BECKHOFF New Automation Technology

Manual | EN

sACN 2016

TwinCAT 3

2021-01-21 | Version: 1.2

Table of Contents

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®, 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

1.2 Safety instructions

Safety regulations

Please note the following safety instructions and explanations!
Product-specific safety instructions can be found on following pages or in the areas mounting, wiring, commissioning etc.

Exclusion of liability

All the components are supplied in particular hardware and software configurations 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 engineering who are familiar with the applicable national standards.

Description of symbols

In this documentation the following symbols are used with an accompanying safety instruction or note. The safety instructions must be read carefully and followed without fail!

⚠ DANGER

Serious risk of injury!

Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.

⚠ WARNING

Risk of injury!

Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.

⚠ CAUTION

Personal injuries!

Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.

NOTE

Damage to the environment or devices

Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.

Tip or pointer

This symbol indicates information that contributes to better understanding.

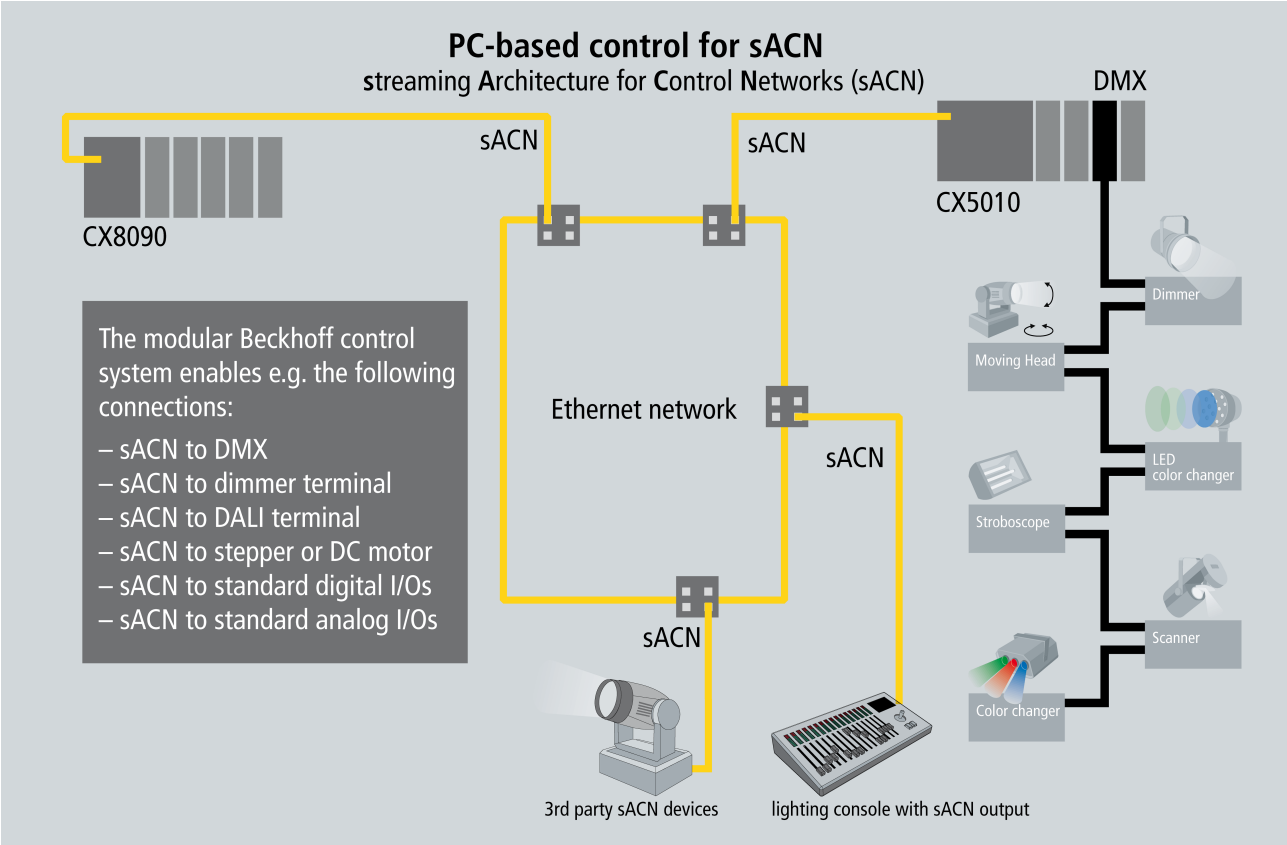
2 Overview

Note This documentation refers to the ANSI standard E1.31-2016 and the use of TwinCAT3.

DMX is a standard that is widely used for controlling lighting elements in stage and building services applications. Beckhoff offers the EL6851 DMX master terminal and the EL6851-0010 DMX slave terminal for this purpose.

ESTA (Entertainment Services and Technology Association) developed Streaming Architecture for Control Networks (sACN in the following) to enable the benefits of Ethernet to be utilised for DMX. It transfers DMX universe data via UDP/IP. The DMX data is embedded in the sACN frame as a complete universe and tunneled through UDP/IP. sACN is defined through ANSI standard E1.31-2009 (in short sACN-2009) and represents a subset of the ACN standard (ANSI standard E1.17).

In 2016, ESTA has enrolled the new version of sACN standard, E1.31-2016 (in short sACN-2016), which added two new frame types, the “Synchronization Frame” and “Universe Discovery Frame”. The Synchronization Frame is a packet that contains only universe synchronization information and it is used to trigger synchronization. The Universe Discovery Frame is a packet that contains a packed list of the universe on which a source is actively operating. This frame allows other devices interested in network traffic to monitor which universes are currently active without to join every multicast group.



sACN data can be mapped to any data types in TwinCAT.

sACN is a protocol specification for transmitting DMX data and was developed by ESTA.

sACN is simplified to Art-Net™ and the information about Art-Net™ can also be found at our website.

Further information about the market activities of Beckhoff in the market stage and show can be found on our website at: PC-based Control for Stage and Show Technology.

System Requirement:

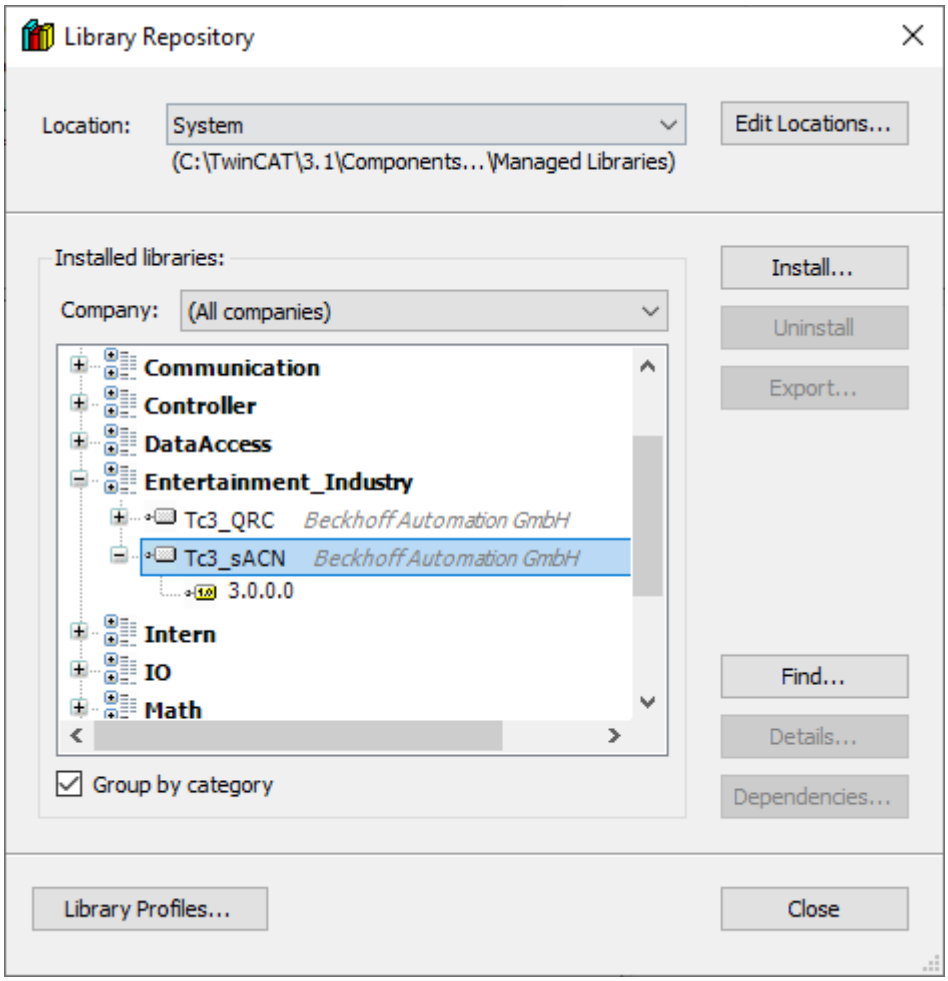
Technical Data	Requirement
TwinCAT version	TwinCAT 3.1 build 4022.20 or higher
Visual Studio version	Visual Studio 2013 or higher
Required TwinCAT license	TF6310 licence

2.1 Update History

[Version 3.0.0.0] – 2020.12.15

Changed:

- Changed the major version number of this library to 3.x.x.x because of TwinCAT 3.
- Move this library file into the library category “Entertainment_Industry”.



3 Programming

This sample generally consists of two modules, an encoder / decoder module and a communication module.

For the sending process, a DMX frame is wrapped by the encoder / decoder module following the sACN structure. Then the wrapped sACN frame is internally transfered through the interface to the communication module. The communication module will send the sACN frame to the target client via UDP.

For the receiving process, an sACN frame is received and validated by the communication module and then transfered to the encoder / decoder function block. This function block will identify the type of sACN frame and extract the DMX frame and the related information about the sending server.

3.1 Function Blocks

The function blocks `FB_sACN` and `FB_sACN_Ex` belong to the encoder / decoder module. The communication module consists of one function block `FB_UDP_Connection`. The two interfaces are `ICommunication` and `I_sACN`.

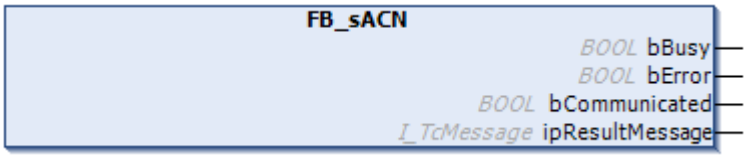
The function block `FB_sACN` implemented sACN-2009, and the function block `FB_sACN_Ex` that derived from `FB_sACN`, implemented sACN-2016.

`FB_UDP_Connection` enbables sending and receiving of frames via UDP. It is an implementation of interface `ICommunication` and it should be instanced and linked to the function block `FB_sACN` or `FB_sACN_Ex`.

According to the sACN specification the usage of sACN frame is carried in UDP packets and in the future, the frame maybe allowed to carry over other networks supported by ACN.

3.1.1 FB_sACN

This function block enables sending and receiving of sACN data frame. With the help of the function block `FB_UDP_Communication`, frames can be sent and received via UDP.



Because `Send` and `Receive` methods are asynchronous, they need more than one cycle to finish working. Only one method could be invoked at the same time. Therefore, always check the output parameter `bBusy` when calling these methods.

Syntax

```
FUNCTION_BLOCK FB_sACN IMPLEMENTS I_sACN
VAR_OUTPUT
    bBusy                               : BOOL;
    bError                              : BOOL;
    bCommunicated                       : BOOL;
    ipResultMessage                     : I_TcMessage;
END_VAR
```

VAR_OUTPUT

bBusy: Is set if an sACN frame is being sent/received.

bError: Is set if an error occurs. You can find error details included Error ID in the “Error List” window.

bCommunicated: Is set after the first sACN frame is successfully sent or received. More information can be found at section Status indicator [▶ 22].

ipResultMessage: Enables error handling with the Tc3_EventLogger.

METHODS

FB_init: Initialization method.

Send: Sending sACN frames (sACN-2009).

Receive: Receiving sACN frames.

INTERFACE

I_sACN: Defines the interface of send and receive methods.

3.1.1.1 FB_init

VAR_INPUT

```
Method FB_init : BOOL
VAR_INPUT
    ipCommunication                 : ICommunication;
    aCID                            : ARRAY[1..16] OF BYTE;
END_VAR
```

ipCommunication: Communication function block that implements the interface `ICommunication`.

aCID: Component Identifier. Sender's unique ID.

Example:

Declaration of the function block `FB_sACN`.

```
PROGRAM MAIN
VAR
    fbsACN    : FB_sACN(fbUDP, aCID);
    fbUDP     : FB_UDP_CONNECTION;
    aCID      : ARRAY[1..16] OF BYTE;
END_VAR
```

3.1.1.2 I_sACN

The function block `FB_sACN` and its derived function block `FB_sACN_Ex` both implemented the interface `I_sACN`. With its help these two function blocks can be switched online only when the `bBusy` is FALSE.

METHODS

Send: Send frames.

Receive: Receive frames.

Example:

```
MAIN
VAR
    fbsACN       : FB_sACN(ipCommunication:= fbUDP, aCID := aCID);
    isACN        : I_sACN := fbsACN;
    fbsACNex     : FB_sACN_Ex(ipCommunication:= fbUDP, aCID:= aCID);
    fbUDP        : FB_UDP_Connection(sLocalHost:= '192.168.1.100', nLocalPort:= 200, sSrvNetId:=
'');
    aCID         : ARRAY[1..16] OF BYTE;
    bSwitch      : BOOL;
    bExtended    : BOOL;
    bBusy        : BOOL;
END_VAR
```

```
IF bSwitch AND NOT bBusy THEN
    bExtended := NOT bExtended;
    bSwitch := FALSE;
END_IF
IF bExtended THEN
    isACN := fbsACNex;
    bBusy := fbsACNex.bBusy OR fbUDP.bBusy;
ELSE
    isACN := fbsACN;
    bBusy := fbsACN.bBusy OR fbUDP.bBusy;
END_IF
```

3.1.1.2.1 Send

This method enables to send UDP frames. The method returns TRUE if a valid sACN frame has been sent.

Syntax

```
Method Send: BOOL
VAR_INPUT
    sSourceName             : STRING;
    sRemoteHost             : T_IPv4Addr := '239.255.0.1';
    nDMXUniverse            : UINT := 1;
    nPriority               : USINT := 100;
    bPreviewData            : BOOL := FALSE;
    bStreamTerminated       : BOOL := FALSE;
    pDMXData                : POINTER TO BYTE;
    bForceSynchron          : BOOL := TRUE;
    nSynchronUniverse       : UINT := 0;
END_VAR
```

VAR_INPUT

sSourceName : Source Name.

sRemoteHost : Target IPv4 address. Unicast and multicast addresses are possible.

nDMXUniverse : Target DMX Universe.

nPriority : Data priority if multiple sources.

bPreviewData : Option Bit: Preview_Data.

bStreamTerminated : Option Bit: Stream_Terminated.

pDMXData : Pointer to DMX data.

bForceSynchron : New Option Bit of sACN-2016: Force_Synchronization.

nSynchronUniverse : Synchronization Universe, ignored in sACN-2009.

1. After the rising edge of `bStreamTerminated` was triggered, the client will continue sending 3 sACN data frames that include the same DMX data, an incremented sequence number, and the terminated flag. After a timeout is expired, the opened socket will be closed and then the sending process is completely terminated unless the falling edge of `bStreamTerminated` is triggered. (More details see this [▶ 22])

2. `bForceSynchron` and `bSynchronUniverse` are new features that are defined in sACN-2016. At function block `FB_sACN` these two variables are set to 0 internally and any other input values will be ignored.

3.1.1.2.2 Receive

This method enables to receive UDP frame. The method returns TRUE if a sACN frame has been received.

Syntax

```
Method Receive : BOOL
VAR_INPUT
    sTargetHost                 : T_IPv4Addr := '239.255.0.1';
```

```
    nTargetUniverse             : UINT;
    sSourceName                 : REFERENCE TO STRING(64);
    nDMXUniverse                : REFERENCE TO UINT;
    aDMXData                    : REFERENCE TO ARRAY[1..512] OF BYTE;
    sSrcHost                    : REFERENCE TO T_IPv4Addr;
    nSrcPort                    : REFERENCE TO UDINT;
    bStreamTerminated           : REFERENCE TO BOOL;
    stFrameCounter              : REFERENCE TO ST_FrameCounter;
    nSynchronUniverse           : REFERENCE TO UINT;
    sDiscoveredUniverse         : REFERENCE TO T_MaxString;
END_VAR
```

VAR_INPUT

sTargetHost: Target IPv4 address from which sACN frames will be received. Unicast and multicast addresses are possible.

nTargetUniverse : Target universe from which the sACN frame will be received.

REFERENCE TO … (OUTPUT)

sSourceName: Source Name of received sACN data frames.

nDMXUniverse : DMX universe of received sACN data frames.

aDMXData : Received DMX Data.

sSrcHost : IPv4 address of remote device from which the sACN frames have been received.

nSrcPort : IPv4 port number of remote device from which the sACN frames have been received.

bStreamTerminated : Is TRUE if data source for the universe has terminated transmission. In consequence this client will stop receiving sACN frames.

stFrameCounter : Counter of received sACN frames.

nSynchronUniverse : Synchronization universe of received sACN data frames.

sDiscoveredUniverse : A list of universes from which sACN data and synchronization frames have been transmitted within 10s.

1. `sDiscoveredUniverse` presents which universes are currently active. It refreshes after a new Universe Discovery frame has been arrived. It presents nothing (empty string) at `FB_sACN`.
2. The Universe Discovery Frame and Synchronization Frame are new types of sACN frames and they are defined in sACN-2016. These two types of frames cannot be recognized by the counter of `FB_sACN` (sACN-2009). These frames can be received but are counted as "Unknown" frames.

3.1.2 FB_sACN_Ex

This function block is derived from `FB_sACN` and it implements the sACN-2016.

Because the `Send`, `Receive`, and `SendSynchronizationFrame` methods are asynchronous and they need more than one cycle to finish working, only one method could be invoked at the same time. Therefore, always check the output parameter `bBusy` when calling these methods.

Syntax

```
FUNCTION_BLOCK FB_sACN_Ex EXTENDS FB_sACN
VAR_OUTPUT
    bBusy                               : BOOL;
    bError                              : BOOL;
```

```
    bCommunicated                       : BOOL;
    ipResultMessage                     : I_TcMessage;
END_VAR
```

VAR_OUTPUT

bBusy: Is set if a valid sACN frame is being sent/received.

bError: Is set if an error occurs. You can find error details included Error ID in logged event.

bCommunicated: Is set after the first streaming ACN frame is successfully sent or received. More information can be found at section Status indicator [▶ 22].

ipResultMessage: Enables error handling with the `Tc3_EventLogger`.

METHODS

FB_init: Initialization method

Send: Sending sACN frames (sACN-2016).

Receive: Receiving sACN frames.

SendSynchronizationFrame: Send sACN synchronization frame.

3.1.2.1 FB_init

```
Method FB_init : BOOL
VAR_INPUT
    ipCommunication                 : ICommunication;
    aCID                            : ARRAY[1..16] OF BYTE;
END_VAR
```

VAR_INPUT

ipCommunication: Communication function block that implements the interface `ICommunication`.

aCID: Component Identifier. Sender's unique ID.

Example:

Declaration of the function block `FB_sACN_Ex`:

```
PROGRAM MAIN
VAR
    fbsACNex        : FB_sACN_Ex(fbUDP, aCID);
    fbUDP           : FB_UDP_CONNECTION;
    aCID            : ARRAY[1..16] OF BYTE;
END_VAR
```

3.1.2.2 SendSynchronizationFrame

This method enables to send sACN synchronization frame that is used to trigger synchronization.

- This method returns TRUE as soon as the request of sending an sACN synchronization frame is terminated.
- This method returns FALSE if the asynchronous request is still active. The method must be called until it was successfully executed, and the return value is TRUE.

```
Method SendSynchronizationFrame : BOOL
VAR_INPUT
    nSynchronUniverse               : UINT := 0;
END_VAR
```

VAR_INPUT

nSynchronUniverse: Universe from which synchronization frames are transmitted.

This Method can only be invoked if `bBusy` is FALSE.

3.1.2.3 I_sACN

The function block `FB_sACN` and its derived function block `FB_sACN_Ex` both implemented the interface `I_sACN`. With its help these two function blocks can be switched online only when the `bBusy` is FALSE.

METHODS

Send: Send frames.

Receive: Receive frames.

Example:

```
MAIN
VAR
    fbsACN       : FB_sACN(ipCommunication:= fbUDP, aCID := aCID);
    isACN        : I_sACN := fbsACN;
    fbsACNex     : FB_sACN_Ex(ipCommunication:= fbUDP, aCID:= aCID);
    fbUDP        : FB_UDP_Connection(sLocalHost:= '192.168.1.100', nLocalPort:= 200, sSrvNetId:=
'');
    aCID         : ARRAY[1..16] OF BYTE;
    bSwitch      : BOOL;
    bExtended    : BOOL;
    bBusy        : BOOL;
END_VAR
```

```
IF bSwitch AND NOT bBusy THEN
    bExtended := NOT bExtended;
    bSwitch := FALSE;
END_IF
IF bExtended THEN
    isACN := fbsACNex;
    bBusy := fbsACNex.bBusy OR fbUDP.bBusy;
ELSE
    isACN := fbsACN;
    bBusy := fbsACN.bBusy OR fbUDP.bBusy;
END_IF
```

3.1.2.3.1 Send

This method enables to send UDP frames. The method returns TRUE if a valid sACN frame has been sent.

Syntax

```
Method Send: BOOL
VAR_INPUT
    sSourceName              : STRING;
    sRemoteHost              : T_IPv4Addr := '239.255.0.1';
    nDMXUniverse             : UINT := 1;
    nPriority                : USINT := 100;
    bPreviewData             : BOOL := FALSE;
    bStreamTerminated        : BOOL := FALSE;
    pDMXData                 : POINTER TO BYTE;
    bForceSynchron           : BOOL := TRUE;
    nSynchronUniverse        : UINT := 0;
END_VAR
```

VAR_INPUT

sSourceName : Source Name.

sRemoteHost : Target IPv4 address. Unicast and multicast addresses are possible.

nDMXUniverse : Target DMX Universe.

nPriority : Data priority if multiple sources.

bPreviewData : Option Bit: Preview_Data.

bStreamTerminated : Option Bit: Stream_Terminated.

pDMXData : Pointer to DMX data.

bForceSynchron : New Option Bit of sACN-2016: Force_Synchronization.

nSynchronUniverse : Synchronization Universe, ignored in sACN-2009.

1. After the rising edge of `bStreamTerminated` was triggered, the client will continue sending 3 sACN data frames that include the same DMX data, an incremented sequence number, and the terminated flag. After a timeout is expired, the opened socket will be closed and then the sending process is completely terminated unless the falling edge of `bStreamTerminated` is triggered. (More details see this [▶ 22])

2. `bForceSynchron` and `bSynchronUniverse` are new features that are defined in sACN-2016. At function block `FB_sACN` these two variables are set to 0 internally and any other input values will be ignored.

3.1.2.3.2 Receive

This method enables to receive UDP frame. The method returns TRUE if a sACN frame has been received.

Syntax

```
Method Receive : BOOL
VAR_INPUT
    sTargetHost                 : T_IPv4Addr := '239.255.0.1';
    nTargetUniverse             : UINT;
    sSourceName                 : REFERENCE TO STRING(64);
    nDMXUniverse                : REFERENCE TO UINT;
    aDMXData                    : REFERENCE TO ARRAY[1..512] OF BYTE;
    sSrcHost                    : REFERENCE TO T_IPv4Addr;
    nSrcPort                    : REFERENCE TO UDINT;
    bStreamTerminated           : REFERENCE TO BOOL;
    stFrameCounter              : REFERENCE TO ST_FrameCounter;
    nSynchronUniverse           : REFERENCE TO UINT;
    sDiscoveredUniverse         : REFERENCE TO T_MaxString;
END_VAR
```

VAR_INPUT

sTargetHost: Target IPv4 address from which sACN frames will be received. Unicast and multicast addresses are possible.

nTargetUniverse : Target universe from which the sACN frame will be received.

REFERENCE TO … (OUTPUT)

sSourceName: Source Name of received sACN data frames.

nDMXUniverse : DMX universe of received sACN data frames.

aDMXData : Received DMX Data.

sSrcHost : IPv4 address of remote device from which the sACN frames have been received.

nSrcPort : IPv4 port number of remote device from which the sACN frames have been received.

bStreamTerminated : Is TRUE if data source for the universe has terminated transmission. In consequence this client will stop receiving sACN frames.

stFrameCounter : Counter of received sACN frames.

nSynchronUniverse : Synchronization universe of received sACN data frames.

sDiscoveredUniverse : A list of universes from which sACN data and synchronization frames have been transmitted within 10s.

1. `sDiscoveredUniverse` presents which universes are currently active. It refreshes after a new Universe Discovery frame has been arrived. It presents nothing (empty string) at `FB_sACN`.

2. The Universe Discovery Frame and Synchronization Frame are new types of sACN frames and they are defined in sACN-2016. These two types of frames cannot be recognized by the counter of `FB_sACN` (sACN-2009). These frames can be received but are counted as "Unknown" frames.

3.1.3 FB_UDP_Connection

This function block enables to create or terminate a UDP connection.

This function block can only be used as the initial variable of function block `FB_sACN` and `FB_sACN_Ex`. It cannot be invoked directly.

Syntax

```
FUNCTION_BLOCK FB_UDP_Connection IMPLEMENTS ICommunication
VAR_OUTPUT
bBusy                   : BOOL;
bError                  : BOOL;
bConnected              : BOOL;
ipResultMessage         : I_TcMessage;
END_VAR
```

VAR_OUTPUT

bBusy: Is TRUE as long as the asynchronous request is still active. Is FALSE if the request is completed. (More information can be found at section Status indicator [▶ 22])

bError: Is set if an error occurs during execution of the function block. Error details can be found at “Error List” window.

bConnected: Is TRUE as long as a socket is opened without error; is FALSE if socket is closed.

ipResultMessage: Enables error handling with the `Tc3_EventLogger`.

METHODS

FB_init: Initialization method

INTERFACE

ICommunication: Defines the interface of communication methods.

3.1.3.1 FB_init

```
Method FB_init : BOOL
VAR_INPUT
    sLocalHost                  : T_IPv4Addr;
    nLocalPort                  : UINT := 200;
    sSrvNetId                   : T_AmsNetId;
END_VAR
```

VAR_INPUT

sLocalHost: The Local IP address (IPv4) of the UDP socket as a string

nLocalPort: The local IP port number of the UDP socket. Default value is 200.

sSrvNetId: AMS Net Id. For the local computer (default) an empty string may be specified.

Variable `nLocalPort` is only for the socket that is used for the sending process. During the receiving process the port number must be 5568, other values will be ignored.

Example:

Declaration of the function block `FB_UDP_CONNECTION`:

```
PROGRAM MAIN
VAR
    fbUDP         : FB_UDP_CONNECTION (sLocalHost:= '192.168.1.100', nLocalPort:= 200, sSrvNetId:= '
');
END_VAR
```

3.1.3.2 ICommunication

This interface defines a communication interface in order to make data exchange between two function blocks possible. This interface has been implemented internally and users do not need to implement it.

3.2 Structures, Interfaces

3.2.1 Structures

3.2.1.1 ST_FrameCounter

```
TYPE ST_FrameCounter:
STRUCT
nDMX_Frames                    : UDINT;
nSynchronization_Frames        : UDINT;
nUniverse_Discovery_Frames     : UDINT;
nUnknown_sACN_Frames           : UDINT;
END_STRUCT
END_TYPE
```

The Universe Discovery frame and Synchronization frame are new types of sACN frames and defined in sACN-2016. They can also be received by `FB_sACN` (sACN-2009), but they are counted as "Unknown" frames.

3.2.2 Interfaces

3.2.2.1 ICommunication

This interface defines a communication interface in order to make data exchange between two function blocks possible. This interface has been implemented internally and users do not need to implement it.

3.2.2.2 I_sACN

The function block `FB_sACN` and its derived function block `FB_sACN_Ex` both implemented the interface `I_sACN`. With its help these two function blocks can be switched online only when the `bBusy` is FALSE.

METHODS

Send: Send frames.

Receive: Receive frames.

Example:

```
MAIN
VAR
    fbsACN       : FB_sACN(ipCommunication:= fbUDP, aCID := aCID);
    isACN        : I_sACN := fbsACN;
    fbsACNex     : FB_sACN_Ex(ipCommunication:= fbUDP, aCID:= aCID);
    fbUDP        : FB_UDP_Connection(sLocalHost:= '192.168.1.100', nLocalPort:= 200, sSrvNetId:=
'');
    aCID         : ARRAY[1..16] OF BYTE;
    bSwitch      : BOOL;
    bExtended    : BOOL;
    bBusy        : BOOL;
END_VAR
```

```
IF bSwitch AND NOT bBusy THEN
    bExtended := NOT bExtended;
    bSwitch := FALSE;
END_IF
IF bExtended THEN
    isACN := fbsACNex;
    bBusy := fbsACNex.bBusy OR fbUDP.bBusy;
ELSE
    isACN := fbsACN;
    bBusy := fbsACN.bBusy OR fbUDP.bBusy;
END_IF
```

3.2.2.2.1 Send

This method enables to send UDP frames. The method returns TRUE if a valid sACN frame has been sent.

Syntax

```
Method Send: BOOL
VAR_INPUT
    sSourceName              : STRING;
    sRemoteHost              : T_IPv4Addr := '239.255.0.1';
    nDMXUniverse             : UINT := 1;
    nPriority                : USINT := 100;
    bPreviewData             : BOOL := FALSE;
    bStreamTerminated        : BOOL := FALSE;
    pDMXData                 : POINTER TO BYTE;
    bForceSynchron           : BOOL := TRUE;
    nSynchronUniverse        : UINT := 0;
END_VAR
```

VAR_INPUT

sSourceName : Source Name.

sRemoteHost : Target IPv4 address. Unicast and multicast addresses are possible.

nDMXUniverse : Target DMX Universe.

nPriority : Data priority if multiple sources.

bPreviewData : Option Bit: Preview_Data.

bStreamTerminated : Option Bit: Stream_Terminated.

pDMXData : Pointer to DMX data.

bForceSynchron : New Option Bit of sACN-2016: Force_Synchronization.

nSynchronUniverse : Synchronization Universe, ignored in sACN-2009.

1. After the rising edge of `bStreamTerminated` was triggered, the client will continue sending 3 sACN data frames that include the same DMX data, an incremented sequence number, and the terminated flag. After a timeout is expired, the opened socket will be closed and then the sending process is completely terminated unless the falling edge of `bStreamTerminated` is triggered. (More details see this [▶ 22])

2. `bForceSynchron` and `bSynchronUniverse` are new features that are defined in sACN-2016. At function block `FB_sACN` these two variables are set to 0 internally and any other input values will be ignored.

3.2.2.2.2 Receive

This method enables to receive UDP frame. The method returns TRUE if a sACN frame has been received.

Syntax

```
Method Receive : BOOL
VAR_INPUT
    sTargetHost                 : T_IPv4Addr := '239.255.0.1';
    nTargetUniverse             : UINT;
    sSourceName                 : REFERENCE TO STRING(64);
    nDMXUniverse                : REFERENCE TO UINT;
    aDMXData                    : REFERENCE TO ARRAY[1..512] OF BYTE;
    sSrcHost                    : REFERENCE TO T_IPv4Addr;
    nSrcPort                    : REFERENCE TO UDINT;
    bStreamTerminated           : REFERENCE TO BOOL;
    stFrameCounter              : REFERENCE TO ST_FrameCounter;
    nSynchronUniverse           : REFERENCE TO UINT;
    sDiscoveredUniverse         : REFERENCE TO T_MaxString;
END_VAR
```

VAR_INPUT

sTargetHost: Target IPv4 address from which sACN frames will be received. Unicast and multicast addresses are possible.

nTargetUniverse : Target universe from which the sACN frame will be received.

REFERENCE TO … (OUTPUT)

sSourceName: Source Name of received sACN data frames.

nDMXUniverse : DMX universe of received sACN data frames.

aDMXData : Received DMX Data.

sSrcHost : IPv4 address of remote device from which the sACN frames have been received.

nSrcPort : IPv4 port number of remote device from which the sACN frames have been received.

bStreamTerminated : Is TRUE if data source for the universe has terminated transmission. In consequence this client will stop receiving sACN frames.

stFrameCounter : Counter of received sACN frames.

nSynchronUniverse : Synchronization universe of received sACN data frames.

sDiscoveredUniverse : A list of universes from which sACN data and synchronization frames have been transmitted within 10s.

1. `sDiscoveredUniverse` presents which universes are currently active. It refreshes after a new Universe Discovery frame has been arrived. It presents nothing (empty string) at `FB_sACN`.

2. The Universe Discovery Frame and Synchronization Frame are new types of sACN frames and they are defined in sACN-2016. These two types of frames cannot be recognized by the counter of `FB_sACN` (sACN-2009). These frames can be received but are counted as "Unknown" frames.

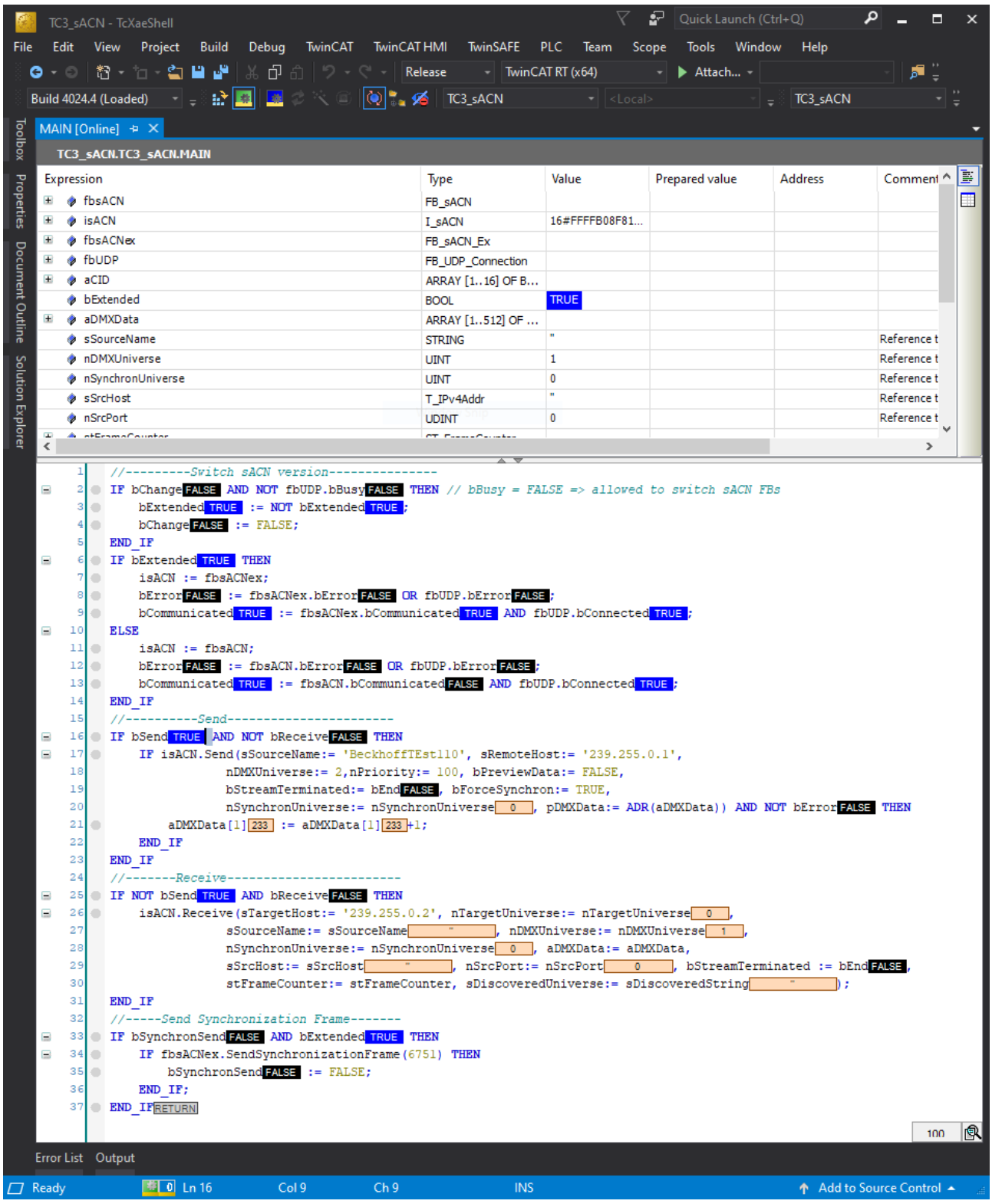
4 Example

This https://infosys.beckhoff.com/content/1033/TF6310_sACN/Resources/zip/9007207188597515.zip contains a PLC program for TwinCAT 3 that presents general sending and receiving of sACN frames. In any use case the user data of the sACN frame can be linked with any PLC variables. This allows a very flexible use of this example. Users can create their own programs based on their needs. The individual function blocks of the example are explained in the Programming section.

For example, a state machine can be used to realize a logic, where listening in the receiving method is basically used and followed by the sending method on a special event or timer. Alternatively, if the sending method and receiving method should be triggered at the same time, two instances could be declared: One instance for sending and the other for receiving.

There is in addition a TwinCAT 2 sample available which support only the sACN version 2009.

Both sACN versions “sACN 2009” and “sACN 2016” are implemented in this TwinCAT 3 example. So that users who want to use the “sACN 2009” version on the TwinCAT 3 platform can do this. The main idea of this sample is to be used for sACN applications using version sACN 2016, nevertheless the online switching of both sACN versions is demonstrated in this sample. It is possible to switch online between these two sACN versions with the help of instantiating the interface “isACN” and triggering the variable “bSwitch”.


TC3_sACN - TcXaeShell
Quick Launch (Ctrl+Q)
File Edit View Project Build Debug TwinCAT TwinCAT HMI TwinSAFE PLC Team Scope Tools Window Help
Release
TwinCAT RT (x64)
Attach...
Build 4024.4 (Loaded)
TC3_sACN
<Local>
TC3_sACN
MAIN [Online]
TC3_sACN.TC3_sACN.MAIN
Expression Type Value Prepared value Address Comment
fbsACN FB_sACN
isACN I_sACN 16#FFFFB08F81...
fbsACNex FB_sACN_Ex
fbUDP FB_UDP_Connection
aCID ARRAY [1..16] OF B...
bExtended BOOL TRUE
aDMXData ARRAY [1..512] OF ...
sSourceName STRING '' Reference t
nDMXUniverse UINT 1 Reference t
nSynchronUniverse UINT 0 Reference t
sSrcHost T_IPv4Addr '' Reference t
nSrcPort UDINT 0 Reference t
//---------Switch sACN version---------------
IF bChange FALSE AND NOT fbUDP.bBusy FALSE THEN // bBusy = FALSE => allowed to switch sACN FBs
bExtended TRUE := NOT bExtended TRUE;
bChange FALSE := FALSE;
END_IF
IF bExtended TRUE THEN
isACN := fbsACNex;
bError FALSE := fbsACNex.bError FALSE OR fbUDP.bError FALSE;
bCommunicated TRUE := fbsACNex.bCommunicated TRUE AND fbUDP.bConnected TRUE;
ELSE
isACN := fbsACN;
bError FALSE := fbsACN.bError FALSE OR fbUDP.bError FALSE;
bCommunicated TRUE := fbsACN.bCommunicated FALSE AND fbUDP.bConnected TRUE;
END_IF
//----------Send-----------------------
IF bSend TRUE AND NOT bReceive FALSE THEN
IF isACN.Send(sSourceName:= 'BeckhoffTEst110', sRemoteHost:= '239.255.0.1',
nDMXUniverse:= 2,nPriority:= 100, bPreviewData:= FALSE,
bStreamTerminated:= bEnd FALSE, bForceSynchron:= TRUE,
nSynchronUniverse:= nSynchronUniverse 0, pDMXData:= ADR(aDMXData)) AND NOT bError FALSE THEN
aDMXData[1] 233 := aDMXData[1] 233 +1;
END_IF
END_IF
//-------Receive-------------------------
IF NOT bSend TRUE AND bReceive FALSE THEN
isACN.Receive(sTargetHost:= '239.255.0.2', nTargetUniverse:= nTargetUniverse 0,
sSourceName:= sSourceName '', nDMXUniverse:= nDMXUniverse 1,
nSynchronUniverse:= nSynchronUniverse 0, aDMXData:= aDMXData,
sSrcHost:= sSrcHost '', nSrcPort:= nSrcPort 0, bStreamTerminated := bEnd FALSE,
stFrameCounter:= stFrameCounter, sDiscoveredUniverse:= sDiscoveredString '');
END_IF
//-----Send Synchronization Frame-------
IF bSynchronSend FALSE AND bExtended TRUE THEN
IF fbsACNex.SendSynchronizationFrame(6751) THEN
bSynchronSend FALSE := FALSE;
END_IF;
END_IF RETURN
100
Error List Output
Ready 0 Ln 16 Col 9 Ch 9 INS Add to Source Control

5 Appendix

5.1 Properties about time

In this project some time properties are defined internally, and they cannot be modified by users. In the following, these time properties and the associated running logic are explained.

Sending process:

- According to sACN-2016, the Universe Discovery Frame is sent every **10s**.
- After data frames with the **terminate flag** have been sent, the socket kept opened in **2.5s**, in order to keep receiving sACN frames arriving from other sources during this period. After time has been expired, this socket is closed and `bCommunicated` and `bConnected` are set back to FALSE. Socket cannot be reopened unless a falling edge of terminate variable `bStreamTerminated` occurs.

Receiving process:

- According to the sACN specification the receiving process will be terminated, and the client will change to idle state, if no new sACN Data Frame has been received in a period of **2.5s.** The socket keeps opened if no error occurs during the idle time.
- If no error occurs during idle state, the receiving process will restart after **5s**.

5.2 Status indicator

bCommunicated

It is an output variable of `FB_sACN` and `FB_sACN_ex`. It indicates the state of the sender / receiver process.

- It sets TRUE if an sACN frame has been successfully sent or received.
- It sets FALSE if terminated flag has been fired. The client goes in idle state if no error occurs.

bConnected

It is an output variable of the function block `FB_UDP_Connection`. It indicates the connection status.

- It sets TRUE if a UDP socket has been successfully opened without an error.
- It sets FALSE if socket has been closed or an error occurred.

bBusy

It is an output variable of all function blocks.

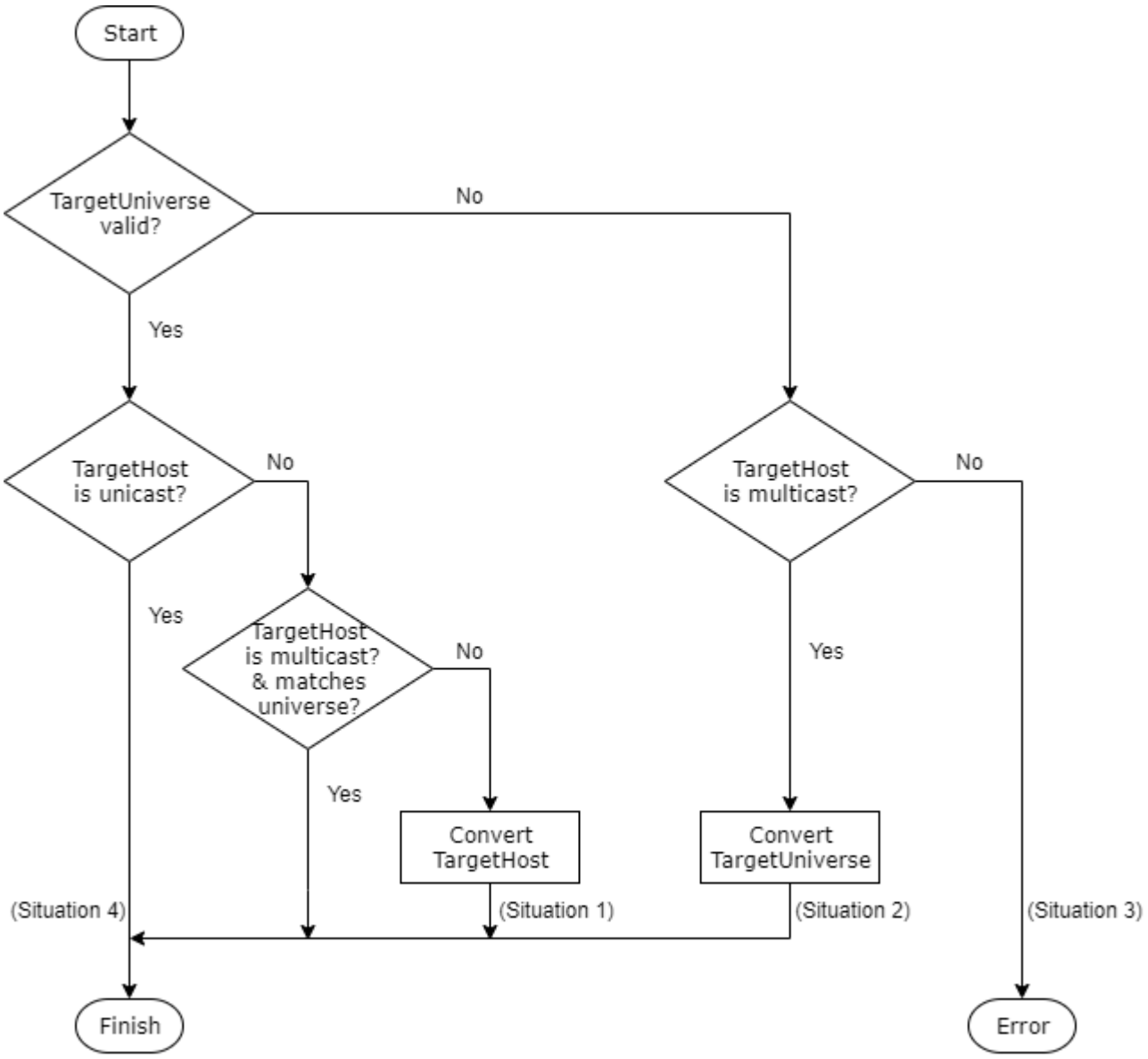
- It sets TRUE as long as the processing is not completed. Each new input variable of methods or switching of sACN function blocks can be accepted ONLY when `bBusy` is FALSE.
- It sets FALSE as long as a processing is completed without error (`bError` is FALSE) or a processing is not completed but an error occurs (`bError` is TRUE).

bError

It sets TRUE if an error occurs.

5.3 Target host and target universe

The target host variable `sTargetHost` and target universe variable `nTargetUniverse` are the target IPv4 address and the port where the sACN frames are sent to and come from.



Text

According to the sACN specification, in multicast mode the last two bytes of a multicast address are defined by high-byte and low-byte of the universe. If the target host is mismatched to the target universe, the universe is dominated. In this case multicast address must be corrected. In consequence the original multicast address is ignored (Situation 1).

However, if the target universe is invalid (universe > 63999 or universe = 0) but the target host is a multicast address, the universe should be converted based on the target host and the invalid target universe will be ignored (Situation 2).

Especially if both variables are invalid, or the target host is a unicast address while the target universe is invalid, then an error occurs, and the error information can be found in the "Error List" window (Situation 3).

If the target universe is valid and target host is a unicast address, these rules are not concerned (Situation 4).

Here are some examples.

Example for Situation 1: The target host is **239.255.0.1** and the target universe is **2**. Because the target universe is valid and the target host is a multicast address, the target host will be changed to **239.255.0.2**.

Example for Situation 2: The target host is **239.255.0.1** and the target universe is **0**. Because the target universe is invalid and the target host is a multicast address, the target universe will be changed to 1.

Example for Situation 3: The target host is **0.0.0.0** or **192.168.0.1** and the target universe is **0**. An error occurs.

Example for Situation 4: The target host is **192.168.0.100** and the target universe is **1**. Both values are accepted without conversion.

5.4 Multicast Group

In order to receive sACN frames from different Multicast Groups (MCGs in short), joining more MCGs is supported in this project. Users could either give the new MCG address at input `sTargetHost` or write associated new universe at input `nTargetUniverse`. All joined MCGs will be saved and users do not need to give new MCG address in 2nd time. The array capacity of saved MCGs is 100, and the first joined MCG will be overwritten if the array is overloaded.

More Information:
www.beckhoff.com/stage

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