DMX devices

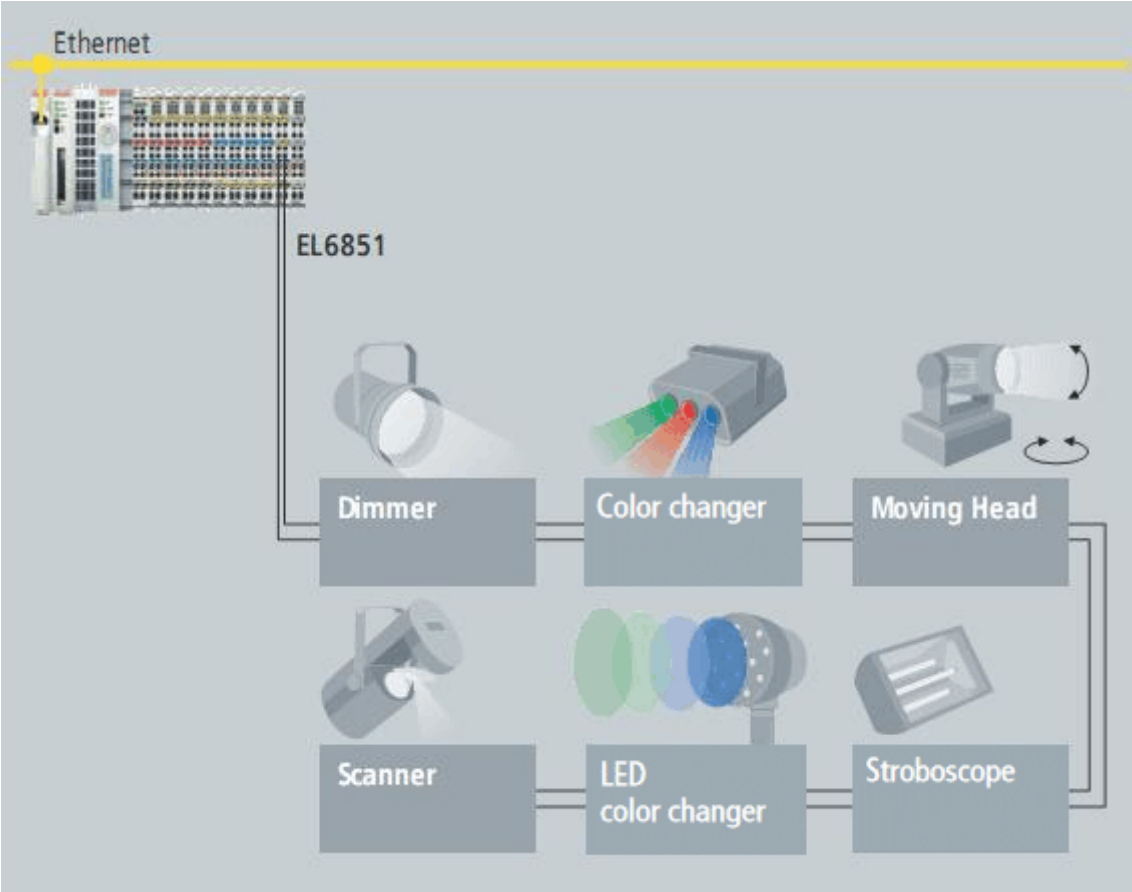
The Tc2_DMX library is usable on all hardware platforms that support TwinCAT 3.1 or higher.

Hardware documentation for EL6851 in the Beckhoff Information System.

3 DMX

DMX is the standard protocol for controlling professional stage and effect lighting equipment, which is used, for example, for the dynamic lighting of showrooms and salesrooms as well as for exclusive plays of light and color in prestigious buildings, such as hotels and event centers. Color mixing and brightness values are transmitted to DMX devices that are static light sources, whilst moving sources of light additionally receive spatial coordinates. EtherCAT's high data transfer rate enables light settings to be updated at a higher rate, with the result that changes of light and color are perceived by the eye as being more harmonious. The EL6851 can be used to control DMX devices with three axes, such as scanners, moving heads or spotlights. TwinCAT function blocks enable implementation of the RDM protocol (**R**emote **D**evice **M**anagement) for internal diagnostics and parameterization in DMX.

The DMX master transmits new settings to the slaves cyclically at 250 kBaud in order to generate dynamic lighting changes and plays of color. In the DMX protocol, a maximum of 32 slaves are allowed in one strand without repeaters. The 512 byte long frame in the DMX protocol is termed a 'universe'. 512 channels are available in it, each of which represents a device setting with 8-bit resolution, i.e. in 256 steps, e.g. for dimming, color, focus etc. In the case of moving light sources, additional settings such as inclination, swiveling and speed (with 8-bit or 16-bit resolution) occupy additional channels, so that the 512 channels are only indirectly sufficient for 32 devices. Furthermore, if the universe is fully utilized a frame will require 22 ms for internal DMX circulation, which means a refresh rate of 44 Hz. Light changes at this frequency are perceived to be inharmonious. The transitions only appear to be harmonious from a frequency of >200 Hz. The circulation period can be shortened by reducing the amount of user data. The optimum has proven to be a utilization of 64 bytes (frequency >300 Hz), with which 64 channels are available per universe.



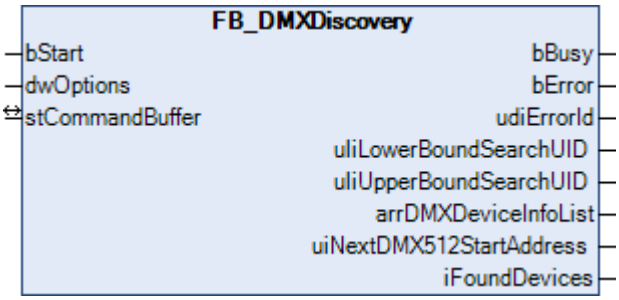
The integration of several universes in a controller becomes simple with the EL6851: EtherCAT can transfer large amounts of data quickly, the EtherCAT protocol is retained until inside the terminal and the terminal supports various mapping sizes (64 to 512 bytes). Hence, if several master terminals are connected, each as its own universe, the time offset in transmitting from the controller to the master can be reduced significantly.

4 Programming

4.1 POUs

4.1.1 High Level

4.1.1.1 FB_DMXDiscovery



This function block searches for up to 50 DMX devices and automatically sets the start address (optional). The most important information for the devices found is displayed in a structure.

Inputs

```
VAR_INPUT
  bStart    : BOOL;
  dwOptions : DWORD;
END_VAR
```

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
dwOptions	DWORD	Options (see table). The individual constants must be linked with OR operators.

Constant	Description
DMX_OPTION_COMPLETE_NEW_DISCOVERY	All DMX devices are taken into account.
DMX_OPTION_SET_START_ADDRESS	The start address is set for all DMX devices that are found. Consecutive, starting with 1.
DMX_OPTION_OPTICAL_FEEDBACK	When a DMX device is found, the function IDENTIFY_DEVICE is called for two seconds.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

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

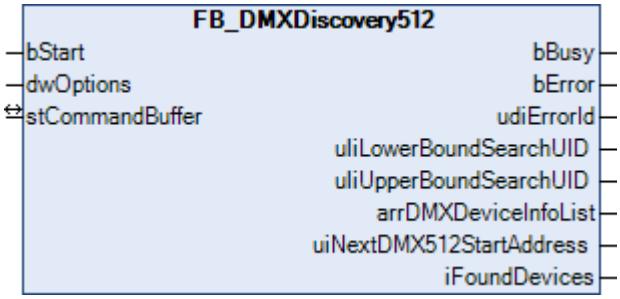
```
  uiNextDMX512StartAddress : UINT;
  iFoundDevices            : INT;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
uliLowerBoundSearchUID	T_ULARGE_INTEGER	During the search, the lower search address is sent to this output.
uliUpperBoundSearchUID	T_ULARGE_INTEGER	During the search, the upper search address is sent to this output.
arrDMXDeviceInfoList	ST_DMXDeviceInfo [▶ 69]	Array with the most important information of the found DMX devices.
uiNextDMX512StartAddress	UINT	If the DMX_OPTION_SET_START_ADDRESS option is activated, then the start address that will be assigned to the next DMX device will be displayed at this output.
iFoundDevices	INT	During the search, the current number of devices found will be available at this output.

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.1.2 FB_DMXDiscovery512



This function block searches for up to 512 DMX devices and automatically sets the start address (optional). The most important information for the devices found is displayed in a structure.

Inputs

```
VAR_INPUT
  bStart    : BOOL;
  dwOptions : DWORD;
END_VAR
```

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
dwOptions	DWORD	Options (see table). The individual constants must be linked with OR operators.

Constant	Description
DMX_OPTION_COMPLETE_NEW_DISCOVERY	All DMX devices are taken into account.
DMX_OPTION_SET_START_ADDRESS	The start address is set for all DMX devices that are found. Consecutive, starting with 1.
DMX_OPTION_OPTICAL_FEEDBACK	When a DMX device is found, the function IDENTIFY_DEVICE is called for two seconds.

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

```
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;
  iFoundDevices            : INT;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
uliLowerBoundSearchUID	T_ULARGE_INTEGER	During the search, the lower search address is sent to this output.
uliUpperBoundSearchUID	T_ULARGE_INTEGER	During the search, the upper search address is sent to this output.
arrDMXDeviceInfoList	ST_DMXDeviceInfo [▶ 69]	Array with the most important information of the found DMX devices.
uiNextDMX512StartAddress	UINT	If the DMX_OPTION_SET_START_ADDRESS option is activated, then the start address that will be assigned to the next DMX device will be displayed at this output.
iFoundDevices	INT	During the search, the current number of devices found will be available at this output.

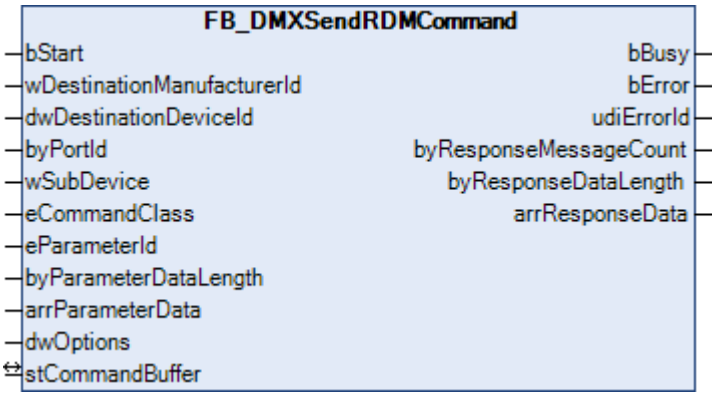
Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2 Low Level

4.1.2.1 Base

4.1.2.1.1 FB_DMXSendRDMCommand



This function block is for the general sending of a RDM command, defined by command number and, if necessary, transfer parameter.

Inputs

```
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;
END_VAR
```

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device.
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device.
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
wSubDevice	WORD	Sub-devices should be used in devices with a recurring number of modules, such as a dimmer rack. The sub-devices input enables parameter messages to be sent to a particular module within the device, in order to read or set module properties.
eCommandClass	E_DMXCommandClass [▶ 62]	Command Class (CC) indicates the message action (see E_DMXCommandClass).
eParameterId	E_DMXParameterId [▶ 63]	Parameter Id is a 16-bit number, which identifies a particular type of parameter data (see E_DMXParameterId).
byParameterDataLength	BYTE	The parameter data length (PDL) is the preceding number of slots, included in the parameter data area. If this input is 0x00, there is no parameter data to follow.
arrParameterData	BYTE	Parameter data of variable length. The format of the content depends on the PID.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

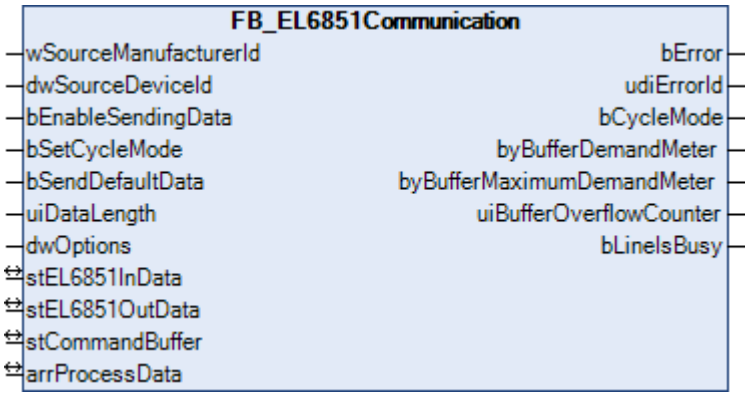
```
VAR_OUTPUT
  bBusy                  : BOOL;
  bError                 : BOOL;
  udiErrorId             : UDINT;
  byResponseMessageCount : BYTE;
  byResponseDataLength   : BYTE;
  arrResponseData        : ARRAY [0..255] OF BYTE;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
byResponseMessageCount	BYTE	This output indicates that the DMX slave contains further messages. The RDM command Get: QUEUES_MESSAG is used to read these messages.
byResponseDataLength	BYTE	Contains the number of bytes returned by the RDM command.
arrResponseData	BYTE	This output contains the data of the response from the RDM command. The length is variable, and the data format depends on the RDM command.

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.1.2 FB_EL6851Communication



Outdated

This function block is outdated. Use FB_EL6851CommunicationEx() [▶ 18] instead.

The EL6851 should always be accessed via this function block. This applies both to the transmission of the cyclic DMX data and to the transmission of the RDM commands.

If data is to be transmitted cyclically to the DMX devices, then set the *bEnableSendingData* input to TRUE, the *bSetCycleMode* input to TRUE, the *bSendDefaultData* input to FALSE and the *uiDataLength* input to the corresponding length (in bytes). The data to be transmitted can be specified via the *arrProcessData* variable.

If RDM commands are to be transmitted, then set the *bEnableSendingData* input to FALSE and the *bSetCycleMode* input to FALSE. The function blocks for the DMX/RDM commands do not directly access the EL6851 process image, but store the individual DMX/RDM commands in a buffer instead. The function block FB_EL6851Communication() reads the commands sequentially from this buffer and forwards them to the EL6851. This prevents multiple function blocks accessing the EL6851 process image at the same time. The buffer in which the DMX/RDM commands are stored is contained in a variable of type ST_DMXCommandBuffer. There is one instance of the function block FB_EL6851Communication() and one variable of type ST_DMXCommandBuffer per EL6851.

The extent to which the buffer is utilized can be determined from the outputs of the function block. If the buffer is regularly overflowing, you should analyze the level of utilization of the PLC task with the aid of the TwinCAT System Manager.

The function block FB_EL6851Communication() can be called in a separate faster task, if necessary. In this case, the faster task in which the FB_EL6851Communication() block is called should have a higher priority than the TASK in which the block for the RDM commands is called.

Examples for both operation modes can be found in the appendix.

Remarks concerning the IDs of DMX devices

Each DMX device has a unique, fixed, 48-bit long address, also called Unique ID or UID for short. This address is composed of the manufacturer ID (16-bit) and the device ID (32-bit). The manufacturer ID identifies the manufacturer of the device and is issued by the ESTA (Entertainment Services and Technology Association). A list of all known manufacturer IDs can be found at http://www.esta.org/tsp/working_groups/CP/mfctrIDs.php. The device ID is freely specified by the manufacturer. This is intended to ensure that each UID exists only once worldwide. The UID cannot normally be changed. The ESTA has given Beckhoff Automation the manufacturer ID 0x4241. Since the DMX master also has a UID, this should be specified in accordance with the ESTA (*wSourceManufacturerId* input).

Inputs

```
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;
END_VAR
```

Name	Type	Description
wSourceManufacturerId	WORD	Unique manufacturer Id of the DMX device. Should be 0x4241 according to the ESTA.
dwSourceDeviceId	DWORD	Unique device Id of the DMX device. Can be freely assigned.
bEnableSendingData	BOOL	If the terminal is in CycleMode (CycleMode output = TRUE), then transmission can be activated (TRUE) or blocked (FALSE) with this function block.
bSetCycleMode	BOOL	Activates the CycleMode. The cyclic process data can be transmitted to the DMX devices in CycleMode. CycleMode must be disabled in order to transmit the RDM/DMX commands.
bSendDefaultData	BOOL	The default values will be transmitted in CycleMode if this input is active (TRUE).
uiDataLength	UINT	This input is only relevant if CycleMode is active. It indicates the length of the DMX512 frame in bytes.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

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

Name	Type	Description
stEL6851InData	ST_EL6851InData [▶ 73]	Structure in the EL6851 input process image. It is used for communication from the EL6851 to the PLC.
stEL6851OutData	ST_EL6851OutData [▶ 73]	Structure in the EL6851 output process image. It is used for communication from the PLC to the EL6851.
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]
arrProcessData	BYTE	The data that are to be transmitted cyclically to the DMX devices are transferred to the function block via this variable. For this the CycleMode must be active (see also input *bSetCycleMode*).

Outputs

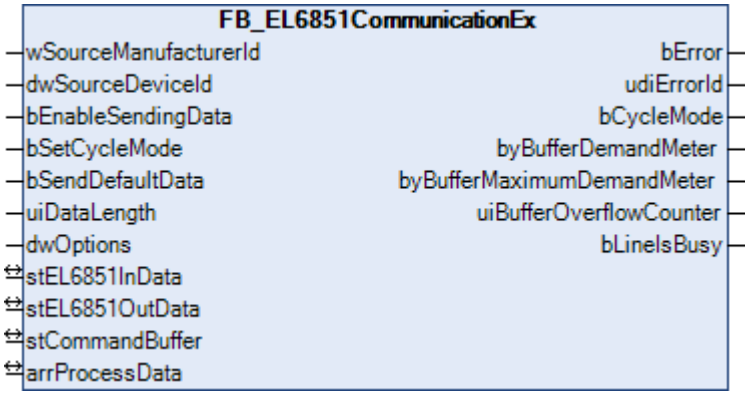
```
VAR_OUTPUT
  bError                      : BOOL;
  udiErrorId                  : UDINT;
  bCycleMode                  : BOOL;
  byBufferDemandMeter         : BYTE;
  byBufferMaximumDemandMeter  : BYTE;
  uiBufferOverflowCounter     : UINT;
  bLineIsBusy                 : BOOL;
END_VAR
```

Name	Type	Description
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
bCycleMode	BOOL	Is TRUE if CycleMode is active (see also *bSetCycleMode* input).
byBufferDemandMeter	BYTE	Demand of the respective buffer (0 - 100%).
byBufferMaximumDemandMeter	BYTE	Previous maximum demand of the respective buffer (0 - 100%).
uiBufferOverflowCounter	UINT	Number of buffer overflows to date.
bLineIsBusy	BOOL	This output is set as long as the FB_EL6851Communication() block is processing DMX/RDM commands.

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.1.3 FB_EL6851CommunicationEx



The EL6851 should always be accessed via this function block. This applies both to the transmission of the cyclic DMX data and to the transmission of the RDM commands.

If data is to be transmitted cyclically to the DMX devices, then set the *bEnableSendingData* input to TRUE, the *bSetCycleMode* input to TRUE, the *bSendDefaultData* input to FALSE and the *uiDataLength* input to the corresponding length (in bytes). The data to be transmitted can be specified via the *arrProcessData* variable.

If RDM commands are to be transmitted, then set the *bEnableSendingData* input to FALSE and the *bSetCycleMode* input to FALSE. The function blocks for the DMX/RDM commands do not directly access the EL6851 process image, but store the individual DMX/RDM commands in a buffer instead. The function block FB_EL6851CommunicationEx() reads the commands sequentially from this buffer and forwards them to the EL6851. This prevents multiple function blocks accessing the EL6851 process image at the same time. The buffer in which the DMX/RDM commands are stored is contained in a variable of type ST_DMXCommandBuffer. There is one instance of the function block FB_EL6851CommunicationEx() and one variable of type ST_DMXCommandBuffer per EL6851.

The extent to which the buffer is utilized can be determined from the outputs of the function block. If the buffer is regularly overflowing, you should analyze the level of utilization of the PLC task with the aid of the TwinCAT System Manager.

The function block FB_EL6851CommunicationEx() can be called in a separate faster task, if necessary. In this case, the faster task in which the function block FB_EL6851CommunicationEx() is called should have a higher priority than the TASK in which the function block for the RDM commands is called.

Examples for both operation modes can be found in the appendix.

Remarks concerning the IDs of DMX devices

Each DMX device has a unique, fixed, 48-bit long address, also called Unique ID or UID for short. This address is composed of the manufacturer ID (16-bit) and the device ID (32-bit). The manufacturer ID identifies the manufacturer of the device and is issued by the ESTA (Entertainment Services and Technology Association). A list of all known manufacturer IDs can be found at http://www.esta.org/tsp/working_groups/CP/mfctrIDs.php. The device ID is freely specified by the manufacturer. This is intended to ensure that each UID exists only once worldwide. The UID cannot normally be changed. The ESTA has given Beckhoff Automation the manufacturer ID 0x4241. Since the DMX master also has a UID, this should be specified in accordance with the ESTA (*wSourceManufacturerId* input).

Inputs

```
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;
END_VAR
```

Name	Type	Description
wSourceManufacturerId	WORD	Unique manufacturer Id of the DMX device. Should be 0x4241 according to the ESTA.
dwSourceDeviceId	DWORD	Unique device Id of the DMX device. Can be freely assigned.
bEnableSendingData	BOOL	If the terminal is in CycleMode (CycleMode output = TRUE), then transmission can be activated (TRUE) or blocked (FALSE) with this function block.
bSetCycleMode	BOOL	Activates the CycleMode. The cyclic process data can be transmitted to the DMX devices in CycleMode. CycleMode must be disabled in order to transmit the RDM/DMX commands.
bSendDefaultData	BOOL	The default values will be transmitted in CycleMode if this input is active (TRUE).
uiDataLength	UINT	This input is only relevant if CycleMode is active. It indicates the length of the DMX512 frame in bytes.
dwOptions	DWORD	Options (currently not used).

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

Inputs/outputs

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

Name	Type	Description
stEL6851InData	ST_EL6851InData [▶ 73]	Structure in the EL6851 input process image. It is used for communication from the EL6851 to the PLC.
stEL6851OutData	ST_EL6851OutData [▶ 73]	Structure in the EL6851 output process image. It is used for communication from the PLC to the EL6851.
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]
arrProcessData	BYTE	The data that are to be transmitted cyclically to the DMX devices are transferred to the function block via this variable. For this the CycleMode must be active (see also input *bSetCycleMode*).

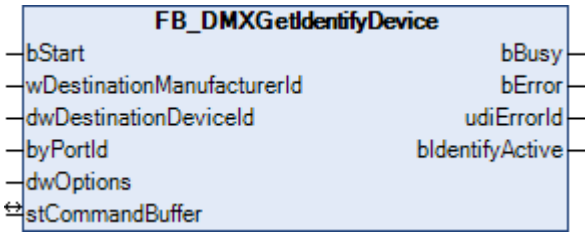
Outputs

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

Name	Type	Description
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
bCycleMode	BOOL	Is TRUE if CycleMode is active (see also *bSetCycleMode* input).
byBufferDemandMeter	BYTE	Demand of the respective buffer (0 - 100%).
byBufferMaximumDemandMeter	BYTE	Previous maximum demand of the respective buffer (0 - 100%).
uiBufferOverflowCounter	UINT	Number of buffer overflows to date.
bLineIsBusy	BOOL	This output is set as long as the FB_EL6851Communication() block is processing DMX/RDM commands.

4.1.2.2 Device Control Parameter Message

4.1.2.2.1 FB_DMXGetIdentifyDevice



This function block queries whether or not the identification of a DMX device is active.

Applying a positive edge to the *bStart* input starts the function block, and the *bBusy* output goes TRUE. The *wDestinationManufacturerId* and *dwDestinationDeviceId* inputs address the DMX device. The *byPortId* input defines the channel within the addressed DMX device. If the execution of the command has been completed, the *bBusy* output goes back to FALSE. The outputs *bError*, *udiErrorId* and *bIdentifyActive* can now be evaluated. Further positive edges at the *bStart* input will be ignored as long as the function block is active (*bBusy* is TRUE).

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device (for details, see DMX device address).
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device (for details, see DMX device address).
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

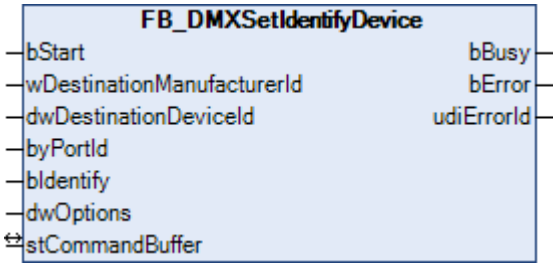
```
VAR_OUTPUT
  bBusy           : BOOL;
  bError          : BOOL;
  udiErrorId      : UDINT;
  bIdentifyActive : BOOL;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
bIdentifyActive	BOOL	If the execution of the command has been completed (*bBusy* is FALSE), then the state of identification of the DMX device is displayed at this output.

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.2.2 FB_DMXSetIdentifyDevice



This function block activates or deactivates the identification of a DMX device.

Applying a positive edge to the *bStart* input starts the function block, and the *bBusy* output goes TRUE. The *wDestinationManufacturerId* and *dwDestinationDeviceId* inputs address the DMX device. The *byPortId* input defines the channel within the addressed DMX device. If the execution of the command has been completed, the *bBusy* output goes back to FALSE. The outputs *bError* and *udiErrorId* can now be evaluated. Further positive edges at the *bStart* input will be ignored as long as the function block is active (*bBusy* is TRUE).

Inputs

```
VAR_INPUT
  bStart                      : BOOL;
  wDestinationManufacturerId  : WORD;
  dwDestinationDeviceId       : DWORD;
  byPortId                    : BYTE;
  bIdentify                   : BOOL := FALSE;
  dwOptions                   : DWORD := 0;
END_VAR
```

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device.
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device.
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
bIdentify	BOOL	This specifies whether the identification is to be activated (TRUE) or deactivated (FALSE).
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

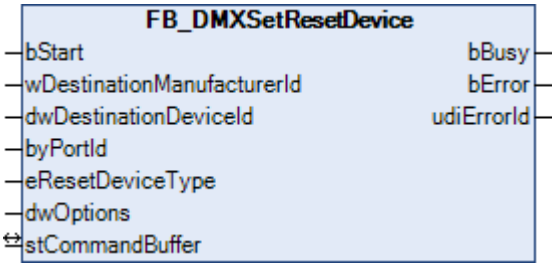
```
VAR_OUTPUT
  bBusy         : BOOL;
  bError        : BOOL;
  udiErrorId    : UDINT;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.2.3 FB_DMXSetResetDevice



This function block activates a reset in a DMX device.

Applying a positive edge to the *bStart* input starts the function block, and the *bBusy* output goes TRUE. The *wDestinationManufacturerId* and *dwDestinationDeviceId* inputs address the DMX device. The *byPortId* input defines the channel within the addressed DMX device. If the execution of the command has been completed, the *bBusy* output goes back to FALSE. The outputs *bError* and *udiErrorId* can now be evaluated. Further positive edges at the *bStart* input will be ignored as long as the function block is active (*bBusy* is TRUE).

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device.
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device.
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
eResetDeviceType	E_DMXResetDeviceType [▶ 65]	This specifies whether a warm start (eDMXResetDeviceTypeWarm) or a cold start (eDMXResetDeviceTypeCold) is to be performed. No other values are possible for this input.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

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

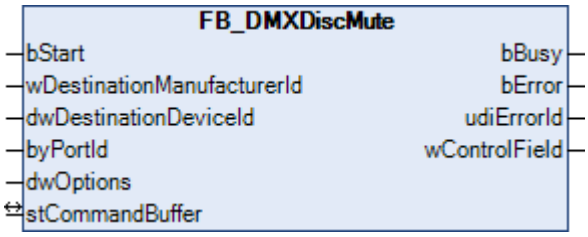
Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.3 Discovery Messages

4.1.2.3.1 FB_DMXDiscMute



This function block sets the mute flag of a DMX device. The mute flag specifies whether a DMX device reacts to the FB_DMXDiscUniqueBranch() [▶ 26] command (mute flag is not set) or not (mute flag is set).

Applying a positive edge to the *bStart* input starts the function block, and the *bBusy* output goes TRUE. The *wDestinationManufacturerId* and *dwDestinationDeviceId* inputs address the DMX device. The *byPortId* input defines the channel within the addressed DMX device. If the execution of the command has been completed, the *bBusy* output goes back to FALSE. The outputs *bError*, *udiErrorId* and *wControlField* can now be evaluated. Further positive edges at the *bStart* input will be ignored as long as the function block is active (*bBusy* is TRUE).

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device (for details, see DMX device address).
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device (for details, see DMX device address).
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

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

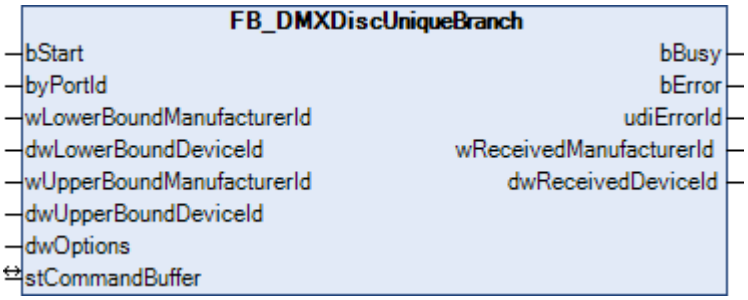
Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
wControlField	WORD	If the execution of the command has been completed (*bBusy* is FALSE), then further information about the DMX device will be available at this output. The meaning of the individual bits is defined as follows (see table):

Bit	Description
0 - Managed Proxy Flag	This bit is set if the DMX device is a proxy device.
1 - Sub-Device Flag	This bit is set if the DMX device supports sub-devices.
2 - Boot-Loader Flag	This bit is set if the DMX device cannot receive any commands (e.g. whilst the firmware is being loaded).
3 - Proxied Device Flag	This bit is set if the response was transmitted by a proxy device.
4 - 15	Reserve (always 0).

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.3.2 FB_DMXDiscUniqueBranch



This function block queries whether DMX devices are located within a certain address range. This command is used for the discovery of the connected DMX devices.

Applying a positive edge to the *bStart* input starts the function block, and the *bBusy* output goes TRUE. The inputs *wLowerBoundManufacturerId*, *dwLowerBoundDeviceId*, *wUpperBoundManufacturerId* and *dwUpperBoundDeviceId* define the address range in which DMX devices are searched. The *byPortId* input defines the channel within the addressed DMX device. If the execution of the command has been completed, the *bBusy* output goes back to FALSE. The outputs *bError*, *udiErrorId*, *wReceivedManufacturerId* and *dwReceivedDeviceId* can now be evaluated. Further positive edges at the *bStart* input will be ignored as long as the function block is active (*bBusy* is TRUE).

If there is only one DMX device in the defined address range, then the 48-bit UID of the DMX device will be returned via the outputs *wReceivedManufacturerId* and *dwReceivedDeviceId*. If no DMX devices are found in this range, the output *bError* is TRUE and *udiErrorId* is 0x8002 (no response from DMX device). If there are two or more DMX devices in the address range, *bError* is TRUE and *udiError* contains a 0x8006 (checksum error).

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
wLowerBoundManufacturerId	WORD	Unique manufacturer Id from the lower address range.
dwLowerBoundDeviceId	DWORD	Unique device Id from the lower address range.
wUpperBoundManufacturerId	WORD	Unique manufacturer Id from the upper address range.
dwUpperBoundDeviceId	DWORD	Unique device Id from upper address range.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

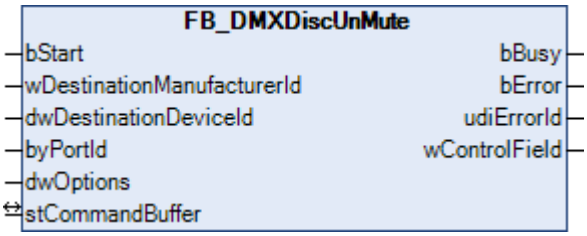
```
VAR_OUTPUT
  bBusy                   : BOOL;
  bError                  : BOOL;
  udiErrorId              : UDINT;
  wReceivedManufacturerId : WORD;
  dwReceivedDeviceId      : DWORD;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
wReceivedManufacturerId	WORD	If the execution of the command has been completed (*bBusy* is FALSE), then the state of identification of the DMX device is displayed at this output.
dwReceivedDeviceId	DWORD	If the execution of the command has been completed (*bBusy* is FALSE), then the state of identification of the DMX device is displayed at this output.

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.3.3 FB_DMXDiscUnMute



This function block resets the mute flag of a DMX device. The mute flag specifies whether a DMX device reacts to the FB_DMXDiscUniqueBranch() [▶ 26] command (mute flag is not set) or not (mute flag is set).

Applying a positive edge to the *bStart* input starts the function block, and the *bBusy* output goes TRUE. The *wDestinationManufacturerId* and *dwDestinationDeviceId* inputs address the DMX device. The *byPortId* input defines the channel within the addressed DMX device. If the execution of the command has been completed, the *bBusy* output goes back to FALSE. The outputs *bError*, *udiErrorId* and *wControlField* can now be evaluated. Further positive edges at the *bStart* input will be ignored as long as the function block is active (*bBusy* is TRUE).

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device (for details, see DMX device address).
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device (for details, see DMX device address).
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

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

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
wControlField	WORD	If the execution of the command has been completed (*bBusy* is FALSE), then further information about the DMX device will be available at this output. The meaning of the individual bits is defined as follows (see table):

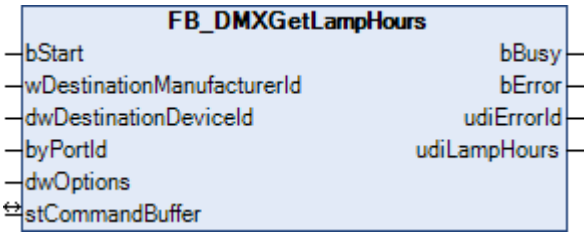
Bit	Description
0 - Managed Proxy Flag	This bit is set if the DMX device is a proxy device.
1 - Sub-Device Flag	This bit is set if the DMX device supports sub-devices.
2 - Boot-Loader Flag	This bit is set if the DMX device cannot receive any commands (e.g. whilst the firmware is being loaded).
3 - Proxied Device Flag	This bit is set if the response was transmitted by a proxy device.
4 - 15	Reserve (always 0).

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.4 Power and Lamp Setting Parameter Messages

4.1.2.4.1 FB_DMXGetLampHours



This function block reads the number of hours in which the lamp was on. The function block FB_DMXSetLampHours() [▶ 32] can be used to edit the hour counter.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device (for details, see DMX device address).
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device (for details, see DMX device address).
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

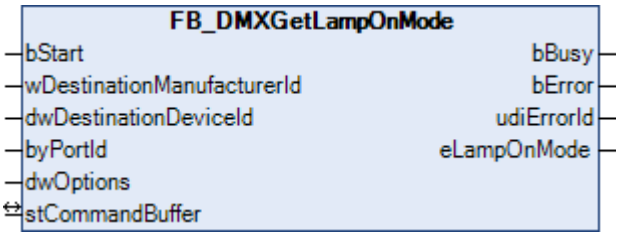
```
VAR_OUTPUT
  bBusy           : BOOL;
  bError          : BOOL;
  udiErrorId      : UDINT;
  udiLampHours    : UDINT;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
udiLampHours	UDINT	Number of hours in which the lamp was switched on.

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.4.2 FB_DMXGetLampOnMode



This function block reads the parameter that defines the switch-on behavior of the DMX device. The function block FB_DMXSetLampOnMode() [▶ 33] can be used to edit the value.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device (for details, see DMX device address).
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device (for details, see DMX device address).
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

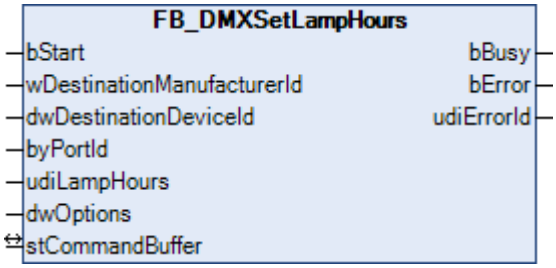
```
VAR_OUTPUT
  bBusy           : BOOL;
  bError          : BOOL;
  udiErrorId      : UDINT;
  eLampOnMode     : E_DMXLampOnMode;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
eLampOnMode	E_DMXLampOnMode [▶ 62]	Contains the current parameter that defines the switch-on behavior of the DMX device.

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.4.3 FB_DMXSetLampHours



This function block sets the operating hours counter for the lamp. The function block FB_DMXGetLampHours() [▶ 30] can be used to read the counter.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device.
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device.
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
udiLampHours	UDINT	New value for the operating hours counter.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

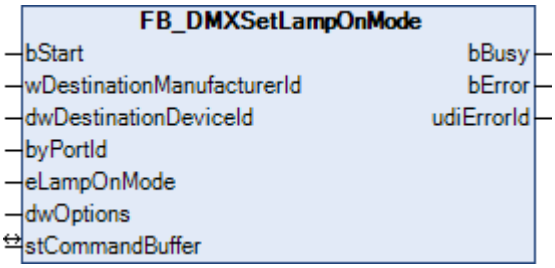
```
VAR_OUTPUT
  bBusy          : BOOL;
  bError         : BOOL;
  udiErrorId     : UDINT;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.4.4 FB_DMXSetLampOnMode



This function block defines the switch-on behavior of the DMX device. The function block FB_DMXGetLampOnMode() [▶ 31] can be used to read the set value.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device.
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device.
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
eLampOnMode	E_DMXLampOnMode [▶ 62]	This parameter defines the switch-on behavior of the DMX device.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

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

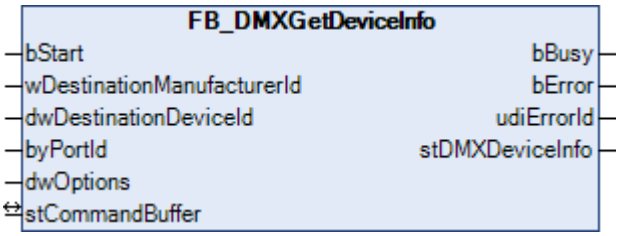
Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.5 Product Information Messages

4.1.2.5.1 FB_DMXGetDeviceInfo



This function block queries all relevant information from a DMX device.

Applying a positive edge to the *bStart* input starts the function block, and the *bBusy* output goes TRUE. The *wDestinationManufacturerId* and *dwDestinationDeviceId* inputs address the DMX device. The *byPortId* input defines the channel within the addressed DMX device. If the execution of the command has been completed, the *bBusy* output goes back to FALSE. The outputs *bError*, *udiErrorId* and *stDMXDeviceInfo* can now be evaluated. Further positive edges at the *bStart* input will be ignored as long as the function block is active (*bBusy* is TRUE).

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device (for details, see DMX device address).
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device (for details, see DMX device address).
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

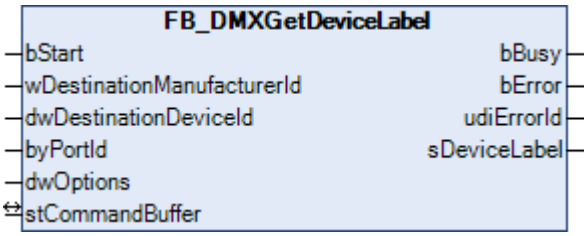
```
VAR_OUTPUT
  bBusy           : BOOL;
  bError          : BOOL;
  udiErrorId      : UDINT;
  stDMXDeviceInfo : ST_DMXDeviceInfo;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
stDMXDeviceInfo	ST_DMXDeviceInfo [▶ 69]	If the execution of the command has been completed (*bBusy* is FALSE), then all relevant information for the DMX device is sent to this output in a structure.

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.5.2 FB_DMXGetDeviceLabel



This function block reads a text from the DMX device, which contains a more detailed description of the device. The function block FB_DMXSetDeviceLabel() [▶ 43] can be used to edit the text.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device (for details, see DMX device address).
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device (for details, see DMX device address).
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

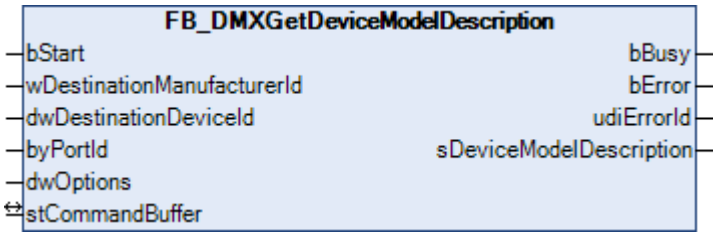
```
VAR_OUTPUT
  bBusy           : BOOL;
  bError          : BOOL;
  udiErrorId      : UDINT;
  sDeviceLabel    : STRING;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
sDeviceLabel	STRING	Description text for the DMX device.

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.5.3 FB_DMXGetDeviceModelDescription



This function block queries the description of the device type.

Applying a positive edge to the *bStart* input starts the function block, and the *bBusy* output goes TRUE. The *wDestinationManufacturerId* and *dwDestinationDeviceId* inputs address the DMX device. The *byPortId* input defines the channel within the addressed DMX device. If the execution of the command has been completed, the *bBusy* output goes back to FALSE. The outputs *bError*, *udiErrorId* and *sDeviceModelDescription* can now be evaluated. Further positive edges at the *bStart* input will be ignored as long as the function block is active (*bBusy* is TRUE).

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device (for details, see DMX device address).
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device (for details, see DMX device address).
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

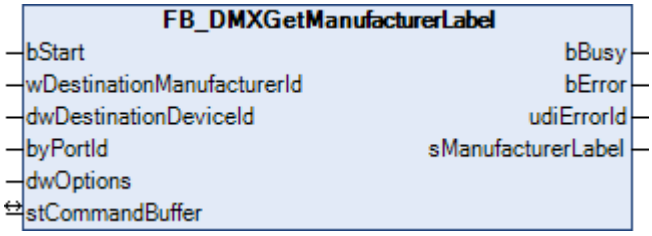
```
VAR_OUTPUT
  bBusy                   : BOOL;
  bError                  : BOOL;
  udiErrorId              : UDINT;
  sDeviceModelDescription : STRING;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
sDeviceModelDescription	STRING	If the execution of the command has been completed (*bBusy* is FALSE), then a description (maximum 32 characters) of the device type will be available at this output. The contents are specified by the DMX device manufacturer.

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.5.4 FB_DMXGetManufacturerLabel



This function block queries the description of the DMX device manufacturer.

Applying a positive edge to the *bStart* input starts the function block, and the *bBusy* output goes TRUE. The *wDestinationManufacturerId* and *dwDestinationDeviceId* inputs address the DMX device. The *byPortId* input defines the channel within the addressed DMX device. If the execution of the command has been completed, the *bBusy* output goes back to FALSE. The outputs *bError*, *udiErrorId* and *sManufacturerLabel* can now be evaluated. Further positive edges at the *bStart* input will be ignored as long as the function block is active (*bBusy* is TRUE).

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device (for details, see DMX device address).
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device (for details, see DMX device address).
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

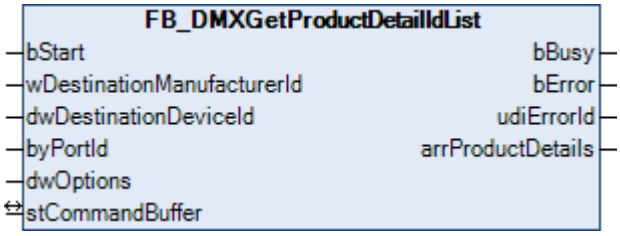
```
VAR_OUTPUT
  bBusy                  : BOOL;
  bError                 : BOOL;
  udiErrorId             : UDINT;
  sManufacturerLabel     : STRING;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
sManufacturerLabel	STRING	If the execution of the command has been completed (*bBusy* is FALSE), then a description (maximum 32 characters) of the DMX device manufacturer will be available at this output. The contents are specified by the DMX device manufacturer.

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.5.5 FB_DMXGetProductDetailIdList



This function block queries the categories to which the DMX device belongs.

RDM defines different device categories. Each DMX device can be assigned to up to 6 categories. The assignment is done by the device manufacturer and cannot be changed via RDM.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device (for details, see DMX device address).
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device (for details, see DMX device address).
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

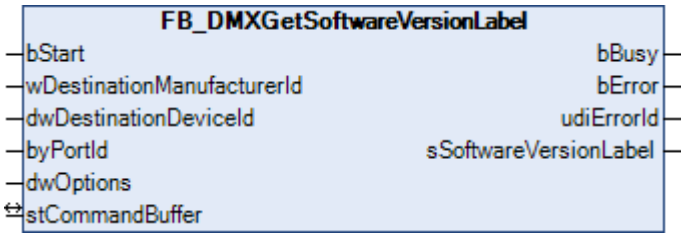
```
VAR_OUTPUT
  bBusy             : BOOL;
  bError            : BOOL;
  udiErrorId        : UDINT;
  arrProductDetails : ARRAY [1..6] OF E_DMXProductDetail;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
arrProductDetails	E_DMXProductDetail [▶ 64]	Contains the list with up to 6 device categories.

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.5.6 FB_DMXGetSoftwareVersionLabel



This function block queries the description of the software version of the DMX device.

Applying a positive edge to the *bStart* input starts the function block, and the *bBusy* output goes TRUE. The *wDestinationManufacturerId* and *dwDestinationDeviceId* inputs address the DMX device. The *byPortId* input defines the channel within the addressed DMX device. If the execution of the command has been completed, the *bBusy* output goes back to FALSE. The outputs *bError*, *udiErrorId* and *sSoftwareVersionLabel* can now be evaluated. Further positive edges at the *bStart* input will be ignored as long as the function block is active (*bBusy* is TRUE).

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device (for details, see DMX device address).
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device (for details, see DMX device address).
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

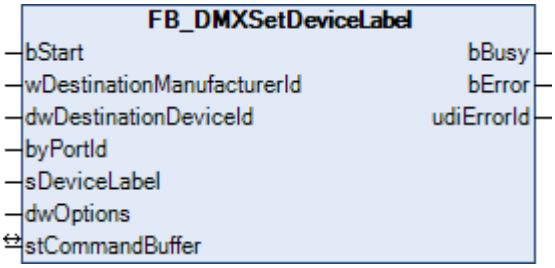
```
VAR_OUTPUT
  bBusy                   : BOOL;
  bError                  : BOOL;
  udiErrorId              : UDINT;
  sSoftwareVersionLabel   : STRING;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
sSoftwareVersionLabel	STRING	If the execution of the command has been completed (*bBusy* is FALSE), then a description (maximum 32 characters) of the software version of the DMX device will be available at this output. The contents are specified by the DMX device manufacturer.

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.5.7 FB_DMXSetDeviceLabel



This function block writes a description text into the DMX device. The function block FB_DMXGetDeviceLabel() [▶ 36] can be used to read the text.

Inputs

```
VAR_INPUT
  bStart                     : BOOL;
  wDestinationManufacturerId : WORD;
  dwDestinationDeviceId      : DWORD;
  byPortId                   : BYTE;
```

```
  sDeviceLabel                : STRING := '';
  dwOptions                   : DWORD := 0;
END_VAR
```

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device (for details, see DMX device address).
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device (for details, see DMX device address).
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
sDeviceLabel	STRING	New description text for the DMX device.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

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

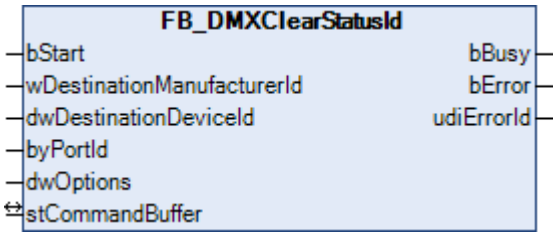
Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.6 Queued and Status Messages

4.1.2.6.1 FB_DMXClearStatusId



This function block clears the message buffer in the DMX device.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device (for details, see DMX device address).
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device (for details, see DMX device address).
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

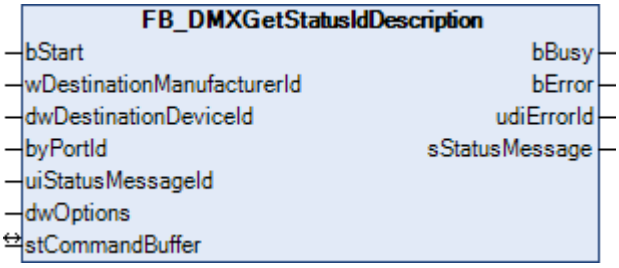
```
VAR_OUTPUT
  bBusy          : BOOL;
  bError         : BOOL;
  udiErrorId     : UDINT;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.6.2 FB_DMXGetStatusIdDescription



This function block reads the text of a certain status Id from the DMX device.

RDM defines some standard messages. Each of these messages has a unique status Id. The associated text can be read out from the DMX device with this function block.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device (for details, see DMX device address).
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device (for details, see DMX device address).
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
uiStatusMessageId	UINT	Status Id for which the text is to be read.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

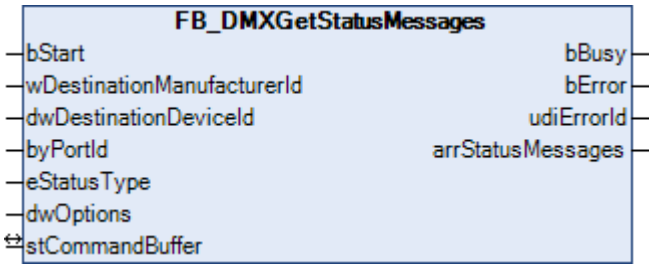
```
VAR_OUTPUT
  bBusy                  : BOOL;
  bError                 : BOOL;
  udiErrorId             : UDINT;
  sStatusMessage         : STRING;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
sStatusMessage	STRING	Status message

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.6.3 FB_DMXGetStatusMessages



This function block collects the status or error information of an DMX device.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device (for details, see DMX device address).
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device (for details, see DMX device address).
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
eStatusType	E_DMXStatusType [▶ 68]	Status type
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

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

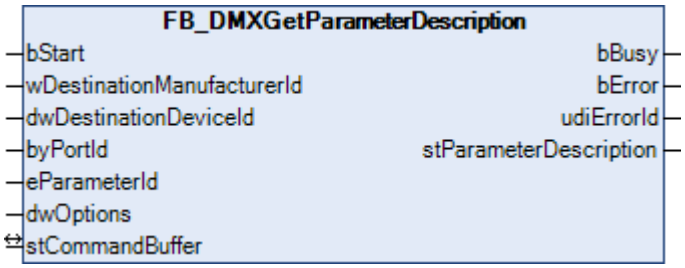
Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
arrStatusMessages	ST_DMXStatusMessage [▶ 72]	Once command execution is complete (*bBusy* is FALSE), all status/error information is available at this output.

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.7 RDM Information Messages

4.1.2.7.1 FB_DMXGetParameterDescription



This function block queries the definition of manufacturer-specific PIDs.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device (for details, see DMX device address).
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device (for details, see DMX device address).
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
eParameterId	E_DMXParameterId [▶ 63]	Requested manufacturer-specific PID.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

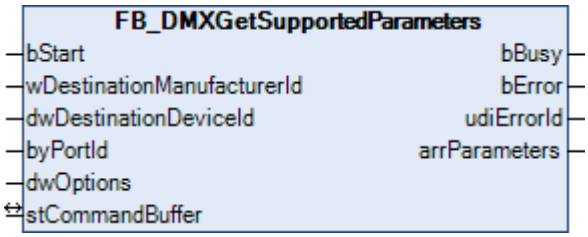
```
VAR_OUTPUT
  bBusy                   : BOOL;
  bError                  : BOOL;
  udiErrorId              : UDINT;
  stParameterDescription  : ST_DMXParameterDescription;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
stParameterDescription	ST_DMXParameterDescription [▶ 70]	Once command execution is complete (*bBusy* is FALSE), the information about the PID is available at this output.

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.7.2 FB_DMXGetSupportedParameters



This function block queries all supported parameters from a DMX device.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device (for details, see DMX device address).
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device (for details, see DMX device address).
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

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

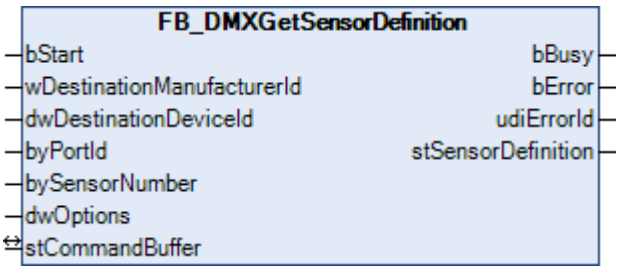
Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
arrParameters	E_DMXParameterId [▶ 63]	Once command execution is complete (*bBusy* is FALSE), all supported parameters for the DMX device are available at this output.

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.8 Sensor Parameter Messages

4.1.2.8.1 FB_DMXGetSensorDefinition



This function block queries the definition of a particular sensor.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device.
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device.
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
bySensorNumber	BYTE	DMX512 sensor number (0 - 254).
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

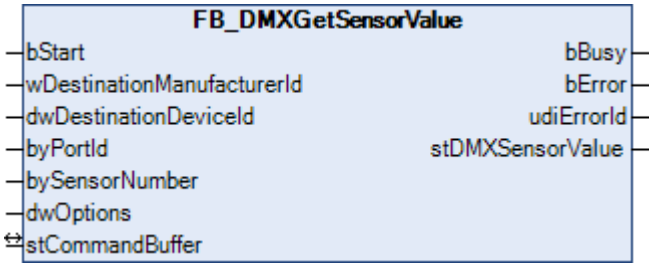
```
VAR_OUTPUT
  bBusy                   : BOOL;
  bError                  : BOOL;
  udiErrorId              : UDINT;
  stSensorDefinition      : ST_DMXSensorDefinition;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
stSensorDefinition	ST_DMXSensorDefinition [▶ 71]	Once command execution is complete (*bBusy* is FALSE), the definition of the sensor is available at this output.

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.8.2 FB_DMXGetSensorValue



This function block queries the current value of a sensor.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device.
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device.
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
bySensorNumber	BYTE	DMX512 sensor number (0 - 254).
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

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

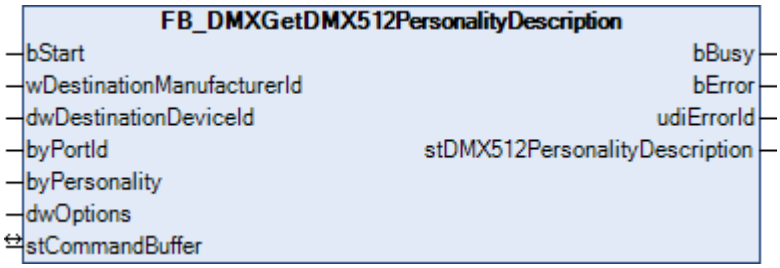
Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
stDMXSensorValue	ST_DMXSensorValue [▶ 72]	Structure with information about the current state of the sensor.

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.9 Setup Messages

4.1.2.9.1 FB_DMXGetDMX512PersonalityDescription



This function block reads further Personality information from the DMX device. Some DMX devices support so-called Personalities. Changing the Personality can influence certain RDM parameters.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device (for details, see DMX device address).
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device (for details, see DMX device address).
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
byPersonality	BYTE	The Personality for which information is queried.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

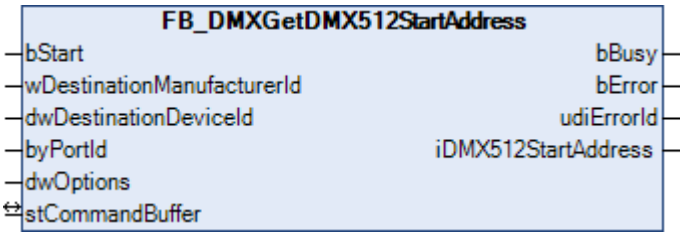
```
VAR_OUTPUT
  bBusy                          : BOOL;
  bError                         : BOOL;
  udiErrorId                     : UDINT;
  stDMX512PersonalityDescription : ST_DMX512PersonalityDescription;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
stDMX512PersonalityDescription	ST_DMX512PersonalityDescription [▶ 68]	Structure with information about the Personality.

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.9.2 FB_DMXGetDMX512StartAddress



This function block queries the DMX512 start address. This lies within the range from 1 – 512. If the DMX device does not occupy any DMX slot, then the DMX512 start address is 0xFFFF (65535). Each sub-device and the root device occupy different DMX512 start addresses.

Applying a positive edge to the *bStart* input starts the function block, and the *bBusy* output goes TRUE. The *wDestinationManufacturerId* and *dwDestinationDeviceId* inputs address the DMX device. The *byPortId* input defines the channel within the addressed DMX device. If the execution of the command has been completed, the *bBusy* output goes back to FALSE. The outputs *bError*, *udiErrorId* and *iDMX512StartAddress* can now be evaluated. Further positive edges at the *bStart* input will be ignored as long as the function block is active (*bBusy* is TRUE).

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device (for details, see DMX device address).
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device (for details, see DMX device address).
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

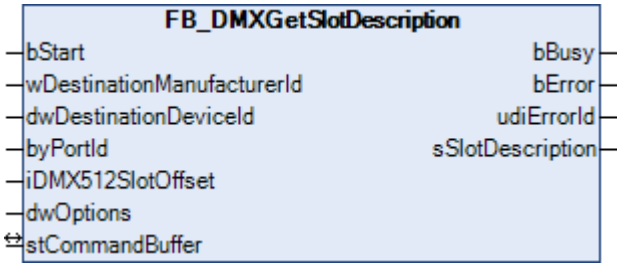
```
VAR_OUTPUT
  bBusy                   : BOOL;
  bError                  : BOOL;
  udiErrorId              : UDINT;
  iDMX512StartAddress     : INT;;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
iDMX512StartAddress	INT	If the execution of the command has been completed (*bBusy* is FALSE), then the DMX512 start address of the DMX device will be available at this output.

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.9.3 FB_DMXGetSlotDescription



This function block queries the text description for slot offsets.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device (for details, see DMX device address).
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device (for details, see DMX device address).
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
iDMX512SlotOffset	INT	DMX512 Slot Offset (0 - 511).
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

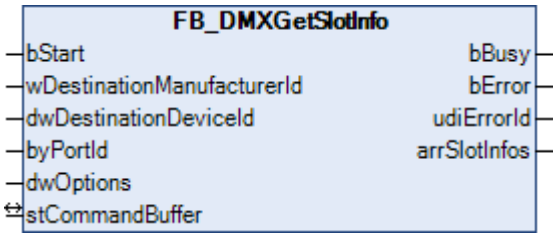
```
VAR_OUTPUT
  bBusy                  : BOOL;
  bError                 : BOOL;
  udiErrorId             : UDINT;
  sSlotDescription       : STRING;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
sSlotDescription	STRING	Once the command execution is complete (*bBusy* is FALSE), the slot description (maximum 32 characters) is available at this output.

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.9.4 FB_DMXGetSlotInfo



This function block queries the basic information of the functionality of the DMX512 slots of a DMX device.

Inputs

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

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device (for details, see DMX device address).
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device (for details, see DMX device address).
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

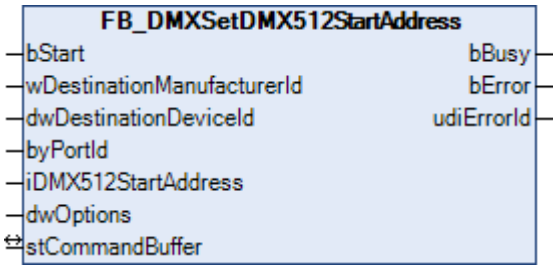
```
VAR_OUTPUT
  bBusy             : BOOL;
  bError            : BOOL;
  udiErrorId        : UDINT;
  arrSlotInfos      : ARRAY [0..45] OF ST_DMXSlotInfo;
END_VAR
```

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).
arrSlotInfos	ST_DMXSlotInfo [▶ 72]	Once the command execution is complete (*bBusy* is FALSE), all relevant information of the DMX512 slots is available as array at this output.

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.2.9.5 FB_DMXSetDMX512StartAddress



This function block sets the DMX512 start address. This lies within the range from 1 – 512. Each sub-device and the root device occupy different DMX512 start addresses.

Applying a positive edge to the *bStart* input starts the function block, and the *bBusy* output goes TRUE. The *wDestinationManufacturerId* and *dwDestinationDeviceId* inputs address the DMX device. The *byPortId* input defines the channel within the addressed DMX device. If the execution of the command has been completed, the *bBusy* output goes back to FALSE. The outputs *bError* and *udiErrorId* can now be evaluated. Further positive edges at the *bStart* input will be ignored as long as the function block is active (*bBusy* is TRUE).

Inputs

```
VAR_INPUT
  bStart                      : BOOL;
  wDestinationManufacturerId  : WORD;
  dwDestinationDeviceId       : DWORD;
  byPortId                    : BYTE;
  iDMX512Startadresse         : INT;
  dwOptions                   : DWORD := 0;
END_VAR
```

Name	Type	Description
bStart	BOOL	The function block is activated by a positive edge at this input.
wDestinationManufacturerId	WORD	Unique manufacturer Id of the DMX device (for details, see DMX device address).
dwDestinationDeviceId	DWORD	Unique device Id of the DMX device (for details, see DMX device address).
byPortId	BYTE	Channel within the addressed DMX device. Sub-devices are addressed through the Port Id. The root device always has the Port Id 0.
iDMX512StartAddress	INT	DMX512 start address (1 to 512).
dwOptions	DWORD	Options (currently not used).

Inputs/outputs

```
VAR_IN_OUT
  stCommandBuffer : ST_DMXCommandBuffer;
END_VAR
```

Name	Type	Description
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Reference to the structure for communication (buffer) with the function block FB_EL6851Communication() [▶ 15]

Outputs

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

Name	Type	Description
bBusy	BOOL	When the function block is activated the output is set, and it remains active until execution of the command has been completed.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in *udiErrorId*. Only valid if *bBusy* is FALSE.
udiErrorId	UDINT	Contains the command-specific error code of the most recently executed command. Only valid if *bBusy* is FALSE (see error codes [▶ 61]).

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.1.3 Error codes

Value (hex)	Value (dez)	Description
0x0000	0	No error.
0x8001	32769	No answer from the DMX terminal.
0x8002	32770	No answer from the DMX device.
0x8003	32771	Communication buffer overflow.
0x8004	32772	No answer from the communication block.
0x8005	32773	The *byPortId* parameter is outside the valid range.
0x8006	32774	Checksum error.
0x8007	32775	The *eResetDeviceType* parameter is outside the valid range.
0x8008	32776	Timeout.
0x8009	32777	The *uliLowerBoundUID* parameter is larger than the *uliUpperBoundUID.* parameter.
0x800A	32778	No RDM commands can be transmitted, because the terminal is in cycle mode.
0x800B	32779	The *iDMX512StartAddress* parameter is outside the valid range (1 – 512).
0x800C	32780	Error in setting the DMX512 start address.
0x800D	32781	No process data can be transmitted, because the terminal is not in cycle mode.
0x800E	32782	It is a RDM telegram received with the data length 0.
0x800F	32783	RDM response: Reply of the RDM telegramm ist invalid.
0x8010	32784	RDM response: The DMX slave cannot comply with request because the message is not implemented in responder.
0x8011	32785	RDM response: The DMX slave cannot interpret request as controller data was not formatted correctly.
0x8012	32786	RDM response: The DMX slave cannot comply due to an internal hardware fault.
0x8013	32787	RDM response: Proxy is not the RDM line master and cannot comply with message.
0x8014	32788	RDM response: SET Command normally allowed but being blocked currently.
0x8015	32789	RDM response: Not valid for *Command Class* attempted. May be used where GET allowed but SET is not supported.
0x8016	32790	RDM response: Value for given Parameter out of allowable range or not supported.
0x8017	32791	RDM response: Buffer or Queue space currently has no free space to store data.
0x8018	32792	RDM response: Incoming message exceeds buffer capacity.
0x8019	32793	RDM response: *Sub-Device* is out of range or unknown.
0x801A	32794	The *iDMX512SlotOffset* lparameter is outside the valid range (0-511).
0x801B	32795	The *bySensorNumber* parameter is outside the valid range (0-254).
0x801C	32796	RDM-Response: The field *Parameter Data (PD)* is too long. It was not possible to receive all the data of the reply. For this purpose, the function block FB_EL6851CommunicationEx() [▶ 18] must be used.
0x801D	32797	The ADS address to access the PDOs is invalid. Was the structure *AdsAddr* of the KL6851 mapped to the corresponding variable?
0x801E	32798	During read access to the PDOs an ADS error has occurred.

4.2 DUTs

4.2.1 Enums

4.2.1.1 E_DMXCommandClass

```
TYPE E_DMXCommandClass :
(
  eDMXCommandClassNotDefined                := 16#0000,
  eDMXCommandClassDiscoveryCommand          := 16#0010,
  eDMXCommandClassDiscoveryCommandResponse := 16#0011,
  eDMXCommandClassGetCommand                := 16#0020,
  eDMXCommandClassGetCommandResponse        := 16#0021,
  eDMXCommandClassSetCommand                := 16#0030,
  eDMXCommandClassSetCommandResponse        := 16#0031
);
END_TYPE
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.1.2 E_DMXDataType

```
TYPE E_DMXDataType :
(
  eDMXDataTypeNotDefined    := 16#0000,
  eDMXDataTypeBitField      := 16#0001,
  eDMXDataTypeASCII         := 16#0002,
  eDMXDataTypeUnsignedByte  := 16#0003,
  eDMXDataTypeSignedByte    := 16#0004,
  eDMXDataTypeUnsignedWord  := 16#0005,
  eDMXDataTypeSignedWord    := 16#0006,
  eDMXDataTypeUnsignedDWord := 16#0007,
  eDMXDataTypeSignedDWord   := 16#0008
);
END_TYPE
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.1.3 E_DMXLampOnMode

```
TYPE E_DMXLampOnMode :
(
  eDMXLampOnModeOff      := 0,
  eDMXLampOnModeDMX      := 1,
  eDMXLampOnModeOn       := 2,
  eDMXLampOnModeAfterCal := 3
);
END_TYPE
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.1.4 E_DMXParameterDescriptionCommandClass

```
TYPE E_DMXParameterDescriptionCommandClass :
(
  eDMXParameterDescriptionCommandClassGet    := 16#0001,
  eDMXParameterDescriptionCommandClassSet    := 16#0002,
```

```
  eDMXParameterDescriptionCommandClassGetSet := 16#0003
);
END_TYPE
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.1.5 E_DMXParameterId

```
TYPE E_DMXParameterId :
(
  eDMXParameterIdNone                          := 16#0000,
  eDMXParameterIdDiscUniqueBranch              := 16#0001,
  eDMXParameterIdDiscMute                      := 16#0002,
  eDMXParameterIdDiscUnMute                    := 16#0003,
  eDMXParameterIdProxiedDevices                := 16#0010,
  eDMXParameterIdProxiedDeviceCount            := 16#0011,
  eDMXParameterIdCommsStatus                   := 16#0015,
  eDMXParameterIdQueuedMessage                 := 16#0020,
  eDMXParameterIdStatusMessages                := 16#0030,
  eDMXParameterIdStatusIdDescription           := 16#0031,
  eDMXParameterIdClearStatusId                 := 16#0032,
  eDMXParameterIdSubDeviceStatusReportThreshold := 16#0033,
  eDMXParameterIdSupportedParamaters           := 16#0050,
  eDMXParameterIdParameterDescription          := 16#0051,
  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,
  eDMXParameterIdDMXPersonality                := 16#00E0,
  eDMXParameterIdDMXPersonalityDescription     := 16#00E1,
  eDMXParameterIdDMXStartAddress               := 16#00F0,
  eDMXParameterIdSlotInfo                      := 16#0120,
  eDMXParameterIdSlotDescription               := 16#0121,
  eDMXParameterIdDefaultSlotValue              := 16#0122,
  eDMXParameterIdSensorDefinition              := 16#0200,
  eDMXParameterIdSensorValue                   := 16#0201,
  eDMXParameterIdRecordSensors                 := 16#0202,
  eDMXParameterIdDeviceHours                   := 16#0400,
  eDMXParameterIdLampHours                     := 16#0401,
  eDMXParameterIdLampStrikes                   := 16#0402,
  eDMXParameterIdLampState                     := 16#0403,
  eDMXParameterIdLampOnMode                    := 16#0404,
  eDMXParameterIdDevicePowerCycles             := 16#0405,
  eDMXParameterIdDisplayInvert                 := 16#0500,
  eDMXParameterIdDisplayLevel                  := 16#0501,
  eDMXParameterIdPanInvert                     := 16#0600,
  eDMXParameterIdTiltInvert                    := 16#0601,
  eDMXParameterIdPanTiltSwap                   := 16#0602,
  eDMXParameterIdRealTimeClock                 := 16#0603,
  eDMXParameterIdIdentifyDevice                := 16#1000,
  eDMXParameterIdResetDevice                   := 16#1001,
  eDMXParameterIdPowerState                    := 16#1010,
  eDMXParameterIdPerformSelftest               := 16#1020,
  eDMXParameterIdSelfTestDescription           := 16#1021,
  eDMXParameterIdCapturePreset                 := 16#1030,
  eDMXParameterIdPresetPlayBack                := 16#1031
);
END_TYPE
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.1.6 E_DMXProductDetail

```
TYPE E_DMXProductDetail :
(
  eDMXProductDetailNotDeclared          := 16#0000,
  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,
  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,
  eDMXProductDetailVideo                := 16#0200,
  eDMXProductDetailSlide                := 16#0201,
  eDMXProductDetailFilm                 := 16#0202,
  eDMXProductDetailOilwheel             := 16#0203,
  eDMXProductDetailLCDGate              := 16#0204,
  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,
  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,
  eDMXProductDetailMirrorballRotator    := 16#0500,
  eDMXProductDetailOtherRotator         := 16#0501,
  eDMXProductDetailKabukiDrop           := 16#0502,
  eDMXProductDetailCurtain              := 16#0503,
  eDMXProductDetailLineset              := 16#0504,
  eDMXProductDetailMotorControl         := 16#0505,
  eDMXProductDetailDamperControl        := 16#0506,
  eDMXProductDetailSplitter             := 16#0600,
  eDMXProductDetailEthernetNode         := 16#0601,
  eDMXProductDetailMerge                := 16#0602,
  eDMXProductDetailDatapatch            := 16#0603,
  eDMXProductDetailWirelessLink         := 16#0604,
  eDMXProductDetailProtocolConvertor    := 16#0701,
  eDMXProductDetailAnalogDemultiplex    := 16#0702,
  eDMXProductDetailAnalogMultiplex      := 16#0703,
  eDMXProductDetailSwitchPanel          := 16#0704,
  eDMXProductDetailRouter               := 16#0800,
  eDMXProductDetailFader                := 16#0801,
  eDMXProductDetailMixer                := 16#0802,
  eDMXProductDetailChangeoverManual     := 16#0900,
  eDMXProductDetailChangeoverAuto       := 16#0901,
  eDMXProductDetailTest                 := 16#0902,
```

```
  eDMXProductDetailGfiRcd                := 16#0A00,
  eDMXProductDetailBattery               := 16#0A01,
  eDMXProductDetailControllableBreaker   := 16#0A02,
  eDMXProductDetailOther                 := 16#7FFF
);
END_TYPE
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.1.7 E_DMXResetDeviceType

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

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.1.8 E_DMXResponseType

```
TYPE E_DMXResponseType :
(
  eDMXResponseTypeAck          := 16#0000,
  eDMXResponseTypeAckTimer     := 16#0001,
  eDMXResponseTypeNackReason   := 16#0002,
  eDMXResponseTypeAckOverflow  := 16#0003
);
END_TYPE
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.1.9 E_DMXSensorType

```
TYPE E_DMXSensorType :
(
  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
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.1.10 E_DMXSensorUnit

```
TYPE E_DMXSensorUnit :
(
  eDMXSensorUnitNone                      := 16#00,
  eDMXSensorUnitCentigrade                := 16#01,
  eDMXSensorUnitVoltsDC                   := 16#02,
  eDMXSensorUnitVoltsACPeak               := 16#03,
  eDMXSensorUnitVoltsACRms                := 16#04,
  eDMXSensorUnitAmpereDC                  := 16#05,
  eDMXSensorUnitAmpereACPeak              := 16#06,
  eDMXSensorUnitAmpereACRms               := 16#07,
  eDMXSensorUnitHertz                     := 16#08,
  eDMXSensorUnitOhm                       := 16#09,
  eDMXSensorUnitWatt                      := 16#0A,
  eDMXSensorUnitKilogram                  := 16#0B,
  eDMXSensorUnitMeters                    := 16#0C,
  eDMXSensorUnitMetersSquared             := 16#0D,
  eDMXSensorUnitMetersCubed               := 16#0E,
  eDMXSensorUnitKilogrammesPerMeterCubed := 16#0F,
  eDMXSensorUnitMetersPerSecond           := 16#10,
  eDMXSensorUnitMetersPerSecondSquared    := 16#11,
  eDMXSensorUnitNewton                    := 16#12,
  eDMXSensorUnitJoule                     := 16#13,
  eDMXSensorUnitPascal                    := 16#14,
  eDMXSensorUnitSecond                    := 16#15,
  eDMXSensorUnitDegree                    := 16#16,
  eDMXSensorUnitSteradian                 := 16#17,
  eDMXSensorUnitCandela                   := 16#18,
  eDMXSensorUnitLumen                     := 16#19,
  eDMXSensorUnitLux                       := 16#1A,
  eDMXSensorUnitIre                       := 16#1B,
  eDMXSensorUnitByte                      := 16#1C
);
END_TYPE
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.1.11 E_DMXSensorUnitPrefix

```
TYPE E_DMXSensorUnitPrefix :
(
  eDMXSensorUnitPrefixNone   := 16#00,
  eDMXSensorUnitPrefixDeci   := 16#01,
  eDMXSensorUnitPrefixCenti  := 16#02,
  eDMXSensorUnitPrefixMilli  := 16#03,
  eDMXSensorUnitPrefixMicro  := 16#04,
  eDMXSensorUnitPrefixNano   := 16#05,
  eDMXSensorUnitPrefixPico   := 16#06,
  eDMXSensorUnitPrefixFempto := 16#07,
  eDMXSensorUnitPrefixAtto   := 16#08,
  eDMXSensorUnitPrefixZepto  := 16#09,
  eDMXSensorUnitPrefixYocto  := 16#0A,
  eDMXSensorUnitPrefixDeca   := 16#11,
```

```
  eDMXSensorUnitPrefixHecto  := 16#12,
  eDMXSensorUnitPrefixKilo   := 16#13,
  eDMXSensorUnitPrefixMega   := 16#14,
  eDMXSensorUnitPrefixGiga   := 16#15,
  eDMXSensorUnitPrefixTerra  := 16#16,
  eDMXSensorUnitPrefixPeta   := 16#17,
  eDMXSensorUnitPrefixExa    := 16#18,
  eDMXSensorUnitPrefixZetta  := 16#19,
  eDMXSensorUnitPrefixYotta  := 16#1A
);
END_TYPE
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.1.12 E_DMXSlotDefinition

```
TYPE E_DMXSlotDefinition :
(
  eDMXSlotDefinitionIntensity       := 16#0001,
  eDMXSlotDefinitionIntensityMaster := 16#0002,
  eDMXSlotDefinitionPan             := 16#0101,
  eDMXSlotDefinitionTilt            := 16#0102,
  eDMXSlotDefinitionColorWheel      := 16#0201,
  eDMXSlotDefinitionColorSubCyan    := 16#0202,
  eDMXSlotDefinitionColorSubYellow  := 16#0203,
  eDMXSlotDefinitionColorSubMagenta := 16#0204,
  eDMXSlotDefinitionColorAddRed     := 16#0205,
  eDMXSlotDefinitionColorAddGreen   := 16#0206,
  eDMXSlotDefinitionColorAddBlue    := 16#0207,
  eDMXSlotDefinitionColorCorrection := 16#0208,
  eDMXSlotDefinitionColorScroll     := 16#0209,
  eDMXSlotDefinitionColorSemaphore  := 16#0210,
  eDMXSlotDefinitionStaticGoboWheel := 16#0301,
  eDMXSlotDefinitionRotoGoboWheel   := 16#0302,
  eDMXSlotDefinitionPrismWheel      := 16#0303,
  eDMXSlotDefinitionEffectsWheel    := 16#0304,
  eDMXSlotDefinitionBeamSizeIris    := 16#0401,
  eDMXSlotDefinitionEdge            := 16#0402,
  eDMXSlotDefinitionFrost           := 16#0403,
  eDMXSlotDefinitionStrobe          := 16#0404,
  eDMXSlotDefinitionZoom            := 16#0405,
  eDMXSlotDefinitionFramingShutter  := 16#0406,
  eDMXSlotDefinitionShutterRotate   := 16#0407,
  eDMXSlotDefinitionDouser          := 16#0408,
  eDMXSlotDefinitionBarnDoor        := 16#0409,
  eDMXSlotDefinitionLampControl     := 16#0501,
  eDMXSlotDefinitionFixtureControl  := 16#0502,
  eDMXSlotDefinitionFixtureSpeed    := 16#0503,
  eDMXSlotDefinitionMacro           := 16#0504,
  eDMXSlotDefinitionUndefined       := 16#FFFF
);
END_TYPE
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.1.13 E_DMXSlotType

```
TYPE E_DMXSlotType :
(
  eDMXSlotTypePrinary        := 0,
  eDMXSlotTypeSecFine        := 1,
  eDMXSlotTypeSecTiming      := 2,
  eDMXSlotTypeSecSpeed       := 3,
  eDMXSlotTypeSecControl     := 4,
  eDMXSlotTypeSecIndex       := 5,
  eDMXSlotTypeSecRotation    := 6,
  eDMXSlotTypeSecIndexRotate := 7,
```

```
  eDMXSlotTypeSecUndefined   := 255
);
END_TYPE
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.1.14 E_DMXStatusType

```
TYPE E_DMXStatusType :
(
  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
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.2 Structures

4.2.2.1 ST_DMX512Personality

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

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.2.2 ST_DMX512PersonalityDescription

```
TYPE ST_DMX512PersonalityDescription :
STRUCT
  iDMX512SlotsRequired : INT;
  sDescription         : STRING;
END_STRUCT
END_TYPE
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.2.3 ST_DMXCommandBuffer

```
TYPE ST_DMXCommandBuffer :
STRUCT
  arrMessageQueue     : ST_DMXMessageQueue;
  stResponseTable     : ST_DMXResponseTable;
```

```
  byTransactionNumber : BYTE;
END_STRUCT
END_TYPE
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.2.4 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
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.2.5 ST_DMXMac

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

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.2.6 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
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.2.7 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
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.2.8 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
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.2.9 ST_DMXProductCategory

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

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.2.10 ST_DMXRDMProtocolVersion

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

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.2.11 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
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.2.12 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
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.2.13 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
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.2.14 ST_DMXSensorValue

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

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.2.15 ST_DMXSlotInfo

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

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.2.16 ST_DMXStatusMessage

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

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.2.17 ST_EL6851AdsAddr

```
TYPE ST_EL6851AdsAddr :
STRUCT
  arrNetId : ARRAY [0..5] OF USINT;
  uiPort   : UINT;
END_STRUCT
END_TYPE
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.2.18 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
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.2.19 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
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.2.2.20 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
```

Requirements

Development environment	required TC3 PLC library
TwinCAT from v3.1.4020.14	Tc2_DMX from 3.5.3.0

4.3 Integration into TwinCAT

4.3.1 EL6851 with CX5120

This sample explains how to write a simple PLC program for DMX in TwinCAT and how to link it with the hardware. Search for DMX devices.

Sample: https://infosys.beckhoff.com/content/1033/tcplclib_tc2_dmx/Resources/6202185099/.zip

Hardware

Setting up the components

The following hardware is required:

- 1x CX5120 Embedded PC
- 1x EL6851 DMX Master terminal
- 1x EL9011 end cap

Set up the hardware and the DMX components as described in the respective documents.

Software

Creation of the PLC program

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

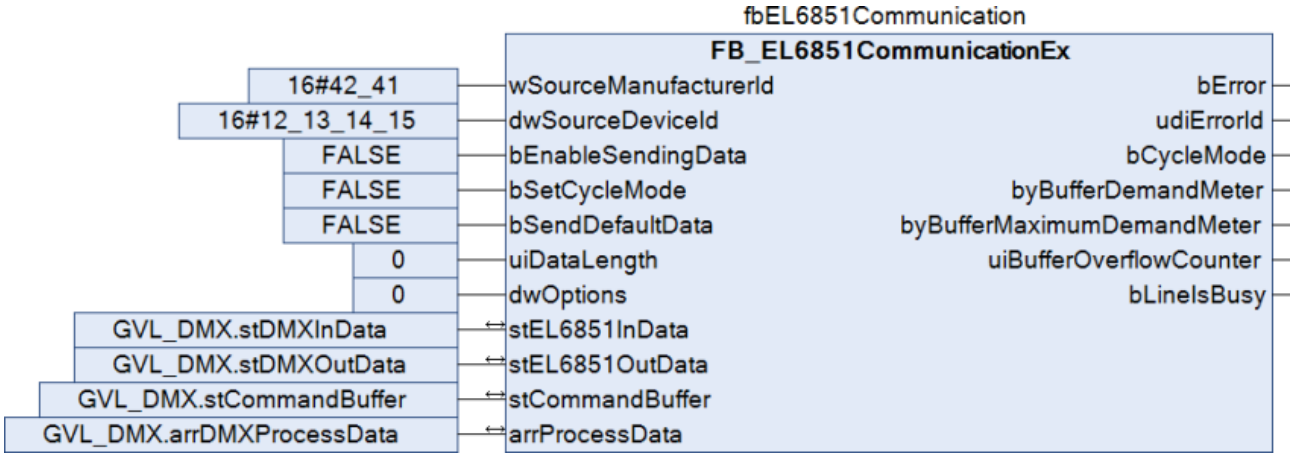
Add the library Tc2_DMX in the PLC project under "References".

Generate a global variable list with the name GVL_EIB and create the following variables:

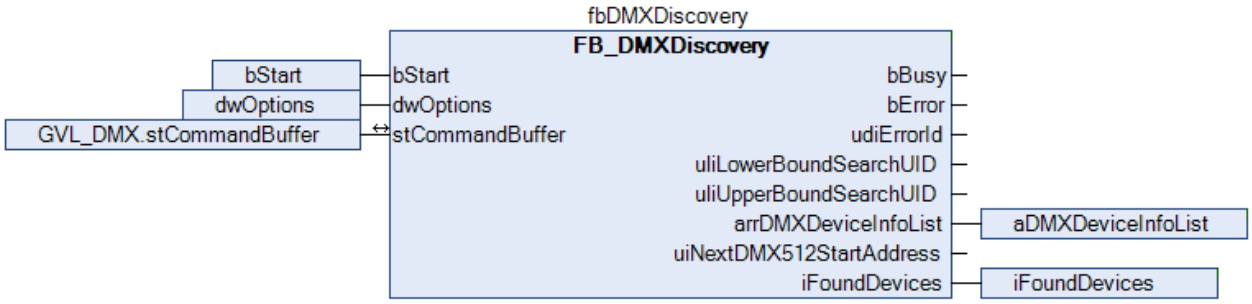
```
VAR_GLOBAL
    stEL6851InData      AT %I* : ST_EL6851InDataEx;
    stKL6851OutData     AT %Q* : ST_EL6851OutData;
    stCommandBuffer            : ST_DMXCommandBuffer;
    arrDMXProcessData          : ARRAY [1..512] OF BYTE;
END_VAR
```

Name	Type	Description
stEL6851InData	ST_EL6851InDataEx [▶ 73]	Input variable for the DMX terminal.
stEL6851OutData	ST_EL6851OutData [▶ 73]	Output variable for the DMX terminal.
stCommandBuffer	ST_DMXCommandBuffer [▶ 68]	Required for communication with DMX.
arrDMXProcessData	BYTE	The data that are to be transmitted cyclically to the DMX devices are transferred to the function block via this variable. For this the CycleMode must be active (see also input *bSetCycleMode*).

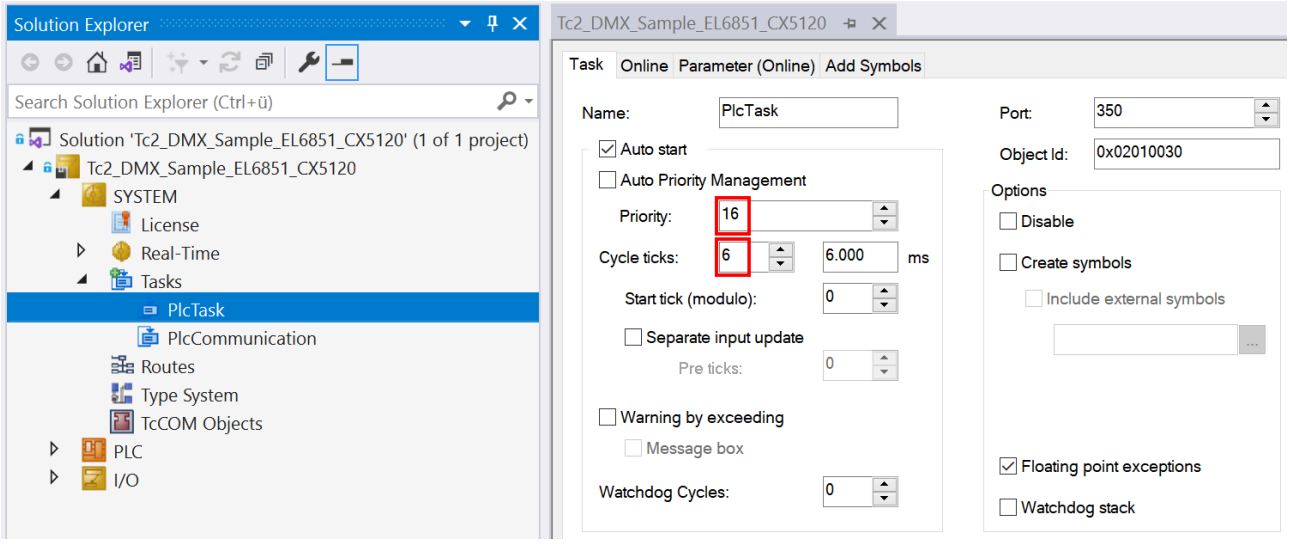
Then create a program (CFC) for background communication with DMX. In this program the function block FB_EL6851CommunicationEx [▶ 18] is called. Make sure to link the communication block with *stEL6851InData*, *stEL6851OutData* and *stCommandBuffer*.



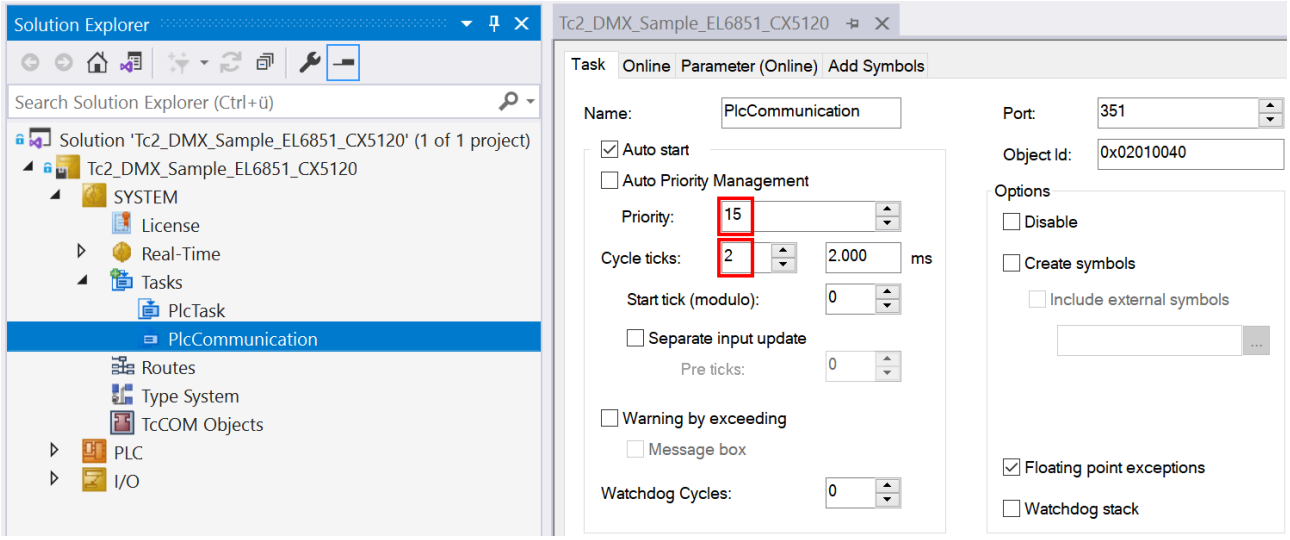
Create a MAIN program (CFC) in which the function block FB_DMXDiscovery512 [▶ 11] is called. The input *stCommandBuffer* of the function block is linked with the global variable *stCommandBuffer*. If the variable *bStart* is TRUE, the search for DMX devices is started. The number of found devices is stored in variable *iFoundDevices* and additional information in *aDMXDeviceInfoList*.



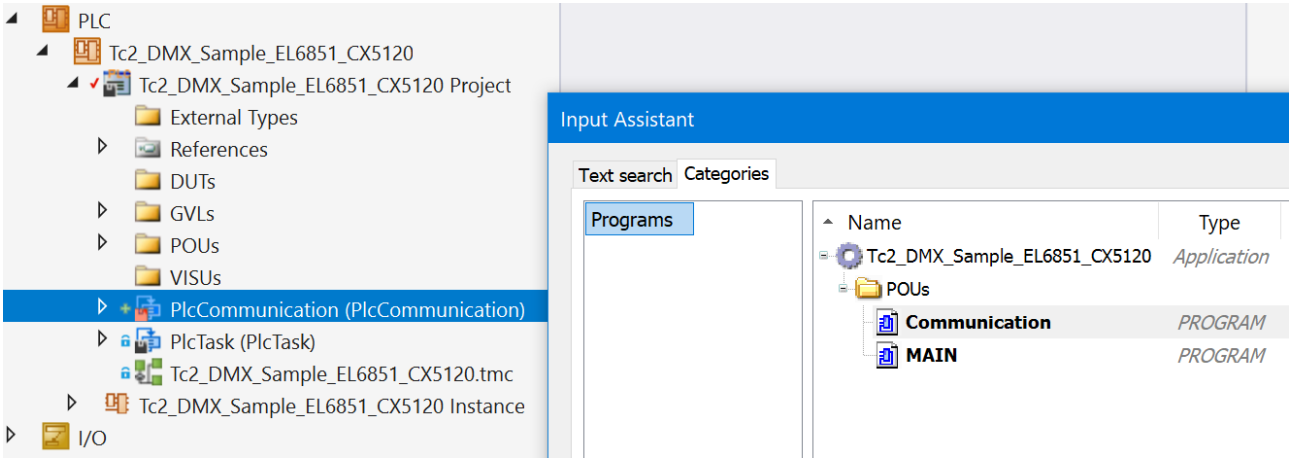
Navigate to the task configuration section and configure the PlcTask. By way of example, the task is assigned priority 16 and a cycle time of 6 ms.



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



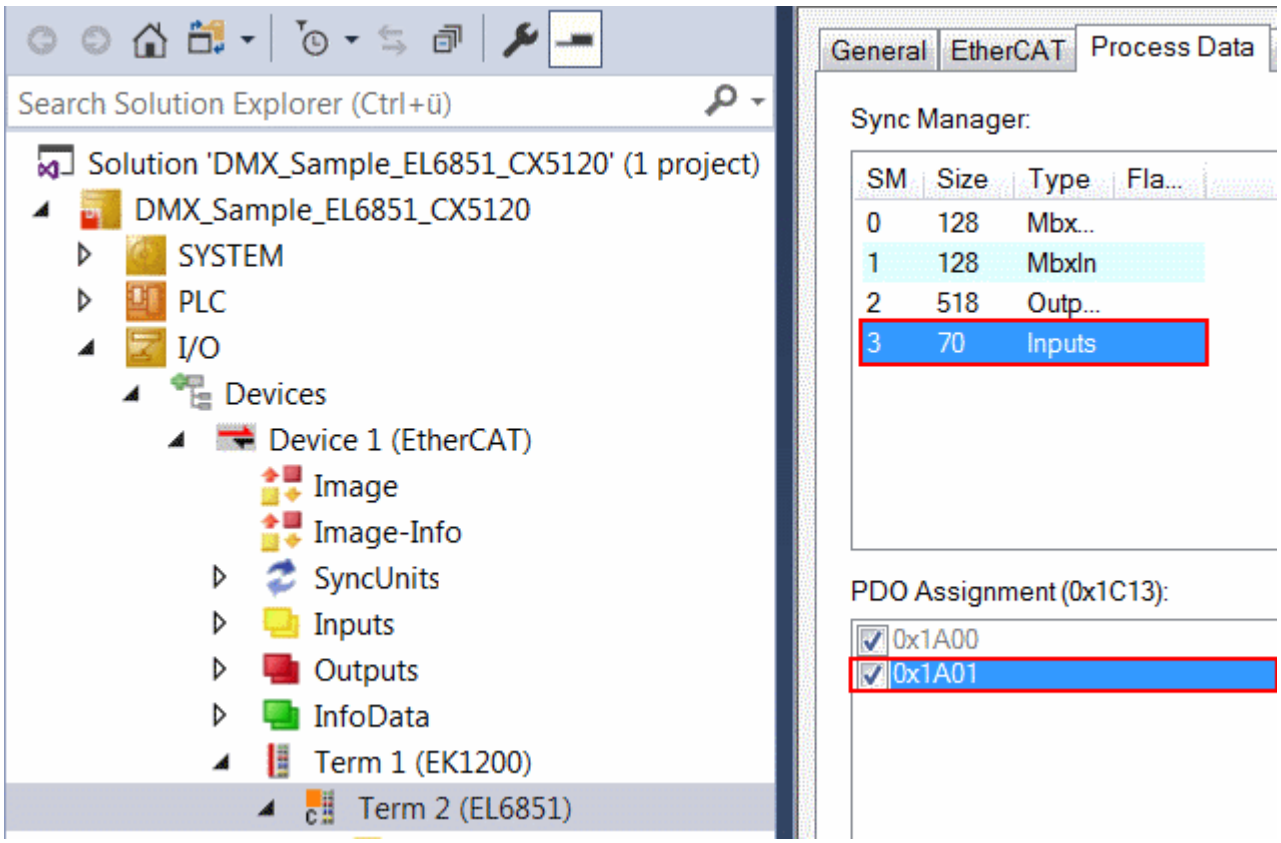
Add the program for the communication to this task. More precise information on the task configuration can be found in the description of the function block FB_EL6851CommunicationEx [▶ 18].



I/O configuration

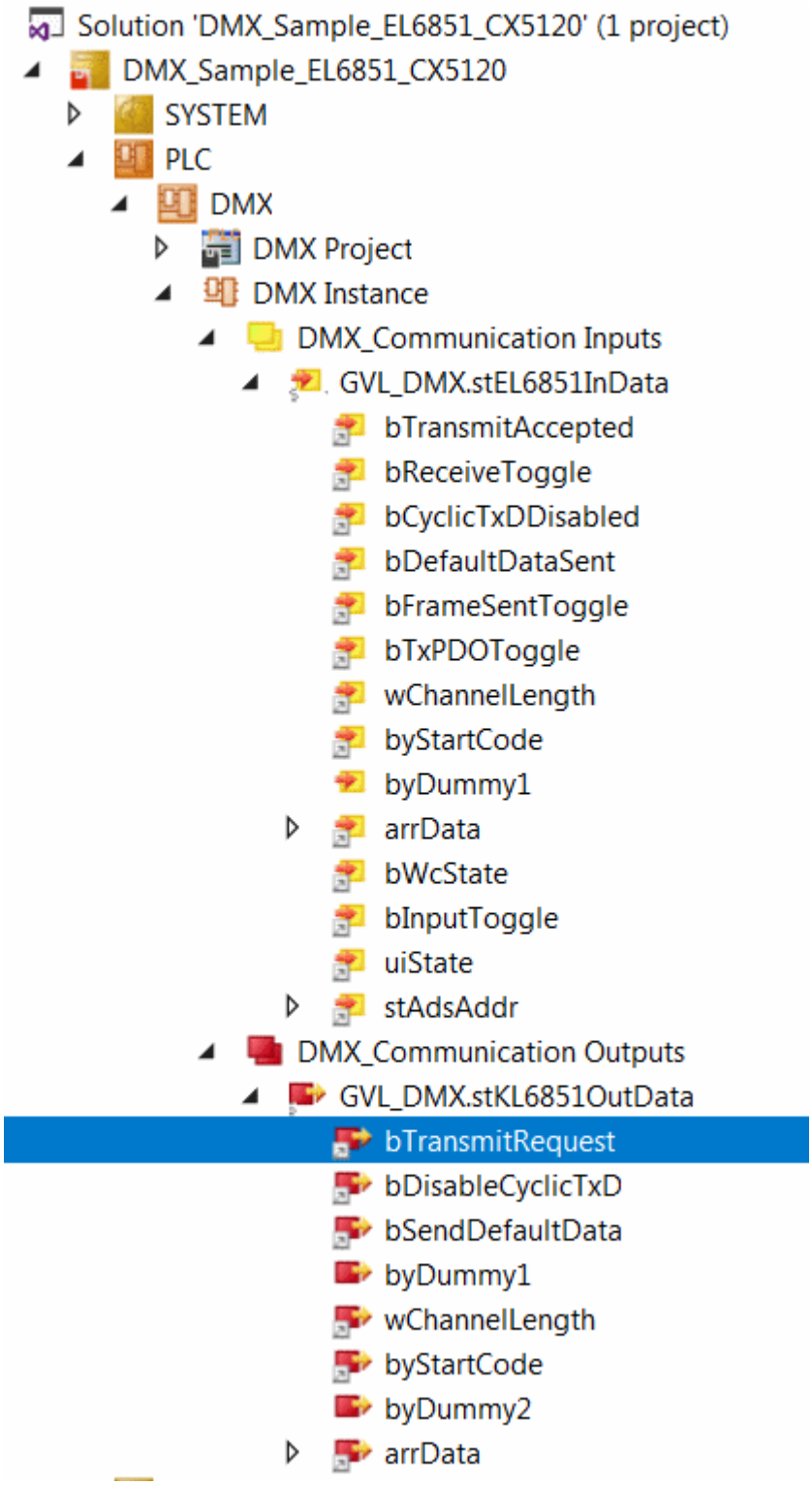
Select the CX as target system and initiate a search for its hardware.

Make the DMX inputs 1 to 64 available by opening the *Process data* tab of the EL6851 and selecting the option *0x1A01* under *Inputs*.



In the project instance within the PLC section, you can see that the input and output variables are assigned to the corresponding tasks.

Link the global variables of the PLC program with the inputs and outputs of the EtherCAT Terminal.



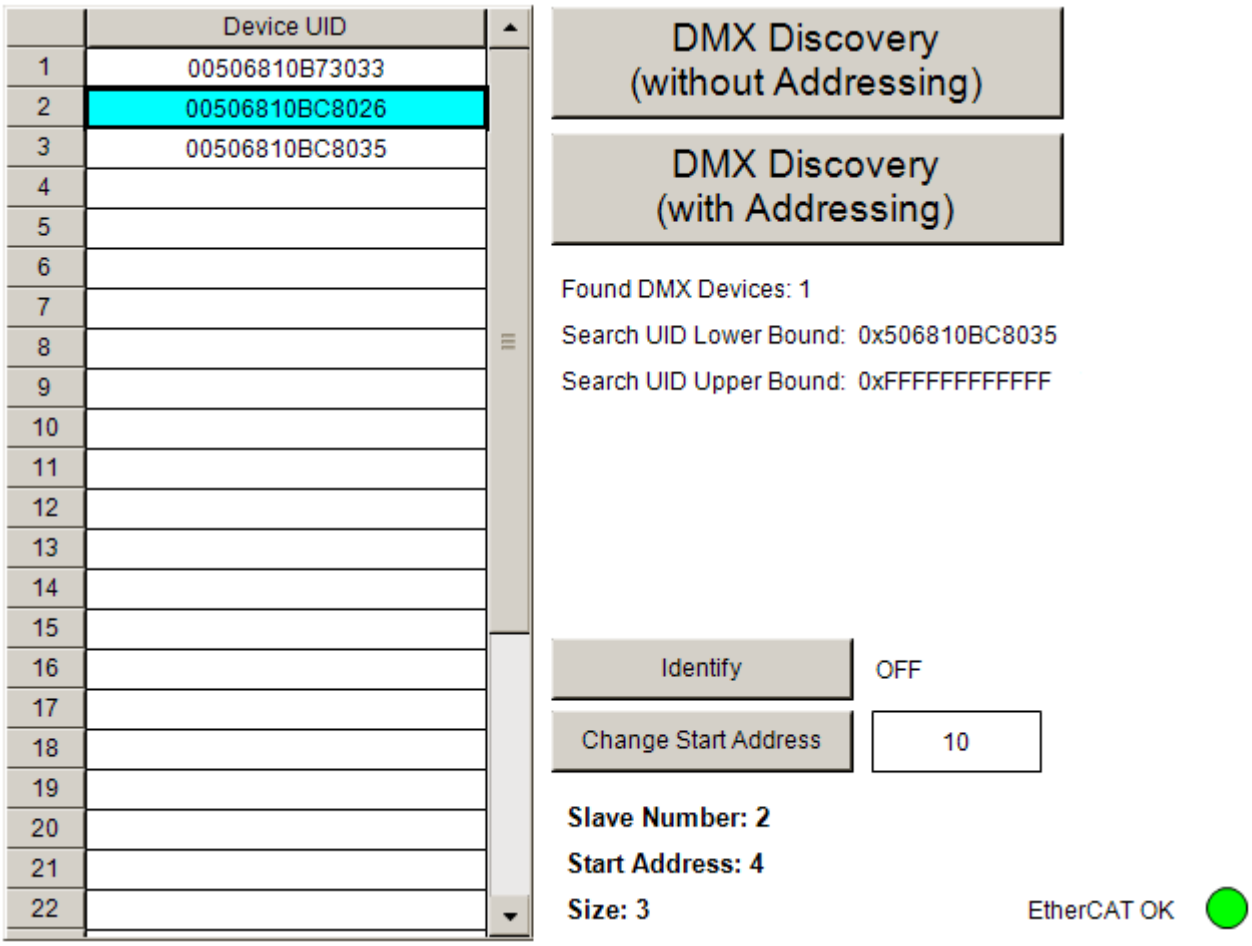
Create the Solution and enable the configuration.

5 Appendix

5.1 Example: Configuration by RDM

Configuration of DMX slaves via Remote Device Management (RDM)

https://infosys.beckhoff.com/content/1033/tcplclib_tc2_dmx/Resources/577712139/.zip for TwinCAT 3.1.



EtherCAT functionality

The dialogue is only functioning when the ‘EtherCAT OK’ LED lights up green. A red LED indicates a fault in the EtherCAT communication.

DMX slaves are sought on the DMX line with the '*DMX Discovery (without Addressing)*' button. All DMX slaves found are displayed in the list on the left. An entry is selected by clicking it. After confirming '*Identify*', the RDM command is sent to identify the respective DMX device. The start address can be entered in the input field adjacent to the '*Change Start Address*' button. After pressing the button, the new start address will be transmitted to the selected DMX device. The number of the DMX device, the start address in the DMX512 frame and the slot size (number of bytes in the DMX frame) for the selected device are displayed in the lower area.

5.2 Example: DMX master

Sending the cyclic process data as a DMX master (EL6851)

https://infosys.beckhoff.com/content/1033/tcplclib_tc2_dmx/Resources/530449035/.zip for TwinCAT 3.1.

Preparation

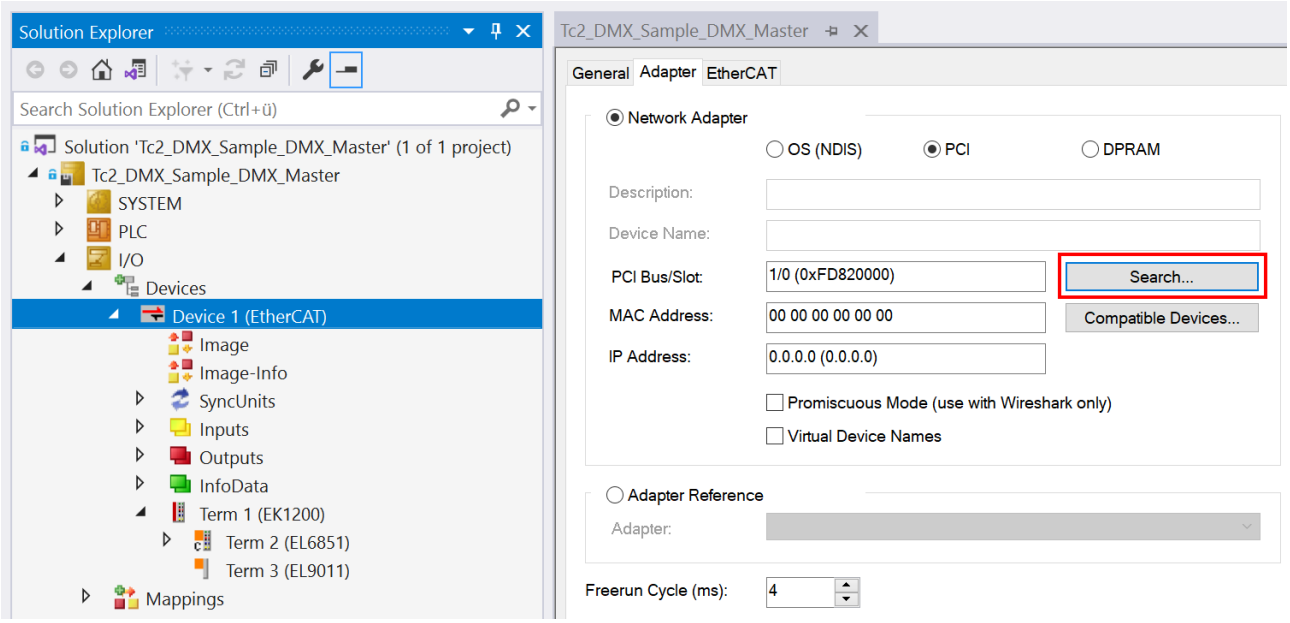
The application examples have been tested with a test configuration and are described accordingly. Certain deviations when setting up actual applications are possible.

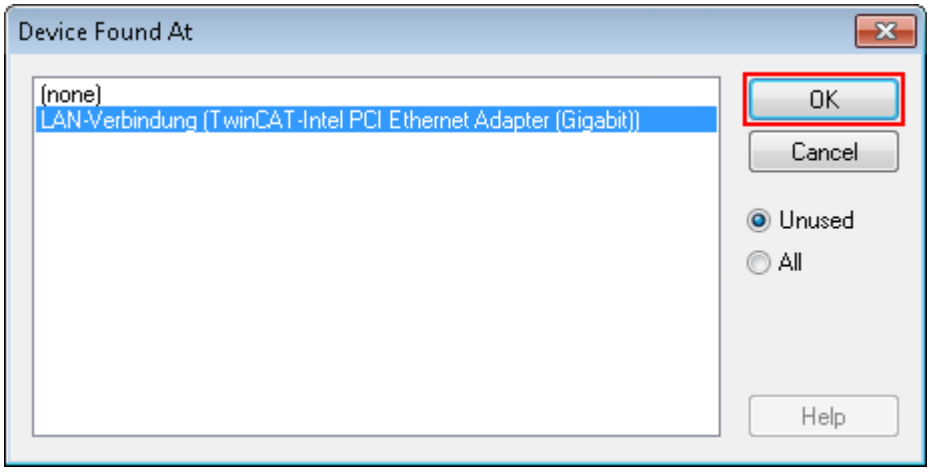
The following hardware and software were used for the test configuration:

- TwinCAT PC with TwinCAT version 3.1 (Build 4014.2) or newer.
- Beckhoff EtherCAT coupler EK1100, EL6851 and EL9011 terminals.
- RGB-LED DMX slave with 3 channels (one for each color). One slot is occupied per channel.

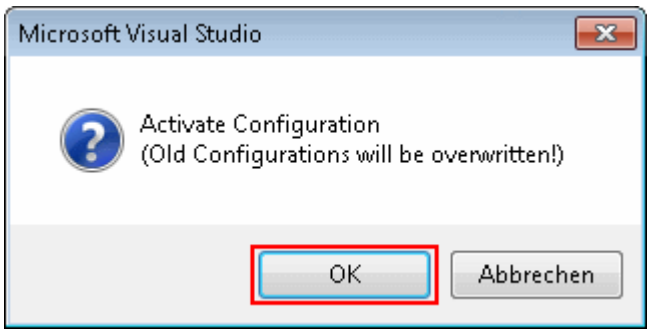
Starting the sample program

1. Save the https://infosys.beckhoff.com/content/1033/tcplclib_tc2_dmx/Resources/530449035/.zip on your hard disk and unzip it.
2. Open the project with TwinCAT XAE.
3. Connect the hardware accordingly and connect the Ethernet adapter of your PCs to the EtherCAT coupler.
4. Select the local Ethernet adapter (with real-time driver, if required) under Tc2_DMX_Sample_DMX_Master > I/O > Devices > Device 1 (EtherCAT) . Then select the appropriate adapter in the **Adapter** tab under **Search...** and confirm it.

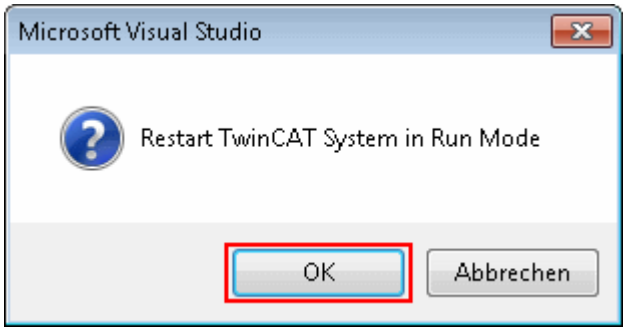




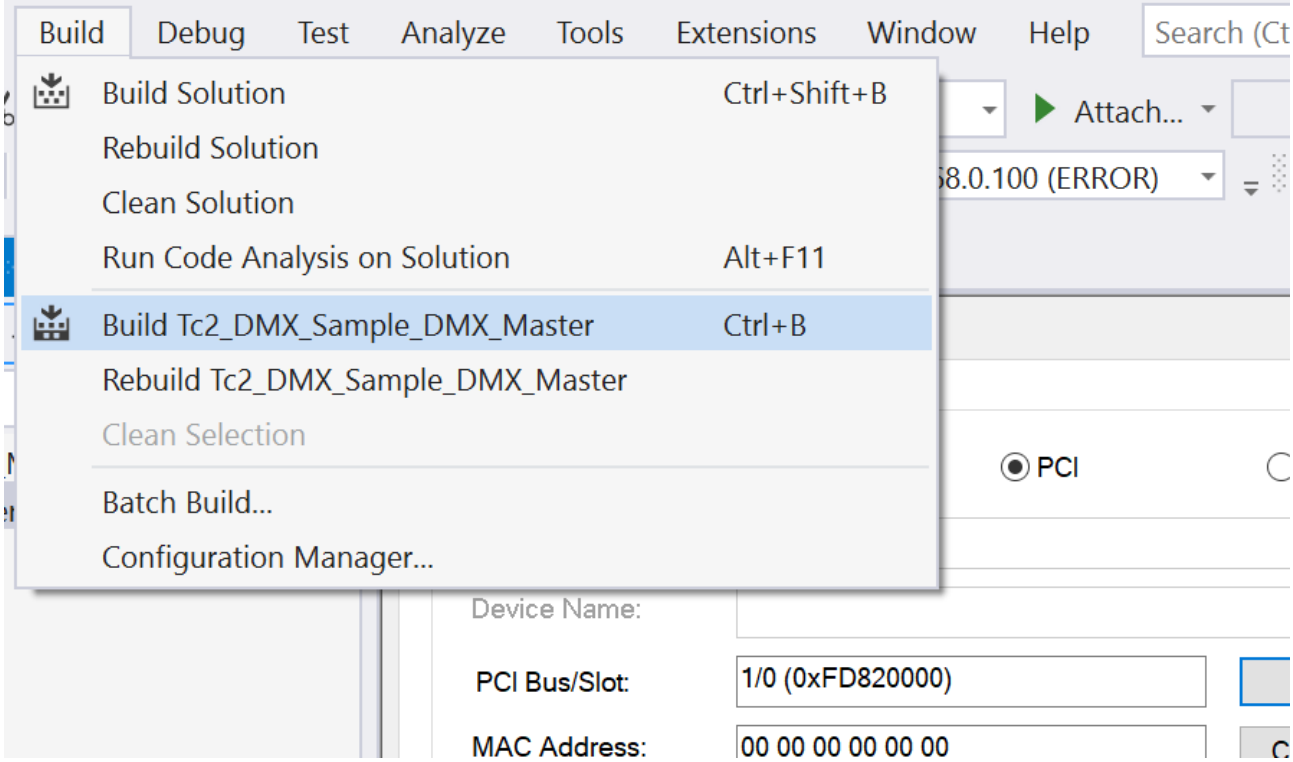
5. Activate the configuration and confirm.



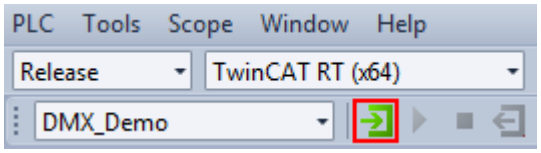
6. Start TwinCAT in RUN mode.

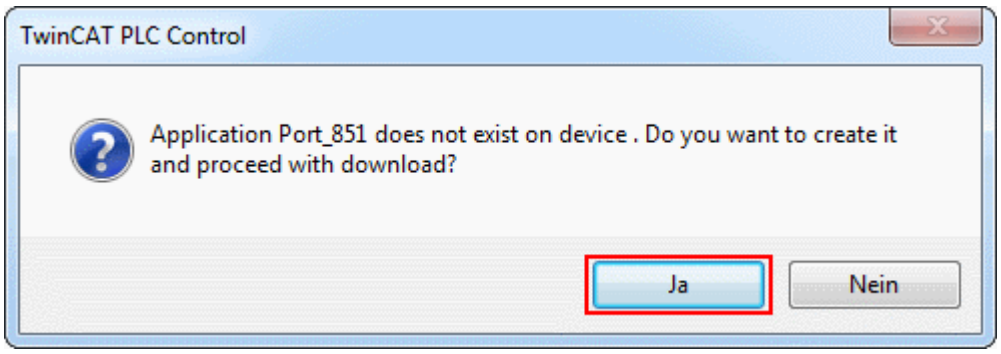


7. Build the project in TwinCAT XAE by selecting the command **Build Tc2_DMX_Sample_DMX_Master** in the **Build** menu.

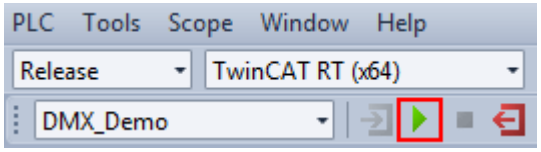


8. Log in and confirm loading of the program.



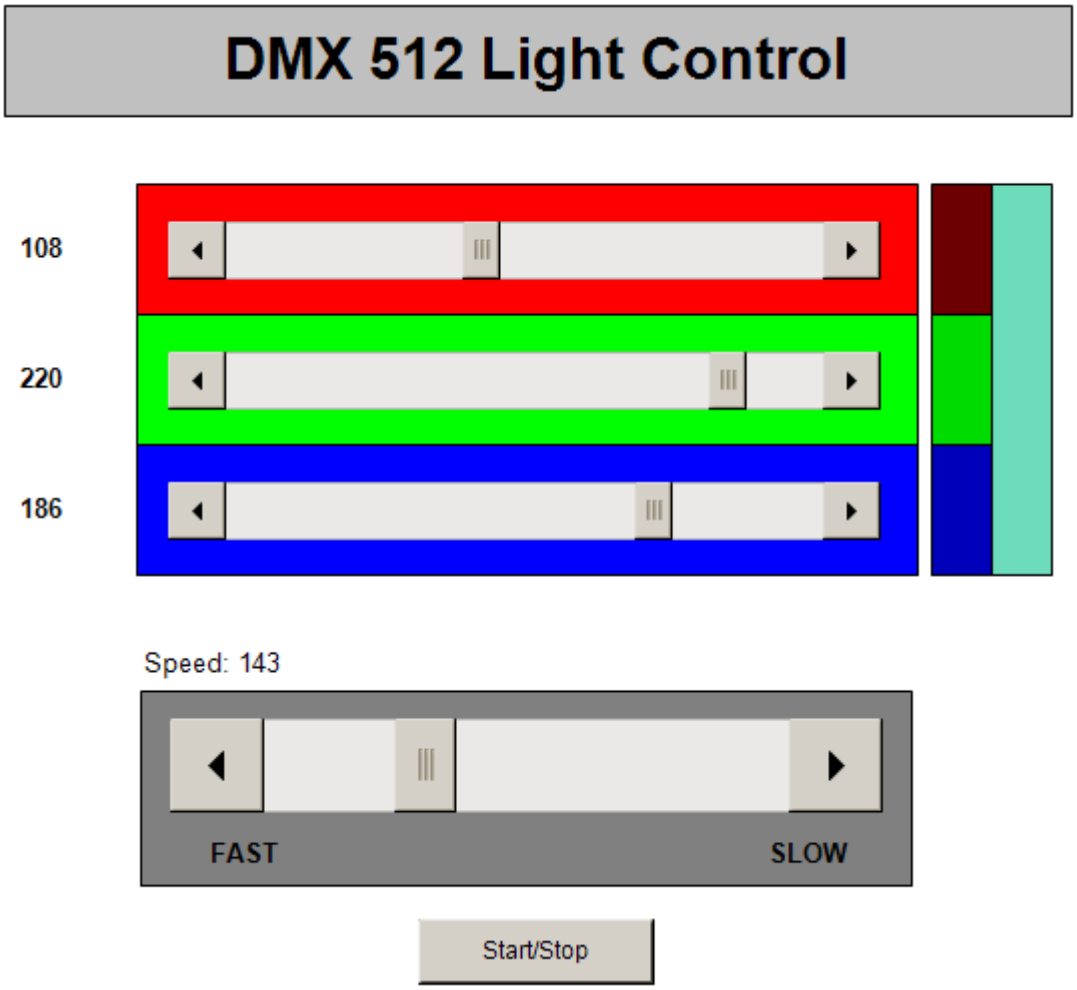


9. Start the program.



Visualization

Specification of the control values for the three colors of the DMX slave in TwinCAT XAE:



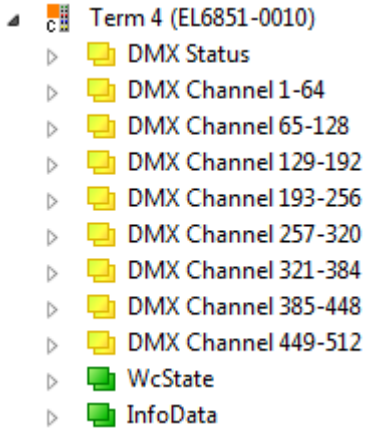
The example transmits the DMX data cyclically to a DMX slave. The DMX device used here occupies three slots (bytes) in the DMX512 frame. Each slot addresses one of the three colors. If the “Start/Stop” button is pressed, then automatically generated data are transmitted to the DMX device. The speed of the changes can be altered using the horizontal slide control. If the “Start/Stop” button is not pressed, you can change the values manually using the three horizontal sliders.

5.3 Example: DMX slave

Receipt of 64 bytes of data by each of two DMX slaves (EL6851-0010)

https://infosys.beckhoff.com/content/1033/tcplclib_tc2_dmx/Resources/578408715/.zip for TwinCAT 3.1.

Arrays with 64 bytes each in full configuration (all PDOs selected):



An EL6851-0010 can read max. 512 bytes (64 bytes in each one of eight arrays, see fig. 1). The arrays can be assigned in the TwinCAT XAE (*Process data* tab) via the PDO 0x1C13.

Example:

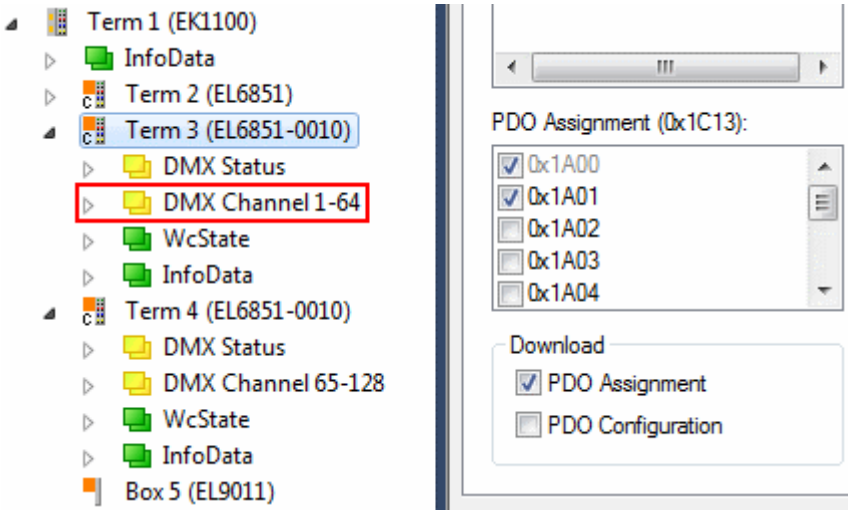
DMX channels 1 - 64 -- > Index 0x1A01

DMX channels 65 - 128 -- > Index 0x1A02

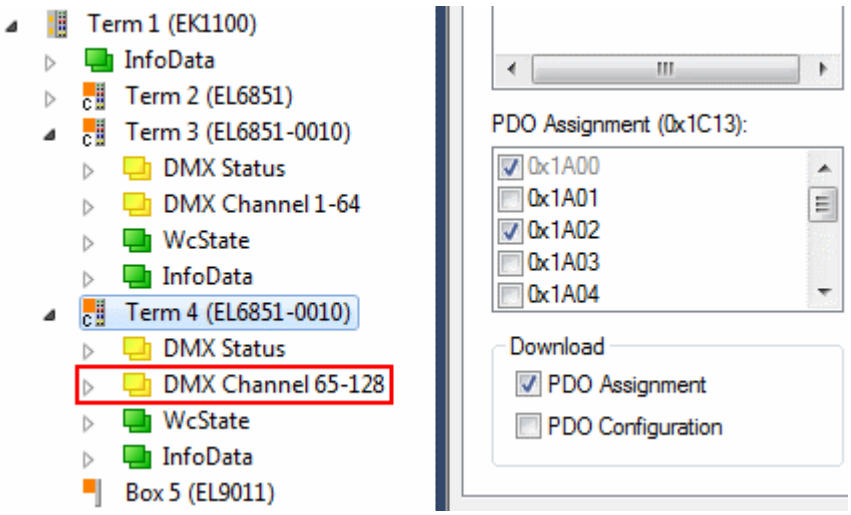
....

DMX channels 449 - 512 -- > Index 0x1A08

- DMX channels 1 - 64 (default) by selection of PDO 0x1A01:

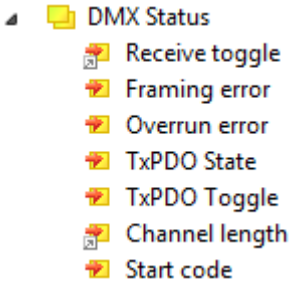


- DMX channels 65 - 128 by selection of PDO 0x1A02:



In the example program the first DMX Slave receives the first 64 bytes sent and the second slave the next 64 bytes (figs. 2 + 3; it is also possible for one EL6851-0010 to receive the entire 128 bytes, the division is chosen deliberately in the example).

DMX status object:



In the DMX status object (index 0x6000, *DMX status*, Fig. 4) a copy counter is created with index 0x6000:11 (*channel length*).

Example:

If PDO 0x1A01 is activated, the value of *Channel length* is 64_{dec}. If PDO 0x1A02 is activated, the value is 128_{dec}. If both PDOs are activated (0x1A01 and 0x1A02) the value is likewise 128_{dec}.

Watchdog DMX Slave 1

	P_DMX_Slave_1.DMX_DATA[INDEX]
1	16#71
2	16#00
3	16#00
4	16#00
5	16#00
6	16#00
7	16#00
8	16#00
9	16#00
10	16#00
11	16#00
12	16#00
13	16#00
14	16#00
15	16#00
16	16#00
17	16#00
18	16#00
19	16#00
20	16#00

Watchdog DMX Slave 2

	P_DMX_Slave_2.DMX_DATA[INDEX]
1	16#71
2	16#00
3	16#00
4	16#00
5	16#00
6	16#00
7	16#00
8	16#00
9	16#00
10	16#00
11	16#00
12	16#00
13	16#00
14	16#00
15	16#00
16	16#00
17	16#00
18	16#00
19	16#00
20	16#00

DMX slave 1 receives 64 bytes of data on channel 1 of the first array (*DMX channels 1 - 64*)

DMX slave 2 receives 64 bytes of data on channel 1 of the second array (*DMX channels 65 - 128*)

The *Receive toggle* bit (index 0x6000:02) is evaluated and displayed in each case via the FB *bMonitorToggleBit* (Watchdog DMX Slave).

5.4 Support and Service

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

Download finder

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

The downloads are available in various formats.

Beckhoff's branch offices and representatives

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

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

You will also find further documentation for Beckhoff components there.

Beckhoff Support

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

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

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

Beckhoff Service

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

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

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

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

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

More Information:
www.beckhoff.com/te1000

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