Manual

TC3 AES70 (OCA) Communication

TwinCAT 3

Version: 1.2
Date: 2018-11-22
Order No.: TF8810
Table of contents

1 Foreword .......................................................................................................................................................... 5
   1.1 Notes on the documentation .......................................................................................................................... 5
   1.2 Safety instructions .......................................................................................................................................... 6

2 Overview .......................................................................................................................................................... 7

3 System requirements ......................................................................................................................................... 8

4 Installation ....................................................................................................................................................... 9

5 Licensing .......................................................................................................................................................... 13

6 Technical introduction .................................................................................................................................... 18

7 PLC API ........................................................................................................................................................... 19
   7.1 Function blocks .............................................................................................................................................. 19
      7.1.1 FB_OcaDevice ......................................................................................................................................... 19
      7.1.2 FB_OcaRoot ........................................................................................................................................... 20
      7.1.3 Worker function blocks .......................................................................................................................... 25
   7.2 Data types ..................................................................................................................................................... 56
      7.2.1 Structures used to represent the properties of OCA objects .................................................................... 56
      7.2.2 E_OcaStatus .......................................................................................................................................... 58
      7.2.3 E_OcaMuteState ................................................................................................................................... 59
      7.2.4 ST_OcaTemperature .............................................................................................................................. 59
      7.2.5 ST_OcaDeviceInfo .................................................................................................................................. 59
      7.2.6 E_OcaMessageType .................................................................................................................................. 59

8 Examples ......................................................................................................................................................... 60
   8.1 Example for using the function block FB_OcaDevice .................................................................................... 60
   8.2 Example for using the function block FB_OcaRoot ........................................................................................ 60
   8.3 Example for using the function block FB_OcaWorker ..................................................................................... 61
   8.4 Example for using the function block FB_OcaGain ....................................................................................... 62

9 Support and Service ....................................................................................................................................... 63
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 the 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®, Safety over EtherCAT®, TwinSAFE®, XFC® and XTS® 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, DE102004044764, DE102007017835
with corresponding applications or registrations in various other countries.

The TwinCAT Technology is covered, including but not limited to the following patent applications and patents:
EP0851348, US6167425 with corresponding applications or registrations in various other countries.

EtherCAT® is 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 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!

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>⚠️ <strong>DANGER</strong></td>
<td>Serious risk of injury! Failure to follow the safety instructions associated with this symbol directly endangers the life and health of persons.</td>
</tr>
<tr>
<td>⚠️ <strong>WARNING</strong></td>
<td>Risk of injury! Failure to follow the safety instructions associated with this symbol endangers the life and health of persons.</td>
</tr>
<tr>
<td>⚠️ <strong>CAUTION</strong></td>
<td>Personal injuries! Failure to follow the safety instructions associated with this symbol can lead to injuries to persons.</td>
</tr>
<tr>
<td><strong>NOTE</strong></td>
<td>Damage to the environment or devices Failure to follow the instructions associated with this symbol can lead to damage to the environment or equipment.</td>
</tr>
</tbody>
</table>

- Tip or pointer
  - This symbol indicates information that contributes to better understanding.
2 Overview

The AES70 standard was published by the Audio Engineering Society, which is based in New York. It defines a scalable control protocol for professional audio devices and describes monitoring and control of such devices, but not the transfer of media data.

The function blocks of the TwinCAT 3 AES70 (OCA) Communication PLC library can be used to establish data exchange between a TwinCAT PLC and a device that supports the AES70 (OCA) standard. The user can query or set properties of different objects in a device (Oca device). Various function blocks such as FB_OcaGain, FB_OcaMute or FB_OcaSwitch are available for this purpose.

The user can thus integrate OCA-compatible audio systems into Beckhoff's PC-based control technology platform and select from a wide range of control panels and various I/Os.

Further information can be found on the Beckhoff website under Application & Solutions > Stage and Show Technology

For the OCA-capable amplifiers from d&b audiotechnik, an example of a higher-level function block with some basic functions such as gain, mute or presets is available for download from the d&b audiotechnik website under www.dbaudio.com > Systems > Networks and Integration > Integration
# 3 System requirements

<table>
<thead>
<tr>
<th>Technical data</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating system</td>
<td>Windows 7/10, Windows Embedded Standard 7, Windows CE 7</td>
</tr>
<tr>
<td>Target platform</td>
<td>PC architecture (x86, x64 or ARM)</td>
</tr>
<tr>
<td>TwinCAT version</td>
<td>TwinCAT 3.1 build 4022.2 or higher</td>
</tr>
<tr>
<td>Required TwinCAT setup level</td>
<td>TwinCAT 3 XAE, XAR</td>
</tr>
<tr>
<td>Required TwinCAT license</td>
<td>TF8810 AES70 (OCA) Communication</td>
</tr>
</tbody>
</table>
4 Installation

The following section describes how to install the TwinCAT 3 Function for Windows-based operating systems.

- The TwinCAT 3 Function setup file was downloaded from the Beckhoff website.

1. Run the setup file as administrator. To do this, select the command Run as administrator in the context menu of the file.

- The installation dialog opens.

2. Accept the end user licensing agreement and click Next.
3. Enter your user data.

4. If you want to install the full version of the TwinCAT 3 Function, select **Complete** as installation type. If you want to install the TwinCAT 3 Function components separately, select **Custom**.
5. Select **Next**, then **Install** to start the installation.

6. Confirm the dialog with **Yes**.
7. Select **Finish** to exit the setup.

℩ The TwinCAT 3 Function has been successfully installed and can be licensed (see Licensing [13]).
Licensing

The TwinCAT 3 Function can be activated as a full version or as a 7-day test version. Both license types can be activated via the TwinCAT 3 development environment (XAE).

The licensing of a TwinCAT 3 Function is described below. The description is divided into the following sections:

- Licensing a 7-day test version [13]
- Licensing a full version [14]

Further information on TwinCAT 3 licensing can be found in the “Licensing” documentation in the Beckhoff Information System (TwinCAT 3 > Licensing).

Licensing a 7-day test version

1. Start the TwinCAT 3 development environment (XAE).
2. Open an existing TwinCAT 3 project or create a new project.
3. If you want to activate the license for a remote device, set the desired target system. To do this, select the target system from the Choose Target System drop-down list in the toolbar.
   - The licensing settings always refer to the selected target system. When the project is activated on the target system, the corresponding TwinCAT 3 licenses are automatically copied to this system.
4. In the Solution Explorer, double-click License in the SYSTEM subtree.
   - The TwinCAT 3 license manager opens.
5. Open the Manage Licenses tab. In the Add License column, check the check box for the license you want to add to your project (e.g. “TF6420: TC3 Database Server”).
6. Open the Order Information (Runtime) tab.
In the tabular overview of licenses, the previously selected license is displayed with the status “missing”.

7. Click 7-Day Trial License... to activate the 7-day trial license.

A dialog box opens, prompting you to enter the security code displayed in the dialog.

8. Enter the code exactly as it appears, confirm it and acknowledge the subsequent dialog indicating successful activation.

   In the tabular overview of licenses, the license status now indicates the expiration date of the license.

9. Restart the TwinCAT system.

The 7-day trial version is enabled.

**Licensing a full version**

1. Start the TwinCAT 3 development environment (XAE).
2. Open an existing TwinCAT 3 project or create a new project.
3. If you want to activate the license for a remote device, set the desired target system. To do this, select the target system from the Choose Target System drop-down list in the toolbar.

   The licensing settings always refer to the selected target system. When the project is activated on the target system, the corresponding TwinCAT 3 licenses are automatically copied to this system.
4. In the **Solution Explorer**, double-click **License** in the **SYSTEM** subtree.

![Solution Explorer](image)

- The TwinCAT 3 license manager opens.

5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. “TE1300: TC3 Scope View Professional”).

![Manage Licenses](image)

6. Open the **Order Information** tab.

- In the tabular overview of licenses, the previously selected license is displayed with the status “missing”.

![Order Information](image)

A TwinCAT 3 license is generally linked to two indices describing the platform to be licensed:
- **System ID**: Uniquely identifies the device
- **Platform level**: Defines the performance of the device

The corresponding **System Id** and **Platform** fields cannot be changed.
7. Enter the order number (License Id) for the license to be activated and optionally a separate order number (Customer Id), plus an optional comment for your own purposes (Comment). If you do not know your Beckhoff order number, please contact your Beckhoff sales contact.

8. Click the Generate File... button to create a License Request File for the listed missing license.
   - A window opens, in which you can specify where the License Request File is to be stored. (We recommend accepting the default settings.)

9. Select a location and click Save.
   - A prompt appears asking whether you want to send the License Request File to the Beckhoff license server for verification:
     - Click Yes to send the License Request File. A prerequisite is that an email program is installed on your computer and that your computer is connected to the internet. When you click Yes, the system automatically generates a draft email containing the License Request File with all the necessary information.
     - Click No if your computer does not have an email program installed on it or is not connected to the internet. Copy the License Request File onto a data storage device (e.g. a USB stick) and send the file from a computer with internet access and an email program to the Beckhoff license server (tclicense@beckhoff.com) by email.

10. Send the License Request File.
    - The License Request File is sent to the Beckhoff license server. After receiving the email, the server compares your license request with the specified order number and returns a License Response File by email. The Beckhoff license server returns the License Response File to the same email address from which the License Request File was sent. The License Response File differs from the License Request File only by a signature that documents the validity of the license file content. You can view the contents of the License Response File with an editor suitable for XML files (e.g. “XML Notepad”). The contents of the License Response File must not be changed, otherwise the license file becomes invalid.

11. Save the License Response File.
12. To import the license file and activate the license, click **License Response File...** in the **Order Information** tab.

13. Select the License Response File in your file directory and confirm the dialog.

![](image)

The License Response File is imported and the license it contains is activated. Existing demo licenses will be removed.

14. Restart the TwinCAT system.

The license becomes active when TwinCAT is restarted. The product can be used as a full version. During the TwinCAT restart the license file is automatically copied to the directory `\TwinCAT\3.1\Target \License` on the respective target system.
The AES70 specification describes objects representing functions and device states. These objects are handled with TwinCAT using function block methods. Each object has a unique object number that is defined by the device manufacturer.

For each AES70-compatible device, an instance of the function block FB_OcaDevice has to be called cyclically. The function block establishes the connection via TCP/IP. The parameterization of the function block (IP address, port etc.) is explained in the Example for using the function block FB_OcaDevice.

After calling the function block instance, you must call the methods of those function blocks that represent objects in the AES70-compatible device, for example so-called worker objects such as OcaMute, OcaSwitch, and so on.
7  PLC API

7.1  Function blocks

7.1.1  FB_OcaDevice

The function block FB_OcaDevice establishes the connection via TcpIp to devices that support the AES70 standard.

**Cyclically single call**

The instance of the function block FB_OcaDevice must be called cyclically once at the start of the PLC program.

**Syntax**

```plaintext
VAR_INPUT
  bEnable : BOOL;
END_VAR

VAR_OUTPUT
  ipResultEvent : Tc3_EventLogger.I_TcResultEvent;
  pLink : POINTER TO ST_Link;
  bBusy : BOOL;
  bActive : BOOL;
  bError : BOOL;
  stDeviceInfo : ST_OcaDeviceInfo;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bEnable</td>
<td>BOOL</td>
<td>Enables/disables execution of the function block and starts the &quot;keep alive&quot; mechanism. A rising edge at this input clears any pending errors (indicated by bError = TRUE). The property sDeviceName should be set before the activation. Parameters such as IP address, port, etc. should also be predefined in the global variable list &quot;GVL_AES70&quot;.</td>
</tr>
</tbody>
</table>
## Outputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ipTcResultEvent</td>
<td>Tc3_EventLogger.I_TcResultEvent</td>
<td>Result interface with detailed information on the return value</td>
</tr>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bBusy</td>
<td>BOOL</td>
<td>TRUE as long as the function block is called with bEnable = TRUE.</td>
</tr>
<tr>
<td>bActive</td>
<td>BOOL</td>
<td>Indicates that the device is ready to operate. This output is generally used to activate OCA objects.</td>
</tr>
<tr>
<td>bError</td>
<td>BOOL</td>
<td>TRUE, if an error occurs.</td>
</tr>
<tr>
<td>stDeviceInfo</td>
<td>ST_OcaDeviceInfo [59]</td>
<td>This structure provides information such as the time of the last sent or received message and the number of messages.</td>
</tr>
</tbody>
</table>

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AmsNetId</td>
<td>T_AmsNetID</td>
<td>Set</td>
<td>Local</td>
<td>AMS network ID of the device For the local computer (default) an empty string may be specified.</td>
</tr>
<tr>
<td>bLocalOcaDevice</td>
<td>BOOL</td>
<td></td>
<td>Local</td>
<td>Currently unused (intended for future extension)</td>
</tr>
<tr>
<td>sDeviceName</td>
<td>STRING</td>
<td>Set</td>
<td></td>
<td>Name of the OCA device to be used Parameters such as IP address and port are defined in the array aOcaDevices, which can be found in the global variable list GVL_AES70.</td>
</tr>
<tr>
<td>tAdsTimeout</td>
<td>TIME</td>
<td>Set</td>
<td></td>
<td>Maximum time allowed for the execution of the function block This input is internally preset to DEFAULT_ADS_TIMEOUT and does not have to be explicitly assigned.</td>
</tr>
</tbody>
</table>

## Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4022.2 or higher</td>
<td>PC or CX (x64, x86, ARM)</td>
<td>Tc3_AES70</td>
</tr>
</tbody>
</table>

See also: Example for using the function block FB_OcaDevice [60]

### 7.1.2 FB_OcaRoot

The function block FB_OcaRoot provides basic OCA functionality. It is the function block from which all other function blocks inherit.

- **No explicit call**
  Since there is no code in the body of the function block, it should not be called explicitly. Instead, use the corresponding methods of the function block.
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetClassIdentification</td>
<td>Local</td>
<td>This method can be used to query the ClassId and ClassVersion of the OCA object. If the query was successful, the result is output in the properties stClassID and stClassVersion.</td>
</tr>
<tr>
<td>GetLockable</td>
<td>Local</td>
<td>This method is used to query whether the OCA object can be locked. If the query was successful, the result is output in the property stLockable.</td>
</tr>
<tr>
<td>LockUnlock</td>
<td>Local</td>
<td>Method for unlocking or locking an OCA object. If the method was executed successfully, the result is output in the property stObjectLocked.</td>
</tr>
<tr>
<td>GetRole</td>
<td>Local</td>
<td>OCA objects can have a text label to make their meaning easier to recognize. If the method call was successful, the result is output in the property stRole.</td>
</tr>
<tr>
<td>Subscription</td>
<td>Local</td>
<td>You can use this method to request OCA objects, generate notifications for value changes, or set up sending. If the method was executed successfully, the result is output in the property stSubscribed.</td>
</tr>
</tbody>
</table>

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sClassId</td>
<td>STRING</td>
<td>Get</td>
<td>Local</td>
<td>Unique class name</td>
</tr>
<tr>
<td>stClassId</td>
<td>ST_CLASSIDPROPERTY[56]</td>
<td>Get</td>
<td>Local</td>
<td>Unique class name</td>
</tr>
<tr>
<td>stClassVersion</td>
<td>ST_UINT16PROPERTY[56]</td>
<td>Get</td>
<td>Local</td>
<td>OCA ClassVersion</td>
</tr>
<tr>
<td>stLockable</td>
<td>ST_BOOLEANPROPERTY[56]</td>
<td>Get</td>
<td>Local</td>
<td>Indicates whether the object can be locked</td>
</tr>
<tr>
<td>stSubscribed</td>
<td>ST_BOOLEANPROPERTY[56]</td>
<td>Get</td>
<td>Local</td>
<td>Shows whether the object was requested to generate notifications when value changes occur</td>
</tr>
<tr>
<td>stRole</td>
<td>ST_STRINGPROPERTY[56]</td>
<td>Get</td>
<td>Local</td>
<td>Task of the object in the device (e.g. Config_InputEnable)</td>
</tr>
<tr>
<td>nONo</td>
<td>UDINT</td>
<td>Get</td>
<td>Local</td>
<td>Unique number with which the instantiated object is uniquely defined</td>
</tr>
<tr>
<td>sOcaObjectDescription</td>
<td>STRING(32)</td>
<td>Set</td>
<td>Local</td>
<td>Arbitrary object name. This is used to assign further object properties to the function block that were defined in the array aOcaDevices (located in the global variable list GVL_AES70).</td>
</tr>
<tr>
<td>stObjectLocked</td>
<td>ST_BOOLEANPROPERTY[56]</td>
<td>Get</td>
<td>Local</td>
<td>Indicates whether the object is locked.</td>
</tr>
</tbody>
</table>

### AES70 standard

The names of variables and function blocks have been adapted to the AES70 standard where possible. Information about this communication protocol can be found at [www.aes.org](http://www.aes.org) and [www.ocauliance.com](http://www.ocauliance.com).
PLC API

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4022.2 or higher</td>
<td>PC or CX (x64, x86, ARM)</td>
<td>Tc3_AES70</td>
</tr>
</tbody>
</table>

See also: Example for using the function block FB_OcaRoot [60]

7.1.2.1 GetClassIdentification

The method GetClassIdentification can be used to query the ClassId and ClassVersion of the OCA object. If the query was successful, the result is output in the properties stClassID and stClassVersion.

**Cyclic method call**

Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

**Syntax**

```
VAR_INPUT
    pLink : POINTER TO ST_Link;
    bExecute : BOOL;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
</tbody>
</table>

7.1.2.2 GetLockable

The method GetLockable is used to query whether the OCA object can be locked. If the query was successful, the result is output in the property stLockable.

**Cyclic method call**

Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

**Syntax**

```
VAR_INPUT
    pLink : POINTER TO ST_Link;
    bExecute : BOOL;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
</tbody>
</table>

7.1.2.3 **LockUnlock**

The LockUnlock method can be used to unlock or lock an OCA object. If the method was executed successfully, the result is output in the property stObjectLocked.

**Cyclic method call**

Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

Syntax

```
VAR_INPUT
  pLink     :  POINTER TO ST_Link;
  bExecute  :  BOOL;
  bLock     :  BOOL; // If the Method is executed this Boolean Input decides whether the object
  eCmdMode  :  E_OcaMessageTypeCmd
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
<tr>
<td>bLock</td>
<td>BOOL</td>
<td>Variable that determines whether to lock (TRUE) or unlock (FALSE)</td>
</tr>
<tr>
<td>eCmdMode</td>
<td>E_OcaMessageTypeCmd</td>
<td>Depending on whether confirmation is required or not, this input variable is set to E_OcaMessageTypeCmd.OcaCmdRrq or E_OcaMessageTypeCmd.OcaCmd.</td>
</tr>
</tbody>
</table>

7.1.2.4 **GetRole**

OCA objects can have a text label to make their meaning easier to recognize. If the method call was successful, the result is output in the property stRole.
Cyclic method call
Since several PLC cycles may pass between sending and the response from the device, this
method should be executed cyclically.

Syntax
VAR_INPUT
  pLink : POINTER TO ST_Link;
  bExecute : BOOL;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
</tbody>
</table>

7.1.2.5 Subscription

You can use the Subscription method to request OCA objects, generate notifications for value changes, or set up sending of notifications. If the method was executed successfully, the result is output in the property stSubscribed.

Cyclic method call
Since several PLC cycles may pass between sending and the response from the device, this
method should be executed cyclically.

Syntax
VAR_INPUT
  pLink : POINTER TO ST_Link;
  bExecute : BOOL;
  eMode : E_SubscriptionMode;
  nSubscriptionNo : UDINT;
  eCmdMode : E_OcaMessageTypeCmd;
END_VAR
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
<tr>
<td>eMode</td>
<td>E_SubscriptionMode</td>
<td>Depending on whether a notification request is to be created or deleted, the input of the method can be set to E_SubscriptionMode.ADD_Subscription or E_SubscriptionMode.DELETE_Subscription.</td>
</tr>
<tr>
<td>nSubscriptionNo</td>
<td>UDINT</td>
<td>Unique number for localizing the respective subscription</td>
</tr>
<tr>
<td>eCmdMode</td>
<td>E_OcaMessageTypeCmd</td>
<td>Depending on whether confirmation is required or not, this input variable is set to E_OcaMessageTypeCmd.OcaCmdRrq or E_OcaMessageTypeCmd.OcaCmd.</td>
</tr>
</tbody>
</table>

### 7.1.3 Worker function blocks

Worker function blocks are used to control OCA worker objects.

#### AES70 standard

The names of variables and function blocks have been adapted to the AES70 standard where possible. Information about this communication protocol can be found at [www.aes.org](http://www.aes.org) and [www.ocaaliance.com](http://www.ocaaliance.com).

#### 7.1.3.1 FB_OcaWorker

The function block FB_OcaWorker extends the function block FB_OcaRoot with properties and methods for handling the OCA objects.

#### No explicit call

Since there is no code in the body of the function block, it should not be called explicitly. Instead, use the corresponding methods of the function block.

**Inheritance hierarchy**

FB_OcaRoot

FB_OcaWorker
## Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetEnabled / SetEnabled</td>
<td>Local</td>
<td>Method for querying or setting the Enabled property of the OCA object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the query was successful, the result is output in the property stEnabled.</td>
</tr>
<tr>
<td>GetLabel / SetLabel / RNtfLabel</td>
<td>Local</td>
<td>This method is used to query whether the OCA object can be locked.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the query was successful, the result is output in the property stLockable.</td>
</tr>
<tr>
<td>LockUnlock</td>
<td>Local</td>
<td>Method for unlocking or locking an OCA object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the method was executed successfully, the result is output in the property stObjectLocked.</td>
</tr>
<tr>
<td>GetRole</td>
<td>Local</td>
<td>OCA objects can have a text label to make their meaning easier to recognize.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the method call was successful, the result is output in the property stRole.</td>
</tr>
<tr>
<td>Subscription</td>
<td>Local</td>
<td>You can use this method to request OCA objects, generate notifications for value changes, or set up sending.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the method was executed successfully, the result is output in the property stSubscribed.</td>
</tr>
</tbody>
</table>

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stEnabled</td>
<td>ST_BooleanProperty</td>
<td>Get</td>
<td>Local</td>
<td>Indicates whether the OCA object in the corresponding OCA device is enabled.</td>
</tr>
<tr>
<td>stLabel</td>
<td>ST_StringProperty</td>
<td>Get</td>
<td>Local</td>
<td>OCA objects can have a description text that is stored in this property to facilitate recognition.</td>
</tr>
<tr>
<td>stLatency</td>
<td>ST_FLOAT32Property</td>
<td>Get</td>
<td>Local</td>
<td>Processing latency of the OCA object</td>
</tr>
<tr>
<td>stOwner</td>
<td>ST_UINT32Property</td>
<td>Get</td>
<td>Local</td>
<td>OCA object number of the higher-level object</td>
</tr>
</tbody>
</table>

## AES70 standard

The names of variables and function blocks have been adapted to the AES70 standard where possible. Information about this communication protocol can be found at [www.aes.org](http://www.aes.org) and [www.ocaalliance.com](http://www.ocaalliance.com).

## Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4022.2 or higher</td>
<td>PC or CX (x64, x86, ARM)</td>
<td>Tc3_AES70</td>
</tr>
</tbody>
</table>

See also: [Example for using the function block FB_OcaWorker](#)
7.1.3.1.1 GetEnabled / SetEnabled

GetEnabled

The GetEnabled method is used to query the Enabled property of the OCA object. If the query was successful, the result is output in the property stEnabled.

Cyclic method call
Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

Syntax

```plaintext
VAR_INPUT
  pLink      : POINTER TO ST_Link;
  bExecute   : BOOL;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
</tbody>
</table>

SetEnabled

The SetEnabled method can be used to enable or disable an OCA object. If the method was executed successfully, the result is output in the property stEnabled.

Cyclic method call
Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

Syntax

```plaintext
VAR_INPUT
  pLink      : POINTER TO ST_Link;
  bExecute   : BOOL;
  bValEnabled: BOOL;
  eCmdMode   : E_OcaMessageTypeCmd;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device.</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
<tr>
<td>bValEnabled</td>
<td>BOOL</td>
<td>Depending on whether the OCA object is to be enabled or disabled, this input must be assigned TRUE or FALSE.</td>
</tr>
<tr>
<td>eCmdMode</td>
<td>E_OcaMessageTypeDef</td>
<td>Depending on whether confirmation is required or not, this input variable is set to E_OcaMessageTypeDef.OcaCmdRqq or E_OcaMessageTypeDef.OcaCmd.</td>
</tr>
</tbody>
</table>

#### 7.1.3.1.2 GetLabel / SetLabel / RNtfLabel

**GetLabel**

OCA objects can be labeled to facilitate recognition. The GetLabel method is used to query this property of the OCA object. If the query was successful, the result is output in the property stLabel.

**Syntax**

```plaintext
VAR_INPUT
    pLink : POINTER TO ST_Link;
    bExecute : BOOL;
END_VAR
```

**Cyclic method call**

Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

**SetLabel**

The SetLabel method can be used to set the labeling of an OCA object. If the method was executed successfully, the result is output in the property stLabel.

**Syntax**

```plaintext
VAR_INPUT
    pLink : POINTER TO ST_Link;
    bExecute : BOOL;
    sLabel : STRING(255);
END_VAR
```

**Cyclic method call**

Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

---

**Version:** 1.2  
**TC3 AES70 (OCA) Communication**
### Syntax

```plaintext
VAR_INPUT
    pLink : POINTER TO ST_Link;
    bExecute : BOOL;
    sLabel : STRING;
    eCmdMode : E_OcaMessageTypeCmd;
END_VAR
```

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
<tr>
<td>sLabel</td>
<td>STRING</td>
<td>Label for the OCA object</td>
</tr>
<tr>
<td>eCmdMode</td>
<td>E_OcaMessageTypeCmd</td>
<td>Depending on whether confirmation is required or not, this input variable is set to E_OcaMessageTypeCmd.OcaCmdRrq or E_OcaMessageTypeCmd.OcaCmd.</td>
</tr>
</tbody>
</table>

---

**RNtfLabel**

```plaintext
VAR_INPUT
    pLink : POINTER TO ST_Link;
    bEnable : BOOL;
    nSubscribtionNo : UDINT;
END_VAR
```

#### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device.</td>
</tr>
<tr>
<td>bEnable</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bEnable input.</td>
</tr>
<tr>
<td>nSubscribtionNo</td>
<td>UDINT</td>
<td>Unique number for localizing the respective subscription.</td>
</tr>
</tbody>
</table>

---

### Cyclic method call

Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

---

### 7.1.3.1.3 GetLatency / SetLatency / RNtfLatency

#### GetLatency

```plaintext
VAR_INPUT
    pLink : POINTER TO ST_Link;
    bExecute : BOOL;
END_VAR
```
The `GetLatency` method is used to query the processing latency of the OCA object. If the query was successful, the result is output in the property `stLatency`.

**Cyclic method call**
Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

**Syntax**

```plaintext
VAR_INPUT
    pLink : POINTER TO ST_Link;
    bExecute : BOOL;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
</tbody>
</table>

The `SetLatency` method can be used to set the processing latency of an OCA object. If the method was executed successfully, the result is output in the property `stLatency`.

**Cyclic method call**
Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

**Syntax**

```plaintext
VAR_INPUT
    pLink : POINTER TO ST_Link;
    bExecute : BOOL;
    fLatency : REAL;
    eCmdMode : E_OcaMessageTypeCmd;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
<tr>
<td>fLatency</td>
<td>REAL</td>
<td>Value to be assigned to this property of the OCA object</td>
</tr>
<tr>
<td>eCmdMode</td>
<td>E_OcaMessageTypeCmd</td>
<td>Depending on whether confirmation is required or not, this input variable is set to E_OcaMessageTypeCmd.OcaCmdRrq or E_OcaMessageTypeCmd.OcaCmd.</td>
</tr>
</tbody>
</table>
RNtfLatency

If the OCA object was prompted to notify changes (using the Subscription method), incoming notifications are read using the RNtfLatency method.

**Cyclic method call**
Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

**Syntax**

```plaintext
VAR_INPUT
  pLink : POINTER TO ST_Link;
  bEnable : BOOL;
  nSubscriptionNo : UDINT;
END_VAR
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device.</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
<tr>
<td>nSubscriptionNo</td>
<td>UDINT</td>
<td>Unique number for localizing the respective subscription.</td>
</tr>
</tbody>
</table>

7.1.3.1.4 GetOwner

The GetOwner method is used to query the higher-level element. If the query was successful, the result is output in the property stOwner.

**Cyclic method call**
Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

**Syntax**

```plaintext
VAR_INPUT
  pLink : POINTER TO ST_Link;
  bExecute : BOOL;
END_VAR
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device.</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
</tbody>
</table>
7.1.3.2 Actuator function blocks

Actuator objects are worker objects whose purpose is the control of electronic functions.

- **AES70 standard**
  
  The names of variables and function blocks have been adapted to the AES70 standard where possible. Information about this communication protocol can be found at [www.aes.org](http://www.aes.org) and [www.ocaalliance.com](http://www.ocaalliance.com).

### 7.1.3.2.1 FB_OcaActuator

The function block FB_OcaActuator extends the function block FB_OcaWorker. It is the function block from which all function blocks that are assigned to the Actuator category inherit. FB_OcaActuator has neither properties nor methods.

- **No explicit call**
  
  Since there is no code in the body of the function block, it should not be called explicitly. Instead, use the corresponding methods of the function block.

**Inheritance hierarchy**

FB_OcaRoot

FB_OcaWorker

FB_OcaActuator

**Requirements**

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4022.2 or higher</td>
<td>PC or CX (x64, x86, ARM)</td>
<td>Tc3_AES70</td>
</tr>
</tbody>
</table>

### 7.1.3.2.2 FB_OcaGain

The function block FB_OcaGain provides properties and methods for handling OCA objects for gain adjustment.

- **No explicit call**
  
  Since there is no code in the body of the function block, it should not be called explicitly. Instead, use the corresponding methods of the function block.

**Inheritance hierarchy**

FB_OcaRoot

FB_OcaWorker

FB_OcaActuator

FB_OcaGain
Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetGain</td>
<td>Local</td>
<td>Used to query the properties Gain, GainMAX and GainMIN of the OCA object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the query was successful, the result is output in the properties stGain,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>stGainMAX and stGainMIN.</td>
</tr>
<tr>
<td>SetGain</td>
<td>Local</td>
<td>Use this method to set the gain of an OCA object in dB.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the method was executed successfully, the result is output in the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>property stGain.</td>
</tr>
<tr>
<td>RNtfGain</td>
<td>Local</td>
<td>If the OCA object was prompted to notify changes (using the Subscription</td>
</tr>
<tr>
<td></td>
<td></td>
<td>method), this method is used to read incoming notifications.</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stGain</td>
<td>ST_FLOAT32Property [56]</td>
<td>Get</td>
<td>Local</td>
<td>Gain in dB</td>
</tr>
<tr>
<td>stGainMAX</td>
<td>ST_FLOAT32Property [56]</td>
<td>Get</td>
<td>Local</td>
<td>Maximum gain value in dB</td>
</tr>
<tr>
<td>stGainMIN</td>
<td>ST_FLOAT32Property [56]</td>
<td>Get</td>
<td>Local</td>
<td>Minimum gain value in dB</td>
</tr>
</tbody>
</table>

AES70 standard

The names of variables and function blocks have been adapted to the AES70 standard where possible. Information about this communication protocol can be found at www.aes.org and www.ocaaliance.com.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4022.2 or higher</td>
<td>PC or CX (x64, x86, ARM)</td>
<td>Tc3_AES70</td>
</tr>
</tbody>
</table>

See also: Example for using the function block FB_OcaGain [62]

GetGain / SetGain / RNtfGain

GetGain

The GetGain method is used to query the properties Gain, GainMAX and GainMIN of the OCA object. If the query was successful, the result is output in the properties stGain, stGainMAX and stGainMIN.

Cyclic method call

Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.
Syntx

VAR_INPUT
  pLink : POINTER TO ST_Link;
  bExecute : BOOL;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
</tbody>
</table>

SetGain

Use the SetGain method to set the gain of an OCA object in dB. If the method was executed successfully, the result is output in the property stGain.

- **Cyclic method call**
  Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

Syntx

VAR_INPUT
  pLink : POINTER TO ST_Link;
  bExecute : BOOL;
  fGain : REAL;
  eCmdMode : E_OcaMessageTypeCmd;
END_VAR

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
<tr>
<td>fGain</td>
<td>REAL</td>
<td>Gain value in dB to which the OCA object is to be set.</td>
</tr>
<tr>
<td>eCmdMode</td>
<td>E_OcaMessageTypeCmd</td>
<td>Depending on whether confirmation is required or not, this input variable is set to E_OcaMessageTypeCmd.OcaCmdRrq or E_OcaMessageTypeCmd.OcaCmd.</td>
</tr>
</tbody>
</table>
RNtfGain

If the OCA object was prompted to notify changes (using the Subscription [24] method), incoming notifications are read using the RNtfGain method.

**Cyclic method call**
Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

**Syntax**

```
VAR_INPUT
  pLink            : POINTER TO ST_Link;
  bEnable          : BOOL;
  nSubscriptionNo : UDINT;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device.</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
<tr>
<td>nSubscriptionNo</td>
<td>UDINT</td>
<td>Unique number for localizing the respective subscription.</td>
</tr>
</tbody>
</table>

7.1.3.2.3 FB_OcaSwitch

The function block FB_OcaSwitch extends the function block FB_OcaActuator with properties and methods and for handling OCA switch objects. These objects can have multiple switch positions. Individual positions can be enabled or disabled and given names.

**No explicit call**
Since there is no code in the body of the function block, it should not be called explicitly. Instead, use the corresponding methods of the function block.

**Inheritance hierarchy**

FB_OcaRoot
  FB_OcaWorker
    FB_OcaActuator
    FB_OcaSwitch
Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetPosition</td>
<td>Local</td>
<td>This method is used to query the Position, PositionMAX, and PositionMIN properties of the OCA object.</td>
</tr>
<tr>
<td>SetPosition</td>
<td>Local</td>
<td>Use this method to set the switch position of an OCA object.</td>
</tr>
<tr>
<td>RNtfPosition</td>
<td>Local</td>
<td>This method reads incoming notifications</td>
</tr>
<tr>
<td>GetPositionEnabled</td>
<td>Local</td>
<td>Method for querying a switch position</td>
</tr>
<tr>
<td>SetPositionEnabled</td>
<td>Local</td>
<td>Method for enabling or disabling a switch position</td>
</tr>
<tr>
<td>GetPositionName</td>
<td>Local</td>
<td>Method for querying the name of a switch position</td>
</tr>
<tr>
<td>SetPositionName</td>
<td>Local</td>
<td>Use this method to assign a name to a switch position.</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stPosition</td>
<td>ST_UINT16Property</td>
<td>Get</td>
<td>Local</td>
<td>Indicates the switch position of the OCA object.</td>
</tr>
<tr>
<td>stPosition MAX</td>
<td>ST_UINT16Property</td>
<td>Get</td>
<td>Local</td>
<td>Indicates the lowest switch position of the OCA object.</td>
</tr>
<tr>
<td>stPosition MIN</td>
<td>ST_UINT16Property</td>
<td>Get</td>
<td>Local</td>
<td>Indicates the highest switch position of the OCA object.</td>
</tr>
<tr>
<td>stPosition Enabled</td>
<td>ST_PositionEnabledProperty</td>
<td>Get</td>
<td>Local</td>
<td>Indicates whether a particular switch position is enabled or disabled.</td>
</tr>
<tr>
<td>stPosition Name</td>
<td>ST_PositionNameProperty</td>
<td>Get</td>
<td>Local</td>
<td>Indicates the position name of a particular switch position.</td>
</tr>
</tbody>
</table>

AES70 standard

The names of variables and function blocks have been adapted to the AES70 standard where possible. Information about this communication protocol can be found at [www.aes.org](http://www.aes.org) and [www.ocaalliance.com](http://www.ocaalliance.com).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4022.2 or higher</td>
<td>PC or CX (x64, x86, ARM)</td>
<td>Tc3_AES70</td>
</tr>
</tbody>
</table>
GetPosition / SetPosition / RNtfPosition

GetPosition

The GetPosition method is used to query the Position, PositionMAX, and PositionMIN properties of the OCA object. If the query was successful, the result is output in the properties stPosition, stPositionMAX and stPositionMIN.

**Cyclic method call**

Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

### Syntax

```plaintext
VAR_INPUT
    pLink : POINTER TO ST_Link; //Pointer to address of the structure which links the OCA objects to the OCA device
    bExecute : BOOL; //The Method is triggered by a rising edge at this input.
END_VAR
```

### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
</tbody>
</table>

SetPosition

The SetPosition method can be used to set the switch position of an OCA object. If the method was executed successfully, the result is output in the property stPosition.

**Cyclic method call**

Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

### Syntax

```plaintext
VAR_INPUT
    pLink : POINTER TO ST_Link; //Pointer to address of the structure which links the OCA objects to the OCA device
    bExecute : BOOL; //The Method is triggered by a rising edge at this input.
    nPosition : UINT;
    eCmdMode : E_OcaMessageTypeCmdbLock;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device.</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
<tr>
<td>nIndex</td>
<td>UINT</td>
<td>Switch position for the OCA object.</td>
</tr>
<tr>
<td>eCmdMode</td>
<td>E_OcaMessageTypeCmd</td>
<td>Depending on whether confirmation is required or not, this input variable is set to E_OcaMessageTypeCmd.OcaCmdRqr or E_OcaMessageTypeCmd.OcaCmd.</td>
</tr>
</tbody>
</table>

### RNtfPosition

If the OCA object was prompted to notify changes (using the `Subscription` method), incoming notifications are read using the RNtfPosition method.

- **Cyclic method call**
  Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

### Syntax

```plaintext
VAR_INPUT
  pLink : POINTER TO ST_Link;
  bEnable : BOOL;
  nIndex : UDINT;
END_VAR
```

## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device.</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
<tr>
<td>nIndex</td>
<td>UDINT</td>
<td>Unique number for localizing the respective subscription.</td>
</tr>
</tbody>
</table>

## GetPositionEnabled / SetPositionEnabled

### GetPositionEnabled

The GetPositionEnabled method is used to query whether the respective switch position determined by nIndex is enabled or disabled. If the query was successful, the result is output in the property stPositionEnabled.
Cyclic method call
Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

Syntax

```plaintext
VAR_INPUT
  pLink      : POINTER TO ST_Link; //Pointer to address of the structure which links the OCA objects to the OCA device
  bExecute   : BOOL; //The Method is triggered by a rising edge at this input.
  nIndex     : UINT; //The Index of the queried Position
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
<tr>
<td>nIndex</td>
<td>UINT</td>
<td>Index of the switch position to be queried</td>
</tr>
</tbody>
</table>

SetPositionEnabled

The SetPositionEnabled method can be used to enable or disable a specific switch position specified by nIndex. If the query was successful, the result is output in the property stPositionEnabled.

Cyclic method call
Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

Syntax

```plaintext
VAR_INPUT
  pLink      : POINTER TO ST_Link; //Pointer to address of the structure which links the OCA objects to the OCA device
  bExecute   : BOOL; //The Method is triggered by a rising edge at this input.
  bValEnabled: BOOL; //Set this Input to TRUE to enable the Position specified by nIndex or FALSE to disable this Position
  nIndex     : UINT; //The Index of the Position which should be modified
  eCmdMode   : E_OcaMessageTypeCmdLock;
END_VAR
```
Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
<tr>
<td>bValEnabled</td>
<td>BOOL</td>
<td>If this input is set to TRUE, the switch position selected by nIndex is enabled when this method is executed. If the input is set to FALSE, the switch position is disabled.</td>
</tr>
<tr>
<td>nIndex</td>
<td>UINT</td>
<td>Switch position for the OCA object.</td>
</tr>
<tr>
<td>eCmdMode</td>
<td>E_OcaMessageTypeCmdbLock</td>
<td>Depending on whether confirmation is required or not, this input variable is set to E_OcaMessageTypeCmd.OcaCmdRrq or E_OcaMessageTypeCmd.OcaCmd.</td>
</tr>
</tbody>
</table>

7.1.3.2.4 FB_OcaMute

The function block FB_OcaMute provides properties and methods for handling OCA objects for muting.

- **No explicit call**
  Since there is no code in the body of the function block, it should not be called explicitly. Instead, use the corresponding methods of the function block.

Inheritance hierarchy

FB_OcaRoot
  - FB_OcaWorker
    - FB_OcaActuator
      - FB_OcaMute

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetMuteState</td>
<td>Local</td>
<td>This method is used to query the Position, PositionMAX, and PositionMIN properties of the OCA object.</td>
</tr>
<tr>
<td>SetMuteState</td>
<td>Local</td>
<td>Use this method to set the switch position of an OCA object.</td>
</tr>
<tr>
<td>RNtfMuteState</td>
<td>Local</td>
<td>This method reads incoming notifications.</td>
</tr>
</tbody>
</table>

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stMuteState</td>
<td>ST_MuteProperty</td>
<td>Get</td>
<td>Local</td>
<td>Indicates whether the OCA object is muted or not.</td>
</tr>
</tbody>
</table>
AES70 standard
The names of variables and function blocks have been adapted to the AES70 standard where possible. Information about this communication protocol can be found at [www.aes.org](http://www.aes.org) and [www.ocaaliance.com](http://www.ocaaliance.com).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4022.2 or higher</td>
<td>PC or CX (x64, x86, ARM)</td>
<td>Tc3_AES70</td>
</tr>
</tbody>
</table>

GetMuteState / SetMuteState / RNtfMuteState

GetMuteState

```plaintext
VAR_INPUT
    pLink    : POINTER TO ST_Link;
    bExecute : BOOL;
END_VAR
```

The GetPosition method is used to query the MuteState property of the OCA object. If the query was successful, the result is output in the property stMuteState.

Cyclic method call
Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

Syntax

```plaintext
VAR_INPUT
    pLink    : POINTER TO ST_Link;
    bExecute : BOOL;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
</tbody>
</table>

SetMuteState

```plaintext
VAR_INPUT
    pLink          : POINTER TO ST_Link;
    bExecute       : BOOL;
    eMuteState     : E_OcaMuteState;
    eChildMode     : E_OcaMessageTypeEnd;
END_VAR
```

The SetMuteState method can be used to set the muting feature of an OCA object. If the method was executed successfully, the result is output in the property stMuteState.

Cyclic method call
Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.
**Syntax**

```
VAR_INPUT
   pLink : POINTER TO ST_Link;
   bExecute : BOOL;
   eMuteState : E_OcaMuteState;
   eCmdMode : E_OcaMessageTypeCmd;
END_VAR
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
<tr>
<td>eMuteState</td>
<td>E_OcaMuteState[59]</td>
<td>Muting setting (muted/unmuted) to be assigned to the OCA object.</td>
</tr>
<tr>
<td>eCmdMode</td>
<td>OcaMessageTypeCmd</td>
<td>Depending on whether confirmation is required or not, this input variable is set to E_OcaMessageTypeCmd.OcaCmdRrq or E_OcaMessageTypeCmd.OcaCmd.</td>
</tr>
</tbody>
</table>

** RNtfMuteState **

```
RNtfMuteState
   pLink : POINTER TO ST_Link;
   bEnable : BOOL;
   nSubscriptionNo : UDINT;
```

If the OCA object was prompted to notify changes (using the Subscription[24] method), incoming notifications are read using the RNtfMuteState method.

---

** Cyclic method call **

Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

---

**Syntax**

```
VAR_INPUT
   pLink : POINTER TO ST_Link;
   bEnable : BOOL;
   nSubscriptionNo : UDINT;
END_VAR
```

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bEnable</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
<tr>
<td>nSubscriptionNo</td>
<td>UDINT</td>
<td>Unique number for localizing the respective subscription.</td>
</tr>
</tbody>
</table>

7.1.3.2.5  FB_OcaBasicActuator
The function block FB_OcaBasicActuator extends the function block FB_OcaActuator. It is the function block from which all function blocks that are assigned to the BasicActuator category inherit. FB_OcaBasicActuator has neither properties nor methods.

Inheritance hierarchy

FB_OcaRoot
   FB_OcaWorker
      FB_OcaActuator
         FB_OcaBasicActuator

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4022.2 or higher</td>
<td>PC or CX (x64, x86, ARM)</td>
<td>Tc3_AES70</td>
</tr>
</tbody>
</table>

7.1.3.2.6 FB_OcaBooleanActuator

The function block FB_OcaBooleanActuator extends the function block FB_OcaBasicActuator with properties and methods for handling OcaBooleanActuator objects.

- **No explicit call**

  Since there is no code in the body of the function block, it should not be called explicitly. Instead, use the corresponding methods of the function block.

Inheritance hierarchy

FB_OcaRoot
   FB_OcaWorker
      FB_OcaActuator
         FB_OcaBasicActuator
            FB_OcaBooleanActuator

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetSetting</td>
<td>Local</td>
<td>This method is used to query a Boolean property.</td>
</tr>
<tr>
<td>SetSetting</td>
<td>Local</td>
<td>This method can be used to set the value of a Boolean property.</td>
</tr>
<tr>
<td>RNtfSetting</td>
<td>Local</td>
<td>This method reads incoming notifications</td>
</tr>
</tbody>
</table>
## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stSetting</td>
<td>ST_BooleanProperty [56]</td>
<td>Get</td>
<td>Local</td>
<td>Indicates the switch position of the OCA object.</td>
</tr>
</tbody>
</table>

### AES70 standard

The names of variables and function blocks have been adapted to the AES70 standard where possible. Information about this communication protocol can be found at [www.aes.org](http://www.aes.org) and [www.ocaalliance.com](http://www.ocaalliance.com).

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4022.2 or higher</td>
<td>PC or CX (x64, x86, ARM)</td>
<td>Tc3_AES70</td>
</tr>
</tbody>
</table>

### GetSetting / SetSetting / RNtfSetting

#### GetSetting

The GetSetting method is used to query the Boolean property of the OCA object. If the query was successful, the result is output in the property stSetting.

**Cyclic method call**

Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

**Syntax**

```plaintext
VAR_INPUT
    pLink : POINTER TO ST_Link;
    bExecute : BOOL;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
</tbody>
</table>

#### SetSetting

The SetSetting method can be used to set the muting feature of an OCA object. If the method was executed successfully, the result is output in the property stMuteState.
Cyclic method call
Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

Syntax

```plaintext
VAR_INPUT
  pLink : POINTER TO ST_Link;
  bExecute : BOOL;
  bSetting : BOOL;
  eCmdMode : E_OcaMessageTypeCmd;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device.</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
<tr>
<td>bSetting</td>
<td>BOOL</td>
<td>Value to be assigned to the property of the OCA object.</td>
</tr>
<tr>
<td>eCmdMode</td>
<td>OcaMessageTypeCmd</td>
<td>Depending on whether confirmation is required or not, this input variable is set to E_OcaMessageTypeCmd.OcaCmdRrq or E_OcaMessageTypeCmd.OcaCmd.</td>
</tr>
</tbody>
</table>

RNtfMuteState

If the OCA object was prompted to notify changes (using the Subscription [24] method), incoming notifications are read using the RNtfSetting method.

Cyclic method call
Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

Syntax

```plaintext
VAR_INPUT
  pLink : POINTER TO ST_Link;
  bEnable : BOOL;
  nSubscriptionNo : UDINT;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device.</td>
</tr>
<tr>
<td>bEnable</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
<tr>
<td>nSubscriptionNo</td>
<td>UDINT</td>
<td>Unique number for localizing the respective subscription.</td>
</tr>
</tbody>
</table>
7.1.3.2.7 FB_OcaStringActuator

The function block FB_OcaStringActuator extends the function block FB_OcaBasicActuator with properties and methods for handling OcaStringActuator objects.

### No explicit call

Since there is no code in the body of the function block, it should not be called explicitly. Instead, use the corresponding methods of the function block.

### Inheritance hierarchy

FB_OcaRoot
- FB_OcaWorker
  - FB_OcaActuator
    - FB_OcaBasicActuator
    - FB_OcaStringActuator

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>getValue [47]</td>
<td>Local</td>
<td>This method is used to query a signed property (integer, signed data type, 32 bits).</td>
</tr>
<tr>
<td>setValue [47]</td>
<td>Local</td>
<td>This method can be used to set the value of the property.</td>
</tr>
<tr>
<td>RntfValue [47]</td>
<td>Local</td>
<td>This method reads incoming notifications</td>
</tr>
<tr>
<td>getMaxLen [48]</td>
<td>Local</td>
<td>Use this method to query the maximum length of the string that the OCA object accepts.</td>
</tr>
</tbody>
</table>

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stSetting</td>
<td>ST_STRINGProperty [56]</td>
<td>Get</td>
<td>Local</td>
<td>Contains the string</td>
</tr>
<tr>
<td>stMaxLen</td>
<td>ST_UINT16Property [56]</td>
<td>Get</td>
<td>Local</td>
<td>Contains the maximum length of the string</td>
</tr>
</tbody>
</table>

### AES70 standard

The names of variables and function blocks have been adapted to the AES70 standard where possible. Information about this communication protocol can be found at [www.aes.org](http://www.aes.org) and [www.oca-alliance.com](http://www.oca-alliance.com).

### Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4022.2 or higher</td>
<td>PC or CX (x64, x86, ARM)</td>
<td>Tc3_AES70</td>
</tr>
</tbody>
</table>
GetValue / SetValue / RNtfValue

GetValue

The `GetValue` method is used to query the property of the OCA object. If the query was successful, the result is output in the property `stSetting`.

**Cyclic method call**

Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

**Syntax**

```plaintext
VAR_INPUT
    pLink    : POINTER TO ST_Link;
    bExecute : BOOL;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
</tbody>
</table>

SetValue

The `SetValue` method can be used to set the property of an OCA object. If the method was executed successfully, the result is output in the property `stSetting`.

**Cyclic method call**

Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

**Syntax**

```plaintext
VAR_INPUT
    pLink    : POINTER TO ST_Link;
    bExecute : BOOL;
    sValue   : STRING;
    eCmdMode : E_OcaMessageTypeCmd;
END_VAR
```
## Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device.</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
<tr>
<td>nSetting</td>
<td>UDINT</td>
<td>Value to be assigned to the property of the OCA object.</td>
</tr>
<tr>
<td>eCmdMode</td>
<td>OcaMessageTypeCmd</td>
<td>Depending on whether confirmation is required or not, this input variable is set to E_OcaMessageTypeCmd.OcaCmdRrq or E_OcaMessageTypeCmd.OcaCmd.</td>
</tr>
</tbody>
</table>

### RNtfValue

If the OCA object was prompted to notify changes (using the Subscription method), incoming notifications are read using the RNtfValue method.

### Cyclic method call

Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

### Syntax

```plaintext
VAR_INPUT
  pLink : POINTER TO ST_Link;
  bEnable : BOOL;
  nSubscriptionNo : UDINT;
END_VAR
```

## GetMaxLen

The GetMaxLen method is used to query the maximum accepted length of the string of the OCA object (OcaStringActuator). If the query was successful, the result is output in the property stMaxLen.

### Cyclic method call

Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.
Syntax

VAR_INPUT

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
</tbody>
</table>

7.1.3.2.8 FB_OcaInt32Actuator

The function block FB_OcaInt32Actuator extends the function block FB_OcaBasicActuator with properties and methods for handling OcaInt32Actuator objects.

**No explicit call**

Since there is no code in the body of the function block, it should not be called explicitly. Instead, use the corresponding methods of the function block.

Inheritance hierarchy

FB_OcaRoot

FB_OcaWorker

FB_OcaActuator

FB_OcaBasicActuator

FB_OcaInt32Actuator

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetValue</td>
<td>Local</td>
<td>This method is used to query a signed property (integer, signed data type, 32 bits).</td>
</tr>
<tr>
<td>SetValue</td>
<td>Local</td>
<td>This method can be used to set the value of the property.</td>
</tr>
<tr>
<td>RNotfValue</td>
<td>Local</td>
<td>This method reads incoming notifications.</td>
</tr>
</tbody>
</table>
Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stSetting</td>
<td>ST_INT32Property</td>
<td>Get</td>
<td>Local</td>
<td>Contains the value (integer, signed data type, 32 bit).</td>
</tr>
<tr>
<td>stSettingMAX</td>
<td>ST_INT32Property</td>
<td>Get</td>
<td>Local</td>
<td>Contains the maximum value (integer, signed data type, 32 bit).</td>
</tr>
<tr>
<td>stSettingMIN</td>
<td>ST_INT32Property</td>
<td>Get</td>
<td>Local</td>
<td>Contains the minimum value (integer, signed data type, 32 bit).</td>
</tr>
</tbody>
</table>

AES70 standard

The names of variables and function blocks have been adapted to the AES70 standard where possible. Information about this communication protocol can be found at [www.aes.org](http://www.aes.org) and [www.ocalliance.com](http://www.ocalliance.com).

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4022.2 or higher</td>
<td>PC or CX (x64, x86, ARM)</td>
<td>Tc3_AES70</td>
</tr>
</tbody>
</table>

GetValue / SetValue / RNtfValue

GetValue

```plaintext
VAR_INPUT
    pLink    : POINTER TO ST_Link;
    bExecute : BOOL;
END_VAR
```

The GetValue method is used to query the property of the OCA object. If the query was successful, the result is output in the properties stSetting, stSettingMAX and stSettingMIN.

Cyclic method call

Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

Syntax

```plaintext
VAR_INPUT
    pLink    : POINTER TO ST_Link;
    bExecute : BOOL;
END_VAR
```

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
</tbody>
</table>
SetValue

The SetValue method can be used to set the property of an OCA object. If the method was executed successfully, the result is output in the property stSetting.

**Cyclic method call**
Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

**Syntax**

```plaintext
VAR_INPUT
  pLink  : POINTER TO ST_Link;
  bExecute : BOOL;
  nSetting : UDINT;
  eCmdMode : E_OcaMessageTypeCmd;
END_VAR
```

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device.</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
<tr>
<td>nSetting</td>
<td>UDINT</td>
<td>Value to be assigned to the property of the OCA object.</td>
</tr>
<tr>
<td>eCmdMode</td>
<td>OcaMessageTypeCmd</td>
<td>Depending on whether confirmation is required or not, this input variable is set to E_OcaMessageTypeCmd.OcaCmdRrq or E_OcaMessageTypeCmd.OcaCmd.</td>
</tr>
</tbody>
</table>

RntfValue

If the OCA object is prompted to notify changes (using the Subscription [p.24] method), incoming notifications are read using the RntfValue method.

**Cyclic method call**
Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

**Syntax**

```plaintext
VAR_INPUT
  pLink  : POINTER TO ST_Link;
  bEnable : BOOL;
  nSubscriptionNo : UDINT;
END_VAR
```
### Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device.</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
<tr>
<td>nSubscriptionNo</td>
<td>UDINT</td>
<td>Unique number for localizing the respective subscription.</td>
</tr>
</tbody>
</table>

### 7.1.3.3 Sensor function blocks

Sensors enable querying of different parameters.

- **AES70 standard**
  - The names of variables and function blocks have been adapted to the AES70 standard where possible. Information about this communication protocol can be found at [www.aes.org](http://www.aes.org) and [www.ocaal-liance.com](http://www.ocaal-liance.com).

### 7.1.3.3.1 FB_OcaSensor

The function block FB_OcaSensor extends the function block FB_OcaWorker. It is the function block from which all function blocks that are assigned to the Sensor category inherit.

- **No explicit call**
  - Since there is no code in the body of the function block, it should not be called explicitly. Instead, use the corresponding methods of the function block.

#### Inheritance hierarchy

- FB_OcaRoot
- FB_OcaWorker
- FB_OcaSensor

#### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetReadingState</td>
<td>Local</td>
<td>You can use this method to query whether the value that was read is valid or not. If the query was successful, the result is output in the property stReadingState.</td>
</tr>
</tbody>
</table>

#### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Access</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stReadingState</td>
<td>ST_SensorReadingState</td>
<td>Get</td>
<td>Local</td>
<td>Indicates whether the queried value is valid or not.</td>
</tr>
</tbody>
</table>
Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4022.2 or higher</td>
<td>PC or CX (x64, x86, ARM)</td>
<td>Tc3_AES70</td>
</tr>
</tbody>
</table>

**GetReadingState**

The `GetReadingState` method is used to query the `ReadingState` property of the OCA object. If the query was successful, the result is output in the property `stReadingState`.

*Cyclic method call*

Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

**Inputs**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
</tbody>
</table>

### 7.1.3.3.2 FB_OcaTemperatureSensor

The function block `FB_OcaTemperatureSensor` provides properties and methods for querying OCA objects that map a temperature value.

*No explicit call*

Since there is no code in the body of the function block, it should not be called explicitly. Instead, use the corresponding methods of the function block.

**Inheritance hierarchy**

FB_OcaRoot

    FB_OcaWorker

    FB_OcaSensor

    FB_OcaTemperatureSensor
Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition location</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetReading</td>
<td>Local</td>
<td>Used to query the properties Reading, minReading and maxReading of the OCA object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the query was successful, the result is output in the properties stReading, stReadingMAX and stReadingMIN.</td>
</tr>
<tr>
<td>RNtfGain</td>
<td>Local</td>
<td>If the OCA object was prompted to notify changes (using the Subscription [24] method), this method is used to read incoming notifications.</td>
</tr>
</tbody>
</table>

AES70 standard

The names of variables and function blocks have been adapted to the AES70 standard where possible. Information about this communication protocol can be found at www.aes.org and www.ocaalliance.com.

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4022.2 or higher</td>
<td>PC or CX (x64, x86, ARM)</td>
<td>Tc3_AES70</td>
</tr>
</tbody>
</table>

GetReading / RNtfReading

GetReading

The GetReading method is used to query the property of the OCA object. If the query was successful, the result is output in the properties stReading, stReadingMAX and stReadingMIN.

Cyclic method call

Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
</tbody>
</table>

RNtfReading

If the OCA object was prompted to notify changes (using the Subscription [24] method), incoming notifications are read using the RNtfReading method.
Cyclic method call

Since several PLC cycles may pass between sending and the response from the device, this method should be executed cyclically.

Inputs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pLink</td>
<td>POINTER TO ST_Link</td>
<td>Pointer for establishing a connection between OCA objects and the OCA device.</td>
</tr>
<tr>
<td>bExecute</td>
<td>BOOL</td>
<td>The method is executed with a positive edge at the bExecute input.</td>
</tr>
<tr>
<td>nSubscriptionNo</td>
<td>UDINT</td>
<td>Unique number for localizing the respective subscription.</td>
</tr>
</tbody>
</table>

7.1.3.3.3 FB_OcaBasicSensor

The function block FB_OcaBasicSensor extends the function block FB_OcaSensor [52]. It is the function block from which all function blocks that are assigned to the BasicSensor category inherit. FB_OcaBasicSensor has neither properties nor methods.

Inheritance hierarchy

FB_OcaRoot
  FB_OcaWorker
    FB_OcaSensor
      FB_OcaBasicSensor

Requirements

<table>
<thead>
<tr>
<th>Development environment</th>
<th>Target platform</th>
<th>PLC libraries to include</th>
</tr>
</thead>
<tbody>
<tr>
<td>TwinCAT v3.1.4022.2 or higher</td>
<td>PC or CX (x64, x86, ARM)</td>
<td>Tc3_AES70</td>
</tr>
</tbody>
</table>

7.1.3.3.4 FB_OcaBooleanSensor

In terms of functionality, the function block FB_OcaBooleanSensor is similar to the function block FB_OcaBooleanActuator [43]. However, since sensors can only be read, there is no corresponding set method.

Inheritance hierarchy

FB_OcaRoot
  FB_OcaWorker
    FB_OcaSensor
      FB_OcaBasicSensor
7.1.3.3.5 FB_OcaStringSensor

In terms of functionality, the function block FB_OcaBooleanSensor is similar to the function block FB_OcaStringActuator. However, since sensors can only be read, there is no corresponding set method.

Inheritance hierarchy

FB_OcaRoot
  FB_OcaWorker
    FB_OcaSensor
      FB_OcaBasicSensor
      FB_OcaStringSensor

7.1.3.3.6 FB_OcaInt32Sensor

In terms of functionality, the function block FB_OcaBooleanSensor is similar to the function block FB_OcaInt32Actuator. However, since sensors can only be read, there is no corresponding set method.

Inheritance hierarchy

FB_OcaRoot
  FB_OcaWorker
    FB_OcaSensor
      FB_OcaBasicSensor
      FB_OcaInt32Sensor

7.2 Data types

7.2.1 Structures used to represent the properties of OCA objects

ST_OcaProperty

All structures listed below inherit from this structure.

```plaintext
TYPE ST_OcaProperty :
  STRUCT
    eState: E_OcaStatus := E_OcaStatus.Undefined;
    sPropString: STRING(5) := '00p00';
  END_STRUCT
END_TYPE
```

See also: E_OcaStatus [58]
### ST_BooleanProperty

```lisp
TYPE ST_BooleanProperty EXTENDS ST_OcaProperty :
STRUCT
  bVal: BOOL := FALSE;
END_STRUCT
END_TYPE
```

### ST_ClassIdProperty

```lisp
TYPE ST_ClassIdProperty EXTENDS ST_OcaProperty :
STRUCT
  nVal: ST_OcaClassId;
END_STRUCT
END_TYPE
```

### ST_FLOAT32Property

```lisp
TYPE ST_FLOAT32Property EXTENDS ST_OcaProperty:
STRUCT
  fVal: REAL;
END_STRUCT
END_TYPE
```

### ST_INT16Property

```lisp
TYPE ST_INT16Property EXTENDS ST_OcaProperty :
STRUCT
  nVal: INT;
END_STRUCT
END_TYPE
```

### ST_INT32Property

```lisp
TYPE ST_INT32Property EXTENDS ST_OcaProperty :
STRUCT
  nVal: DINT;
END_STRUCT
END_TYPE
```

### ST_INT8Property

```lisp
TYPE ST_INT8Property EXTENDS ST_OcaProperty :
STRUCT
  nVal: BYTE;
END_STRUCT
END_TYPE
```

### ST_MuteStateProperty

```lisp
TYPE ST_MuteStateProperty EXTENDS ST_OcaProperty :
STRUCT
  eVal: E_OcaMuteState := E_OcaMuteState.Unmuted ;
END_STRUCT
END_TYPE
```

See also: `E_OcaMuteState` [59]

### ST_PolarityStateProperty

```lisp
TYPE ST_PolarityStateProperty EXTENDS ST_OcaProperty :
STRUCT
END_STRUCT
END_TYPE
```

### ST_PositionEnabledProperty

```lisp
TYPE ST_PositionEnabledProperty EXTENDS ST_BooleanProperty:
STRUCT
  nIndex: UINT;
END_STRUCT
END_TYPE
```
ST_PositionNameProperty

TYPE ST_PositionNameProperty EXTENDS ST_StringProperty:
STRUCT
  nIndex: UINT;
END_STRUCT
END_TYPE

ST_SensorReadingState

TYPE ST_SensorReadingState EXTENDS ST_OcaProperty:
STRUCT
  eVal: E_OcaSensorReadingState := E_OcaSensorReadingState.eUnknown;
END_STRUCT
END_TYPE

ST_StringProperty

TYPE ST_StringProperty EXTENDS ST_OcaProperty:
STRUCT
  stVal: ST_OcaString;
END_STRUCT
END_TYPE

ST_SubscriptionManagerState

TYPE ST_SubscriptionManagerState EXTENDS ST_OcaProperty:
STRUCT
  eVal: E_OcaSubscriptionManagerState;
END_STRUCT
END_TYPE

ST_TemperatureProperty

TYPE ST_TemperatureProperty EXTENDS ST_OcaProperty:
STRUCT
  stVal: ST_OcaTemperature;
END_STRUCT
END_TYPE

See also: ST_OcaTemperature [59]

ST_UINT16Property

TYPE ST_UINT16Property EXTENDS ST_OcaProperty:
STRUCT
  nVal: UINT;
END_STRUCT
END_TYPE

ST_UDINT32Property

TYPE ST_UDINT32Property EXTENDS ST_OcaProperty:
STRUCT
  nVal: UDINT;
END_STRUCT
END_TYPE

ST_UINT8Property

TYPE ST_UINT8Property EXTENDS ST_OcaProperty:
STRUCT
  nVal: SINT;
END_STRUCT
END_TYPE

7.2.2 E_OcaStatus

The status code that identifies the result of the method invocation the response belongs to. E_OcaStatus has a size of 1 byte.

(*Status codes returned from method calls*)
(attribute 'qualified_only')
TYPE E_OcaStatus:
  { Ok := 0,
ProtocolVersionError := 1,
DeviceError := 2,
Locked := 3,
BadFormat := 4,
BadONo := 5,
ParameterError := 6,
ParameterOutOfRange := 7,
NotImplemented := 8,
InvalidRequest := 9,
ProcessingFailed := 10,
BadMethod := 11,
PartiallySucceeded := 12,
Timeout := 13,
BufferOverflow := 14,
DecodingError := 20,
Undefined := 21
)BYTE;
END_TYPE

7.2.3 E_OcaMuteState

{attribute 'qualified_only'}
{attribute 'strict'}
TYPE E_OcaMuteState:
{
    Muted:= 1,
    Unmuted:= 2
};
END_TYPE

7.2.4 ST_OcaTemperature

TYPE ST_OcaTemperature :
STRUCT
    fDegreesCelsius: REAL; //Value in Degrees Celsius
END_STRUCT
END_TYPE

7.2.5 ST_OcaDeviceInfo

Structure used to show informations about OCA devices.

TYPE ST_OcaDeviceInfo:
STRUCT
    tDeviceEnabledSince: DATE_AND_TIME;
    tClientConnectedSince: DATE_AND_TIME; //yet not used - for further extensions
    tLastSentMsg: DATE_AND_TIME;
    tLastReceivedMsg: DATE_AND_TIME;
    aReceivedMessages: ARRAY[E_OcaMessageType.OcaCmd..E_OcaMessageType.OcaKeepAlive] OF UDINT;
END_STRUCT
END_TYPE

7.2.6 E_OcaMessageType

Indicates the type of the message

{attribute 'qualified_only'}

TYPE E_OcaMessageType:
{
    OcaCmd:= 0, // Command - no Response Required
    OcaCmdRrq:= 1, // Command - Response Required
    OcaNtf:= 2, // Notification
    OcaRsp:= 3, // Response (to a command or notification)
    OcaKeepAlive:= 4, // Keep-alive message used for device supervision.
    Idle:= 7
);BYTE;
END_TYPE
8 Examples

8.1 Example for using the function block FB_OcaDevice

The example shows how to handle and parameterize the function block FB_OcaDevice. The function block FB_OcaDevice forms the basis for the use of further function blocks that can be used to read and modify OCA objects of an OCA device.

This example assumes that a device that supports the AES70 standard is connected.

Download: https://infosys.beckhoff.com/content/1033/tf8810_tc3_aes70/Resources/zip/4223793163.zip

8.2 Example for using the function block FB_OcaRoot

The example shows how the function block FB_OcaRoot can be used. The function block FB_OcaRoot provides basic functionalities and passes these on to all function blocks of the library that are used to read and modify OCA objects such as FB_OcaMute, FB_OcaSwitch etc.
This example assumes that a device that supports the AES70 standard is connected.

Download: https://infosys.beckhoff.com/content/1033/tf8810_tc3_aes70/Resources/zip/4223020171.zip

8.3 Example for using the function block FB_OcaWorker

The example shows how the function block FB_OcaWorker can be used.

This example assumes that a device that supports the AES70 standard is connected.
8.4 Example for using the function block FB_OcaGain

The example shows how to handle and parameterize the function block FB_OcaGain. Methods and properties that are inherited from FB_OcaWorker and therefore also from FB_OcaRoot are also used.

This example assumes that a device that supports the AES70 standard is connected.

Download: https://infosys.beckhoff.com/content/1033/tf8810_tc3_aes70/Resources/zip/4223791499.zip
9 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.

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 her internet pages:

http://www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG
Huelshorstweg 20
33415 Verl
Germany

Phone: +49(0)5246/963-0
Fax: +49(0)5246/963-198
e-mail: info@beckhoff.com

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(0)5246/963-157
Fax: +49(0)5246/963-9157
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(0)5246/963-460
Fax: +49(0)5246/963-479
e-mail: service@beckhoff.com